RESTful API v4.8 User Guide

RESTful API v4.8 User Guide

Version history

Version	Date	Edits
4.0	January 8, 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 14, 2019	• User Details • Global User Settings • Global Password Complexity Settings • Inventory Details
4.2	May 29, 2019	• Auto Discovery Status • Auto Discovery Profiles • Auto Discovery Profile
4.3	June 27, 2019	• User Details
4.4	July 22, 2019	• Inventory Summary • Servers • Servers Details • Zone Details • Object Attribute Details

4.5	January 17, 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 28, 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 2, 2020	• Inventory Details
4.8	January 7, 2021	• Inventory Summary • Inventory Details • Auto Discovery Profiles • Auto Discovery Profile

Overview

Introduction

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

ENA 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.

Version

Shows the version of the RESTful API in use.

URL: version

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	**curl -u admin:admin https://<ENA Server> /api/version?media=json**

OUTPUT	{
	"version" : "v5"
	}

INPUT	**curl -u admin:admin https://<ENA Server>/api/version?media=xml**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <versionInfo version="5" 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"/>

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	**curl -u admin:admin https://<ENA Server>/api/v1/version?media=json**

OUTPUT	{
	"version" : "v1"
	}

Domain Filters

Lists the filter id, name and serverId information of available domain filters.

URL: domainFilters (example http://localhost/api/domainFilters?media=json)

Methods Summary

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

GET Method detail

Lists available domain 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	`curl -u admin:admin  http://localhost/api/domainFilters?media=json`

OUTPUT	{
	"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
	}

INPUT	**curl -u admin:admin http://localhost/api/domainFilters?media=xml**
OUTPUT	<?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>

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	**curl -u admin:admin http://localhost/api/domainFilters?media=json -X POST -H "content-type: application/json" -d \
	'{
	  "name" : "A",
	  "rules" : [{
	  "SRCTYPE" : "4",
	  "DEVNAME" : "Two",
	  "ZONENAME" : "None"
	  }]
	}'**

OUTPUT	{
	  "name" : "A",
	  "systemFilter" : false,
	  "rules" : [{
	  "SRCTYPE" : "4",
	  "DEVNAME" : "Two",
	  "ZONENAME" : "None"
	  }]
	}

INPUT	curl -u admin:admin http://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="http://www.w3.org/2001 /XMLSchema-instance">SRCTYPE</key> <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001 /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">4< /value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="http://www.w3.org/2001 /XMLSchema-instance">DEVNAME</key> <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001 /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Two< /value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="http://www.w3.org/2001 /XMLSchema-instance">ZONENAME</key> <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001 /XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">None< /value> </entries> </rules> </domainFilterInfo>'

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

Domain Filter Details

Represents a set of operations on a particular domain filter.

URL: domainFilters/{filterId}

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 selectedomain 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	**curl -u admin:admin http://localhost/api/domainFilters/1002?media=json**

OUTPUT	{
	"name" : "A",
	"systemFilter" : false,
	"rules" : [{
	"SRCTYPE" : "DEVICE",
	"DEVNAME" : "Two",
	"ZONENAME" : "All"
	}]
	}

INPUT	**curl -u admin:admin http://localhost/api/domainFilters/1002?media=xml**

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

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	**curl -u admin:admin http://localhost/api/domainFilters/1002?media=json -X PUT -H "content-type: application/json" -d ** **'{** **"name" : "B",** **"rules" : [{** **"SRCTYPE" : "APPLICATION",** **"DEVTYPE" : "158",** **"ZONENAME" : "None"** **}]** **}'**

OUTPUT	{
	"name" : "B",
	"rules" : [{
	"SRCTYPE" : "APPLICATION",
	"DEVTYPE" : "158",
	"ZONENAME" : "None"
	}]
	}

INPUT	curl -u admin:admin http://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="http://www.w3.org/2001/XMLSchema-instance">SRCTYPE</key> <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">APPLICATION</value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">ZONENAME</key> <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">None</value> </entries> <entries> <key xsi:type="opCode" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">DEVNAME</key> <value xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Two</value> </entries> </rules> </domainFilterInfo>'

OUTPUT

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

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	**curl -u admin:admin http://localhost/api/domainFilters/1002?media=json -X DELETE**

OUTPUT	{
	"message" : "OK"
	}

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

Event Filters

Lists minimal information about available event filters.

URL: eventFilters

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	`curl -u admin:admin  http://localhost/api/eventFilters?media=json`

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

INPUT	**curl -u admin:admin http://localhost/api/eventFilters?media=xml**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="namedItem" name="All Events" id="1" serverId="98eb1254-0d30-4c7d-8910-ac610cd5e351" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

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	

Examples

INPUT

```
curl -u admin:admin  http://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
}'
```

OUTPUT

```
{
  "name" : "A",
  "selectedNames" : [
    "AvailMonitor Application Unavailable",
    "AvailMonitor High Latency Reaching Application Cleared"
  ],
  "systemFilter" : false,
  "passIP" : true
}
```

INPUT	**curl -u admin:admin http://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>'**
OUTPUT	<?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>

Event Filter Details

Represents a set of operations on a particular event filter.

URL: eventFilters/{filterId}

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	`curl -u admin:admin  http://localhost/api/eventFilters/1001?media=json`

OUTPUT	{
	"name": "A",
	"selectedNames": [
	"AvailMonitor Application Unavailable",
	"AvailMonitor High Latency Reaching Application Cleared"
	],
	"systemFilter": false,
	"passIP": true
	}

INPUT	**curl -u admin:admin http://localhost/api/eventFilters/1001?media=xml**
OUTPUT	<?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>

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

```
curl -u admin:admin  http://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
}'
```

OUTPUT

```
{
  "name" : "B",
  "selectedNames" : [
    "WAN Port Low Outbound Utilization Cleared",
    "AvailMonitor Application Unavailable",
    "AvailMonitor High Latency Reaching Application Cleared"
  ],
  "systemFilter" : false,
  "passIP" : false
}
```

INPUT	**curl -u admin:admin http://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>'**
OUTPUT	<?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>

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	**curl -u admin:admin http://localhost/api/eventFilters/1001?media=json -X DELETE**

OUTPUT	{
	"message" : "OK"
	}

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

Incident Filters

List available incident filters, and create new filters.

URL: incidentFilters

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	`curl -u admin:admin http://boron/api/incidentFilters?media=json`

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

INPUT	**curl -u admin:admin http://boron/api/incidentFilters?media=xml**
OUTPUT	<?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>

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

```
curl -u admin:admin http://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
}'
```

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

INPUT	**curl -u admin:admin http://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>'**
OUTPUT	<?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>

Incident Filter Details

Show details about a particular incident filter, modify the parameters of a filter, and delete a filter.

URL: incidentFilters/{filterId}

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	`curl -u admin:admin http://localhost/api/incidentFilters/2?media=json`

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 http://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

```
curl -u admin:admin http://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
}'
```

OUTPUT

```
{
  "name" : "B",
  "selectedNames" : [
    "AP Antenna Host Count Abnormality",
    "AP Antenna Power Change Frequency High",
    "AP Not Associated With Controller"
  ],
  "systemFilter" : false,
  "passIP" : true
}
```

INPUT

```
curl -u admin:admin http://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>'
```

OUTPUT	<?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>

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 http://localhost/api/incidentFilters/2?media=json -X DELETE "content-type: application/json"**

OUTPUT	{
	"message" : "OK"
	}

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

Events

List available events, or raise a new event.

URL: events

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 : swId StormWorks Identifier (integer) compId 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	`curl -u admin:admin http://localhost/api/events?media=json`

OUTPUT


```
{
  “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} ]
```


INPUT

```
curl -u admin:admin http://localhost/api/events?media=xml
```

OUTPUT

```
<?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>
```

POST Method details

Raises an event.

Request

Name	Description
id	Event type

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
sourceDescription	Source description

Examples

INPUT	**curl -u admin:admin http://localhost/api/events?media=json -X POST -H "content-type: application/json" -d ** **'{** **"reason" : "API test",** **“name” : “New Event”** **“id” : 786441,** **“impactDescription” : “Entuity service”** **“severity” : 6,** **“objectKeyInfo” : { “swId” : 12345, “compId” : { “ids” : [4096, 0, 0, 0] } }** **}'**

OUTPUT	{
	"message" : "Event sent"
	}

INPUT	**curl -u admin:admin http://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>’**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>Event sent</message> </statusInfo>

Event Types

List available event types.

