blueplanet
a division of ciena

Unified Assurance and Analytics

SNMP RA Development User Guide

Release 23.08.64
Issue 1.0 | November 21, 2023 | 450-3704-710-2308

Table of Contents

Publication History

The following table lists the 23.08 UAA SNMP RA Development User Guide publication history.

Table 1: Publication History

DATE	VERSION	NOTES
21-Nov-2023	1.0	Maintenance Patch Release for 23.08.64 (MR2)
25-Oct-2023	1.0	Maintenance Patch Release for 23.08.61 (MR1)
14-Aug-2023	1.0	Initial 23.08 Release

About the Document

This document explains the procedure to develop an SNMP RA, different files, and its tags. Different files explained in this document are -

- Profile
- Device
- Trap
- Inventory
- Layer2
- Performance
- Performance template
- Java class Implementation

Pre-requisite knowledge to develop SNMP RAs

To develop an SNMP RA using SNMP SDK kit, the user must have -

- Good understanding of SNMP and MIBs
- Good understanding of the BPUAA functionality and capabilities
- Knowledge of the Network Management
- Clear understanding of the Device and its monitoring capabilities
- Good understanding on core java concepts
- Understanding of XMLs
- Basic knowledge of Linux commands

Introduction to RAs

RA abbreviates to Resource Adapter, it's a software (combination of XMLs and java code) which is coded with all the SNMP monitoring information required to be fetched from a device.

The XMLs contain MIB details that specify -

- Rules to discover a device [device XML& discovery tree XML]
- The way a device reports traps and how they must be interpreted as mentioned [fault XMLs]
- Inventory that needs to be fetched from the device [inventory XMLs]
- Performance that the device reports and can be collected [Performance XMLs]
- Layer2 service tables information that the device supports

The XMLs are the complete set of monitoring information of device from MIBs in the format understandable by BPUAA platform. The platform does the job of Polling the device as per the logic in XMLs and to create all the fault, inventory, and Performance data on the UI.

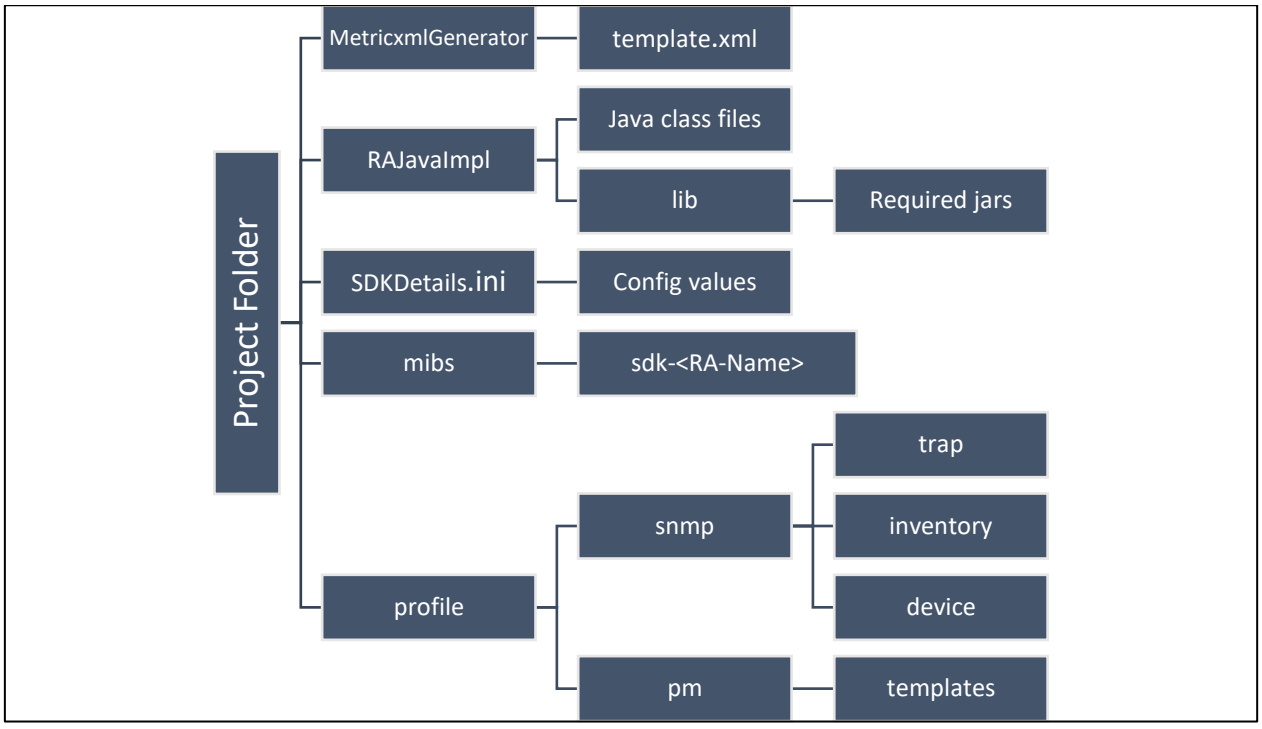
RA stores the information in XMLs related to which UAA object and the logic to create it using the MIB information.

Getting started with RA Development

BPUAA can be used to monitor any number of devices independent of vendor and protocol. To develop and update multiple RAs overtime SDK Docker image assumes certain folder structure as explained below.

Project Folder setup

The RA Development process using the SDK Kit involves different types of files (XML, txt and Java). To organize and maintain the files overtime the folder structure represented in the below image is used by the SDK Kit.



Folder and its role in detail below -

Project Folder

This is the parent/base folder to store all the files related to SNMP RA development using SDK Kit. This folder is to be mounted to the docker container to perform any script actions offered by SDK Docker image.

NOTE	All the project structure and the folders can be created using the SDK docker container by choosing the option 1 in the interactive prompt while running the container and this operation is to be done only once to setup the environment.

mibs

mibs folder is the starting point of RA development process, to develop any RA all its supported mibs are to be copied into a folder (with sdk-<RA-name>) inside the mibs folder as SDK docker container will look for mibs in this location to generate the RA templates to start the development process.

Once the template generation process is successful the profile folder will get created and all the required folders and templates will be created in the corresponding folders

profile

This folder contains different folders for different XMLs based on their functionality. All these XMLs are to be filled/handled as needed by the device monitoring capabilities or requirement. Clear explanation on how to handle the XMLs is provided in later sections

RAJavaImpl

This folder is used to store the Java source files as required based on requirement

MetricxmlGenerator

This folder is to store the excel sheet which is read by the pm file generator to create the PM and PM template XMLs

Generating template XMLs from MIBs

To start the RA development process one must have all the SNMP mibs required for device monitoring in **.mib** or **.my** format. MIBs provided by vendor tend to have syntactical errors so all these must be rectified before running template generation action.

TIP	Use any IDE which supports MIB file extension, so the errors are easily identified and rectified using the suggestions provided by IDE. (Ex. IntelliJ Idea has a plugin "MIB Support" that can be installed).

Once all the MIBs are available follow below steps to generate the template XMLs

1. Navigate to mibs folder in project folder
2. Create a folder with RA Name prefixed with **"sdk-"** (**Example: sdk-**< RA-Name>)
3. Copy all the required MIBs into the RA folder created above
4. Navigate back to the project folder, run the SDK docker container and select the option 2 presented in the interactive prompt to start the template generation process.
5. Once the above option is selected it will prompt for RA name.
6. Enter the name which is created in step 2
7. The template generation process will start and if no errors are encountered it will display message BUILD SUCCESSFUL and the process is complete
8. In case any errors are encountered, a log file with name SnmpProfileTemplateGenerator.log inside the logs folder (created under the project folder) can be referred.
9. Fix the errors logged in the above-mentioned file and re-run the docker container (from step 4) to start the template generation process

NOTE	Refer SNMP SDK Kit user guide for on how to use docker container

The above template generation process once successful will create multiple xml files required for RA in the profiles directory present in the project folder.

The XMLs generated include the following

- Profile XMLs
- Traps XMLs
- Device XMLs
- Mib-repository

The XMLs are empty templates which are to be filled and the guidance to fill each XMLs is provided in the later sections.

Profile XML

A profile xml is the entry point or the main file which contains device meta data and all other xml file paths which are required for the RA.

Location: XMLs are located at **ProjectFolder/profiles/**

This file contains following file paths -

- Device
- Mib-repository
- Trap
- Inventory
- pm (performance)
- pm-templates
- Jar file
- Instructions txt file

Complete Example:

```
<module id="sonicwall-tz-170">
  <!-- Meta property values for sonicwall-tz-170 -->
  <meta>
    <vendor name="SonicWALL"/>
    <protocol name="SNMP"/>
    <product name="SonicWALL TZ 170"/>
    <version name="*"/>
    <capabilities>
      <discovery>true</discovery>
      <fault>true</fault>
    </capabilities>
    <ne-template name="Managed NE"/>
    <pne-template/>
    <sne-template/>
    <am-template name="ICMP ping only"/>
  </meta>
  <dependencies>
    <file path="snmp/device/sonicwall-tz-170.xml" version="7621a158e899"/>
    <file path="snmp/mib-repository/all-nodes.xml" version="c540e23d1c41"/>
    <file path="snmp/mib-repository/sonicwall-tz-170-mib-repository.xml"
version="7621a158e899"/>
    <file path="snmp/trap/generic-traps-mib.xml" version="f6f1c864b115"/>
    <file path="snmp/trap/sonicwall-firewall-trap-mib.xml"
version="75a38e5ea9d7"/>
    <file path="snmp/trap/catch-all-trap.xml" version="f1cd5298301d"/>
    <file path="snmp/inventory/if-mib-32.xml" version="572bf2870626"/>
    <file path="snmp/inventory/sonicwall-firewall-ip-statistics-mib.xml"
version="4d469da28547"/>
```

```
    <file path="snmp/inventory/invProfile.dtd" version="6657e3866df0"/>
    <file path="pm/if-mib.xml" version="f70af1606d17"/>
    <file path="pm/sonicwall-firewall-ip-statistics-mib.xml"
version="5b255280a5a4"/>
    <file path="pm/pmProfile.dtd" version="3c6cf4f00fa9"/>
    <file path="pm/templates/pmTemplate.dtd" version="a8da80f0b74c"/>
    <file path="pm/templates/if-mib-32.xml" version="ea849f513e21"/>
    <file path="pm/templates/sonicwall-firewall-ip-statistics-mib-
sonicwallFwStats.xml" version="ff1b6e05f34d"/>
  </dependencies>
</module>
```

The xml tags are explained below -

module

It is the root tag of the profile xml.

Child tags

- meta
- dependencies

Attributes

Attribute	Required	Description
id	yes	Name of the RA (auto filled during template generation)
version	no	(Optional) Alphanumeric version to identify the file (EX: MD5 sum)

Example

```
<module id="sonicwall-tz-170">
</module>
```

meta

This section is to update meta properties of the device and node

Child tags

- vendor
- protocol
- product
- version
- capabilities
- ne-template
- pne-template
- sne-template

- am-template

Example

```
<meta>
</meta>
```

vendor

Attributes

Attribute	Required	Description
name	yes	Name of the vendor

Example

```
<vendor name="SonicWALL"/>
```

protocol

Attributes

Attribute	Required	Description
name	yes	Specifies protocol of the RA. Should be SNMP, will be auto filled during template generation

Example

```
<protocol name="SNMP"/>
```

product

Attributes

Attribute	Required	Description
name	yes	Name of the device.

Example

```
<product name="SBC"/>
```

version

Attributes

Attribute	Required	Description
name	yes	Version of the device.

Example

```
<version name="6000"/>
```

capabilities

Lists all the capabilities that the RA supports (true/false)

Child tags

- discovery
- fault
- ip-sla

Example

```
<capabilities>
      <discovery>false</discovery>
      <fault>true</fault>
</capabilities>
```

discovery

Specifies if the profile is capable of discovery.

Example

```
<discovery>false</discovery>
```

fault

Specifies if the profile is capable of fault.

Example

```
<fault>true</fault>
```

ip-sla

Specifies if the profile is capable of IPSLA.

Example

```
<ip-sla>true</ip-sla>
```

ne-template

Specifies the node template.

Attributes

Attribute	Required	Description
name	no	Name of the node template Default value: **Managed NE**

Example

```
<ne-template name="Managed NE"/>
```

pne-template

Specifies the passive node template. This will take the default value as shown in the example

Example

```
<pne-template/>
```

sne-template

Specifies the subtended node templates.

Example

```
<sne-template/>
```

am-template

Specifies the application monitoring templates. This will take the default value as shown in the example

Attributes

Attribute	Required	Description
name	no	Name of the application monitoring template Default value: **ICMP ping only**

Example

```
<am-template name="ICMP ping only"/>
```

dependencies

Lists all the dependent files of the profile. These list of file paths will be used during the build phase to package and create the RA Profile

Child tags

- file

Example

```
<dependencies>
</dependencies>
```

file

Attributes

Attribute	Required	Description
path	yes	Specifies the path of the dependent file. The path should start from the profiles folder. **Example:** for a trap xml located at project folder -> profiles -> snmp -> trap the path variable should specify path from profiles i.e., snmp/trap/<filename>.xml
version	yes	MD5 sum of file generated during the build process and need not change manually
reference	No	This element can be used to add the reference file name from which this file is created

Example

```
<file path="snmp/device/sonicwall-tz-170.xml" version="7621a158e899"/>
```

Device XML

There can only be one device xml for any RA. This xml contains the meta data similar to profile xml, device level details that has to polled and also holds the fault related information.

Location: XMLs are located at **ProjectFolder/snmp/device/**

An example device XML below

```
<?xml version="1.0" encoding="UTF-8"?>

<snmp-profile id="sonicwall-tz-170">

<!-- ========================================================================= -->
    <!-- Meta -->
<!-- ========================================================================= -->

    <!-- Discovery capabilities for sonicwall-tz-170 -->
    <meta>
        <capabilities>
            <discovery>true</discovery>
        </capabilities>
    </meta>

    <!-- NetworkElement property values for sonicwall-tz-170 -->
    <device object="centina.sa.model.topology.NetworkElement">
        <explicit>
            <property name="vendorName" value="SonicWALL"/>
            <property name="deviceType" value="FIREWALL"/>
            <property name="productVersion" value="170"/>
        </explicit>
     <on-inventory-sync>
        <query-agent>
            <property name="productName" oid="1.3.6.1.2.1.1.1.0"/>
            <property name="sysObjectID" oid="1.3.6.1.2.1.1.2.0"/>
            <property name="name" oid="1.3.6.1.2.1.1.5.0"/>
            <property name="location" oid="1.3.6.1.2.1.1.6.0"/>
        </query-agent>
     </on-inventory-sync>

    </device>
    <trap-group-list>
        <trap-group id="generic-traps"/>
        <trap-group id="sonicwall-firewall-trap-mib"/>
```

```
        <trap-group id="catch-all-trap"/>
    </trap-group-list>
</snmp-profile>
```

The tags in the XML are explained below

snmp-profile

This is the root tag of the xml

Attributes

Attribute	Required	Description
id	yes	This is the Profile Id. Should be same as the one given in profile XML
profileClass	no	Java implementation class name implemented in RAJavaImpl. **Example:** class name including the package name as packaged inside the jar
version	no	*(Optional)* An alphanumeric version to identify the file

child tags

- meta
- device
- on-inventory-sync
- trap-group-list

Example

```
<snmp-profile id="alcatel-7210"
profileClass="centina.sa.plugins.ALUPluginImpl">
```

meta

child tags

- capabilities
- attributes

Example

```
      <meta>
        <capabilities>
            <discovery>false</discovery>
        </capabilities>
        <attributes>
            <check-agent-connection-mode>ICMP</check-agent-connection-mode>
        </attributes>
    </meta>
```

capabilities

Set the device capabilities (like the capabilities section from the profile xml). **Example:** UAA does the nodal discovery when the discovery capability is enabled in the discovery tag.

child tags

- discovery
- ip-sla
- mta

Child tag	Required	Description
discovery	false	true/false
ip-sla	false	true/false

Example

```
<capabilities>
            <discovery>false</discovery>
 </capabilities>
```

Attributes

child tags

- Check-agent-connection-mode (NONE, SNMP, ICMP)

Child Tag	Required	Description
check-agent-connection-mode	false	• **NONE** - It will not check the connection periodically. • **SNMP** - checks the connection by polling for SysOid for snmpv2 or starting oid for snmpv1. If SNMP NE connection type is DORMANT (agent is not running on the NE), it will not poll the device. • **ICMP** - ping the device to check the keep-alive. Default timeout to check the keep-alive is 3000ms.

Example

```
<attributes>
            <check-agent-connection-mode>ICMP</check-agent-connection-mode>
</attributes>
```

device

Map the Device properties to Network Element object (Node) by setting the properties values explicitly and get some values by querying the device with OIDs

child tags

- explicit

- query-agent

Attribute	Required	Description
object	True	Value = “centina.sa.model.topology.NetworkElement”should not be changed

Example

```
<device object="centina.sa.model.topology.NetworkElement">
      <explicit>
          <property name="vendorName" value="Alcatel"/>
          <property name="deviceType" value="SWITCH"/>
          <property name="productVersion" value="7210"/>
      </explicit>
      <query-agent>
          <property name="productName" oid="1.3.6.1.2.1.1.1.0"/>
          <property name="sysObjectID" oid="1.3.6.1.2.1.1.2.0"/>
          <property name="name" oid="1.3.6.1.2.1.1.5.0"/>
          <property name="location" oid="1.3.6.1.2.1.1.6.0"/>
      </query-agent>
  </device>
```

explicit

The values for certain properties of objects have to be specified explicitly. For such properties the explicit tag is used. Any property can be set explicitly, generally, the following properties are explicit:

- Vendorname
- Devicetype
- Productversion

Device type is to be picked from one of the following Enums

- UNKNOWN
- SWITCH
- ROUTER
- DCS
- ADM
- DWDM
- DSLAM
- MSPP
- MSSP
- MSTP
- AMPLIFIER
- MANAGER
- AGENT
- DLC

- OSTP
- SERVER
- SSU
- OAC
- ESP
- FIREWALL
- SBC
- SAN
- OLT
- ONU
- MUX
- MSAP
- VPN
- LOAD_BALANCER
- VIDEO_PROCESSOR
- PACKETSHAPER
- UPS
- CMTS
- DTV_RECEIVER
- BLC
- RECTIFIER
- POWER_EQUIPMENT
- ACCESS_POINT
- SERVICE_CONTROLLER
- IP_PBX
- COLLABORATION_SERVER
- DEMARCATION_DEVICE
- MICROWAVE_RADIO
- GATEWAY
- MODEM
- VIDEO_PLATFORM
- SWITCH_ROUTER
- HVAC
- CRAC
- HVAC_CRAC
- MANAGED_NODE
- CABLE_MODEM
- MTA
- CLOUD_HOST
- VIRTUAL_MACHINE
- RPD

Attributes explained

Property Name	Required	Description
vendorName	false	This is the device vendor Name. Example: Cisco, Juniper, Alcatel
deviceType	true	Must be one of the Enum values mentioned in the above list
productVersion	false	The version/model of the device/software being supported by the plug-in.