URL: eventTypes

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	`curl -u admin:admin http://localhost/api/eventTypes?media=json`

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

INPUT	**curl -u admin:admin http://localhost/api/eventTypes?media=xml**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> ...removed for brevity... <item xsi:type="eventType" severity="2" name="WAN Port Low Outbound Utilization Cleared" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" /> </items>

Incidents

Lists available incidents.

URL: incidents

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 : swId StormWorks Identifier (integer) compId 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	`curl -u admin:admin http://localhost/api/incidents?media=json`

OUTPUT


```
{
  "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
}
```


INPUT

```
curl -u admin:admin http://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 Details

Change state, annotation and attributes of an incident.

URL: incidents/{incidentID}

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	**curl -X PUT -u admin:admin http://localhost/api/incidents/69?media=json -H "content-type:application/json" -d ** **'{ "state" : "open",** **"annotation" : "New annotation",** **"attribute" : {"key" : "attribKey", "value": "New value" }** **}'**

OUTPUT	{
	"message" : "Incident opened, Attribute 'attribKey' updated"
	}

INPUT	**curl -X PUT -u admin:admin http://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>'**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>Incident opened, Attribute 'attribKey' updated</message> </statusInfo

Incident Types

Lists incident types.

URL: incidentTypes

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	`curl -u admin:admin http://localhost/api/incidentTypes?media=json`

OUTPUT	`[ {`
	`  "name" : "AP Antenna Channel Change Frequency High"`
	`}, {`
	`  ...removed for brevity...`
	`}, {`
	`  "name" : "Wireless Controller High Number of Connected APs"`
	`} ]`

INPUT	**curl -u admin:admin http://localhost/api/incidentTypes?media=xml**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> ...removed for brevity... <item xsi:type="incidentType" name="Wireless Controller High Number of Connected APs" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

Inventory Summary

List the inventory on a server, or add a new device to the inventory.

URL: inventory

Method Summary

- 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).

GET Method detail

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	`curl –u admin:admin http://localhost/api/inventory?media=json`

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

INPUT	**curl –u admin:admin http://localhost/api/inventory?media=xml**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> </items>

POST Method detail

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
duplicateIps	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 behaiour set the flag to false or omit the flag altogether.

Request Parameters

Name	Description
authKey	SNMP v3 authentication key.
authPass	*authType* password.
authType	SNMP v3 authentication type: • none • MD5 • SHA • SHA224 • SHA256 • SHA384 • SHA512
cliMethod	Connection method: SSH, TELNET
cliPassword1	Password 1
cliPassword2	Password 2
cliPort	Connection port
cliUsername	CLI User name
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.
encrKey	SNMP v3 encryption key.

encrPass	*encrType* password.
encrType	SNMP v3 encryption type: • NONE. • DES. • DES3. • AES. • AES192. • AES256.
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.
polledName	DNS name or IP Address.
protocol	transport protocol: IPv4, IPv6.
readCommunity	SNMP v1/v2 read community string.
writeCommunity	SNMP v1/v2 write community string.
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).
userName	SNMP v3 user name.
webAccessKey	access key (Amazon platform only).
webPassword	virtual platform password (non-Amazon virtual platforms).
webPlatformType	virtual platform type: • VMWARE_ESXi • ORACLE_VM_MANAGER • HYPER_V • AMAZON_WEB_SERVICE • AZURE • CISCO_APIC • MERAKI
webSecretKey	secret key (Amazon platform only).
webTenantID	password for Microsoft Azure platform.
webURL	virtual platform URL (non-Amazon virtual platforms).
webUser	virtual platform user name (non-Amazon virtual platforms).

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	**curl -u admin:admin http://localhost/api/inventory?media=json&duplicateIps=false -X POST –H “content-type: application /json” -d ** **‘{** **"polledName" : "10.66.23.1",** **"managementLevel" : "FULL",** **"protocol" : "IPv4",** **"snmpType" : "v2c",** **"readCommunity" : "public"** **}’**

OUTPUT	{
	"message": "Queued"
	}

INPUT	**curl -u admin:admin http://localhost/api/inventory?media=xml&duplicateIps=false -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"** **readCommunity="public"** **/>’**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>Queued</message> </statusInfo>

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

INPUT	**curl -H 'Content-type: application/json' -u admin:admin -X POST 'http://localhost/api/inventory?media=json' -d ** **'{** **"nameUsing" : "POLLEDNAME",** **"polledName" : "10.66.33.2",** **"managementLevel" : "FULL",** **"protocol" : "IPv4",** **"snmpType" : "v2c",** **"readCommunity" : "public",** **}'**

OUTPUT	{
	"message" : "Queued"
	}

To specify a Ping Only device:

INPUT	**curl -H 'Content-type: application/json' -u admin:admin -X POST 'http://localhost/api/inventory?media=json' -d ** **'{** **"nameUsing" : "POLLEDNAME",** **"polledName" : "10.66.33.2",** **"managementLevel" : "PING_ONLY",** **"protocol" : "IPv4"** **}'**

OUTPUT	{
	"message" : "Queued"
	}

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

i). To add an Amazon Web Services platform:

INPUT	**curl -H 'Content-type: application/json' -u admin:admin -X POST 'http://localhost/api/inventory?media=json' -d ** **'{** **"managementLevel" : "WEB",** **"polledName" : "AWS",** **"deviceType" : "VM Platform",** **"webPlatformType" : "AMAZON_WEB_SERVICE",** **"webAccessKey" : "ABCDEFG",** **"webSecretKey" : "xyxyxyxyxyxyxyx"** **}'**

OUTPUT	{
	"message" : "Queued"
	}

ii). To add a Meraki Cloud Controller platform:

INPUT	**curl -H "Content-type: application/json" -u admin:admin -X POST 'http://localhost/api/inventory?media=json' -d ** **'{** **"managementLevel" : "WEB",** **"polledName" : "Meraki",** **"deviceType" : "Cloud Controller",** **"webPlatformType" : "MERAKI",** **"webURL" : "https://api.meraki.com/api/v0/",** **"webPassword" : "123456789"** **}'**

OUTPUT	{
	"message" : "Queued"
	}

iii). To add an Azure platform:

INPUT	**curl -H 'Content-Type:application/json'-u admin:admin -X POST http://localhost/api/inventory -d ** **'{** **"managementLevel" : "WEB",** **"polledName" : "AZURE",** **"deviceType" : "VM Platform",** **"webPlatformType" : "AZURE",** **"webTenantID" : "TenantID",** **"webSecretKey" : "secretKey",** **"webAccessKey" : "vmAccessKey",** **}'**

OUTPUT	{
	"message" : "Queued"
	}

iv). To add a Cisco APIC SDN Controller:

INPUT	**curl -H 'Content-Type:application/json' -u admin:admin -X POST http://localhost/api /inventory -d ** **'{** **"managementLevel" : "WEB",** **"polledName" : "APIC",** **"deviceType" : "SDN Controller",** **"webPlatformType" : "CISCO_APIC",** **"webURL" : "https://myapicname.cisco.com/api/",** **"webUser" : "admin",** **"webPassword" : "password",** **}'**

OUTPUT	{
	"message" : "Queued"
	}

v). To add a VMWare Platform:

INPUT	**curl -H 'Content-Type:application/json' -u admin:admin -X POST http://localhost/api/inventory -d ** **'{** **"managementLevel" : "WEB",** **"polledName" : "vortex",** **"deviceType" : "VM Platform",** **"webPlatformType" : "VMWARE_ESXi",** **"webURL" : "https://vortex/sdk",** **"webUser" : "root",** **"webPassword" : "password",** **"protocol" : "IPv4"** **}'**

OUTPUT	{
	"message" : "Queued"
	}

vi). To add a Viptela device:

INPUT	**curl -H 'Content-Type:application/json' -u admin:admin -X POST http://localhost/api/inventory?media=json' -d ** **'{** **"managementLevel" : "WEB",** **"polledName" : "Viptela1",** **"deviceType" : "Cloud Controller",** **"webPlatformType" : "VIPTELA",** **"webURL" : "https://api.viptela.com/api/v1",** **"webUser" : "entuity",** **"webPassword" : "12345"** **}'**

OUTPUT	{
	"message" : "Queued"
	}

vii). To add a Cisco DNA Center:

INPUT	**curl -H 'Content-Type:application/json' -u admin:admin -X POST http://localhost/api/inventory?media=json' -d ** **'{** **"managementLevel" : "WEB",** **"polledName" : "DNAC1",** **"deviceType" : "Cisco DNA Center",** **"webPlatformType" : "CISCO_DNA_CENTER",** **"webURL" : "https://1.2.3.4",** **"webUser" : "entuity",** **"webPassword" : "12345"** **}'**

OUTPUT	{
	"message" : "Queued"
	}

Inventory Details

List, update or delete an inventory device.

URL: inventory/{deviceId}

Method Summary

- 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.

GET Method detail

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

Request Parameters

None.

Response data keys

Name	Description
authKey	SNMP v3 authentication key.

authType	SNMP v3 authentication type: • none • MD5 • SHA • SHA224 • SHA256 • SHA384 • SHA512
capabilities	device capabilities: routing, switching, switching & routing.
certified	if device has been certified.
deviceType	device type.
dsObjectId	device's stormworks identifier.
encrKey	SNMP v3 encryption key.
encrType	SNMP v3 encryption type: • none • DES • DES3 • AES • AES192 • AES256.
context	SNNP v3 context.
id	device's unique identifier.
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.
polledName	DNS name or IP Address.

protocol	Transport protocol: IPv4 or IPv6.
readCommunity	SNMP read community string.
serverId	server identifier.
snmpPDUSize	maximum size of SNMP PDU, where 0 = system default.
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.
sysDescription	SNMP description field.
sysLocation	SNMP retrieved system Location field.
sysOid	SNMP system identifier field.
username	SNMP v3 user name.
webAccessKey	access key (Amazon platform only).
webPassword	virtual platform password (Non Amazon virtual platforms).
webPlatformType	virtual platform type: • VMWARE_ESXi • ORACLE_VM_MANAGER • HYPER_V • AMAZON_WEB_SERVICE • AZURE • CISCO_APIC • MERAKI
webSecretKey	secret key (Amazon platform only).
webTenantID	password for Microsoft Azure platform.
webURL	virtual platform URL (non-Amazon virtual platforms).
webUser	virtual platform username (non-Amazon virtual platforms).
zoneId	the ID of a zone.

Examples

INPUT	`curl -u admin:admin  http://localhost/api/inventory/2?media=json`

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",
  "sysOid": ".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

```
curl -u admin:admin  http://localhost/api/inventory/2?media=xml
```

OUTPUT	<?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"/>

PUT Method detail

Change the details of an inventory device, in either XML or JSON formats. Please refer to the ENA Help Center for help and information on how to add devices to your network from the ENA UI.

Request Parameters

Name	Description
authKey	SNMP v3 authentication key.
authPass	SNMP v3 authentication password.
authType	SNMP v3 authentication type: • none • MD5 • SHA • SHA224 • SHA256 • SHA384 • SHA512
cliMethod	Connection method: SSH, TELNET.
cliPassword1	password 1.
cliPassword2	password 2.
cliPort	connection port.
cliUsername	CLI User name.
context	SNMP v3 context.
encrKey	SNMP v3 encryption key.
encrPass	SNMP v3 encryption password.
encrType	SNMP v3 encryption type: NONE, DES, AES.
name	display name.

nameUsing	device name is determined using either: • CUSTOMNAME • IPADDRESS • POLLEDNAME • RESOLVABLENAME • RESOLVABLENAMEFQ • SYSTEMNAME.
protocol	Transport protocol: IPv4 or IPv6.
readCommunity	SNMP read community string.
snmpPDUSize	maximum size of snmp PDU, where 0 = system default.
snmpRetry	number of snmp retries, where 0 = system default.
snmpTimeout	SNMP timeout in seconds, where 0 = system default
snmpType	SNMP type: • v1 • v2c • v3 • v1/v2c.
userName	SNMP v3 user name.
webAccessKey	access key (Amazon platform only).
webPassword	virtual platform password (non-Amazon virtual platforms).
webPlatformType	virtual platform type: • VMWARE_ESXi • ORACLE_VM_MANAGER • HYPER_V • AMAZON_WEB_SERVICE • AZURE • CISCO_APIC • MERAKI
webSecretKey	secret key (Amazon platform only).
webTenantID	password for Microsoft Azure platform.
webURL	virtual platform URL (non-Amazon virtual platforms).
webUser	virtual platform user name (non-Amazon virtual platforms).

Response data

200 OK

Examples

INPUT	curl -H -u admin:admin http://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"
	}'

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",
  "sysOid": ".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	**curl -H -u admin:admin http://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"** **/>'**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <inventoryDevice cliUsername="" zoneId="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"/>

INPUT	**curl -H -u admin:admin http://localhost/api/inventory/<deviceId>?media=json -X PUT -H "content-type: application/json" -d ** **'{** **"polledName" : "e2821",** **"snmpType" : "v3",** **"userName" : "e2821user",** **"authType" : "SHA",** **"authKey" : "BE27129CF8E393C2E95C590579188A4824B238B7",** **"encrType" : "AES",** **"encrKey" : "E7DB0CFE9A77DE488A3CC66200377C257A9A9AE2"** **}'**
OUTPUT	

DELETE Method details

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	**curl -u admin:admin http://localhost/api/inventory/2?media=json –X DELETE**

OUTPUT	{
	  "message": "OK"
	}

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

Auto Discovery Status

View a summary of the current status of the Auto Discovery, and start a discovery.

URL: autodiscovery

Method Summary

- GET Method - returns a summary of the current status of the Auto Discovery, and its scheduling (XML, JSON).
- POST Method - given a profile name, will start a discovery with that profile.

GET Method detail

Will show two summaries: a "status" summary and a "scheduler" summary.

The "status" summary will display the current situation of the discovery, such as any running profiles, and any profiles that are queued to run.

The "scheduler" summary will display the schedule details, such as the next scheudled profile to run, when it is going to run.

Request Parameters

None.

Response data keys

Name	Description
status	**this section contains the following items:**
concurrentAllowed	gives the maximum number of concurrent autodiscoveries allowed to run at one time.
queuedProfiles	a list of profiles that are queued to run.

currentlyRunning	a list of statuses of any running Auto Discoveries.
scheduler	**this section contains the following items:**
nextScheduledTime	the soonest schedule to run in millisecond time.
nextScheduledProfile	the soonest profile that would run.
schedules	a map of profiles that have schedules, and their details, consisting of: Key: Profile name. "nextRunTime": the millisecond time of the next run time of this schedule. "scheduleString": a string representation of the chosen schedule, which is a weekday followed by the hour time in which the schedule should start.

The "currentlyRunning" list contains data with these keys:

Name	Description
profile	name of this profile.
running	is true if this profile is running.
haveResults	is true if this profile already has results.
timezoneOffset	shows the timezone offset of the server.
startTime	the time this profile started running.
endTime	the time this profile last finished.
lastUpdateTime	(Web UI only) the time this profile was last queried.
queueInfo	status of the discovery. Consists of: "pingQueueSize": number of devices to ping. "pingItemsProcessed": number of ping devices already processed. "snmpQueneSize

Examples

No Auto Discoveries running, no schedules:

INPUT	`curl -u admin:admin  http://localhost/api/autodiscovery?media=json`

OUTPUT	<pre>{
	"status" : {
	  "concurrentAllowed" : 10,
	  "queuedProfiles" : [ ],
	  "currentlyRunning" : [ ]
	 },
	"scheduler" : {
	  "nextScheduledTime" : null,
	  "nextScheduledProfile" : null,
	  "schedules" : { }
	}
	}</pre>

Multiple Auto Discoveries running, one queued:

INPUT	**curl -u admin:admin http://localhost/api/autodiscovery?media=json**

OUTPUT	<pre> {
	"status" : {
	"concurrentAllowed" : 10,
	"queuedProfiles" : ["One"],
	"currentlyRunning" : [{
	"profile" : "Default (copy 3)",
	"running" : true,
	"haveResults" : false,
	"timezoneOffset" : 3600000,
	"startTime" : 1559134759282,
	"endTime" : null,
	"lastUpdateTime" : 1559134760642,
	"queueInfo" : {
	"pingQueueSize" : 256,
	"pingItemsProcessed" : 0,
	"snmpQueueSize" : 5,
	"snmpItemsProcessed" : 5
	},
	"messages" : null
	}, {</pre>