Example

```
        <explicit>
            <property name="vendorName" value="DELL Inc."/>
            <property name="deviceType" value="SWITCH"/>
            <property name="productVersion" value="5324"/>
        </explicit>
```

query-agent

Query the device's SNMP agent and set the NE/Node properties in UAA. A complete OID must be specified from which the data has to retrieved and mapped.

child tags

- property
- switch-property
- map-property

Example

```
        <query-agent>
            <property name="productName" oid="1.3.6.1.2.1.1.1.0"/>
            <property name="sysObjectID" oid="1.3.6.1.2.1.1.2.0"/>
            <property name="name" oid="1.3.6.1.2.1.1.5.0"/>
            <property name="location" oid="1.3.6.1.2.1.1.6.0"/>
        </query-agent>
```

on-inventory-sync

This tag is introduced to query the mac-address of the device while performing inventory sync for layer2 devices. (mac-address will be useful to discover layer2 adjacency and vLANl services.)

We can query the OIDs and set to device properties while performing inventory sync.

child tags

- query-agent

Example

```
<on-inventory-sync>
```

```
                    <query-agent>
                          <property name="baseBridgeId"
      oid="1.3.6.1.4.1.22420.1.1.2.0"/>
                    </query-agent>
</on-inventory-sync>
```

trap-group-list

The NOTIFICATION-TYPE or TRAP-TYPE objects are handled as traps in the RA trap xmls and all the traps of a MIB are grouped into a trap-group.

This tag will contain list of trap-groups defined in trap xmls. Traps will be processed in the same order as they are listed in trap-group list tag.

NOTE	Add the catch-all-traps xml at the end of the list to catch the alarm which is not handled in the RA trap xmls.

child tags

- trap-group (1 or more)

Attributes

Attribute	Required	Description
synchronizeOnConnect	false	True/false – resync block in trap xml will execute while synchronizing the device. We can set the value of synchronizeOnConnect to false, if we want to disable resync in trap xml.

Example

```
 <trap-group-list synchronizeOnConnect="false">
        <trap-group id="generic-traps"/>
        <trap-group id="asentria-siteboss-420-alarm"
synchronizeOnConnect="true"/>
        <trap-group id="siteboss-420-std-mib"/>
        <trap-group id="catch-all-trap"/>
    </trap-group-list>
```

trap-group

Attributes

Attribute	Required	Description
id	true	Trap group id as mentioned in the trap xml
synchronizeOnConnect	false	True/false

Example

```
 <trap-group-list synchronizeOnConnect="false">
```

```
        <trap-group id="generic-traps"/>
        <trap-group id="asentria-siteboss-420-alarm"
synchronizeOnConnect="true"/>
        <trap-group id="siteboss-420-std-mib"/>
        <trap-group id="catch-all-trap"/>
    </trap-group-list>
```

MIB-repository

MIB repository is created during the template generation process. The compiled mibs data is added in a XML tree format in the mib-repository.

No modifications are required for this file and an should contain a mib-repository as the data is used as mib reference by UAA to resolve the OIDs to strings

RA can contain multiple mib-repository files, UAA aggregates all the mib-repositories to one mib-repository during run time

During template generation the script would automatically add the path to all-nodes.xml to all RAs which carries the MIB information related to generic MIBs.

Example

```
<?xml version="1.0" encoding="UTF-8"?>

<mib-repository id="sdk-sample-RAName">
 <entry id="1" name="iso" type="node">
  <entry id="0" name="std" type="node">
   <entry id="8802" name="iso8802" type="node">
    <entry id="1" name="ieee802dot1" type="node">
     <entry id="1" name="ieee802dot1mibs" type="node">
      <entry id="1" name="ieee8021paeMIB" type="node">
       <entry id="1" name="paeMIBObjects" type="node">
        <entry id="1" name="dot1xPaeSystem" type="node">
         <entry id="1" name="dot1xPaeSystemAuthControl" type="int">
          <list id = "1" value = "enabled(1)"/>
          <list id = "2" value = "disabled(2)"/>
         </entry>
         <entry id="2" name="dot1xPaePortTable" type="node">
          <entry id="1" name="dot1xPaePortEntry" type="node">
           <entry id="1" name="dot1xPaePortNumber" type="int"></entry>
           <entry id="2" name="dot1xPaePortProtocolVersion" type="int"></entry>
           <entry id="3" name="dot1xPaePortCapabilities" type="bits">
            <list id = "0" value = "dot1xPaePortAuthCapable(0)"/>
            <list id = "1" value = "dot1xPaePortSuppCapable(1)"/>

…….. Example doesn't have the complete file
```

Trap

There are two different ways to show the device alarms on UAA.

- Autonomous reporting
- Querying the Device's SNMP tables

Autonomous reporting

The SNMP protocol has three standard version of traps/alarms defined, a device based on the SNMP protocol will report an issue or a problem to NMS/UAA using UDP trap PDUs (Protocol Data Units).

A trap PDU contains multiple varbinds which give information to properly create an alarm in UAA.

An example trap captured from a device (using wireshark) for reference below:

Frame 8: 504 bytes on wire (4032 bits), 504 bytes captured (4032 bits)
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.82.163, Dst: 10.63.114.187
User Datagram Protocol
Simple Network Management Protocol
version: version-1 (0)
community: gAl1le0
data: trap (4)
trap
enterprise: 1.3.6.1.4.1.1824.1 (iso.3.6.1.4.1.1824.1)
agent-addr: 10.0.82.163
generic-trap: enterpriseSpecific (6)
specific-trap: 1
time-stamp: 3016109858
***variable-bindings**: 15 items*
1.3.6.1.4.1.1824.1.1.1.0: 303031424542343430384143
1.3.6.1.4.1.1824.1.2.1.0: 31302e302e38322e313633
1.3.6.1.4.1.1824.1.2.2.0: 1
1.3.6.1.4.1.1824.1.3.1.5.6: 1
1.3.6.1.4.1.1824.1.3.1.1.6: 6
1.3.6.1.4.1.1824.1.3.1.2.6: 47656e657261746f72204d616a6f722053756d6d61727920…
1.3.6.1.4.1.1824.1.3.1.3.6: 47454e455241544f52
1.3.6.1.4.1.1824.1.3.1.4.6: 47656e204d616a6f722053756d6d617279202d2053687574…
1.3.6.1.4.1.1824.1.3.1.6.6: 1
1.3.6.1.4.1.1824.1.3.1.11.6: 1532253684
1.3.6.1.4.1.1824.1.3.1.14.6: 1
1.3.6.1.4.1.1824.1.2.26.0: 4569
1.3.6.1.4.1.1824.1.2.29.0: 1532253684
1.3.6.1.4.1.1824.1.2.58.0: 2
1.3.6.1.4.1.1824.1.2.57.0: 1532474707

A trap PDU has multiple varbinds which has valuable information related to multiple UAA alarm properties that can be mapped.

Querying the device for environment alarms

SNMP MIBs contain a few tables which store alarm information and UAA can poll such tables based on the logic written in trap XMLs and populate alarms to UI. This poll happens during each alarm synchronization. These alarms are referred to as resync alarms in the document and UAA

Properties of an alarm explained

specificProblem
The specific Problem property is the alarm id or the alarm reason. This must be unique.

sourceId
The source Id property is used to identify the source of the alarm. This property is to be mapped with the endpoint SourceId on which this must raise.

sourceName
The source Name property is used to identify the source of the alarm. This property is to be mapped with the endpoint source name on which this has to raise.

Action
This property specifies if the alarm should get raised or if this is clearing an existing alarm
Values that can be set to this property are RAISE/CLEAR

Classification
A classification is to be decided by the developer based on the alarm description and the reason. The classification is used different internal UAA process like the alarm correlation/root cause analysis, so it's important to get the classification right. The classification should be picked from one of the following Enums (Listed in order of their priority from higher to lower)

- NODE_FAILURE
- REACHABILITY_FAILURE
- CARD_FAILURE
- PORT_FAILURE
- EQUIPMENT_FAILURE
- INTERFACE_FAILURE
- LAYER_1_FAILURE
- LAYER_2_FAILURE
- LAYER_3_FAILURE
- APPLICATION_FAILURE
- ABNORMAL_PERFORMANCE
- OTHER_FAILURE - Classifications from this level and below are not used for root cause analysis but can be suppressed by other alarms which have same source as this alarm and has a higher classification
- OTHER_SYMPTOM
- INFORMATION
- INDETERMINATE

perceivedSeverity
This property indicates the severity of alarm being raised. Severity is to be picked from one of the following Enum

- NOT_REPORTED
- INDETERMINATE

- INFO
- WARNING
- MINOR
- MAJOR
- CRITICAL
- COMMUNICATION

probableCause
Devices would generally give multiple Probable causes and these values have to mapped to one the below defined Probable cause Enums

- UNIDENTIFIED,
- INDETERMINATE,
- UNAVAILABLE,
- AIR_COMPRESSOR_FAILURE,
- AIR_CONDITIONING_FAILURE,
- AIR_DRYER_FAILURE,
- ALARM_INDICATION_SIGNAL,
- ANTENNA_FAILURE,
- APPLICATION_SUBSYSTEM_FAILURE,
- BACK_PLANE_FAILURE,
- BANDWIDTH_REDUCED,
- BATTERY_CHARGING_FAILURE,
- BATTERY_DISCHARGING,
- BATTERY_FAILURE,
- BROADCAST_CHANNEL_FAILURE,
- CALL_SETUP_FAILURE,
- COMMERCIAL_POWER_FAILURE,
- COMMUNICATIONS_RECEIVE_FAILURE,
- COMMUNICATIONS_TRANSMIT_FAILURE,
- CONFIGURATION_OR_CUSTOMISATION_ERROR,
- CONGESTION,
- CONNECTION_ESTABLISHMENT_ERROR,
- COOLING_FAN_FAILURE,
- COOLING_SYSTEM_FAILURE,
- CORRUPT_DATA,
- DATA_SET_PROBLEM,
- DATABASE_INCONSISTENCY,
- DEGRADED_SIGNAL,
- DEMODULATION_FAILURE,
- DISK_FAILURE,
- ENCLOSURE_DOOR_OPEN,
- ENGINE_FAILURE,
- EQUIPMENT_IDENTIFIER_DUPLICATION,
- EXCESSIVE_BIT_ERROR_RATE,
- EXCESSIVE_ERROR_RATE,

- EXCESSIVE_RESPONSE_TIME,
- EXCESSIVE_RETRANSMISSION_RATE,
- EXPLOSIVE_GAS,
- EXTERNAL_EQUIPMENT_FAILURE,
- EXTERNAL_IF_DEVICE_PROBLEM,
- EXTERNAL_POINT_FAILURE,
- FAR_END_RECEIVER_FAILURE,
- FILE_ERROR,
- FIRE,
- FIRE_DETECTOR_FAILURE,
- FLOOD,
- FRAMING_ERROR,
- FREQUENCY_HOPPING_FAILURE,
- FUSE_FAILURE,
- GENERATOR_FAILURE,
- HIGH_HUMIDITY,
- HIGH_TEMPERATURE,
- HIGH_WIND,
- ICE_BUILD_UP,
- INTRUSION_DETECTION,
- INVALID_MESSAGE_RECEIVED,
- IO_DEVICE_ERROR,
- LINE_CARD_PROBLEM,
- LOCAL_NODE_TRANSMISSION_ERROR,
- LOSS_OF_FRAME,
- LOSS_OF_MULTI_FRAME,
- LOSS_OF_POINTER,
- LOSS_OF_REAL_TIME,
- LOSS_OF_REDUNDANCY,
- LOSS_OF_SIGNAL,
- LOSS_OF_SYNCHRONIZATION,
- LOW_BATTERY_THRESHOLD,
- LOW_CABLE_PRESSURE,
- LOW_FUEL,
- LOW_HUMIDITY,
- LOW_TEMPERATURE,
- LOW_WATER,
- MEMORY_MISMATCH,
- MODULATION_FAILURE,
- MULTIPLEXER_PROBLEM,
- NE_IDENTIFIER_DUPLICATION,
- OUT_OF_CPU_CYCLES,
- OUT_OF_MEMORY,
- PATH_TRACE_MISMATCH,
- PAYLOAD_TYPE_MISMATCH,

- POWER_PROBLEM,
- POWER_SUPPLY_FAILURE,
- PROCESSOR_PROBLEM,
- PROTECTING_RESOURCE_FAILURE,
- PROTECTION_MECHANISM_FAILURE,
- PROTECTION_PATH_FAILURE,
- PUMP_FAILURE,
- REAL_TIME_CLOCK_FAILURE,
- RECEIVER_FAILURE,
- RECTIFIER_FAILURE,
- RECTIFIER_HIGH_VOLTAGE,
- RECTIFIER_LOW_F_VOLTAGE,
- REDUCED_LOGGING_CAPABILITY,
- REINITIALIZED,
- REMOTE_ALARM_INTERFACE,
- REMOTE_NODE_TRANSMISSION_ERROR,
- REPLACEABLE_UNIT_MISSING,
- REPLACEABLE_UNIT_PROBLEM,
- REPLACEABLE_UNIT_TYPE_MISMATCH,
- ROUTING_FAILURE,
- SIGNAL_LABEL_MISMATCH,
- SIGNAL_QUALITY_EVALUATION_FAILURE,
- SMOKE,
- SOFTWARE_DOWNLOAD_FAILURE,
- SOFTWARE_ENVIRONMENT_PROBLEM,
- SOFTWARE_ERROR,
- STORAGE_CAPACITY_PROBLEM,
- SYNCHRONIZATION_SOURCE_MISMATCH,
- SYSTEM_RESOURCES_OVERLOAD,
- TERMINAL_PROBLEM,
- TIMEOUT_EXPIRED,
- TIMING_PROBLEM,
- TOXIC_GAS,
- TRANSCEIVER_FAILURE,
- TRANSMISSION_ERROR,
- TRANSMITTER_FAILURE,
- TRUNK_CARD_PROBLEM,
- UNDERLAYING_RESOURCES_UNAVAILABLE,
- VENTILATION_SYSTEM_FAILURE,
- VERSION_MISMATCH,
- ADAPTER_ERROR,
- AUTHENTICATION_FAILURE,
- BANDWIDTH_REDUCTION,
- BREACH_OF_CONFIDENTIALITY,
- CABLE_TAMPER,

- COMMUNICATION_PROTOCOL_ERROR,
- COMMUNICATION_SUBSYSTEM_FAILURE,
- CONFIGURATION_OR_CUSTOMIZING_ERROR,
- CPU_CYCLES_LIMIT_EXCEEDED,
- DATA_SET_OR_MODEM_ERROR,
- DELAYED_INFORMATION,
- DENIAL_OF_SERVICE,
- DTE_DCE_INTERFACE_ERROR,
- DUPLICATE_INFORMATION,
- EQUIPMENT_MALFUNCTION,
- EXCESSIVE_VIBRATION,
- FIRE_DETECTED,
- FLOOD_DETECTED,
- HEATING_OR_VENTILATION_OR_COOLING_SYSTEM_PROBLEM,
- HUMIDITY_UNACCEPTABLE,
- INFORMATION_MISSING,
- INFORMATION_MODIFICATION_DETECTED,
- INFORMATION_OUT_OF_SEQUENCE,
- INPUT_DEVICE_ERROR,
- KEY_EXPIRED,
- LAN_ERROR,
- LEAK_DETECTION,
- MATERIAL_SUPPLY_EXHAUSTED,
- NON_REPUDIATION_FAILURE,
- OUT_OF_HOURS_ACTIVITY,
- OUT_OF_SERVICE,
- OUTPUT_DEVICE_ERROR,
- PERFORMANCE_DEGRADED,
- PRESSURE_UNACCEPTABLE,
- PROCEDURAL_ERROR,
- QUEUE_SIZE_EXCEEDED,
- RECEIVE_FAILURE,
- RESOURCE_AT_OR_NEARING_CAPACITY,
- SOFTWARE_PROGRAM_ABNORMALLY_TERMINATED,
- SOFTWARE_PROGRAM_ERROR,
- TEMPERATURE_UNACCEPTABLE,
- THRESHOLD_CROSSED,
- TOXIC_LEAK_DETECTED,
- TRANSMIT_FAILURE,
- UNAUTHORISED_ACCESS_ATTEMPT,
- UNEXPECTED_INFORMATION,
- UNSPECIFIED_REASON,
- A_BIS_TO_BTS_INTERFACE_FAILURE,
- A_BIS_TO_TRX_INTERFACE_FAILURE,
- ANTENNA_PROBLEM,