```
"profile" : "Default (copy 2)",
"running" : true,
"haveResults" : false,
"timezoneOffset" : 3600000,
"startTime" : 1559134759282,
"endTime" : null,
"lastUpdateTime" : 1559134760645,
"queueInfo" : {
"pingQueueSize" : 256,
"pingItemsProcessed" : 0,
"snmpQueueSize" : 5,
"snmpItemsProcessed" : 5
},
"messages" : null
},
. . . . . .
 {
"profile" : "Default (copy)",
"running" : true,
"haveResults" : false,
"timezoneOffset" : 3600000,
"startTime" : 1559134759285,
"endTime" : null,
"lastUpdateTime" : 1559134760548,
"queueInfo" : {
"pingQueueSize" : 256,
"pingItemsProcessed" : 0,
"snmpQueueSize" : 0,
"snmpItemsProcessed" : 0
},
"messages" : null
} ]
},
"scheduler" : {
"nextScheduledTime" : null,
"nextScheduledProfile" : null,
```

	"schedules" : { } } }

Schedules set:

INPUT	**curl -u admin:admin http://localhost/api/autodiscovery?media=json**

OUTPUT	{
	"status" : {
	"concurrentAllowed" : 10,
	"queuedProfiles" : [ ],
	"currentlyRunning" : [ ]
	},
	"scheduler" : {
	"nextScheduledTime" : 1559192400000,
	"nextScheduledProfile" : "Test",
	"schedules" : {
	"One" : {
	"nextRunTime" : 1559577600000,
	"scheduleString" : "MONDAY:17"
	},
	"Test" : {
	"nextRunTime" : 1559192400000,
	"scheduleString" : "ALL:6"
	},
	"Two" : {
	"nextRunTime" : 1559250000000,
	"scheduleString" : "THURSDAY:22"
	}
	}
	}
	}

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
profile	name of the profile to be run.

Response data

Shows the summary if the call was successful. The given profile should be either in the "queuedProfiles" list or the "currentlyRunning" list.

Examples

INPUT	`curl -u admin:admin http://localhost/api/autodiscovery?media=json -X POST –H “content-type: application/json” -d \ ‘{"profile" : "Default"}’`

OUTPUT


```
{
  "status" : {
  "concurrentAllowed" : 10,
  "queuedProfiles" : [ ],
  "currentlyRunning" : [ {
  "profile" : "Default",
  "running" : true,
 "haveResults" : false,
 "timezoneOffset" : 3600000,
 "startTime" : 1559135129610,
 "endTime" : null,
 "lastUpdateTime" : 1559135129642,
 "queueInfo" : {
 "pingQueueSize" : 0,
 "pingItemsProcessed" : 0,
 "snmpQueueSize" : 0,
 "snmpItemsProcessed" : 0
 },
 "messages" : null
 } ]
 },
  "scheduler" : {
 "nextScheduledTime" : null,
  "nextScheduledProfile" : null,
 "schedules" : { }
 }
 }
```

Auto Discovery Profiles

View and create Auto Discovery profiles.

URL: autodiscoveryProfiles

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	`curl -u admin:admin http://localhost/api/autodiscoveryProfiles?media=json`

OUTPUT	<pre>{
	"items" : ["Default (copy 1)", "Default (copy 2)", "Default (copy 3)", "Default (copy 4)", "Default (copy 5)", "Default (copy 6)", "Default (copy)", "Default", "One", "Test", "Two"],
	"count" : 11
	}</pre>

POST Method detail

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

Request Parameters

Name	Description
profile	name of the profile.
managementLevel	value for management level. Possible values are: • Full • Full (No Ports) • Basic • Ping Only • None
snmpPorts	list of ports to try.

snmpAuthOptions	map of authentication details to try. SNMP v3 passwords are hidden. Each option consists of:
	
	• "snmpVersion":
	  • 0: SNMP v1
	  • 1: SNMP v2
	  • 3: SNMP v3
	• "description": A representation of this authentication detail
	
	This value only applies for SNMP v1 and SNMP v2.
	
	• "readCommunity": Community string
	
	The following values only apply for SNMP v3.
	
	• "userName": Username
	• "authTypeName": Description of authentication type.
	• "authTypeId": Id of type of authentication
	  • 1: None
	  • 2: MD5
	  • 3: SHA
	  • 4: SHA224
	  • 5: SHA256
	  • 6: SHA384
	  • 7: SHA512
	• "authPassword": Password for device. Is null if hidden.
	• "encryptionTypeName": Description of authentication type.
	• "encryptionTypeId": Id of type of authentication.
	  • 1: None
	  • 2: DES
	  • 3: 3DES
	  • 4: AES
	  • 20: AES192
	  • 21: AES256
	• "encryptionPassword": Password for encryption. Is null if hidden.
	• "changed": indicates if this value was changed. This must be true to set any SNMP v3 passwords.

includedAddresses	list of ranges of address to ping. An IP range can be in any of these formats: • "<ip1>-<ip2>": devices with IP addresses between ip1 and ip2, inclusive • "<ip1>/<ip2>": subnet with first address ip1 and bit-length of ip2 • "<name>": single device with this IP address or name
excludedAddresses	list of ranges of address to not ping. Uses the same IP ranges as "includedAddresses".
schedule	schedule for repeated autodiscovery runs. Consists of "<weekday>:<hour>"

options	list of arguments to values. Refer to: https://support.entuity.com/hc/en-us/articles/360007731454-How-do-I-discover-what-is-on-my-network-via-the-command-line-#syntax In addition, the argument "-vid" can be used to set a view Id that discovered devices will be added to. If this view Id is shown to be -1, then it indicates that the view needs permissions that the user doesn't have. The argument "-N" is used to set the Name Using field. The allowed values are: • CustomName • SystemName • IpAddress • ResolvableName • ResolvableNameFQ • PolledName
viewPath	path of view to be added to. The path is in the format "<user>::My Network/<path>". If the user has inadequate permissions this value is instead "//INACCESSIBLE//".

Response data

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

Examples

INPUT	**curl -u admin:admin http://localhost/api/autodiscoveryProfiles?media=json -X POST –H “content-type: application/json” -d \ ‘{"profile" : "Three"}’**

OUTPUT	{
	"items" : ["Default (copy 1)", "Default (copy 2)", "Default (copy 3)", "Default (copy 4)", "Default (copy 5)", "Default (copy 6)", "Default (copy)", "Default", "One", "Test", "Three", "Two"],
	"count" : 12
	}

Auto Discovery Profile

View, edit and delete Auto Discovery Profiles.

URL: autodiscoveryProfiles/{profileName}

The profile name should be encoded using URL encoding, such that any non-URL characters are changed to percentage encoding. Plus signs "+" can be used to encode spaces.

For example:

Profile name: "This + That"

URL: "autodiscoveryProfiles/This+%2B+That"

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
profile	name of the profile.
managementLevel	value for management level. Possible values are: • Full • Full (No Ports) • Basic • Ping Only • None
snmpPorts	list of ports to try

snmpAuthOptions	map of authentication details to try. SNMP v3 passwords are hidden. Each option consists of:
	• "snmpVersion":
	  • 0: SNMP v1
	  • 1: SNMP v2
	  • 3: SNMP v3
	• "description": A representation of this authentication detail
	This value only applies for SNMP v1 and SNMP v2.
	• "readCommunity": Community string
	The following values only apply for SNMP v3.
	• "userName": Username
	• "authTypeName": Description of authentication type.
	• "authTypeId": Id of type of authentication
	  • 1: None
	  • 2: MD5
	  • 3: SHA
	  • 4: SHA224
	  • 5: SHA256
	  • 6: SHA384
	  • 7: SHA5123
	• "authPassword": Password for device. Is null if hidden.
	• "encryptionTypeName": Description of authentication type.
	• "encryptionTypeId": Id of type of authentication.
	  • 1: None
	  • 2: DES
	  • 3: 3DES
	  • 4: AES
	  • 20: AES192
	  • 21: AES256
	• "encryptionPassword": Password for encryption. Is null if hidden.
	• "changed": indicates if this value was changed. This must be true to set any SNMP v3 passwords.

includedAddresses	list of ranges of address to ping. An IP range can be in any of these formats: • "<ip1>-<ip2>": devices with IP addresses between ip1 and ip2, inclusive • "<ip1>/<ip2>": subnet with first address ip1 and bit-length of ip2 • "<name>": single device with this IP address or name
excludedAddresses	list of ranges of address to not ping. Uses the same IP ranges as "includedAddresses".
schedule	schedule for repeated autodiscovery runs. Consists of "<weekday>:<hour>"
options	list of arguments to values. Refer to: https://support.entuity.com/hc/en-us/articles/360007731454-How-do-I-discover-what-is-on-my-network-via-the-command-line-#syntax In addition, the argument "-vid" can be used to set a view Id that discovered devices will be added to. If this view Id is shown to be -1, then it indicates that the view needs permissions that the user doesn't have. The argument "-N" is used to set the Name Using field. The allowed values are: • CustomName • SystemName • IpAddress • ResolvableName • ResolvableNameFQ • PolledName
viewPath	path of view to be added to. The path is in the format "<user>::My Network/<path>". If the user has inadequate permissions this value is instead "//INACCESSIBLE//".

Examples

INPUT	**`curl -u admin:admin http://localhost/api/autodiscoveryProfiles/Test?media=json`**

OUTPUT	{
	"profile" : "Test",
	"managementLevel" : "Full",
	"snmpPorts" : [161],
	"snmpAuthOptions" : {
	"1" : {
	"snmpVersion" : 1,
	"readCommunity" : "public",
	"userName" : null,
	"authTypeName" : null,

```
"authTypeId" : null,
"authPassword" : null,
"encryptionTypeName" : null,
"encryptionTypeId" : null,
"encryptionPassword" : null,
"engineId" : null,
"changed" : false,
"description" : "-v 2 -c public"
}
},
"includedAddrs" : [ "10.44.1.0-10.44.1.255" ],
"excludedAddrs" : null,
"schedule" : null,
"options" : {
"-N" : "resolvableName",
"-excludesysoids" :
"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",
"-f" : "30",
"-hn" : "true",
"-ith" : "512",
"-nodb" : "true",
"-pt" : "3",
"-sth" : "64"
},
"viewPath" : null
}
```


PUT Method detail

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

Request Parameters

Name	Description

profile	name of the profile.
managementLevel	value for management level. Possible values are: • Full • Full (No Ports) • Basic • Ping Only • None
snmpPorts	list of ports to try

snmpAuthOptions	map of authentication details to try. SNMP v3 passwords are hidden. Each option consists of:
	• "snmpVersion":
	  • 0: SNMP v1
	  • 1: SNMP v2
	  • 3: SNMP v3
	• "description": A representation of this authentication detail
	
	This value only applies for SNMP v1 and SNMP v2.
	
	• "readCommunity": Community string
	
	The following values only apply for SNMP v3.
	
	• "userName": Username
	• "authTypeName": Description of authentication type.
	• "authTypeId": Id of type of authentication
	  • 1: None
	  • 2: MD5
	  • 3: SHA
	  • 4: SHA224
	  • 5: SHA256
	  • 6: SHA384
	  • 7: SHA512
	• "authPassword": Password for device. Is null if hidden.
	• "encryptionTypeName": Description of authentication type.
	• "encryptionTypeId": Id of type of authentication.
	  • 1: None
	  • 2: DES
	  • 3: 3DES
	  • 4: AES
	  • 20: AES192
	  • 21: AES256
	• "encryptionPassword": Password for encryption. Is null if hidden.
	• "changed": indicates if this value was changed. This must be true to set any SNMP v3 passwords.

includedAddresses	list of ranges of address to ping. An IP range can be in any of these formats: • "<ip1>-<ip2>": devices with IP addresses between ip1 and ip2, inclusive • "<ip1>/<ip2>": subnet with first address ip1 and bit-length of ip2 • "<name>": single device with this IP address or name

excludedAddresses	list of ranges of address to not ping. Uses the same IP ranges as "includedAddresses".
schedule	schedule for repeated autodiscovery runs. Consists of "<weekday>:<hour>"
options	list of arguments to values. Refer to: https://support.entuity.com/hc/en-us/articles/360007731454-How-do-I-discover-what-is-on-my-network-via-the-command-line-#syntax In addition, the argument "-vid" can be used to set a view Id that discovered devices will be added to. If this view Id is shown to be -1, then it indicates that the view needs permissions that the user doesn't have. The argument "-N" is used to set the Name Using field. The allowed values are: • CustomName • SystemName • IpAddress • ResolvableName • ResolvableNameFQ • PolledName
viewPath	path of view to be added to. The path is in the format "<user>::My Network/<path>". If the user has inadequate permissions this value is instead "//INACCESSIBLE//".

Response data

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

Examples

Modifying managementLevel:

INPUT	`curl -u admin:admin http://localhost/api/autodiscoveryProfiles/Test?media=json -X POST –H “content-type: application/json” -d \ ‘{"managementLevel" : "Ping Only"}’`

OUTPUT


```
{
"profile" : "Test",
"managementLevel" : "Ping Only",
"snmpPorts" : [ 161 ],
"snmpAuthOptions" : {
"1" : {
"snmpVersion" : 3,
"readCommunity" : null,
"userName" : "e",
"authTypeName" : "MD5",
"authTypeId" : 2,
"authPassword" : null,
"encryptionTypeName" : null,
"encryptionTypeId" : 1,
"encryptionPassword" : null,
"engineId" : null,
"changed" : false,
"description" : "-u \"e\" -a \"MD5\" -A \"****\""
}
},
"includedAddrs" : [ "10.44.1.0-10.44.1.255" ],
"excludedAddrs" : null,
"schedule" : null,
"options" : {
"-N" : "ipAddress",
"-excludesysoids" :
"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",
"-f" : "30",
"-hn" : "true",
"-ith" : "512",
"-nodb" : "true",
"-pt" : "3",
"-sth" : "64"
},
"viewPath" : null
```

DELETE Method detail

Removes the given profile.

Request Parameters

None.

Response data

None.

Examples

Product Info

Details information about the installed version of Entuity/ENA.

URL: info

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	`curl -u admin:admin http://localhost/api/info?media=json`

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

INPUT	**curl -u admin:admin http://localhost/api/info?media=xml**
OUTPUT	<?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"/>

Servers

List servers.

URL: servers

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	`curl -u admin:admin  http://localhost/api/servers?media=json`

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

INPUT	**curl -u admin:admin http://localhost/api/servers?media=xml**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> </items>

Servers Details

List details of a server.

URL: servers/{serverId}

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	**curl -u admin:admin http://localhost/api/server/34564e92-3b4f-43ba-94a8-04309c0e48fe?media=json**

OUTPUT	{
	"name": "ENTLONDEV08",
	"included": true,
	"role": "Polling",
	"webPort": 80,
	"ssl": false,
	"local": true,
	"serverId": "34564e92-3b4f-43ba-94a8-04309c0e48fe",
	"licensed": true,
	"centralServer": false
	}

INPUT	**curl -u admin:admin http://localhost/api/servers/34564e92-3b4f-43ba-94a8-04309c0e48fe?media=xml**
OUTPUT	<?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"/>

User Groups

This resource lists general information about user groups.

URL: userGroups

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	`curl -u admin:admin http://localhost/api/userGroups?media=json`

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

INPUT	**curl -u admin:admin http://localhost/api/userGroups?media=xml**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="namedItem" name="All Users" id="2" serverId="ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

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

```
curl -u admin:admin http://localhost/api/userGroups?media=json -X POST -H "content-type: application/xml" -d \
'{
  "name" : "Support"
}'
```

OUTPUT


```
{
  "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
}
```


INPUT

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

OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="namedItem" name="All Users" id="2" serverId="ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="namedItem" name="Support" id="6" serverId="ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

User Group Details

Lists a set of operations on a particular group.

URL: userGroups/{groupID}

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	**`curl -u admin:admin http://localhost/api/userGroups?media=json -X DELETE`**

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

INPUT	**curl -u admin:admin http://localhost/api/userGroups?media=xml -X DELETE**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="namedItem" name="All Users" id="2" serverId="ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

User Group Permissions

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

URL: userGroups/{groupID}/tools

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	`curl -u admin:admin http://localhost/api/userGroups/6/tools?media=json`

OUTPUT	{
	"items": [],
	"count": 0
	}

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

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	**curl -u admin:admin http://localhost/api/userGroups/6?media=json X PUT -H "content-type: application/json" -d ** **'{** **"tools" : [{** **"id" : 15** **}, {** **"id" : 89** **}]** **}'**

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

INPUT	**curl -u admin:admin http://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="http://www.w3.org/2001/XMLSchema-instance"/>** **<tool id="89" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>** **</groupToolsWrapper>'**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="2"> <item xsi:type="toolId" groupName="Tools" id="15" toolName="Ticker" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="toolId" groupName="Administrator Tools" id="89" toolName="Data Export" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

Tools

Lists tools available on the server.

URL: tools

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	`curl -u admin:admin http://localhost/api/tools?media=json`

OUTPUT

```
{
  "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
}
```

INPUT

```
curl -u admin:admin  http://localhost/api/tools?media=xml
```

OUTPUT

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<items count="6">
  <item xsi:type="toolsGroup" xmlns:xsi="http://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="http://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>
```

Users

Lists information about users.