- BATTERY_BREAKDOWN,
- BATTERY_CHARGING_FAULT,
- CLOCK_SYNCHRONIZATION_PROBLEM,
- COMBINER_PROBLEM,
- DISK_PROBLEM,
- EQUIPMENT_FAILURE,
- EXCESSIVE_RECEIVER_TEMPERATURE,
- EXCESSIVE_TRANSMITTER_OUTPUT_POWER,
- EXCESSIVE_TRANSMITTER_TEMPERATURE,
- EXTERNAL_POWER_SUPPLY_FAILURE,
- EXTERNAL_TRANSMISSION_DEVICE_FAILURE,
- FAN_FAILURE,
- FILE_SYSTEM_CALL_UNSUCCESSFUL,
- FREQUENCY_HOPPING_DEGRADED,
- FREQUENCY_REDEFINITION_FAILED,
- INPUT_PARAMETER_OUT_OF_RANGE,
- INTRUSION_DETECTED,
- INVALID_MSU_RECEIVED,
- INVALID_PARAMETER,
- INVALID_POINTER,
- LAPD_LINK_PROTOCOL_FAILURE,
- LINE_INTERFACE_FAILURE,
- LINK_FAILURE,
- LOCAL_ALARM_INDICATION,
- MAINS_BREAKDOWN_WITH_BATTERY_BACK_UP,
- MAINS_BREAKDOWN_WITHOUT_BATTERY_BACK_UP,
- MESSAGE_NOT_EXPECTED,
- MESSAGE_NOT_INITIALIZED,
- MESSAGE_OUT_OF_SEQUENCE,
- RECEIVER_ANTENNA_FAULT,
- RECEIVER_MULTICOUPLER_FAILURE,
- REDUCED_ALARM_REPORTING,
- REDUCED_EVENT_REPORTING,
- REDUCED_TRANSMITTER_OUTPUT_POWER,
- REMOTE_ALARM_INDICATION,
- SMOKE_DETECTED,
- SS7_PROTOCOL_FAILURE,
- SYSTEM_CALL_UNSUCCESSFUL,
- TIMESLOT_HARDWARE_FAILURE,
- TRANSCEIVER_PROBLEM,
- TRANSCODER_OR_RATE_ADAPTER_PROBLEM,
- TRANSCODER_PROBLEM,
- TRANSMITTER_ANTENNA_FAILURE,
- TRANSMITTER_ANTENNA_NOT_ADJUSTED,
- TRANSMITTER_LOW_VOLTAGE_OR_CURRENT,

- TRANSMITTER_OFF_FREQUENCY,
- VARIABLE_OUT_OF_RANGE,
- WATCH_DOG_TIMER_EXPIRED,
- ALTERNATE_MODULATION_SIGNAL,
- AUTOMATIC_LASER_SHUTDOWN,
- AUTOMATIC_MESSAGES_INHIBITED,
- AUTOMATIC_PROTECTION_SWITCH,
- CENTRALIZED_POWER_FAILURE,
- CODING_VIOLATION,
- COLDBOOT,
- COMMUNICATION_FAILURE,
- DATA_COMMUNICATION_CHANNEL_FAILURE,
- EMS_SYSTEM_ALARM,
- END_OF_LIFE,
- EQUIPMENT_DEGRADE,
- ERRORED_SECONDS,
- FACILITY_LOOPBACK,
- FAILURE_TO_RELEASE_FROM_PROTECTION,
- FAILURE_TO_SWITCH_TO_PROTECTION,
- FUEL_LEAK,
- GAS_MONITOR_FAILURE,
- HATCH_FAILURE,
- HIGH_AIRFLOW,
- HIGH_LASER_BIAS,
- HIGH_RECEIVE_POWER,
- HIGH_VOLTAGE,
- HIGH_WATER,
- IMPROPER_REMOVAL,
- INTRUSION,
- LEVEL_CONVERTER,
- LIGHT,
- LINE_LOOPBACK,
- LOG_FULL,
- LOG_INITIALIZED,
- LOG_OVERFLOW,
- LOOPBACK,
- LOSS_OF_ALIGNMENT,
- LOSS_OF_CELL_DELINEATION,
- LOSS_OF_CHANNEL,
- LOSS_OF_CONTINUITY,
- LOSS_OF_MULTIFRAME,
- LOSS_OF_POWER,
- LOW_BATTERY_VOLTAGE,
- LOW_RECEIVE_POWER,
- LOW_VOLTAGE,

- LOW_VOLTAGE_DISCONNECT,
- MANUAL_PROTECTION_SWITCH,
- MANUAL_RESET,
- MANUAL_SWITCH,
- MEMORY_LOW,
- MISCELLANEOUS,
- MISMATCH_OF_EQUIPMENT_AND_ATTRIBUTES,
- OPEN_DOOR,
- OUTGOING_DEFECT_INDICATION,
- PAYLOAD_DEFECT_INDICATION,
- PAYLOAD_LABEL_MISMATCH,
- PAYLOAD_LOOPBACK,
- PAYLOAD_UNEQUIPPED,
- POWER_FAILURE,
- POWER_SUPPLY,
- PROTECTION_NOT_AVAILABLE,
- REBOOT,
- RECTIFIER_LOW_VOLTAGE,
- REMOTE_DEFECT_INDICATION,
- REMOTE_FAILURE_INDICATION,
- RINGING_GENERATOR,
- SECURITY_VIOLATION,
- SERVER_SIGNAL_FAIL,
- SEVERELY_ERRORED_FRAMING_SECONDS,
- SEVERELY_ERRORED_SECONDS,
- SIGNAL_DEGRADE,
- SIGNAL_FAILURE,
- SOFTWARE_DOWNLOAD_IN_PROGRESS,
- SPRINKLER,
- STANDBY_ENGINE_FAILURE,
- SWITCH_TO_PROTECTION_INHIBITED,
- SWITCH_TO_WORKING_INHIBITED,
- TERMINAL_LOOPBACK,
- TERMINATION_FAILURE,
- TRACE_IDENTIFIER_MISMATCH,
- TRANSMIT_UNDERRUN,
- TRANSMITTER_DEGRADE,
- UNAVAILABLE_SECONDS,
- UNAVAILABLE_TIME,
- UNEQUIPPED,
- VENTELATION_SYSTEM_FAILURE,
- WAIT_TO_RESTORE,
- ACTIVE_ALARMS_INIT_COMPLETE

Resyncable
This property is to let the UAA know that an alarm is resyncable. When this property is set to true this alarm can be cleared/raised during the alarm synchronization
Default value: false

serviceAffecting
This property can be set to true/false based on requirement

networkElementName
This property is used to identify the source of alarm. When this property is mapped UAA will automatically create a node with the mapped name if it not existing already.

addittionalInformation
This property is usually mapped automatically by software to show all the varbinds information.

NOTE

SpecificProblem is the mandatory field in the alarm properties

An alarm is uniquely identified using the specificProblem, sourceId and sourceName properties

To clear an alarm which is already an active alarm in UAA, an alarm should have all the above-mentioned properties the same

If the sourceId/sourceName is not mapped in the xml the alarm will by default raise on the node

A few properties have defined Enums so only those are to be used to map the alarm properties

All the XML tags defined in the below section are not mandatory, only the ones required to meet the requirement can be used in the XML

Handling the Trap XML

A profile xml may contain any number of trap xml files, and all the profiles contain two common trap xml files. Those xml files are:

- Catch-all-trap.xml
- generic-traps-mib.xml: In generic-traps-mib.xml all the traps regarding if-mib and snmpv2-mib are included.

The trap XMLs contain all the trap/alarm related details grouped per MIB after the template generation process is done. These files must be filled with logic to parse the alarms/traps sent from device

Location: Project folder -> profiles -> snmp -> trap

trap-group

This tag is used to group all the traps from a MIB.

Child tags

- Trap

Attributes

Attribute	Required	Description
id	Yes	Unique id. **Default:** The MIB name will be set during template generation

Example

```
<trap-group id="argus-mib">
</trap-group>
```

trap

This tag corresponds to a NOTIFICATION-TYPE or TRAP-TYPE from the mib.

Child tags

- trap-condition
- alarm
- synchronize

Attributes

Attribute	Required	Description
id	Yes	Unique id under a trap-group, that signifies the trap. **Default:** This will be the NOTIFICATION-TYPE/TRAP-TYPE objects name as defined in the MIB

Example

```
<trap id="acdAlarmActiveState">
</trap>
```

trap-condition

<trap-condition> tag determines which incoming trap must the block following it should process.

Child tags

- <trap-oid>
- <enterprise> and <specific-trap>
- <var-bind-oid-value>
- <var-bind-index-value>
- <var-bind-index-oid>
- <or>

trap-oid

The Oid of the NOTIFICATION-TYPE is specified here

Attributes

Attribute	Required	Description
value	Yes	OID of the trap. It is a regular expression

Example

```
<trap-oid value="1\.3\.6\.1\.4\.1\.22420\.2\.1\.12\.1"/> (backslash to escape ".")
```

enterprise and specific-trap

Used in case of SNMPv1 traps, this combination is used to determine which incoming trap must be processed by the alarm block under it.

Attributes

Attribute	Required	Description
value	Yes	For enterprise tag, value refers to enterprise OID specified in the mib for SNMPv1 trap. For specific-trap tag, value refers to the specific trap OID specified in the mib for SNMPv1 trap.

Example

```
<enterprise value="1\.3\.6\.1\.4\.1\.22420\.2\.1\.12"/>
<specific-trap value="1"/>
```

var-bind-oid-value

A var bind is the combination of OID and its value.

This tag is used in combination with <trap-oid> or "<enterprise> and <specific-trap>" tags. This tag is used to create more filters, even though a trap satisfies the <trap-oid> or <enterprise> and <specific-trap> tags it may not be processed by the corresponding alarm block unless it satisfies <var-bind-oid-value> tag.

To handle different versions of devices which has an alarm with same OID but different associated Objects in the MIB, this filter can be used

Attributes

Attribute	Required	Description
Oid/starting-oid	Yes	OID = exact oid that would be picked from the varbind list of the trap, whose value will be compared against the value specified in value attribute.

Attribute	Required	Description
		starting-oid = first oid that has the sequence specified in the starting-oid will be picked from trap.
value	Yes	Whatever value is specified against this tag will be compared to the value fetched using the above attribute.

Example

In the below Example the alarm block will be processed when the received trap is with OID satisfying the regular expression 1\.3\.6\.1\.4\.1\.5003\.9\.10\.1\.21\.2\.0\.([1-9]|[1-6][0-9]|7[0-2] and has the value “5” in the trap PDU’s Varbind OID 1.3.6.1.4.1.5003.9.10.1.21.1.6

```
<trap-oid value="1\.3\.6\.1\.4\.1\.5003\.9\.10\.1\.21\.2\.0\.([1-9]|[1-6][0-9]|7[0-2])"/>
<var-bind-oid-value starting-oid="1.3.6.1.4.1.5003.9.10.1.21.1.6" value="5"/>
```

var-bind-index-value

Same use case as <var-bind-oid-value>, but in this case this tag fetches the value based on the position of an oid in the incoming trap PDUs varbind list.

Attributes

Attribute	Required	Description
index	Yes	The number specified against this tag corresponds to the position of the var-bind in the var-bind list of the incoming trap.
value	Yes	Whatever value is specified against this tag will be compared to the value fetched using the above attribute.

Example

This example checks for value 5 in the second varbind OID of the trapPDU

```
<trap-oid value="1\.3\.6\.1\.4\.1\.5003\.9\.10\.1\.21\.2\.0\.([1-9]|[1-6][0-9]|7[0-2])"/>
<var-bind-index-value index=”2” value="5"/>
```

var-bind-index-oid

Same as <var-bind-index-value> instead of fetching the value of the varbind specified by the index, <var-bind-index-oid> fetches the oid.

Attributes

Attribute	Required	Description
index	Yes	The number specified against this tag corresponds to the position of the varbind in the varbind list of the incoming trap.

value	Yes	Whatever value is specified against this tag will be compared to the oid fetched using the above attribute.

Example

In this case the alarm block is processed only when the trap OID satisfies the OID Regular expression and has the OID “1\.3\.6\.1\.4\.1” in the second position of varbinds list of trapPDU

```
<trap-oid value="1\.3\.6\.1\.4\.1\.5003\.9\.10\.1\.21\.2\.0\.([1-9]|[1-6][0-9]|7[0-2])"/>
<var-bind-index-oid index=”2” value="1\.3\.6\.1\.4\.1"/>
```

or

This tag is used to construct a complex trap-condition. More than one OID of the incoming trap will be considered to decipher whether the alarm block following this trap-condition should process that trap.

Child tags

- set

Example

```
<or>
      <set>
            <trap-oid value="1\.3\.6\.1\.4\.1\.22420\.2\.1\.12\.1"/>
      </set>
      <set>
            <enterprise value="1\.3\.6\.1\.4\.1\.22420\.2\.1\.12"/>
            <specific-trap value="1"/>
      </set>
</or>
```

The above example is used to allow support for both SNMPv1 and V2 traps.\

set

Multiple set blocks can be used under the <or> tag, meaning if the incoming trap satisfies one of the set block properties, the trap will be processed by the corresponding alarm block.

Child tags

- trap-oid
- enterprise and specific-trap
- var-bind-oid-value
- var-bind-index-value
- var-bind-index-oid

Example

```
<set>
```

```
      <enterprise value="1\.3\.6\.1\.4\.1\.22420\.2\.1\.12"/>
      <specific-trap value="1"/>
</set>
```

Alarm

This tag contains all the information required to map the incoming trap information to UAA alarm properties. If we need to create more than one alarm for single trap received, we can define more than one alarm blocks in a single trap block.

Child tags

- explicit
- trap-pdu
- var-bind-index-value
- var-bind-index-oid
- var-bind-oid-value
- var-bind-oid-oid
- evaluate-property
- switch-evaluate-property
- switch
- map

Attributes

Attribute	Required	Description
object	Yes	UAA object to which the incoming trap should get translated to. **NOTE:** Leave the default value generated during the template generation process

Example

```
<alarm object="centina.sa.model.fault.EquipmentAlarm">
</alarm>
```

explicit

Assign certain alarm properties explicitly(hard coding the values).

Child tags

- property

property

Assign a property of the object specified in alarm object attribute.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Alarm object property name
value	Yes	The value that must be assigned to the property mentioned above.

Example

```
<property name="serviceAffecting" value="false"/>
<property name="action" value="RAISE"/>
```

property-condition

Only when this condition passes, the super tag is applicable.

Child tags

Refer < trap-condition> for child tags.

Example

```
<property name="serviceAffecting" value="false">
      <property-condition>
            <var-bind-index-oid index=”2” value="1\.3\.6\.1\.4\.1"/>
      </property-condition>
</property>
```

Only when the <var-bind-index-oid> condition is true, <property> gets applied.

trap-pdu

This tag provides access to trap/enterprise/specific-trap fields of the trap pdu.

Child tags

- property
- switch-property
- map-property
- switch-map-property

property

Assign trap/enterprise/specific-trap and other trap pdu fields to a UAA alarm property.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Alarm's property name
field	Yes	Can take one of the following values • pdu-type = takes 1 or 2 signifying snmpv1 and v2 respectively. • Enterprise = enterprise oid of v1 trap. • Generic-trap = generic trap name like coldstart, warmstart etc. • Specific-trap = specific trap value of v1 trap-condition • trap-oid = trap oid field available in the pdu. • timeStamp = time stamp field of the trap pdu. • Community = community field of the trap pdu. • Var-bind = convert all the incoming varbind list to objectname = value pair, where objectname is the name corresponding to a given oid as specified by the mib.

Example

```
<trap-pdu>
      <property name="additionalInformation" field="var-bind"/>
</trap-pdu>
```

<switch-property>

This tag provides access to trap pdu fields which in-turn will be manipulated using <swtich> tag and assigned to an alarm property.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Alarm object property name
field	No	Can take one of the following values • pdu-type = takes 1 or 2 signifying snmpv1 and v2 respectively. • Enterprise = enterprise oid of v1 trap. • Generic-trap = generic trap name like coldstart, warmstart etc. • Specific-trap = specific trap value of v1 trap-condition • trap-oid = trap oid field available in the pdu. • timeStamp = time stamp field of the trap pdu. • Community = community field of the trap pdu. • Var-bind = convert all the incoming varbind list to objectname = value pair, where objectname is the name corresponding to a given oid as specified by the mib.
id	Yes	(userdefined) This value is later used in the <switch> tag to assign a desired value to the property specified against name field.

Example

```
<trap-pdu>
      <switch-property id="specificProblem" name="specificProblem"
field="trap-oid"/>
</trap-pdu>
<switch id="specificProblem">
      <case input=".*\.1$" output="SYS-HOUSEKEEP1" type="REG_EXP"/>
      <case input=".*\.2$" output="SYS-HOUSEKEEP2" type="REG_EXP"/>
</switch>
```

In the above example, when the value of the trap-oid ends with 1, the specific-problem property takes SYS-HOUSEKEEP1 as the value. Switch block will be discussed in detail in the later portion of the document.

map-property

Assigns trap PDU variable to a non-alarm field such as a variable. This variable can be used to set any alarm property.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	(userdefined) Name of a variable which later will be used in <map> tag.
field	Yes	Can take one of the following values • pdu-type = takes 1 or 2 signifying snmpv1 and v2 respectively. • Enterprise = enterprise oid of v1 trap. • Generic-trap = generic trap name like coldstart, warmstart etc. • Specific-trap = specific trap value of v1 trap-condition • trap-oid = trap oid field available in the pdu. • timeStamp = time stamp field of the trap pdu. • Community = community field of the trap pdu. • Var-bind = convert the entire incoming varbind list to objectname = value pair, where objectname is the name corresponding to a given oid as specified by the mib.

Example

```
<trap-pdu>
      <map-property name="PhysicalIndex"   field="specific-trap"/>
</trap-pdu>
<map>
      <property name="sourceId"
value="1.3.6.1.2.1.47.1.1.1.1.7.$PhysicalIndex$"/>
</map>
```

The $PhysicalIndex$ will be replaced by specific-trap value of pdu.

<switch-map-property>

This tag is a combination of switch-property and map-property.

Attributes

Attribute	Required	Description
name	Yes	(userdefined) Variable used to store the output
field	Yes	Can take one of the following values • pdu-type = takes 1 or 2 signifying snmpv1 and v2 respectively. • Enterprise = enterprise oid of v1 trap. • Generic-trap = generic trap name like coldstart, warmstart etc. • Specific-trap = specific trap value of v1 trap-condition • trap-oid = trap oid field available in the pdu. • timeStamp = time stamp field of the trap pdu. • Community = community field of the trap pdu. • Var-bind = convert all the incoming varbind list to objectname = value pair, where objectname is the name corresponding to a given oid as specified by the mib.
id	Yes	(userdefined) This value is later used in the <switch> tag to assign a desired value to the property specified against name field.