URL: users

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	`curl -u admin:admin http://localhost/api/users?media=json`

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

INPUT	**curl -u admin:admin http://localhost/api/users?media=xml**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> </items>

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	**curl -u admin:admin http://localhost/api/users?media=json -X POST -H "content-type: application/json" -d ** **'{** **"name" : "John",** **"password" : "little"** **}'**

OUTPUT	{
	"items": [{
	"serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a",
	"id": "6",
	"name": "John"
	}, {
	"serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a",
	"id": "3",
	"name": "admin"
	}],
	"count": 2
	}

INPUT	**curl -u admin:admin http://localhost/api/users?media=xml -X POST -H "content-type: application/xml" -d ** **'<userPasswordWrapper>** **<name>John</name>** **<password>little</password>** **</userPasswordWrapper>'**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="namedItem" name="admin" id="3" serverId="ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

User Details

List detailed user information, modify parameters of a user, and delete a user.

URL: users/{userID}

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.
lockDurationFailed Attempts	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	**curl -u admin:admin http://localhost/api/users/6?media=json**

OUTPUT	{
	"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,
	}

INPUT	**curl -u admin:admin http://localhost/api/users/6?media=xml -X GET**

OUTPUT	<?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>

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	**curl -u admin:admin http://localhost/api/users/6?media=json -X PUT -H "content-type: application/json" -d ** **'{** **"lockAttempts" : 3,** **"expiryDays" : 1579800600,** **}'**

OUTPUT	{
	"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,
	}

INPUT	**curl -u admin:admin http://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>'**

OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <userAccountInfo id="6" admin="false" locked="false" expired="false" id="19" xmlns:ns2="http://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>

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 http://localhost/api/users/6?media=json -X DELETE**
OUTPUT	{ "items": [{ "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a", "id": "3", "name": "admin" }], "count": 1 }

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

OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> </items>

User's Groups

List the groups a particular user is a member of.

URL: users/{userID}/groups

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	**curl -u admin:admin https://<ENA Server>/api/users/3/groups?media=json**

OUTPUT	{
	"items" : ["All Users", "Administrators"],
	"count" : 2
	}

INPUT	**curl -u admin:admin https://bvt/<ENA Server>/users/3/groups?media=xml**

OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <items count="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"> <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">All Users</item>> <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Administrators< /item>> </items>

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 - 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	**curl -u admin:admin http://localhost/api/settings/globalUserSettings?media=json -X GET**

OUTPUT	{
	"pwChangeDays" : 30,
	"pwChangeNoticePeriod" : 10,
	"timeoutMinutes" : 20,
	"lockAttempts" : 3,
	"lockDurationFailedAttempts" : 5,
	}

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

OUTPUT	{
	"pwChangeDays" : 30,
	"pwChangeNoticePeriod" : 10,
	"timeoutMinutes" : 20,
	"lockAttempts" : 3,
	"lockDurationFailedAttempts" : 5,
	}

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	**curl -u admin:admin http://localhost/api/settings/globalUserSettings?media=json -X GET**

OUTPUT	{
	"pwChangeDays" : 30,
	"pwChangeNoticePeriod" : 10,
	"timeoutMinutes" : 20,
	"lockAttempts" : 3,
	"lockDurationFailedAttempts" : 5,
	}

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

OUTPUT	{
	"pwChangeDays" : 30,
	"pwChangeNoticePeriod" : 10,
	"timeoutMinutes" : 20,
	"lockAttempts" : 3,
	"lockDurationFailedAttempts" : 5,
	}

POST Method details

Updates existing users to the global user settings.

Response

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

Examples

INPUT	**curl -u admin:admin http://localhost/api/settings/globalUserSettings /resetAllUsersSettings -X POST -H**

OUTPUT	{
	"message" : "Success. All existing users settings now match the global user settings. User-specific settings are not affected."
	}

Global Password Complexity Settings

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

URL:

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	`curl -u admin:admin http://localhost/api/settings /globalPasswordComplexitySettings -X GET`

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 http://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	{
	"pwdHistoryLimit" : 4,
	"pwdMinLength" : 8,
	"pwdReqNumeric" : true,
	"pwdReqLowerCase" : true,
	"pwdReqUpperCase" : true,
	"pwdReqSpecial" : true,
	"pwdReqOTP" : true
	}

View List

List available Views or create a new View.

URL: views

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	`curl -u admin:admin http://localhost/api/views?media=json`

OUTPUT	`{`
	`  "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
}
```


INPUT	**curl -u admin:admin http://localhost/api/views?media=xml**

OUTPUT	`<?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>`

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

```
curl -u admin:admin -H "content-type: application/json" http://localhost
/api/views -X POST -d \
'{
  "name" : "Simple View"
}'
```

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"
  } ],
  "count" : 3
}
```


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

INPUT

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

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",
```

<table>
<tr><td></td><td>


```
    "id" : "6",
    "displayName" : "My Network/MyView",
    "path" : "admin::My Network/MyView"
  } ],
  "count" : 4
}
```


</td></tr>
</table>

To create a Subview:

<table>
<tr><td>INPUT</td><td>

```
curl -u admin:admin -H "content-type: application/json" http://localhost
/api/views -X POST -d \
'{
  "name" : "SubView",
  "parentViewPath" : "MyView"
}'
```

</td></tr>
<tr><td>OUTPUT</td><td>


```
{
  "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"
```


</td></tr>
</table>

```
  }, {
    "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

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

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

```
curl -u admin:admin -H "content-type: application/json" http://localhost
/api/views -X POST -d \
'{
  "name" : "Intersection View",
  "baseViewAggregation" : "INTERSECTION",
  "baseViewPaths" : [ "MyView", "MyView/SubView" ]
}'
```

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"
  }, {

    "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	**curl -u admin:admin -H "content-type: application/json" http://localhost /api/views -X POST -d **

	'{
	"name" : "John's View",
	"owner" : "John",
	"accessGroups" : [{
	"userGroupName" : "Support",
	"editable" : "true"
	}, {
	"userGroupName" : "All Users",
	"editable" : "false"
	}]
	}'

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"
  }, {
    "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

```
curl -u admin:admin -H "content-type: application/json"
http://localhost/api/views -X POST -d \
'{
  "name" : "London Branch",
  "owner" : "admin",
  "accessGroups" : [ {
  "userGroupName" : "All Users",
  "editable" : "true"
  } ],
  "location" : "london, uk"
}'
```

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"
  }, {
    "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.

URL: views/{id}

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	**`curl -u admin:admin http://localhost/api/views/48?media=json`**

OUTPUT	{
	"serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96",
	"id" : "48",
	"path" : "Simple View",
	"displayName" : "Simple View",
	"baseViewAggregation" : "NONE",
	"baseViewPaths" : [ ],
	"domainFilterName" : "All Objects",
	"manuallyPopulated" : true,
	"eventFilterName" : "All Events",
	"incidentFilterName" : "All Incidents",
	"owner" : "admin",
	"accessGroups" : [{

```
    "userGroupName" : "Administrators",
    "editable" : true
  } ],
  "implicitAccessGroups" : [ ],
  "implicitAccessUsers" : [ ]
}
```


INPUT	**curl -u admin:admin http://localhost/api/views/48?media=xml**
OUTPUT	<?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>

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 http://localhost/api/views/48

INPUT	**curl -u admin:admin -X PUT -H "content-type: application/json" -d '{ "name" : "myView" }' http://localhost/api/views/4**
OUTPUT	{ "serverId" : "d251cf76-a753-41a7-9631-91baac1d644f", "id" : "4",

```
  "path" : "admin::My Network/myView",
  "displayName" : "My Network/myView",
  "baseViewAggregation" : "NONE",
  "baseViewPaths" : [ ],
  "domainFilterName" : "All Objects",
  "manuallyPopulated" : true,
  "eventFilterName" : "All Events",
  "incidentFilterName" : "All Incidents",
  "owner" : "admin",
  "accessGroups" : [ {
    "userGroupName" : "Administrators",
    "editable" : true
  } ],
  "implicitAccessGroups" : [ ],
  "implicitAccessUsers" : [ ]
}
```


DELETE Method detail

Delete existing View.

Request

No request expected.

Response

Empty response.

Examples

Delete View with id 48.

curl -u admin:admin -X DELETE http://localhost/api/views/48

View Objects

Inspect, add or delete View objects.

URL: views/{id}/objects

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.

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 http://localhost/api/views/49/objects?media=json**

OUTPUT	{
	"items" : [{
	"serverId" : "821c3e87-bcdf-4fef-a5e8-7d2524928d96",
	"id" : 769,
	"typeName" : "ManagedHost",
	"typeDisplayName" : "Managed Host",
	"displayName" : "bvt"
	}],
	"count" : 1
	}

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

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 -X PUT "http://localhost/api/views/49/objects?
media=json&id=2024&id=3279"
```

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 -X DELETE "http://localhost/api/views/49/objects?
media=json&id=2024&id=3279"
```

Remove all items from a View:

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

Object Attributes

Lists attributes of an object.

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

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
 http://localhost/api/objects/611/attributes?media=json&includeDetails=y
```

OUTPUT

```
{
  "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
}
```

INPUT

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

OUTPUT

```
<?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="http://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="http://www.w3.org/2001/XMLSchema-
instance">
    <name>category</name>
    <value>device</value>
  </item>
</items>
```

Object Attribute Details

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

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

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	`curl -u admin:admin http://localhost/api/objects/611/attributes/typeName?media=json`
OUTPUT	{

```
  "name" : "typeName",
  "displayName" : "typeName",
  "userEditable" : false,
  "userOverriden" : false,
  "values" : [ "SwitchDevice" ]
}
```

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

PUT Method details

Modifies an attribute of the object.

Request

An attribute definition:

Name	Description
name	name of attribute to be changed.
userOverridden	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

```
curl -u admin:admin http://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" ]
}'
```

OUTPUT

```
{
  "name" : "TransferServer",
  "displayName" : "Configuration Transfer Server",
  "userEditable" : true,
  "userOverriden" : true,
  "values" : [ "10.44.2.101" ]
}
```

INPUT

```
curl -u admin:admin http://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>'
```

OUTPUT

```
<?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>
```

Object Ports

Lists the ports on an object.

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

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	`curl -u admin:admin http://localhost/api/objects/655/ports?includeVirtual=yes&includeUnmanaged=yes&media=json`

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	**curl -u admin:admin http://localhost/api/objects/655/ports?includeVirtual=yes&includeUnmanaged=yes&media=xml**

OUTPUT


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<items count="8">
  <item xsi:type="devicePortInfo" xmlns:xsi="http://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>
      <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="http://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>
    ...removed for brevity...
      <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

- 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	**curl -u admin:admin http://localhost/api/objects/655/associations?showEmpty=yes&media=json**
OUTPUT	{ "items": ["AdaptorUnits", "Annotation", "ChassisList",

```
    "ComputeBlades",
    "DeviceMIBPolls",
    "FabricExtenders",
    "FanModules",
    "Fans",
    "HostEthernets",
    "IPSLABaseCreators",
    "IPSLABasePollers",
    "IpAddresses",
    "LocalDisks",
  ...removed for brevity...
    "uDComponents19",
    "uDComponents20"
  ],
  "count": 60
}
```


INPUT	**curl -u admin:admin http://localhost/api/objects/655/associations?showEmpty=yes&media=xml**

OUTPUT


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<items count="60">
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">AdaptorUnits</item>
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Annotation</item>
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">ChassisList</item>
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">ComputeBlades</item>
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">DeviceMIBPolls<
/item>
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">FabricExtenders<
/item>
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">FanModules</item>
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Fans</item>
```

```
  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">HostEthernets</item>

  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">IPSLABaseCreators<
/item>

  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">IPSLABasePollers<
/item>

  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">IpAddresses</item>

  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">LocalDisks</item>

...removed for brevity...

  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">uDComponents19<
/item>

  <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">uDComponents20<
/item>

</items>
```

Association Details

Display details of an association.

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

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	**curl -u admin:admin http://localhost/api/objects/655/associations/fans?media=json**

OUTPUT	<pre>{
	  "items": [
	  {
	  "objectId": 656,</pre>

```
      "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
}
```


INPUT	**curl -u admin:admin http://localhost/api/objects/655/associations/fans?media=xml**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="associationInfo" hasServiceStatus="true" displayType="" displayName="Fan 2" typeName="Fan" objectId="657" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="associationInfo" hasServiceStatus="true" displayType="" displayName="Fan 3" typeName="Fan" objectId="658" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

Configuration Management

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

URL: objects/{swID}/configManagement

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 http://localhost/api/objects/605/ports?includeVirtual=y&includeUnmanaged=y&media=json
```

OUTPUT


```
{
  "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" : " [ Vl1 ] Vlan1",
      "ifDescr" : " [ Vl1 ] 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",
```

```
    "portOperationalStatus" : "up",
    "portOutSpeed" : 0,
    "portShortDescr" : "[ Vl1 ]",
    "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	**curl -u admin:admin http://localhost/api/objects/611/configManagement?media=json -X PUT -H "content-type: application/json" -d ** **‘{** **"name" : "cliMethod",** **"values" : ["ssh"]** **}’**

OUTPUT	{
	"message" : "OK"
	}

INPUT	**curl -u admin:admin http://localhost/api/objects/611/configManagement?media=xml -X PUT -H "content-type: application/xml" -d ** **‘<objectAttribute>** **<name>cliMethod</name>** **<value>ssh</value>** **</objectAttribute>’**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo

Port Management

Set the managed/unmanaged state of a port.

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

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	**curl -u admin:admin http://localhost/api/portManagement/2/18?unManage=y&media=json -X PUT -H "content-type: application/json"**

OUTPUT	{
	"message" : "Successfully marking port as unmanaged"
	}

INPUT	**curl -u admin:admin http://localhost/api/portManagement/2/18?unManage=y&media=xml -X PUT -H "content-type: application/xml"**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo>

```
    <message>Successfully marking port as unmanaged</message>
</statusInfo
```

Zones

Lists configured zones and creates new zones.

URL: zones

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	**`curl -u admin:admin  http://localhost/api/zones?media=json`**

OUTPUT	`{`
	`  "items": [`
	`    {`
	`      "serverId": "ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a",`

```
      "id": "0",
      "name": "None"
    }
  ],
  "count": 1
}
```

INPUT	`curl -u admin:admin  http://localhost/api/zones?media=xml`

OUTPUT	`<?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="http://www.w3.org/2001/XMLSchema-instance"/>`
	`</items>`

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

```
curl -u admin:admin  http://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" ]
}’
```

OUTPUT

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

INPUT	**curl -u admin:admin http://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>'**
OUTPUT	<?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="http://www.w3.org/2001/XMLSchema-instance"/> <item xsi:type="namedItem" name="None" id="0" serverId="ee934fe2-1f4d-4f1e-a0b5-a87b261eb30a" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/> </items>

Zone Details

This resource implements operations acting on a single zone.

URL: zones/{zoneID}

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	**curl -u admin:admin http://localhost/api/zones/1?media=json**

OUTPUT	{
	"id": 1,
	"name": "A",
	"flags": 0,
	"description": "",
	"v4Interface": "10.44.2.103",
	"v6Interface": "",
	"domainSuffix": "",
	"proxy": "",
	"devicePrefix": "",
	"hostFile": "",
	"dnsServers": [
	"a.b.c.d"
	]
	}

INPUT	**curl -u admin:admin http://localhost/api/zones/1?media=xml**
OUTPUT	<?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>
```

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.
domainSuffx	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

```
curl -u admin:admin http://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" **]** **}'**

OUTPUT	{
	"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"
	]
	}

INPUT	**curl -u admin:admin http://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>'**
OUTPUT	<?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>
```

DELETE Method details

Deletes a zone.

Response

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

Examples

INPUT	`curl -u admin:admin http://localhost/api/zones/1?media=json -X DELETE`

OUTPUT	`{`
	`  "message" : "Zone deleted successfully"`
	`}`

INPUT	`curl -u admin:admin http://localhost/api/zones/1?media=xml -X DELETE`

OUTPUT	`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
	`<statusInfo>`
	`    <message>Zone deleted successfully</message>`
	`</statusInfo>`

IP SLA Management

View information about, modify and delete an IP SLA.

URL: ipsla/{id}

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	**`curl -u admin:admin http://localhost/api/ipsla/772?media=json`**

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 http://localhost/api/ipsla/772?media=xml`

OUTPUT	`<?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>`

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	`curl -u admin:admin http://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"> }’

OUTPUT	{
	"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"
	}
	}

INPUT	curl -u admin:admin http://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>'
OUTPUT	<?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>

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	**curl -u admin:admin http://localhost/api/ipsla/772?media=json -X DELETE**

OUTPUT	{
	"message" : "OK"
	}

INPUT	**curl -u admin:admin http://localhost/api/ipsla/772?media=xml -X DELETE**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <statusInfo> <message>OK</message> </statusInfo

IP SLA Creators

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

URL: objects/{swId}/ipslaCreators

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	**curl -u admin:admin http://localhost/api/objects/600/ipslaCreators?media=json**
OUTPUT	{ "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
}
```


INPUT	**curl -u admin:admin http://localhost/api/objects/600/ipslaCreators?media=xml**

OUTPUT


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<items count="1">
  <item xsi:type="ipslaAttributesMap" xmlns:xsi="http://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>
```

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

```
curl -u admin:adminhttp://localhost/api/objects/600/ipslaCreators?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"
}'
```

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