Switch uses a simple case tag with input and output attributes. Where input is matched with the id value and the output value will be returned.

Example

```
<trap-pdu>
      <switch-map-property id="causeCode"         name="causeCode"
field="enterprise"/>
</trap-pdu>
<switch id="causeCode">
      <case input="1"  output="unknown"/>
      <case input="2"  output="heartBeatStopped"/>
      <case input="3"  output="routerThreadDied"/>
      <case input="4"  output="timerThreadDied"/>
      <default output="INFO"/> <!—default can be used to output a value when
no case mentioned above matches -->

</switch>
<map>
      <property name="specificProblem"
value="ccmCallManagerFailed:$causeCode$"/>
</map>
```

var-bind-index-value

Fetch the value of the nth var-bind and assign it to a variable.

Child tags

- property
- switch-property
- map-property
- switch-map-property

property

Assign the value portion of the nth var-bind to an alarm property.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Alarm object property name
index	Yes	Index of the nth varbind, whose value will be used.

Example

to Assign the value of 1st varbind to the additionalInformation property

```
<var-bind-index-value>
      <property name="additionalInformation" index="1"/>
</var-bind-index-value>
```

switch-property

Retrieve the value of the nth varbind and assign to a variable, which in turn will be used in switch block to be assigned to an alarm property.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Alarm object property name
index	Yes	Index of the nth varbind in trap pdu varbind list whose value will be stored against the variable named after id attribute.
id	Yes	This value is later used in the <switch> tag to assign a desired value to the property specified against name field.

Example

```
<var-bind-index-value>
      <switch-property id="action" name="action" index="1"/>
</var-bind-index-value>
<switch id="action">
      <case input="6" output="CLEAR"/>
      <default output="RAISE"/>
</switch>
```

map-property

Assign value of the var-bind fetched based on index to a variable.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Name of a variable which later will be used in <map> tag.
index	Yes	index of the nth varbind in trap pdu varbind list whose value will be fetched

Example

```
<var-bind-index-value>
      <map-property name="PhysicalIndex"   index="1"/>
<var-bind-index-value>
<map>
      <property name="sourceId"
value="1.3.6.1.2.1.47.1.1.1.1.7.$PhysicalIndex$"/>
</map>
```

The value of 1st varbinds is first save to the variable **"PhysicalIndex"** and then this is used in assigning the sourceId property of the alarm

switch-map-property

This tag is a combination of switch-property and map-property.

Attributes

Attribute	Required	Description
name	Yes	Variable used to store the output.
index	Yes	Index of the nth varbind in trap pdu varbind list whose value will be fetched.

id	Yes	This value is later used in the <switch> tag to assign a desired value to the property specified against name field.

Example

```
<var-bind-index-value>
      <switch-map-property id="causeCode"         name="causeCode"
index="1"/>
</var-bind-index-value>
<switch id="causeCode">
      <case input="1"  output="unknown"/>
      <case input="2"  output="heartBeatStopped"/>
      <case input="3"  output="routerThreadDied"/>
      <case input="4"  output="timerThreadDied"/>
</switch>
<map>
      <property name="specificProblem"
value="ccmCallManagerFailed:$causeCode$"/>
</map>
```

var-bind-index-oid

Similar to <var-bind-index-value>, the only difference being instead of fetching the value of the varbind based on the index given, this tag fetches the oid. Allows an extra set of operation through translators.

The translator concept will be explained below using map-property.

Example

```
<var-bind-index-oid>
      <map-property name="id" index="2">
            <oid-index-translator type="OID_VARIABLE_LENGTH_IX_ONLY" start-
      index="8"/>
      </map-property>
</var-bind-index-oid>
```

The above example map-property fetches the oid from the 2nd varbind, this oid will be further processed by the oid-index-translator, start-index = 14 means it would chop the oid from 8th number till end. Let us say the oid from the 2nd index is: 1.3.6.1.2.1.47.1.4.5.6; The output will be .4.5.6.

Let us consider one more example that takes fixed length type.

```
<var-bind-index-oid>
      <map-property name="id" index="2">
            <oid-index-translator type="OID_FIXED_LENGTH_IX_ONLY" start-
      index="8" length="1"/>
      </map-property>
</var-bind-index-oid>
```

In this case instead of the chopping the oid from the 8th number till end, the translator chops the oid from 8th number upto a length specified by length attribute. So, let us say the second oid is: 1.3.6.1.2.1.47.1.4.5.6; The output here would be .4.

Translator type	Description
OID_VARIABLE_LENGTH_IX_ONLY	Fetches the remainder of the oid starting from start-index length.
OID_FIXED_LENGTH_IX_ONLY	Fetches the remainder of the oid starting from start-index length till the length specified using length attribute
OCTETS_FIXED_LENGTH_IX_ONLY	Used when an oid contains octet indices. Converts all the octet indices post the length specified by start-index till the length specified in length attribute.
OCTETS_VARIABLE_LENGTH_IX_ONLY	Used when an oid contains octet indices. Converts all the octet indices post the length specified by start-index till end.
ASCII_FIXED_LENGTH_IX_ONLY	Used when an oid contains ascii indices. Converts all the ascii indices post the length specified by start-index
ASCII_VARIABLE_LENGTH_IX_ONLY	Used when an oid contains ascii indices. Converts all the ascii indices post the length specified by start-index till the length specified in length attribute
IP_ADDR_V4_IX_ONLY	Used when an oid contains ip version 4 indices. Converts all the ip indices post the length specified by start-index
HEX_FIXED_LENGTH_IX_ONLY	Used when an oid contains hexadecimal indices. Converts all the hexadecimal indices post the length specified by start-index till the length specified in length attribute

var-bind-oid-value

Fetches the value from the trap pdu based on the OID specified in the oid/starting-oid attribute.

Child tags

- property
- switch-property
- map-property
- switch-map-property

property

Assign value from the first varbind that matches the oid specified in the oid/starting-oid tag to an alarm property specified in the name attribute.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Specifies the alarm property for which a value will be assigned.

Attribute	Required	Description
oid/starting-oid	Yes	oid = exact oid that would be picked from the varbind list of the trap pdu, whose value will be assigned to the alarm property specified in the name attribute. starting-oid = first oid that has the sequence specified in the starting-oid will be picked from trap.

Example

```
<var-bind-index-value>
      <property name="additionalInformation" oid="1.3.6.1.2.1.47.1"/>
</var-bind-index-value>
```

switch-property

Retrieve the value from a varbind based on the oid/starting oid match and assign the value to a variable, which in turn will be used in switch block to be assigned to an alarm property.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Alarm object property name
Oid/starting-oid	Yes	oid = exact oid that would be picked from the varbind list of the trap pdu, whose value will be stored against the variable named after id attribute. starting-oid = first oid that has the sequence specified in the starting-oid will be picked from trap.
id	Yes	This value is later used in the <switch> tag to assign a desired value to the property specified against name field.

Example

```
<var-bind-index-value>
      <switch-property id="action" name="action" starting-
oid="1.3.6.1.2.1.47.1"/>
</var-bind-index-value>
<switch id="action">
      <case input="6" output="CLEAR"/>
      <default output="RAISE"/>
</switch>
```

map-property

Assign value of the varbind fetched based on oid/starting-oid match to a variable.

Child tags

- property-condition

Attributes

Attribute	Required	Description
name	Yes	Name of a variable which later will be used in <map> tag.
Oid/starting-oid	Yes	oid = exact oid that would be picked from the varbind list of the trap pdu, whose value will be stored against the variable named after name attribute. starting-oid = first oid that has the sequence specified in the starting-oid will be picked from trap.

Example

```
<var-bind-index-value>
      <map-property name="PhysicalIndex"   oid="1.3.6.1.2.1.47.1.1.1.1.7"/>
<var-bind-index-value>
<map>
      <property name="sourceId"
value="1.3.6.1.2.1.47.1.1.1.1.7.$PhysicalIndex$"/>
</map>
```

switch-map-property

This tag is a combination of switch-property and map-property.

Attributes

Attribute	Required	Description
name	Yes	Variable used to store the output
Oid/starting-oid	Yes	oid = exact oid that would be picked from the varbind list of the trap pdu, whose value will be stored against the variable named after name attribute. starting-oid = first oid that has the sequence specified in the starting-oid will be picked from trap.
id	Yes	This value is later used in the <switch> tag to assign a desired value to the property specified against name field.

Example

```
<var-bind-oid-value>
      <switch-map-property id="causeCode"          name="causeCode"
starting-oid="1.3.6.1.4.1.9.9.156.1.10.2"/>
</var-bind-oid-value>
<switch id="causeCode">
      <case input="1"  output="unknown"/>
      <case input="2"  output="heartBeatStopped"/>
      <case input="3"  output="routerThreadDied"/>
      <case input="4"  output="timerThreadDied"/>
</switch>
<map>
      <property name="specificProblem"
value="ccmCallManagerFailed:$causeCode$"/>
</map>
```

var-bind-oid-oid

Same as <var-bind-oid-value> instead of fetching value, OID will be fetched and as a result, translators can be used.

switch

This tag is a mandate when switch-property tag is used. This property can be compared to a switch case, where in for each output pattern derived from the switch-property a user defined value will be fetched and assigned.

Child tags

- Case = case has 3 attributes in turn - **input, output, and type**. Input is the value against which the output returned by <swtich-property> will be compared, while **output** is the user defined value. Type attribute can take one of the following values
    - REG_EXP = meaning a regular expression pattern is used in the input
    - equals = this is the default value when not specified, performs an equals operation.
- Default = when none of the case matches then default value will be assigned to the property. Therefore, default has only one attribute that is **output**.

Attributes

Attribute	Required	Description
id	Yes	This attribute value must be the same as the id attribute specified in the <switch-property> tag.

Example

```
<var-bind-index-value>
      <switch-property id="action" name="action" starting-
oid="1.3.6.1.2.1.47.1"/>
</var-bind-index-value>
```

```
<switch id="action">
      <case input="6" output="CLEAR"/>
      <default output="RAISE"/>
</switch>
```

map

This tag is used to assign output returned by <map-property> tag to an alarm property, by attaching some user defined post and pre values.

Child tags

- <property> = tags two attributes namely the **name** tag, which is the alarm property and the **value** tag, where the value of the variable from map-property can be accessed by enclosing it between $ symbol.

Example

```
<var-bind-index-value>
      <map-property name=”PhysicalIndex”   oid=”1.3.6.1.2.1.47.1.1.1.1.7”/>
<var-bind-index-value>
<map>
      <property name=”sourceId”
value=”1.3.6.1.2.1.47.1.1.1.1.7.$PhysicalIndex$”/>
</map>
```

switch-evaluate-property

This tag is used to compare 2 values fetched from var-bind list of the trap PDU.

Child tags

- Argument = this tag has a few attributes namely
    - name = a name for a variable that will store one of the values to be compared.
    - map-name = the output from the map-property tag is stored in this variable.

Attributes

Attribute	Required	Description
name	Yes	Alarm property name.
evaluator	Yes	The evaluate java class. “centina.common.snmp.evaluator.CompareEvaluator”
Id	Yes	This value is later used in the <switch> tag to assign a desired value to the property specified against name field.

Example

```
<var-bind-oid-value>
      <map-property name=”threshold” starting-
oid=”1.3.6.1.4.1.738.1.5.200.2.1.8.1.3”/>
```

```
      <map-property name="current" starting-
oid="1.3.6.1.4.1.738.1.5.200.2.1.8.1.4"/>
</var-bind-oid-value>

<switch-evaluate-property id="action" name="action"
evaluator="centina.common.snmp.evaluator.CompareEvaluator">
      <argument name="arg1" map-name="threshold"/>
      <argument name="arg2" map-name="current"/>
</switch-evaluate-property>

<switch id="action">
      <case input="-1" output="CLEAR"/>
      <default output="RAISE"/>
</switch>

<map>
      <property name="sourceId"   value="1.3.6.1.2.1.2.2.1.1$ifIndex$"/>
</map>
```

evaluate-property

This property is better understood with the following example. The java class used is "centina.sa.server.adapter.snmp.vendor.common.RegExpEvaluator"

```
<var-bind-oid-value>
      <map-property name="RectAlrm" starting-
oid="1.3.6.1.4.1.7309.4.1.5.1.2.1.4"/>
      <map-property name="DigAlrm"  starting-
oid="1.3.6.1.4.1.7309.4.1.5.2.2.1.4"/>
</var-bind-oid-value>
<evaluate-property name="specificProblem"
evaluator="centina.sa.server.adapter.snmp.vendor.common.RegExpEvaluator">
      <argument name="pattern"      explicit-value="(.*)rect"/>
      <argument name="input"  map-name="RectAlrm"/>
</evaluate-property>
<evaluate-property name="specificProblem"
evaluator="centina.sa.server.adapter.snmp.vendor.common.RegExpEvaluator">
      <argument name="pattern"      explicit-value="(.*)Dig"/>
      <argument name="input"  map-name="DigAlrm"/>
</evaluate-property>
```

- map-property tag stores the value of the respective oid in RectAlrm and DigAlrm variables.
- Consider the first evaluate property, **name** is the alarm property, **and evaluator** is the java code packaged with UAA. Now if the value of RectAlrm matches the **pattern** specified in the **explicit-value,** the specific problem property of alarm will take **map-name** as the value, in this case RectAlrm.

Trap synchronize block

synchronize

child tags

- rule-list
- trap-object

Attribute	Required	Description
useAlarmBlock	no	Default value is true. If the value is true, • Alarm block of the trap-id in which this synchronize block is defined is used for parsing and creating the alarm • the trapOid in the explicit block of this synchronize section is to be set the same as the trap OID of this trap-id If the value is set to false, • This condition will iterate through all the trap-id alarm blocks present in the trap xml only if the alarm block has the property resync set to true • The list of OIDs present in the query-agent section of the synchronize block will be queried and a varbinds list is formed and validated against the alarm objects based on the useAlarmBlock condition
instance-list-oid	yes	Snmp table column oid
instance-list-retrieval-type	no	value = index it will append the index of instance-list-oid index to each varbind oid before querying the device.
filter-reg-exp	no	Filter the values based on give regular expression

Example

```
   <synchronize instance-list-oid="1.3.6.1.4.1.868.2.4.1.2.2.5.1.43" instance-
list-retrieval-type="index" filter-reg-exp="[3-6]">
<synchronize useAlarmBlock="true" instance-list-oid="1.3.6.1.2.1.10.18.6.1.1">
<synchronize useAlarmBlock="true" instance-list-
oid="1.3.6.1.4.1.8072.1.3.2.3.1.4" instance-list-retrieval-type="index" >
<synchronize useAlarmBlock="false" instance-list-
oid="1.3.6.1.4.1.1751.2.102.2.1.1.1.1" instance-list-retrieval-type="index">
```

Rulelist

Rule list is used to filter the index's fetched from the above synchronize tag. This is an optional tag and not mandatory. Multiple rules can be added to check for the operational status or to check for raise condition in the SNMP table

child tags

- rule

Example

To add a rule to check if the device responds to certain OID below rule can be used

```
<rulelist>
      <rule object="centina.common.snmp.SnmpRuleRespondsToOID">
            <property name="oid" value="1.3.6.1.2.1.1.2.0"/>
      </rule>
</rulelist>
```

rule

child tags

- property

Attribute	Required	Description
object	True	centina.common.snmp.SnmpRuleOIDIndexRegExp

Example

```
<rule object="centina.common.snmp.decisiontree.SnmpRuleOIDIndexRegExp">
      <property name="oid" value="1.3.6.1.2.1.2.2.1.7"
appendIndex="true"/>
      <property name="regexp" value="1|3"/>
</rule>
```

property

Attribute	Required	Description
name	True	• oid • regexp • invert-result
value	true	Oid value which has to query the device or regular exp.
appendIndex	false	True/false

Example

```
<rule object="centina.common.snmp.decisiontree.SnmpRuleOIDIndexRegExp">
      <property name="oid" value="1.3.6.1.2.1.2.2.1.7"
appendIndex="true"/>
      <property name="regexp" value="1|3"/>
</rule>
```

trap-object

child tags

- explicit
- query-agent
- evaluate-property
- switch-evaluate-property
- switch
- map

Attribute	Required	Description
object	True	Java trap object class

Example

```
<trap-object object="centina.common.snmp.trap.TrapObject">
```

Explicit

This is like other explicit blocks.

```
<explicit>
       <property name="trapPduType" value="2"/>
       <property name="trapOid" value="1.3.6.1.6.3.1.1.5.3"/>
</explicit>
```

query-agent

child tags

- varbind

varbind

Attribute	Required	Description
oid	true	Varbind oid.
appendIndex	false	Default value is false.
mandatory	false	Default value is true.
oid-override	false	Overrides the oid in trap-pdu.