```
curl -u admin:admin http://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>'
```

OUTPUT

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<items count="2">
  <item xsi:type="ipslaAttributesMap" xmlns:xsi="http://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="http://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>
```

IP SLA Pollers

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

URL: objects/{swId}/ipslaPollers

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	**curl -u admin:admin http://localhost/api/objects/600/ipslaPollers?media=json**

OUTPUT	{
	"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
}
```


INPUT	**curl -u admin:admin http://localhost/api/objects/600/ipslaPollers?media=xml**

OUTPUT


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<items count="4">
  <item xsi:type="ipslaAttributesMap" xmlns:xsi="http://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="http://www.w3.org/2001
/XMLSchema-instance">
  ...removed for brevity...
  </item>
  <item xsi:type="ipslaAttributesMap" xmlns:xsi="http://www.w3.org/2001
/XMLSchema-instance">
    <attributes>
  ...removed for brevity...
    </attributes>
  </item>
</items>
```

Data Access Interface

General description

Data Access is inteded 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** statem ent 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** (pr esented 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	*XML element* 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	*XML attribute* node defining the name of the template. One server can only store templates with different names. Renaming of a template is not supported.

description	*XML element* 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	*XML element node* 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	*XML element node* 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	*XML element node* 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	*XML attribute* 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	*XML attribute* 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 only if 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: <object asElement=”true” type=”DeviceEx” tag=”DEVICE”> will get mapped to: <DEVICE>*data depending on the children of* ***object*** </DEVICE>

	If the template fragment were changed to: <object asElement=”true” type=”DeviceEx” tag=”DeviceParameters” > then the corresponding export fragments would have been changed to: <DeviceParameters>… </ DeviceParameters >
type	*XML attribute* node that defines the StormWorks type of object. The type of objects defines what attributes and streams may be access and exported.
selector	*XML element* 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	*XML element* 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.

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 - 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	`curl -u admin:admin http://localhost/api/dataAccessTemplates?media=xml -X GET -H "content-type: application/xml"`
OUTPUT	`<?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="http://www.w3.org/2001
/XMLSchema-instance"/>

  <item xsi:type="dataAccessTemplateSummary" description="" type="USER"
name="B" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>

</items>
```

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

```
curl -u admin:admin  "http://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>'
```

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 - 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	**curl -u admin:admin http://localhost/api/dataAccessTemplates/C?media=xml -X GET -H "content-type: application/xml"**
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>
```

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

```
curl -u admin:admin  http://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>'
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=""/> <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>

DELETE Method Details

Removes the existing Data Access Template.

Response

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

Example

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

Data Access – Export Triggering

Export user-defined Data Access templates.

URL: dataAccess/{templateName}

Methods Summary

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

GET Method details

Exports user-defined Data Access templates.

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 http://localhost/api/dataAccess/C?maxResults=5&view=All%20Objects&media=xml -X GET -H "content-type: application/xml"**
OUTPUT	<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=" [Vl1] 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>

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 http://localhost/api/dataAccess/C?position=565:2,575:0&maxResults=5&view=All%20Objects&media=xml -X GET -H "content-type: application/xml"**
OUTPUT	

```
<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>
```

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

INPUT	**curl -u admin:admin http://localhost/api/dataAccess/C?position=646:2,805:0&maxResults=5&view=All%20Objects&media=xml -X GET -H "content-type: application/xml"**
OUTPUT	<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>

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 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.
	string representation of the management IP address of the device. IPv4 and IPv6 are supported.

IpAddress	
deviceName	string representation of the device's display name.
swObId	StormWorks Object ID. All objects, including devices, are assigned a stormworks object ID. As with **deviceId**, **swObId** 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	**curl -u admin:admin https://localhost/api/flowDevices?media=json -X GET**

OUTPUT	{
	"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]
	}
	]
	}

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 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	`curl -u admin:admin  https://testld.entuity.com/api/options/flowApplications?media=json -X GET`

OUTPUT	`{`
	`  "applications" : [ {`
	`  "name" : "1ci-smcs"`
	`  }, {`
	`  "name" : "3Com-nsd"`
	`  }, {`

```
    "name" : "3PC"
  }, {
    "name" : "3com-amp3"
  }, {
...
```

Flow Data - Time Series History

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

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

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.

Response

...

Examples

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

INPUT	**curl -u admin:admin http://localhost/api/flowHistory/1?media=xml&startTime=-1800&interval=300**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="true"?> <ns2:flowHistoryResult xmlns:ns2="http://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
```

Grouped by ingress interface:

INPUT	**curl -u admin:admin http://localhost/api/flowHistory/1?media=xml&startTime=-900&interval=300&groups=ifIn**

OUTPUT

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<ns2:flowHistoryResult xmlns:ns2="http://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>
```

Grouped by egress interface:

INPUT	**curl -u admin:admin http://localhost/api/flowHistory/1?media=xml&startTime=-900&interval=300&groups=ifOut**

OUTPUT

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<ns2:flowHistoryResult xmlns:ns2="http://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>
```

Grouped by interface (note the double counting):

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

OUTPUT

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<ns2:flowHistoryResult xmlns:ns2="http://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="9842.563333333334" 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="7776.736666666667" timestamp="1507544100" volume="
2333021"/>
  </sampleSet>
</ns2:flowHistoryResult>
```

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

INPUT	**curl -u admin:admin http://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.*))**
OUTPUT	<?xml version="1.0" encoding="UTF-8" standalone="true"?> <ns2:flowHistoryResult xmlns:ns2="http://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>

Maintenance List

List available maintenance schedules or create a new maintenance schedule.

URL: maintenance

Method Summary

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

GET Method detail

Listss available maintenance schedules.

Response

A map of maintenance ids to their names.

Example

INPUT	`curl -u admin:admin localhost/api/maintenance`

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**

Example

INPUT

```
curl -u admin:admin 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",
  }
}
```

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 - 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.
suppesEvents	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	**curl -u admin:admin localhost/api/maintenance/2?media=json**

OUTPUT	{
	"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
	}

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.
suppesEvents	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.
suppesEvents	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

```
curl -X PUT -H "content-type: application/json" -u admin:admin localhost
/api/maintenance/2 -d \
{
"addDevices" : {
  "server-id" : [ 12 ]
 }
}
```

OUTPUT

```
{
  "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

}
```

Delete Method detail

Remove this maintenance schedule. Returns OK if successful.

Response

Returns OK status if successful.

Example

INPUT	**curl -X DELETE -u admin:admin localhost/api/maintenance/2?media=json**

OUTPUT	{
	"message" : "OK"
	}

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 - 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).
secretKeyIsSet	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 -X GET`**
OUTPUT	`{ "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 }`

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 - 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).
secretKeyIsSet	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	`curl -u admin:admin https://localhost/api/webhook/cloudAlert/Meraki/secretKey?objectId=5073 -X GET`
OUTPUT	`{ "serverId" : "9a55e715-3c18-4ef1-9cc9-f1b7f29ea139", "objectId" : 5073, "secretKey" : null }`

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

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).
secretKeyIsSet	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	**curl -u admin:admin https://localhost/api/webhook/cloudAlert/Meraki /secretKey?objectId=5073 -X DELETE**

OUTPUT	{
	"serverId" : "9a55e715-3c18-4ef1-9cc9-f1b7f29ea139",
	"objectId" : 5073,
	"secretKey" : ""
	}

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 - 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	`curl -u admin:admin https://localhost/api/webhook/cloudAlert/Meraki/enabled?objectId=5073 -X GET`
OUTPUT	`true`

PUT Method details

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

Example

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

Cisco DNA Center Webhooks

List DNA Center webhooks.

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

Methods Summary

- GET Method - list DNA Center webhooks

GET Method details

Returns a list of DNA Center webhooks.

Example

INPUT	**curl -k -u admin:admin https://localhost/api/webhook**

OUTPUT	{
	"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
	}

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 - enable or disable webhook events on a DNA Center.

PUT Method details

Enable or disable webhook events on a DNA Center.

Example

INPUT

```
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" ],
```

	<pre>"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" : "http://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 - 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	`curl -k -u admin:admin https://localhost/api/webhook/cloudAlert/enabled/<objectId>`
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	`curl -k -u admin:admin https://localhost/api/webhook/cloudAlert/enabled/<ObjectId> -X PUT -d "false"`
OUTPUT	false

Cisco DNA Centers

Return a list of DNA Centers.

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

Methods Summary

- GET Method - return a list of DNA Centers.

GET Method details

Return a list of DNA Centers.

Example

INPUT	`curl -k -u admin:admin https://localhost/api/DNACenters`

OUTPUT	{
	"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
	}