Example

```
 <varbind oid="1.3.6.1.4.1.6321.1.2.2.3.1.1.1.10" appendIndex="true" oid-
override="1.3.6.1.4.1.6321.1.2.2.4.1.2.0"/>


<varbind oid="1.3.6.1.2.1.31.1.1.1.1" appendIndex="true" mandatory="false"/>
```

Alarm Synchronization can fail in the below cases -

- If all the varbinds used in the query-agent do not responds for a given index

- If one of the varbinds form the query-agent fails with an error during polling

Complete Synchronize block example

Example 1

Refer comments in the example for more details

```
 <synchronize instance-list-oid="1.3.6.1.2.1.2.2.1.8" instance-list-retrieval-
type="index" filter-reg-exp="2">
        <rulelist>
              <!-- rule : ifAdminStatus is up(1) or testing(3) -->
           <rule
object="centina.common.snmp.decisiontree.SnmpRuleOIDIndexRegExp">
              <property name="oid" value="1.3.6.1.2.1.2.2.1.7"
appendIndex="true"/>
              <property name="regexp" value="1|3"/> <!— or condition to match
the value 1 or 3 -->
           </rule>
           <!—rule: if the OID 1.3.6.1.2.1.2.2.1.8 has the value 2 -->
           <rule
object="centina.common.snmp.decisiontree.SnmpRuleOIDIndexRegExp">
              <property name="oid" value="1.3.6.1.2.1.2.2.1.8"
appendIndex="true"/>
              <property name="regexp" value="2"/>
           </rule>
        </rulelist>

     <!—the following trap PDU will be created when above rules are passed --
>
        <trap-object object="centina.common.snmp.trap.TrapObject">
           <explicit>
              <property name="trapPduType" value="1"/>
              <property name="genericTrap" value="LINKDOWN"/>
              <property name="trapOid" value="1.3.6.1.6.3.1.1.5.3"/>
           </explicit>
           <query-agent>
              <!--
                 Querying the device for the following OID to create a
varbinds list
              -->
              <varbind oid="1.3.6.1.2.1.2.2.1.1" appendIndex="true"/> <!—this
OID requires index while querying so the appendIndex is marked true -->
              <varbind oid="1.3.6.1.2.1.2.2.1.7" appendIndex="true"/>
              <varbind oid="1.3.6.1.2.1.2.2.1.8" appendIndex="true"/>
              <varbind oid="1.3.6.1.2.1.2.2.1.2" appendIndex="true"/>
              <varbind oid="1.3.6.1.2.1.2.2.1.3" appendIndex="true"/>
```

```
                    <varbind oid="1.3.6.1.2.1.31.1.1.1.1" appendIndex="true"
mandatory="false"/>
<!—The OID is set to false as this OID is not mandatory or not required to
construct an alarm -->
           </query-agent>
        </trap-object>
     </synchronize>
```

Example 2

A simple use case to construct the PDU or varbinds list. The below example has useAlarmblock is true so only the alarm blocks (if resync property set to true) present in this trap-id are evaluated

```
      <synchronize useAlarmBlock="true" instance-list-
oid="1.3.6.1.4.1.6321.1.2.2.3.1.1.1.1">
        <trap-object object="centina.common.snmp.trap.TrapObject">
           <explicit>
              <property name="trapPduType" value="2"/>
              <property name="trapOid" value="1.3.6.1.4.1.6321.1.2.2.4.2.1"/>
           </explicit>
           <query-agent>
              <varbind oid="1.3.6.1.4.1.6321.1.2.2.3.1.1.1.10"
appendIndex="true" oid-override="1.3.6.1.4.1.6321.1.2.2.4.1.2.0"/>
              <varbind oid="1.3.6.1.4.1.6321.1.2.2.3.1.1.1.1"
appendIndex="true" oid-override="1.3.6.1.4.1.6321.1.2.2.4.1.4.0"/>
              <varbind oid="1.3.6.1.4.1.6321.1.2.2.3.1.1.1.11"
appendIndex="true" oid-override="1.3.6.1.4.1.6321.1.2.2.4.1.13.0"/>
           </query-agent>
        </trap-object>
     </synchronize>
```

Inventory

UAA supports monitoring of device entities both physical and logical. The information for inventory/endpoints can be found from the SNMP tables.

SNMP MIBs contain huge number of tables and the RA may not need to include all the tables information. First the tables which must be monitored are to be identified to create the endpoints in RA/UAA

There will be an SNMP table for each different type of entity and all the relevant information will be available from the table columns.

The RA uses the values from the SNMP table columns to map them to the UAA endpoint object properties

The following section explains the properties of the endpoints that UAA supports

- SourceId
- SourceName
- Source Index
- Source Type
- Source Type Id
- endpoint template
- Performance monitoring template
- IP Address
- Mac address
- Operational state
- Admin state
- Description
- Parent

SourceId

This property is used to identify an endpoint uniquely and is generally one of the table Column OIDs but can be mapped as per requirement from one of the table column response. This property is unique and is a mandatory property.

Source Name

This property is the used to map the name of the entity an end point must represent.

Source Index

This property is used to map the SNMP table source Index.

Source Type

This property is to be decided based on the entity the endpoint is created for. Ex: if the endpoint is created from shelf table, source type can be shelf or if the table represent fan information the source type property can be mapped to Fan. This property is used to represent the type of endpoint and can be mapped using the type-name tag in the xml

Source Type Id

The source Type Id should be unique(string/Integer). This property is used to map the Performance metrics to a particular entity. An endpoint can have multiple source type Id's. This property can be mapped using the type-values tag in the xml

NOTE The values from 1 to 301 and reserved for use by IF-MIB and the values 2000, 2001 & 2002 are reserved for use by some of the default endpoints by UAA. So, avoid re using the above-mentioned source Type Ids.

End Point Template:

An endpoint template contains multiple configurable properties which include controlling the performance poll frequency (Refer UAA end point templates page for more details). UAA includes a few default endpoint templates so one of the end point template from the below list can be used based on requirement.

Name	Description
Ethernet FLT/Binned-PM	This template supports Fault and PM features. The endpoints are designated as "ethernet" interfaces and will appear in the ethernet table. PM is stored in binned format on the NE.
Ethernet FLT/PM	This template supports Fault and PM features. The endpoints are designated as "ethernet" interfaces and will appear in the ethernet table.
FLT	This template supports Fault only.
FLT/Binned-PM	This template supports Fault and PM features, with PM in binned format.
FLT/PM	This template supports Fault and PM features.
PM	This template supports PM features but not fault.
System Default EndPoint Template	The System Default EndPoint Template will be assigned to inventoried EPs when no other template is available. This template should not be modified as it may be replaced during software upgrades. The default template disables PM functions.
Unmanaged EndPoint	This template disables Fault, Inventory and PM features.

Performance Monitoring Template

A performance monitoring template is the collection of metric group Ids which enables PM collection on the endpoint. This endpoint property carries the name of Performance monitoring template. An endpoint can have multiple performance monitoring templates.

IP Address

This property is used to map the IP address to the endpoint

Mac Address

This property is used to map the Mac address to the endpoint

Operational state

This endpoint property is to be used to map the operational state of the endpoint. Endpoint also requires a truth value to be set to indicate the proper running state (running) of the endpoint

Administrative state

This endpoint property is used to map the administrative state of the endpoint. Endpoint also requires a truth value to be set to indicate the proper running state (UP) of the endpoint

Description

This property can be mapped with the value which describes the entity

Parent

This property is used to display the endpoints parent endpoint. This property is used to show the parent-child relationship among the device endpoints and is also used for alarm correlation and root cause analysis by UAA. Parent property can be mapped from the java class added to RA

Example: Chassis endpoint is shown in the parent property of its underlying shelf endpoints

Handling Inventory XMLs

Inventory xml is created for those with tables containing relevant end point information.

Location: Project folder -> profiles -> snmp -> inventory

Example Inventory file

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE invProfile PUBLIC "snmp/inventory/invProfile.dtd" "invProfile.dtd">

<!--
   SNMP if-mib inventory
   Copyright (c) 2009 Centina Systems, inc. All rights reserved
-->

<invProfile name="if-mib" version="UNCLEAN">

   <table name="ifTable" oid="1.3.6.1.2.1.2.2.1">
    <types>
      <type type-values="1-5,7-17,20-29,31-52,54-251" inv="y" type-
template="FLT/PM" />
      <type type-values="6" inv="y" type-template="Ethernet FLT/PM" />
      <type type-values="18" inv="y" type-template="FLT" />
      <type type-values="19" inv="y" type-template="FLT" />
      <type type-values="30" inv="y" type-template="FLT" />
      <type type-values="53" inv="y" type-template="FLT" />
      <type type-values="253,255,257-260,262,263,265-272,277-280" inv="y"
type-template="FLT/PM"/>
      <type type-values="252,254,256,261,264,281-296" inv="y" type-
template="FLT/PM"/>
    </types>
      <column name="ifIndex" suffix="1" required="true"/>
      <column name="ifDescr" suffix="2" required="true"/>
      <column name="ifType" suffix="3" required="true"/>
      <column name="ifSpeed" suffix="5"/>
      <column name="ifPhysAddress" suffix="6"/>
      <column name="ifAdminStatus" suffix="7"/>
      <column name="ifOperStatus" suffix="8"/>

      <end-point>
         <property name="sourceId" column="ifIndex" value="OID"/>
         <property name="sourceIndex" column="ifIndex" value="INDEX"/>
         <property name="sourceName" column="ifDescr" value="STRING"/>
         <property name="sourceIfName" column="ifDescr" value="STRING"/>
         <property name="sourceType" column="ifType" value="NAME"/>
         <property name="sourceTypeId" column="ifType" value="STRING"/>
         <property name="speed" column="ifSpeed" value="LONG"/>
         <property name="macAddress" column="ifPhysAddress" value="STRING"/>
         <property name="administrativeStateName" column="ifAdminStatus"
value="NAME"/>
         <property name="administrativeState" column="ifAdminStatus"
value="BOOLEAN" column-regex="1"/>
         <property name="operationalStateName" column="ifOperStatus"
value="NAME"/>
```

```
         <property name="operationalState" column="ifOperStatus"
value="BOOLEAN" column-regex="1"/>
      </end-point>

   </table>

   <!-- ================================================================== --
>

   <table name="ifxTable" oid="1.3.6.1.2.1.31.1.1.1" appends="ifTable">
     <types>
      <type type-values="1-5,7-17,20-29,31-52,54-251" inv="y" type-
template="FLT/PM" pm-template="if-mib" />
      <type type-values="6" inv="y" type-template="Ethernet FLT/PM" pm-
template="if-mib" />
      <type type-values="18" inv="y" type-template="FLT"/>
      <type type-values="19" inv="y" type-template="FLT"/>
      <type type-values="30" inv="y" type-template="FLT"/>
      <type type-values="53" inv="y" type-template="FLT" />
     <type type-values="253,255,257-260,262,263,265-272,277-280" inv="y" type-
template="FLT/PM"  pm-template="if-mib" />
      <type type-values="252,254,256,261,264,281-296" inv="y" type-
template="FLT/PM"  pm-template="if-mib" />
     </types>
      <column name="ifName" suffix="1" required="true"/>
      <column name="ifHighSpeed" suffix="15"/>
      <column name="ifAlias" suffix="18"/>

      <end-point>
         <property name="sourceIndex" column="ifName" value="INDEX"/>
         <property name="sourceName" column="ifName" value="STRING"/>
         <property name="speed" column="ifHighSpeed" value="LONG"
rpn="ifHighSpeed,1000000,*"/>

         <map-property name="OCTETSTRING_TO_ASCII_REPLACEMENT_CHAR" value=""/>
         <property name="description" column="ifAlias"
value="OCTETSTRING_TO_ASCII"/>

      </end-point>

   </table>

</invProfile>
```

Detailed explanation of Tags below

invProfile

This is the root tag for inventory profile xml.

child tags

- table

Attributes

Attribute	Required	Description
name	TRUE	Name of the inventory profile
version	FALSE	(Optional) Alphanumeric version to identify the file

Example

```
<invProfile name="if-mib" version="UNCLEAN">
```

table

This tag is used to map information available in SNMP mib table to endpoint properties. There can be multiple table tags to map multiple tables from the same mib to end points

child tags

- column
- types
- end-point

Attributes

Attribute	Required	Description
name	yes	Table name - maintain unique name across all XMLs in the RA
oid	yes	Table oid – snmp table oid until xxxEntry object = tableOid.1
type-name	no	sourceType name, if type name is null, type-value will be the type-name.
type-values	no	Comma separated integer values or integer range. Type-value should be unique.
type-template	no	Endpoint template name. Should be one from the list explained in the properties List. Usually FLT/PM is used which enables fault and Performance support
pm-template	no	Perf-template name. PM templates are explained in the later sections
appends	no	Appends tag takes the name of the table to which this table information is to be appended. To append the data from one table to other the source Index of both tables should be same Multiple tables can be appended to same table. If the table specifies the appends tag then the sourceIndex is a mandatory property. All the other properties specified in this endpoint tag will overwrite the properties of the parent table/endpoint
inv	no	y|Y|n|N|true|false This tables let the UAA know that the endpoint is disabled. The disabled endpoints are polled but the endpoints are not shown in the endpoints page. An end point can be enabled or disabled from UI settings page

Example

```
1. <table name="moduleTable" oid="1.3.6.1.4.1.231.6.22.1.2.1" table-
processors="fetchNeNameFromNeTable" type-template="FLT/PM" type-values="1200">


2. <table name="deviceTempstatsTable" oid="1.3.6.1.4.1" appends="node" type-
values="2000" inv="y" type-template="FLT/PM" pm-template="device-temperature-
stats-template">
<!—The able table tag does not create any endpoint but appends the PM template
to the node source type 2000 -->
```

Types

Types tag is optional, all type values can be specified in the table tag itself. In case there are multiple type-values or names then the types tag is used to specify the type properties to the endpoint

child tags

- type

Example

```
  <types>
      <type type-values="1-5,7-52,54-251" inv="y" type-template="FLT/PM" />
    </types>
```

type

This tag is used to define the following properties to end point

Attributes

Attribute	Required	Description
type-name	no	SourceType name
type-values	no	SourceTypeId values -Unique Integer numbers
type-template	no	Endpoint template name
pm-template	no	Pm template name
inv	no	Enable/disable inventory, Possible values -y|Y|n|N|true|false Default value is true

Example

```
1.      <types>
      <type type-values="1-5,7-52,54-251" inv="y" type-template="FLT/PM" pm-
template="if-mib" />
      <type type-values="6" inv="y" type-template="Ethernet FLT/PM" pm-
template="if-mib" />
      <type type-values="53" inv="y" type-template="FLT" />
```

```
      <type type-values="301" inv="y" type-template="FLT/PM" pm-template="if-mib" />
     </types>
```

```
2. <types>
          <type type-values="21795" type-name="Slot"       inv="y" type-template="FLT" />
          <type type-values="21796" type-name="Port"       inv="y" type-template="FLT" />
      </types>
```

column

This tag is for SNMP table column. Multiple column tags can be added as required based on the entries present in the SNMP table

It contains name, suffix and required attributes.

Attributes

Attribute	Required	Description
name	yes	Column name – usually SNMP table column name
suffix	yes	Suffix to the table oid, (table oid + .suffix = column oid).
required	no	True/false – If required = true, column should have value in device snmp table, otherwise UAA will not create endpoints from this table. If this column is mapping to key property of endpoint like sourceId, sourceName, sourceType, sourceTypeId, sourceIndex then the column should be marked required true. If the column is mapping to optional property of endpoint sourceDescription, operational state – we can mention required = false.

Example

```
<column name="ifName" suffix="1" required="true"/> <!-- Marked true as this column is used to map the sourceName -->
      <column name="ifHighSpeed" suffix="15"/>
```

end-point

Endpoint is the tag used to map all the end point properties using the columns mapped in the columns tags

child tags

- property
- map-property
- script-property

Example

```
<end-point>
         <property name="sourceId" column="ifIndex" value="OID"/>
         <property name="sourceIndex" column="ifIndex" value="INDEX"/>
         <property name="sourceName" column="ifDescr" value="STRING"/>
         <property name="sourceIfName" column="ifDescr" value="STRING"/>
         <property name="sourceType" column="ifType" value="NAME"/>
         <property name="sourceTypeId" column="ifType" value="STRING"/>
         <property name="speed" column="ifSpeed" value="LONG"/>
         <property name="macAddress" column="ifPhysAddress" value="STRING"/>
         <property name="administrativeStateName" column="ifAdminStatus"
value="NAME"/>
         <property name="administrativeState" column="ifAdminStatus"
value="BOOLEAN" column-regex="1"/>
         <property name="operationalStateName" column="ifOperStatus"
value="NAME"/>
         <property name="operationalState" column="ifOperStatus"
value="BOOLEAN" column-regex="1"/>
      </end-point>

   </table>
```

property

Attributes

Attribute	Required	Description
name	yes	Name of the endpoint property
column	yes	Column name from column tag

value	yes	Value can take one of the below values
		• STRING
		  ○ picks the columns response used
		• OID
		  ○ picks the complete OID(Including the index) of the column
		• INDEX
		  ○ Picks the Index of the column. (The values in the OID after the column value is taken as index)
		• NAME
		  ○ Picks the response and tries to find an alternate or alias from the mib-repository.
		  ○ **Example:** Consider IANAIfType Object from the IANAIfType-MIB which has syntax as integer but specifies a list of string names corresponding to each source type. In this case to pick the string from the response NAME is to be used in the value tag
		
		IANAifType ::= TEXTUAL-CONVENTION
		STATUS current
		DESCRIPTION
		""
		SYNTAX INTEGER {
		other(1), -- none of the following

		regular1822(2),
		hdh1822(3),
		ddnX25(4),
		rfc877x25(5), ----
		
		In this case when the response returns an integer 1 the property will be set to other(1)
		• BOOLEAN
		○ Boolean is used for operational and admin state mapping. This tag should be used in combination with column-regex which should specify the truth value i.e., a proper working condition so UAA knows the working condition
		○ **Example:** consider Object ifOperstatus
		
		ifOperStatus OBJECT-TYPE
		SYNTAX INTEGER {
		up(1), -- ready to pass packets
		down(2),
		testing(3), -- in some test mode
		unknown(4), -- status can not be determined
		-- for some reason.
		dormant(5),
		notPresent(6), -- some component is missing
		lowerLayerDown(7) -- down due to state of
		-- lower-layer interface(s)
		}
		
		○ The column regex here should specify the value 1 as this says operational state is UP
		○ UAA would consider any value other than 1 as a down condition
		• LONG
		○ Long is specified to pick the speed value from the response
		• CUSTOM_STRING
		○ This is used in combination with format to set custom string with replaceable variables

column-regex	no	Regular expression specifying the UP condition
rpn	no	Reverse polish notation used to do mathematical computations
format	no	This is used to specify the variables in $'s Ex: $variableName$ which are created using map property so the value is translated

map-property

child tags

- oid-index-translator: this tag can be repeated more than once

Attributes

Attribute	Required	Description
name	True	Name of the property to map in the inventory file

Attribute	Required	Description
column	False	Name of column of map-property

value	False	Value can take one of the below values
		• STRING
		○ picks the columns response used
		• OID
		○ picks the complete OID (Including the index) of the column
		• INDEX
		○ Picks the Index of the column. (The values in the OID after the column value is taken as index)
		• NAME
		○ Picks the response and tries to find an alternate or alias from the mib-repository.
		○ **Example:** Consider IANAIfType Object from the IANAIfType-MIB which has syntax as integer but specifies a list of string names corresponding to each source type. In this case to pick the string from the response NAME is to be used in the value tag
		
		IANAifType ::= TEXTUAL-CONVENTION
		STATUS current
		DESCRIPTION
		""
		SYNTAX INTEGER {
		other(1), -- none of the following
		regular1822(2),
		hdh1822(3),
		ddnX25(4),
		rfc877x25(5), ----
		
		○ In this case when the response returns an integer 1 the property will be set to other(1)
		• BOOLEAN
		○ Boolean is used for operational and admin state mapping. This tag should be used in combination with column-regex which should specify the truth value i.e., a proper working condition so UAA knows the working condition
		○ **Example:** consider Object ifOperstatus
		
		ifOperStatus OBJECT-TYPE
		SYNTAX INTEGER {
		up(1), -- ready to pass packets
		down(2),
		testing(3), -- in some test mode
		unknown(4), -- status can not be determined
		-- for some reason.
		dormant(5),
		notPresent(6), -- some component is missing
		lowerLayerDown(7) -- down due to state of
		-- lower-layer interface(s)
		}
		
		○ The column regex here should specify the value **1** as this says

Attribute	Required	Description
		operational state is **UP** ○ UAA would consider any value other than 1 as a down condition • LONG ○ Long is specified to pick the speed value from the response • CUSTOM_STRING ○ This is used in combination with format to set custom string with replaceable variables
column-regex	False	if column regex is specified, use it as a truth test
rpn	False	Reverse polish notation used to do mathematical computations
format	False	This is used to specify the variables in $'s **Example:** $variableName$ which are created using map property so the value is translated

Example

Example 1:

```
        <map-property column="adGenSlotInfoIndex" name="index"
value="INDEX"/>
        <property format="Slot-$index$" name="sourceName"
value="CUSTOM_STRING"/>
```

Example 2:

```
<map-property column="bsnRogueAPDot11MacAddress" name="mac" value="OID">
               <oid-index-translator delim=":" length="6" start-index="12"
type="HEX_FIXED_LENGTH_IX_ONLY"/>
           </map-property>


 <property format="BSN Rogue AP-$mac$" name="sourceName"
value="CUSTOM_STRING"/>
               <property format="$mac$" name="macAddress"
value="CUSTOM_STRING"/>
```

oid-index-translator

Following are the OID Index Translators:

- ASCII_FIXED_LENGTH,
- ASCII_VARIABLE_LENGTH,
- IP_ADDR_V4,
- ASCII_FIXED_LENGTH_IX_ONLY,
- ASCII_VARIABLE_LENGTH_IX_ONLY,
- IP_ADDR_V4_IX_ONLY,
- OID_FIXED_LENGTH_IX_ONLY,
- OCTETS_FIXED_LENGTH,
- OCTETS_FIXED_LENGTH_IX_ONLY,

- OCTETS_VARIABLE_LENGTH,
- OCTETS_VARIABLE_LENGTH_IX_ONLY,
- OID_VARIABLE_LENGTH_IX_ONLY,
- HEX_FIXED_LENGTH_IX_ONLY,
- HEX_FIXED_LENGTH;

Attributes

Attribute	Required	Description
type	yes	Enum values mentioned above
start-index	yes	Specifying the start index
length	no	The count to pick the number of values separated by delimiter
delim	no	Default is "." Not required to specify when working on OID's

Example

```
<oid-index-translator delim=":" length="6" start-index="12"
      type="HEX_FIXED_LENGTH_IX_ONLY"/>
```

script-property

Attributes

Attribute	Required	Description
name	True	Name of the script property
scriptType	True	javascript
scriptFunction	True	Definition of the script function
scriptInvokeName	False	Name of the script function

Example

```
  <script-property name="desc" scriptType="JavaScript"
scriptFunction="function getDesc(){
      var original = discreteAlarmStateDescription;
        var replace_normal = original.replace(/NORMAL$/, '');
        return replace_normal;
      }"
      scriptInvokeName="getDesc,discreteAlarmStateDescription"/>
      <property name="description" value="CUSTOM_STRING" format="$desc$"/>
```

Complete Inventory Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE invProfile PUBLIC "snmp/inventory/invProfile.dtd" "invProfile.dtd">
```

```
<!--
   SNMP if-mib inventory
   Copyright© 2020 Ciena® Corporation. All Rights Reserved
-->

<invProfile name="if-mib" version="UNCLEAN">

   <table name="ifTable" oid="1.3.6.1.2.1.2.2.1">
    <types>
      <type type-values="1-5,7-52,54-251" inv="y" type-template="FLT/PM" />
      <type type-values="6" inv="y" type-template="Ethernet FLT/PM" />
      <type type-values="53" inv="y" type-template="FLT" />
      <type type-values="301" inv="y" type-template="FLT/PM" />
    </types>
      <column name="ifIndex" suffix="1" required="true"/>
      <column name="ifDescr" suffix="2" required="true"/>
      <column name="ifType" suffix="3" required="true"/>
      <column name="ifSpeed" suffix="5"/>
      <column name="ifPhysAddress" suffix="6"/>
      <column name="ifAdminStatus" suffix="7"/>
      <column name="ifOperStatus" suffix="8"/>

      <end-point>
         <property name="sourceId" column="ifIndex" value="OID"/>
         <property name="sourceIndex" column="ifIndex" value="INDEX"/>
         <property name="sourceName" column="ifDescr" value="STRING"/>
         <property name="sourceIfName" column="ifDescr" value="STRING"/>
         <property name="sourceType" column="ifType" value="NAME"/>
         <property name="sourceTypeId" column="ifType" value="STRING"/>
         <property name="speed" column="ifSpeed" value="LONG"/>
         <property name="macAddress" column="ifPhysAddress" value="STRING"/>
         <property name="administrativeStateName" column="ifAdminStatus"
value="NAME"/>
         <property name="administrativeState" column="ifAdminStatus"
value="BOOLEAN" column-regex="1"/>
         <property name="operationalStateName" column="ifOperStatus"
value="NAME"/>
         <property name="operationalState" column="ifOperStatus"
value="BOOLEAN" column-regex="1"/>
      </end-point>

   </table>
```

```
 <!-- ================================================================== -->


   <table name="ifxTable" oid="1.3.6.1.2.1.31.1.1.1" appends="ifTable">
     <types>
      <type type-values="1-5,7-52,54-251" inv="y" type-template="FLT/PM" pm-
template="if-mib" />
      <type type-values="6" inv="y" type-template="Ethernet FLT/PM" pm-
template="if-mib" />
      <type type-values="53" inv="y" type-template="FLT" />
      <type type-values="301" inv="y" type-template="FLT/PM" pm-template="if-
mib" />
     </types>
      <column name="ifName" suffix="1" required="true"/>
      <column name="ifHighSpeed" suffix="15"/>
      <column name="ifAlias" suffix="18"/>
      <end-point>
         <property name="sourceIndex" column="ifName" value="INDEX"/>
         <property name="sourceName" column="ifName" value="STRING"/>
         <property name="speed" column="ifHighSpeed" value="LONG"
rpn="ifHighSpeed,1000000,*"/>
      </end-point>


   </table>
</invProfile>\
```

Layer2 connections/services

RA supports the discovery of following services in UAA

- Ethernet Virtual Connections (EVC)
- Layer 2 adjacencies
- Virtual LANs

Ethernet Virtual Connections

Carrier Ethernet Services operate over Ethernet Virtual Connections or EVCs. An **EVC** is an association of two or more UNIs that limits the exchange of Service Frames to UNIs in the Ethernet Virtual Connection (EVC). A given UNI can support more than one EVC via the Service Multiplexing attribute.

To create an Ethernet service from MIB tables the information for the table columns or MIB objects, has to be mapped to the two of the below UAA Java Objects either from the XML or from the Java class that is created in RAJavaImpl folder in the project folder

- centina.sa.model.profile.layer2.EvcInfo
- centina.sa.model.profile.layer2.EvcLink

centina.sa.model.profile.layer2.EvcInfo

This object is used by RA to create an ethernet service in UAA. The properties of this object are listed below

- evcId
- logicalId
- description

An empty ethernet service will be created when the above properties are mapped and the endpoint associations are mapped from the EvcLink class

centina.sa.model.profile.layer2.EvcLink

EvcLink object is used to add the associated endpoints to the ethernet service already created using EvcInfo Object.

The properties of the EvcInfo are listed below

- logicalId
  - The logical Id of the service to which an endpoint is to be associated
- ADeviceId
  - This property stores the value to identify the node (self-node property)
  - This property can use one of the following node properties
    - Ip
    - baseBridgeId - a node property which is used to map the Mac address
- ADeviceIdPropName
  - This property identifies the value stored in the ADeviceId. Uses one of the following Values
    - IP
    - MAC
  - If this property is set as **ip** then the value in ADeviceId property is considered as node IP, else if, this property is set to **MAC** then the value of ADeviceId property is considered as MAC address of node

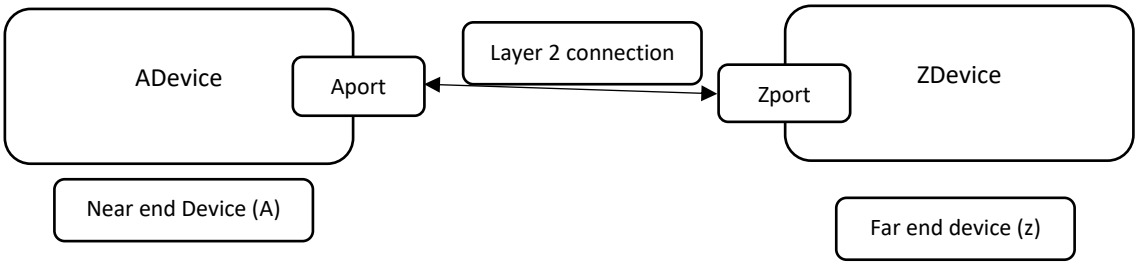
- APort
    - This property stores the endpoint information which is to be associate to the service
- APortPropName
    - This property identifies the value stored in the APort
        - sourceId
        - sourceName
    - if this property is set to sourceId then the value in APort is considered as endpoint sourceId else if, this property is set to sourceName then the value in APort is considered as endpoint sourceName
- stateString
    - This property is used to map the Operational state string
- StateFromOid
    - This property picks value from OID to map the Operational state

Layer 2 adjacencies

MIB tables contains the information regarding the layer2 connections and the device use two protocols to store the connections information

- LLDP
    - The link layer discovery protocol (LLDP) is an open and extendable part of the Internet protocol suite used in IEEE 802 to advertise a device's identity and abilities, as well as other devices connected within the same network.
    - LLDP is mainly used in wired Ethernet-connected devices to facilitate management of network resources and simplify networking tasks for administrators in a multi-vendor network.
- CDP
    - The Cisco Discovery Protocol or commonly called as just CDP is a Cisco Proprietary Protocol. It is a Data Link Layer Protocol. CDP is used to collect and share information about directly connected Cisco devices.

UAA uses the following terminology to represent devices and links



- ADevice – Identifies the self-node properties or near end side device property
- Aport – Identifies the endpoint on the near end side
- Zdevice - Identifies the node on the far end side
- Zport – identifies the port on the far end side

The following UAA Java Object is to be created either in RA or from the Java Impl

- centina.sa.model.profile.layer2.Link

centina.sa.model.profile.layer2.Link

The properties of the Link Object are explained below -

- ADeviceId
    - This property stores the value to identify the node (self-node property)
    - This property can use one of the following node properties
        - Ip
        - baseBridgeId - a node property which is used to map the Mac address
- ADeviceIdPropName
    - This property identifies the value stored in the ADeviceId. Uses one of the following Values
        - IP
        - MAC
    - If this property is set as **ip** then the value in ADeviceId property is considered as node IP, else if, this property is set to **MAC** then the value of ADeviceId property is considered as MAC address of node
- APort
    - This property stores the endpoint information which is to be associate to the service
- APortPropName
    - This property identifies the value stored in the APort
        - sourceId
        - sourceName
    - if this property is set to sourceId then the value in APort is considered as endpoint sourceId else if, this property is set to sourceName then the value in APort is considered as endpoint sourceName
- ZDeviceId
    - This property identifies the far end device
    - This property can use one of the following node properties
        - IP
        - MAC
- ZDeviceIdPropName
    - This property identifies the value stored in the ZDeviceId. Uses one of the following Values
        - Ip
        - baseBridgeId - a node property which is used to map the Mac address
    - If this property is set as **ip** then the value in ZDeviceId property is considered as node IP, else if, this property is set to **baseBridgd** then the value of ZDeviceId property is considered as MAC address of node

Virtual LANs

To create VLANs from SNMP MIB information, RA uses the following two classes to create a VLAN and add associate corresponding ports or endpoints to the VLANs created

- centina.sa.model.profile.layer2.Vlan
- centina.sa.model.profile.layer2.VlanPorts

centina.sa.model.profile.layer2.Vlan

This UAA java object is used to create a layer 2 service. Using RA XMLs or Java class the following properties are to be mapped from the MIB tables

- deviceId
  - This value is used to identify the node has similar mapping as in ADeviceId
- deviceIdProp
  - Identifies the value stored in deviceId property and has similar mapping as in ADeviceIdPropName
- name
  - Name of the VLAN can be mapped to this property
- vlanIndex
  - VLAN index based on the VLAN interface index
- vlanId
  - VLAN ID which should be unique across the topology

centina.sa.model.profile.layer2.VlanPorts

This Object is used to map the proper interface endpoints to the corresponding VLANs. The following properties are to be mapped in either the XMLs or in the java class

- deviceId
  - deviceId property should have the value as mapped in Vlan object
- taggedPorts
  - List of tagged port values
- taggedPortsProp
  - Identifies the end point property of values in taggedPorts
- untaggedPorts
  - List of untagged port values
- untaggedPortsProp
  - Identifies the end point property of values in taggedPorts
- vlanIndex
  - VLAN index based on the VLAN interface index to associate the endpoints

NOTE

The tagged ports can be shown or picked only when that port is part of any of the layer2 adjacency

The tagged ports are shown the topology diagram but not in the endpoints section of the VLAN service in the UI

The untagged ports are shown in the endpoints section of the VLAN service

Handling Layer2 files from XMLs

The layer2 files root tag is to be selected based on the functionality.

- EVC services use the evcProfile as the root tag
- VLans and layer2adjacencies use the layer2Profile as root tag

The other table and the corresponding child tags are common for all the services and are to be used as required

layer2Profile

This tag is used to support the layer2 feature. This is the root tag of layer2 xml.

Child Tags

- Table

Attributes

Attributes	Required	Description
name	True	Name of the layer2profile
version	False	(Optional) Version to identify the file

Example

```
<layer2Profile name="ciena-lldp-layer2-adj-discovery" version="UNCLEAN">
```

evcProfile

This tag is used to support EVC functionality.

Child Tags

- Table

Attributes

Attributes	Required	Description
name	True	Name of the EVC profile
version	False	

Example

```
<evcProfile name="accedian-nid-coe-evc-discovery" version="UNCLEAN">
```

table

This tag is used to map information available in snmp mib table to VLAN, EVCs and adjacencies.

child Tags

- column
- object

Attributes

Attributes	Required	Description
name	True	Name of the table
oid	True	To identify every table uniquely

Example

```
<table name="acdPaaL2CfgTable" oid="1.3.6.1.4.1.22420.2.5.4.1">
```

column

This tag is used to represent the object of a table.

Attributes

Attributes	Required	Description
name	True	Name of the column in a table
suffix	True	Adding suffix to entry and allows to reach the different columns of table
required	False	Required column

Example

```
 <column name="acdPaaL2CfgID" suffix="1" required="true"/>
```

object

Child Tags

- property
- map-property
- script-property
- operstate-property

Attributes

Attributes	Required	Description
class	True	The type of java class object to be created. Refer the Layer2 files/service section for list of objects to be created

Example

```
<object class="centina.sa.model.profile.layer2.VlanPorts">  </object>
```

```
<object class="centina.sa.model.profile.layer2.Link"> </object>
```

property

Attributes

Attributes	Required	Description
name	True	Name of property
value	True	It can be • CUSTOM_STRING • NE_FIELD • HEX_TO_IF_LIST – This converts the obtains hex string to the port numbers/interface indexes and this is used while mapping VLANs ports when the portsprop is sourceId
format	False	Use property name as variable – using dollar symbols mac
rpn	False	Used to specify the field from which Network element property is to be picked
column	False	Name of the property column

Example

```
1)<property name="ADeviceIdPropName" value="CUSTOM_STRING" format="MAC" />

2)<property name="ADeviceId" value="NE_FIELD" rpn="baseBridgeId" />

3)<property name="APort" value="OID" column="acdPaaL2CfgName" />
```

map-property

Child Tags

- oid-index-translator

Attributes

Attributes	Required	Description
name	True	Name of the property to map in the layer2 file
column	False	Name of column of map-property
value	False	
column-regex	False	If column regex is specified, use it as a truth test
rpn	False	Used to do some calculation
format	False	Use property name as variable – using dollar symbols mac/name

Example

```
1)  <map-property name="paaIndex" value="LONG" column="acdPaaL2CfgPeerID"/>
```

```
2)<map-property name="ifIndex" column="cdpCacheAddress">
      <oid-index-translator type="OID_FIXED_LENGTH_IX_ONLY" start-index="14"
length="1" />           </map-property>
```

script-property

Attributes

Attributes	Required	Description
name	True	Name of the script property
scriptType	True	JavaScript
scriptFunction	True	Definition of the script function
scriptInvokeName	False	Name of the script function

Example

```
<script-property name="macString" scriptType="JavaScript"
                  scriptFunction="
                        function getMacString()
                        {
                              var macs = acdPaaL2CfgMacDst;
                              macs = macs.replace(/(^\s+|\s+$)/g,'');
                              macs = macs.toLowerCase();
                              macs = macs.replace(/\s+/g, ':');
                              return macs;
                        }"
scriptInvokeName="getMacString,acdPaaL2CfgMacDst"/>
```

operstate-property

Attributes

Attributes	Required	Description
column-regex	True	Regular Expression

Example

```
<operstate-property column-regex="2"/>
```

oid-index-translator

This tag is used by map-property tag to get sub-OID string from start-index to the length specified. Works in the same way as used in inventory.

Example

```
<map-property name="ifIndex" column="cdpCacheAddress">
      <oid-index-translator type="OID_FIXED_LENGTH_IX_ONLY" start-index="14"
length="1" />
</map-property>
```

Complete Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE evcProfile PUBLIC "snmp/layer2/layer2Profile.dtd"
"layer2Profile.dtd">

<evcProfile name="accedian-nid-coe-evc-discovery" version="UNCLEAN">

  <table name="acdPaaL2CfgTable" oid="1.3.6.1.4.1.22420.2.5.4.1">
        <column name="acdPaaL2CfgID" suffix="1" required="true"/>
        <column name="acdPaaL2CfgName" suffix="2" required="true"/>
      <column name="acdPaaL2CfgMacDst" suffix="21" required="true"/>
      <column name="acdPaaL2CfgPeerID" suffix="32" required="true"/>

            <object class="centina.sa.model.profile.layer2.EvcLink">

                  <property name="ADeviceIdPropName" value="CUSTOM_STRING"
format="MAC" />
                  <property name="ADeviceId" value="NE_FIELD"
rpn="baseBridgeId" />

                  <property name="APortPropName" value="CUSTOM_STRING"
format="sourceId" />
                  <property name="APort" value="OID" column="acdPaaL2CfgName"
/>



                  <property name="ZDeviceIdPropName" value="CUSTOM_STRING"
format="MAC" />
                  <script-property name="macString" scriptType="JavaScript"
                  scriptFunction="
                        function getMacString()
                        {
                              var macs = acdPaaL2CfgMacDst;

                              macs = macs.replace(/(^\s+|\s+$)/g,'');
                              macs = macs.toLowerCase();
                              macs = macs.replace(/\s+/g, ':');
                              return macs;
```

```
                        }"
scriptInvokeName="getMacString,acdPaaL2CfgMacDst"/>
                  <property name="ZDeviceId" value="CUSTOM_STRING"
format="$macString$" />

                  <property name="ZPortPropName" value="CUSTOM_STRING"
format="sourceId"/>
                    <map-property name="paaIndex" value="LONG"
column="acdPaaL2CfgPeerID"/>
                  <property name="ZPort" value="CUSTOM_STRING"
format="1.3.6.1.4.1.22420.2.5.4.1.2.$paaIndex$"/>

            </object>
      </table>

</evcProfile>
```

```
2)<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE layer2Profile PUBLIC "snmp/layer2/layer2Profile.dtd"
"layer2Profile.dtd">

<layer2Profile name="ciena-layer2-vlan-discovery" version="UNCLEAN">

      <table name="wwpLeosVlanTable" oid="1.3.6.1.4.1.6141.2.60.5.1.1.4.1">
            <column name="wwpLeosVlanId" suffix="1" required="true"/>
            <column name="wwpLeosVlanName" suffix="2" required="true"/>

            <object class="centina.sa.model.profile.layer2.Vlan">

                  <!-- Device's MAC address on which this VLAN is queried -->
                  <property name="deviceIdProp" value="CUSTOM_STRING"
format="MAC" />
                  <property name="deviceId" value="NE_FIELD"
rpn="baseBridgeId" />

                  <!-- Setting the name of VLAN based on the VLAN description
-->
                  <property name="name" value="STRING"
column="wwpLeosVlanName" />

                  <!-- Setting of VLAN index based on the VLAN interface index
-->
                  <property name="vlanIndex" value="INDEX"
column="wwpLeosVlanId" />
```

```
                <!-- Setting the VLAN ID which should be unique across the
topology -->
                <property name="vlanId" value="STRING"
column="wwpLeosVlanId" />
            </object>

      </table>
```

Example mapping to map VLAN Ports from the extremeVlanOpaqueTable (EXTREME-VLAN-MIB)

The tagged ports and untagged ports information is available from the below table columns

```
extremeVlanOpaqueTaggedPorts OBJECT-TYPE
        SYNTAX  PortList
        MAX-ACCESS      read-only
        STATUS  current
        DESCRIPTION
                "Each bit in the octet string represents one port.
                        A 1 means that the port is a tagged port in that vlan.
                The bit value for a port is 0 otherwise."
        ::= { extremeVlanOpaqueEntry 1}
extremeVlanOpaqueUntaggedPorts OBJECT-TYPE
                SYNTAX PortList
                MAX-ACCESS  read-only
                STATUS current
                DESCRIPTION
                "Each bit in the octet string represents one port.
                        A 1 means that the port is an untagged port in that
vlan.
                The bit value for a port is 0 otherwise."
        ::= { extremeVlanOpaqueEntry 2 }
```

Using sourceId as the PortsProp and converting the value from these columns to interface endpoints index's using the value HEX_TO_IF_LIST

```
<table name="extremeVlanOpaqueTable" oid="1.3.6.1.4.1.1916.1.2.6.1.1" >

   <column name="extremeVlanOpaqueTaggedPorts" suffix="1" required="true"/>
   <column name="extremeVlanOpaqueUntaggedPorts" suffix="2" required="true"/>
```

```
   <object class="centina.sa.model.profile.layer2.VlanPorts">

      <!-- Device's MAC address on which this VLAN is queried -->
      <property name="deviceId" value="NE_FIELD" rpn="baseBridgeId" />

      <!-- Below map properties extract index into two variables, vlanIndex
and slotNumber -->

      <!-- First part of the index is a Vlan interface index, used to identify
the VLAN -->
      <!-- The Hex value of this column is convereted to binary. The bit
position whose value is '1' indicates
           a tagged port. The slot of this port is as part of this table's
index -->
      <property name="taggedPorts" value="HEX_TO_IF_LIST"
column="extremeVlanOpaqueTaggedPorts" />
      <property name="taggedPortsProp" value="CUSTOM_STRING" format="sourceId"
/>


      <!-- The Hex value of this column is convereted to binary. The bit
position whose value is '1' indicates
           an untagged port. The slot of this port is as part of this table's
index -->
      <property name="untaggedPorts" value="HEX_TO_IF_LIST"
column="extremeVlanOpaqueUntaggedPorts" />
      <property name="untaggedPortsProp" value="CUSTOM_STRING"
format="sourceId" />

      <map-property name="vlanIfIndex" column="extremeVlanOpaqueTaggedPorts">
         <oid-index-translator type="OID_FIXED_LENGTH_IX_ONLY" start-
index="13" length="1" />
      </map-property>
      <property name="vlanIndex" value="CUSTOM_STRING" format="$vlanIfIndex$"
/>

      <!-- Second part of the index is the slot number. -->
      <map-property name="slotNumberIdx"
column="extremeVlanOpaqueTaggedPorts">
         <oid-index-translator type="OID_FIXED_LENGTH_IX_ONLY" start-
index="14" length="1" />
      </map-property>
      <property name="slot" value="CUSTOM_STRING" format="$slotNumberIdx$" />

   </object>
</table>
```

Performance Monitoring

UAA supports performance monitoring on all the endpoints created and the node itself. The performance monitoring is enabled by creating metric objects in RA for a SNMP MIB object/column which gives the metric related information

Performance metrics are shown either on endpoints or on the node, so performance metrics monitoring is possible only when a related endpoint is monitored by UAA

The concept of performance monitoring involves the following terms

- Performance metrics
- Metric Groups
- Performance monitoring templates

The terms are explained in detail below -

Performance metrics

SNMP MIBs contain the details of objects which measure the performance stats. Performance metrics for the device physical or logical entities can be found in the corresponding tables.

RA can accommodate any number of metrics per source type and Metrics can be mapped to an endpoint from multiple different tables from different MIBs.

There should be one metric object/tag created in RA for one snmp mib object. A metric tag carries all the essential information to properly poll the device and show it on the assigned endpoint.

Properties of metric object or tag are as follows

- id
- name
- desc
- protocol
- units
- conversion-function
- consolidation-function
- display-type
- display-color
- location
- direction
- min
- max

id

A metric id is the value that is used by the software to identify the metric uniquely. This value has to be unique among all the performance metric xml files and across RAs.

NOTE	The Id should be prefixed with **"sdk-"** while developing SNMP RAs using SDK Kit to avoid the name space conflict for the RAs developed by Blue Planet RA team.

name

The metric property is a string which will be displayed in the UAA UI. An appropriate name has to be set based on the metric description by the developer.

desc

The description property is to be set by the developer appropriately.

protocol

Protocol is to be set as SNMP

Units

Appropriate metric units which indicate the value being polled is to be set by developer

conversion-function

A conversion function has following Enums defined -

- NONE
- PER_PERIOD
- PER_PERIOD_PER_SECOND
- PER_PERIOD_TIMES_SECOND
- PER_SECOND
- TIMES_SECOND

consolidation-function

The metric collection in UAA is based on the poll frequency set in the end point template and will always poll the values and store it to compute the values for last 1 hour and last 1 day. The last 1 hour and the last 1 day data cannot be polled from devices.

To aggregate or calculate the values for last 1 hour and last 1 day, a method has to be specified in the consolidation function tag.

Based on the method specified the last 1 hour and last 1 day values stored are aggregated/calculated and the value is shown in UI

The available functions are -

- AVG
- SUM
- MIN
- MAX

displayType

This property sets the graph style that the metric has to be displayed. The available options are

- verticalBar-hist

    - Metric will be displayed as a bar graph. Generally used for the counter metrics like the packet stats
- lineSeries-hist
    - Metric will be displayed as a linear line graph. Generally used for the gauge metrics like the temperature measurement.

displayColor

This property sets the color for the graph. The available options are -

- BLUE
- RED
- ORANGE
- YELLOW
- MAGENTA
- INDIGO
- GREEN
- VIOLET
- DARK_BLUE
- DARK_RED
- DARK_ORANGE
- DARK_YELLOW
- DARK_MAGENTA
- DARK_INDIGO
- DARK_GREEN
- DARK_VIOLET

location

Specifies the location of the metric and the available options are listed below -

- NA
- NEAR_END
- FAR_END

direction

Specifies the direction of the metric and the available options are listed below -

- NA
- RECEIVE
- TRANSMIT
- BOTH

min

Specifies the minimum value for the metric.

max

Specifies the maximum value for the metric.

Metric Groups

RA groups multiple related metrics into one metric group. Any metric should be a part of some metric group. A performance xml will generally have multiple metric groups and metrics in turn have multiple metrics in them

A metric group has a metric group id which is used in the performance monitoring templates.

NOTE | Metric group name and Id should be unique

Performance monitoring templates

Performance monitoring templates are the collection of pm metric group ids. Performance monitoring templates have an id which is specified in the pm-template property of endpoint to assign all the metrics present in the underlying metrics groups.

Handling Performance XML

This xml is used to define performance metric groups/metrics.

pmProfile

This is the root tag of the pm xml.

Child tags

- MetricGroup

Attributes

Attribute	Required	Description
name	yes	Name of the mib.
version	no	(optional) Version of the file.

Example

```
<pmProfile name="sonicwall-firewall-ip-statistics-mib" version="UNCLEAN">
</pmProfile>
```

metricGroup

This tag is used to define group of related metrics.

Child tags

- Metric

Attributes

Attribute	Required	Description
id	yes	The unique value of the metric group
name	yes	Name of the metric group
protocol	yes	SNMP
displayType	yes	It is the display type of the metric group. It can have the following values: NORMAL - Display all metrics in the metric group on individual charts COMPOSITE - Display all metrics in the metric group in one chart.

Example

```
  <metricGroup name="SonicWALL CPU Usage" id="sonicwall-firewall-ip-
statistics-mib.CPU Usage" protocol="SNMP" displayType="Normal">
</metricGroup>
```

metric

This tag defines all the attributes of performance metric through which the performance of the device is collected.

Child tags

- parameter
- min
- value
- max
- history-value
- history-parameter

Attributes

Attribute	Required	Description
id	yes	A unique value of the metric
name	yes	Name of the metric
parmType	no	It is the metric type. Few of the values it can have are as follows: • VENDOR_SPECIFIC • ETH_RX_BANDWIDTH • ETH_RX_BYTES
desc	no	Details about the metric.
protocol	yes	SNMP
units	no	Unit of measurement used for the metric
conversion-function	yes	It is the performance derivative type. It can have the following values • NONE – it is the actual performance value collected from the device • PER_PERIOD - Difference between polls • PER_SECOND - Average per second • PER_PERIOD_PER_SECOND - Difference between polls, average per second • TIMES_SECOND - • PER_PERIOD_TIMES_SECOND -
consolidation-function	yes	Consolidation functions are used to consolidate performance values to a larger bin size i.e., last 1 hour and last 1 day It can have the following values: • SUM - The sum of all the values collected in the specified frequency will be set as the performance value • AVG - The average of all the values collected in the specified frequency will be set as the performance value • MIN - The minimum value out of all the values collected in the specified frequency will be set as the performance value

Attribute	Required	Description
		• MAX - The maximum value out of all the values collected in the specified frequency will be set as the performance value
location	no	It can have the following values: • NA – not applicable • NEAR_END • FAR_END
direction	no	It can have the following values: • NA – not applicable • RECEIVE • TRANSMIT • BOTH
min	no	This is used to set the scale on the graph. It is the minimum value from which the graph has to start. Example 0.
max	no	This is used to set the scale on the graph. It is the maximum value for the graph. Example 100.
displayType	no	Type used to display the metric in graphs. lineSeries-hist verticalBar-hist
displayColor	no	Color used to display the metric in graphs. It can have the following values: BLUE, RED, ORANGE, YELLOW, MAGENTA, INDIGO, GREEN, VIOLET, DARK_BLUE, DARK_RED, DARK_ORANGE, DARK_YELLOW, DARK_MAGENTA, DARK_INDIGO, DARK_GREEN, DARK_VIOLET
alwaysDisplayInPmPage	no	It can have the following values: y, Y, n, N, true, false If true, in details page the metric is displayed even if it does not collect pm.

Example

```
<metric id="sonicwall-firewall-ip-statistics-mib.CPU Used"
            name="SonicWALL CPU Usage"
            protocol="SNMP"
            units="%"
            direction="NA"
            location="NA"
            min="0"
            max="100"
            displayType="lineSeries-hist"
            displayColor="DARK_GREEN"
            conversion-function="NONE"
            consolidation-function="AVG"
            desc="Instantaneous CPU Utilization in percent."

            alwaysDisplayInPmPage="true"
```

parameter

It is the metric variable which has the information to collect performance value

Child tags

- Discriminator

Attributes

Attribute	Required	Description
name	yes	Name of the metric variable
collector	no	SNMP
oid	no	Oid on which the device is queried for the performance value
type	no	Performance value type. It can have the following values • COUNTER - Ever increasing counter value that only resets when it overflows. • GAUGE - Absolute value that rises and falls.
indexed	no	Specifies whether to pick the index from the endpoint or not. It can have following values: true, TRUE, 1, false, 0, FALSE
regex	no	A regex can be applied on the response and the always the first match group is picked to display in UI
aggregation-function	no	When the response of the oid is a bulk data, to select which value to be considered, this attribute is used. It can have the following values • COUNT • SUM • AVG • MIN • MAX • VALUE_1 • VALUE_2 • VALUE_3 • VALUE_4 • VALUE_5 • VALUE_6 • VALUE_7 • VALUE_8
discriminator-function	no	It is the logical function applied on more than one discriminator tag. It can have the following values: • AND • OR
onResynchOid	no	Oid specified will be queried during resynchronization. Ex: OID of a metric like speed
oidSuffix	no	Indicates the value that must be appended to the oid (after appending index)
index-type	no	This attribute will be specified only if indexed = “true"

Attribute	Required	Description
		It can have the following values • DEFAULT – appends the endpoint index • OBJECT_PROPERTY – allows to append the customized index (i.e to append any one of the endpoint's property as an index to the oid)
index	no	This attribute will be specified only if index-type = ”OBJECT_PROPERTY” It specifies which property of the endpoint must be appended as an index to the oid
index-converter	no	This attribute will be specified only if index-type = ”OBJECT_PROPERTY” converts the specified index. It can have the following values - • DECIMAL_TO_HEX • HEX_TO_DECIMAL • NONE

Example

```
<parameter name="sonicCurrentCPUUtil"
      collector="SNMP"
      oid=".1.3.6.1.4.1.8741.1.3.1.3.0"
      type="GAUGE"
      indexed="false"/>
<parameter name="sfpDiagnosticTemperature"
      collector="SNMP"
      oid=".1.3.6.1.4.1.18070.2.8.100.1.2.9.4.1.4"
      regex="&amp;quot;*[ ]*([-.0-9]+).*&amp;quot;*"
      type="GAUGE"
      indexed="true"/>
<parameter name="cadIfCmtsCmStatusRangFlaps"
      collector="SNMP"
      oid=".1.3.6.1.4.1.4998.1.1.20.2.2.1.13"
      type="COUNTER"
      indexed="true"
      index-type="OBJECT_PROPERTY"
      index-converter="HEX_TO_DECIMAL"
      index="macAddress"
      index-input-separator=":"
      index-output-separator="."/>
<parameter name="speed"
      collector="OBJECT"
      oid="speed"
      onResynchOid=".1.3.6.1.2.1.2.2.1.5"
      type="GAUGE"
```

```
      indexed="true"/>
<parameter name="genEquipPmTrafficSLMinRsl"
      collector="SNMP"
      oid=".1.3.6.1.4.1.2281.10.6.3.2.1.1.4.2"
      type="GAUGE"
      indexed="true"
      oidSuffix="0"/>
<parameter name="hwDocsIf3CmtsCmUsStatusTxPower"
      collector="SNMP"
      oid=".1.3.6.1.4.1.2011.6.180.1.1.20.4.1.1"
      type="GAUGE"
      indexed="true"
      aggregation-function="VALUE_1"/>
```

discriminator

This tag is used to append the customized index to the polling oid. Only used when the aggregation-function property is defined in the parameter.

For example, to get all the modems which are online of specific upstream/downstream can be achieved by this.

Attributes

Attribute	Required	Description
name	yes	Name of the discriminator
type	yes	It has the value as COLUMN
oid	no	Specifies which oid must be queried
regex	yes	Indicates which value must be picked from the oid response

Example

```
<discriminator name="downstreamIndex"
      type="COLUMN"
      oid=".1.3.6.1.2.1.10.127.1.3.3.1.4"
      regex="${sourceIndex}"/>
2. <parameter name="cmRegistered" collector="SNMP"
oid=".1.3.6.1.2.1.10.127.1.3.3.1.9" regex="6"   type="GAUGE" indexed="false"
aggregation-function="COUNT">
            <discriminator name="upstreamIndex" type="COLUMN"
oid=".1.3.6.1.2.1.10.127.1.3.3.1.5"
      regex="${sourceIndex}"/>
  </parameter>
```

min

This tag is used to set the metric minimum value.

Attributes

Attribute	Required	Description
parameter	no	Variable which stores the collected pm
rpn	no	Indicates the calculations that has to be performed on the collected pm

Example

```
<min rpn="cLApLinkLatencyStatsMin,10,*"/>
<min parameter="OPRMIN"/>
```

value

This tag is used to represent the metric value.

Attributes

Attribute	Required	Description
parameter	no	Variable which stores the collected pm
rpn	no	Indicates the calculations that has to be performed on the collected pm

Example

```
<value rpn="sonicCurrentCPUUtil"/>
<value parameter="cmRegistered"/>
```

max

This tag is used to set the metric maximum value.

Attributes

Attribute	Required	Description
parameter	no	Variable which stores the collected pm
rpn	no	Indicates the calculations that has to be performed on the collected pm

Example

```
<max rpn="cLApLinkLatencyStatsMax,10,*"/>
<max parameter="OPRMAX"/>
```

history-parameter

The oid in this tag will be polled if binned is true in the inventory xml. This tag is used to poll the binned/history values.

Attributes

Attribute	Required	Description
name	yes	Name of the metric history variable.
collector	no	SNMP
oid	no	Oid on which the device is queried by the history variable for the performance value.
type	no	History performance value type. It can have the following values • COUNTER - Ever increasing counter value that only resets when it overflows. • GAUGE - Absolute value that rises and falls.
indexed	no	It tells whether to pick the index from the endpoint or not. It can have following values: true, TRUE, 1, false, 0, FALSE
onResynchOid	no	Oid specified will be queried by the history variable during resynchronization
oidSuffix	no	Indicates the value that has to be appended to the oid (after appending index)

Example

```
<history-parameter name="f3PtpSOOCHistoryAvgOffsetFromMaster"
      collector="SNMP"
      oid=".1.3.6.1.4.1.2544.1.12.18.2.8.1.7"
      type="COUNTER"
      indexed="true"
      oidSuffix="1"/>
```

history-value

This tag is used to set the metric binned/history value.

Attributes

Attribute	Required	Description
parameter	no	Variable which stores the collected pm
rpn	no	Indicates the calculations that must be performed on the collected pm

Example

```
<history-value parameter="f3PtpSOOCHistoryAvgOffsetFromMaster"/>
```

```
<history-value rpn="necIntervalStpS"/>
```

Complete Example

```
<pmProfile name="if-mib" version="UNCLEAN">

   <metricGroup name="Ethernet Utilization" id="if-mib.percent utilization"
protocol="SNMP" displayType="Composite">

      <metric
         id="if-mib.Receive average utilization"
         name="Ethernet Rx Avg Utilization"
         parmType="ETH_RX_UTILIZATION"
         desc="Average percent utilization received on this interface"
         protocol="SNMP"
         units="%"
         conversion-function="PER_PERIOD_PER_SECOND"
         consolidation-function="AVG"

         displayType="lineSeries-hist"
         displayColor="DARK_GREEN"
         min="0"
      >
         <parameter name="ifInOctets" collector="SNMP"
oid=".1.3.6.1.2.1.2.2.1.10" type="COUNTER" indexed="true"/>
         <parameter name="speed" collector="OBJECT" oid="speed"
onResynchOid=".1.3.6.1.2.1.2.2.1.5" type="GAUGE" indexed="true"/>
         <value rpn="ifInOctets,8,*,speed,/,100,*"/>
      </metric>

      <metric
         id="if-mib.Transmit average utilization"
         name="Ethernet Tx Avg Utilization"
         parmType="ETH_TX_UTILIZATION"
         desc="Average percent utilization transmitted on this interface"
         protocol="SNMP"
         units="%"
         conversion-function="PER_PERIOD_PER_SECOND"
         consolidation-function="AVG"

         displayType="lineSeries-hist"
         displayColor="BLUE"
         min="0"
      >
```

```
         <parameter name="ifOutOctets" collector="SNMP"
oid=".1.3.6.1.2.1.2.2.1.16" type="COUNTER" indexed="true"/>
         <parameter name="speed" collector="OBJECT" oid="speed"
onResynchOid=".1.3.6.1.2.1.2.2.1.5" type="GAUGE" indexed="true"/>
         <value rpn="ifOutOctets,8,*,speed,/,100,*"/>
      </metric>


   </metricGroup>
</pmProfile>
```

Performance Template

The performance template xml contains the tags that define the metric groups and performance thresholds of a performance template.

objects

This is the root tag of pm template.

Child Tags

- perf-template

Attributes

Attributes	Required	Description
version	False	Unique identification for the Pm threshold xmls. UNCLEAN-No proper version

Example

```
 <objects version="UNCLEAN">
</object>
```

perf-template

This tag is used to identify the pm templates.

Child Tags

- description
- templateType
- pm-metric-groups
- pm-threshold-policies

Attributes

Attributes	Required	Description
name	True	Name of Pm Template

Example

```
  <perf-template name="adc-hdsl">
 </perf-template>
```

description

This tag is used to mention the description of the performance monitoring template.

Example

```
    <description>sonet-path tl1 rx tx and fe</description>
```

templateType

This tag describes type of the template.

Example

```
 <templateType>PLUGIN</templateType>
```

pm-metric-groups

This tag is used to describe the set of performance metric groups.

Child Tags

- value

Attributes

Attributes	Required	Description
type	CDATA #FIXED "String"	Type of pm-metric-groups input. It can be String.

Example

```
    <pm-metric-groups type="String">
      <value>sonet-path-tl1</value>
      <value>sonet-path-tl1.rx</value>
   </pm-metric-group>
```

Value
Value tag contains the metric group Id. Multiple value tags are allowed.

Example:

```
<value>sonet-path-tl1</value>
<value>sonet-path-tl1.rx</value>
```

pm-threshold-policies

This tag is used to define the pm Threshold Policy.

Attributes

Attributes	Required	Description
type	CDATA #FIXED "String"	Type of pm-metric-groups input. It can be String.

Example

```
 <pm-threshold-policies type="String">
 </pm-threshold-policies>
```

Complete Example

```
<objects version="UNCLEAN">

    <perf-template name="adc-hdsl">
        <description>sonet-path tl1 rx tx and fe</description>
        <templateType>PLUGIN</templateType>

        <pm-metric-groups type="String">
            <value>sonet-path-tl1</value>
            <value>sonet-path-tl1.rx</value>
            <value>sonet-path-tl1.tx</value>
            <value>sonet-path-tl1.fe</value>
            <value>sonet-path-tl1.rx.fe</value>
            <value>sonet-path-tl1.tx.fe</value>
            <value>adc-soneplex-tl1.ne</value>
            <value>adc-soneplex-tl1.fe</value>
        </pm-metric-groups>

        <pm-threshold-policies type="String">
        </pm-threshold-policies>

    </perf-template>

</objects>
```

RAJavaImpl

Using java implementation in the RAJavaImpl a few features listed below can be handled

- Parent child modelling
- LAG
- Layer2 services

Java class implements an interface class "**PluginInterface**" from the model jar. This provides multiple methods to be implemented but Out of all available methods RA uses the two following methods to implement all the above-mentioned features.

- getProcessedEpList
- toGenericBeans

PluginInterface class for reference below -

```
public interface PluginInterface
{

    Long SERIAL_VERSION_UID = 0L;

    String IP_ADDRESS_DELIMITTER = ".";

    public HashMap getOidList();

    public ArrayList<SNMPLayer2Table> getSnmpTables();

    public ArrayList<GenericBean> processData();

    public ArrayList<GenericBean> toGenericBeans(
        SNMPLayer2Table layer2Table, NetworkElement ne, Target remoteTarget,
Snmp snmp,
        SnmpMessageDispatcher dispatcher, SnmpMIBRepository mibRepository,
        CommunicationSettings commSettings)
            throws Exception;

    public GenericTL1CommandSet getCommandSet();

    public ArrayList<EndPoint> getProcessedEpList(
                NetworkElement ne, Target remoteTarget, Snmp snmp,
                SnmpMessageDispatcher dispatcher,CommunicationSettings
commSettings,ArrayList<EndPoint> newEpList);

}
```

Details of functions that are used in RA implementation

getProcessedEpList

UAA processes all the table tags from the inventory XMLs and creates EndPoint objects based on the logic provided. Once all the objects are created these Objects are passed as an argument to this method to further process these endpoints. Developer can use custom logic to update the endpoint properties and return the complete set of endpoints that are to be monitored/displayed in the UI.

Method Arguments

```
public ArrayList<EndPoint> getProcessedEpList(
               NetworkElement ne, Target remoteTarget, Snmp snmp ,
               SnmpMessageDispatcher dispatcher,CommunicationSettings
commSettings,ArrayList<EndPoint> newEpList)
```

ne

This is an instance of NetworkElement Object which is created based on the details provided from the device/profile XML and contains all the NE level Details and can be accessed by using the get methods

Useful get methods -

- ne.getIp() returns the IP of the node
- ne.getprofileId() returns the RA name
- ne.getDeviceType() returns the Device type
- ne.getSysObjectID() returns the sysOID of the node

remoteTarget, snmp, dispatcher, commSettings

These arguments are related to the snmp4j which are used to query/poll the device. The usages are explained in the examples section

newEpList

List of all the Endpoint objects of the node. These Object properties are to be updated if required and the complete set of endpoints with updated properties is to be returned.

toGenericBeans

This method is used to create the layer2 services. The method arguments are explained below

```
public ArrayList<GenericBean> toGenericBeans(
        SNMPLayer2Table layer2Table, NetworkElement ne, Target remoteTarget,
Snmp snmp,
        SnmpMessageDispatcher dispatcher, SnmpMIBRepository mibRepository,
        CommunicationSettings commSettings)
```

layer2Table

This argument contains the information of table tag from the XML, or the table created in the getSnmpTables method.

mibRepository

This is the mibrepository of the RA which has all the details of the MIB objects and OIDs

commSettings

This Object provides the details of the communications setting of UAA. Required setting values can be obtained by using the get method

A few methods for reference below -

- getSnmpConnectionTimeout() returns snmp connection timeout
- getSnmpConnectionRetryCount() returns snmp connection retry count

Implemention and other helper functions explained using a few examples below

Helper Methods in Java Impl

To create an OID instance

OID is provided by the snmp4j jar

```
import org.snmp4j.smi.OID;
OID snmptableOID = new OID("1.3.6.1.2.1.4.34.1.3"); // constructing an OID
from string
```

To poll/query device using snmp OID

The centina.common.snmp.GenericSNMPTableMapper provides useful methods to poll OID

getTableColumnVariableBindings

This method takes OID & other objects related to snmp4j as input and queries the device and returns an ArrayList with the device response.

Example: Polling a device using the snmp table OID

```
ArrayList<VariableBinding> evcvarbinds  =
GenericSNMPTableMapper.getTableColumnVariableBindings( snmptableOID, null,
remoteTarget, snmp, dispatcher, commSettings, ne );
```

VaribaleBinding

VaribaleBinding is provided by the org.snmp4j.smi

The response is stored in a ArrayList of Variablebinding objects. A variable binding is the combination of OID and its response.

vb.getOid() – returns the OID

vb.getVariable() – returns the response of the OID

```
for( VariableBinding vb : evcvarbinds )
{
   if( vb.getOid() == null || vb.getOid().toString() == null )
    {
         continue;
    }
     System.out.println("OID from VariableBinding: " + vb.getOid());
     System.out.println("Response value from VariableBinding: " +
vb.getVariable());

 }
```

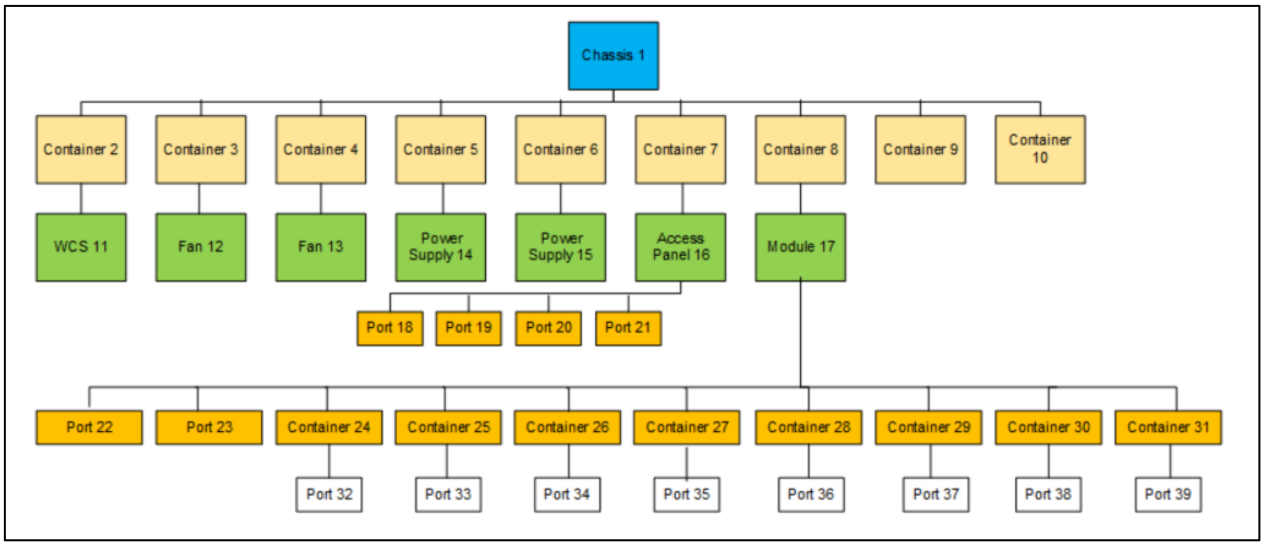
Extracting the Index from VariableBinding

```
//Example OID
OID oid = new OID("1.3.6.1.4.1") // Creating the table OID instance

for( VariableBinding vb : evcvarbinds )
{
   if( vb.getOid() == null || vb.getOid().toString() == null )
    {
         continue;
    }
     //chop the index
     // Below line of code get the index of the table OID using the oid
variable created in line 1
    //+1 is done to get the index value without the dot
String indx = vb.getOid().toString().substring( oid.toString().length() + 1 );
System.out.println("Index: " + indx);
}
```

Creating Parent-child relationship

An endpoint has a property to identify its parent object. Parent child relationship can be best explained using below image



The parent child relationship is used in alarm correlation and root cause analysis by the UAA. The above image represents the parent child relationship of a device in the top-down model. Each box in the above image is created as an endpoint.

All the endpoint Objects have a setter method to set its parent.

```
childEP = < Child end point Object >
ParentEP = <Parent end point Object >
childEP.setParent(ParentEP)
```

NOTE	Parent-child relationship is set in the getProcessedEpList method This method is called by UAA once all the endpoint objects are created as per the logic provided in RA inventory XMLs

The argument to this method has the Network element details and the list of endpoints

The logic should update the parent (if any) to the endpoints objects using the setParent() method. setParent method takes the parent endpoint as argument

Example: Mapping parent property to the endpoints

Consider the case to map the Parent child relationship between the Virtual connection endpoints and the pseudo wire endpoints from the MIBs.

Virtual connection table from the CISCO-IETF-PW-MIB

```
cpwVcTable   OBJECT-TYPE
  SYNTAX      SEQUENCE OF CpwVcEntry
  MAX-ACCESS   not-accessible
  STATUS      current
  DESCRIPTION
     "This table specifies information for connecting various
     emulated services to various tunnel type."
  ::= { cpwVcObjects 2 }

cpwVcEntry   OBJECT-TYPE
  SYNTAX      CpwVcEntry
  MAX-ACCESS   not-accessible
  STATUS      current
  DESCRIPTION
      "A row in this table represents an emulated virtual
      connection (VC) across a packet network. It is indexed by
      cpwVcIndex, which uniquely identifying a singular
      connection.
     "
  INDEX  { cpwVcIndex }

    ::= { cpwVcTable 1 }

CpwVcEntry ::= SEQUENCE {
    cpwVcIndex           CpwVcIndexType,
    cpwVcType            CpwVcType,
    cpwVcOwner            INTEGER,
    cpwVcPsnType          INTEGER,
    cpwVcSetUpPriority      Integer32,
    cpwVcHoldingPriority     Integer32,
    cpwVcInboundMode        INTEGER,
    cpwVcPeerAddrType       InetAddressType,
    cpwVcPeerAddr          InetAddress,

    cpwVcID              CpwVcIDType,
    cpwVcLocalGroupID       CpwGroupID,
    cpwVcControlWord        TruthValue,
    cpwVcLocalIfMtu        Unsigned32,
    cpwVcLocalIfString      TruthValue,
    cpwVcRemoteGroupID      CpwGroupID,
    cpwVcRemoteControlWord   INTEGER,
    cpwVcRemoteIfMtu        Unsigned32,
    cpwVcRemoteIfString     SnmpAdminString,
    cpwVcOutboundVcLabel     Unsigned32,
    cpwVcInboundVcLabel      Unsigned32,
```

```
    cpwVcName             SnmpAdminString,
    cpwVcDescr            SnmpAdminString,
    cpwVcCreateTime       TimeStamp,
    cpwVcUpTime           TimeTicks,
    cpwVcAdminStatus      INTEGER,
    cpwVcOperStatus       CpwOperStatus,
    cpwVcInboundOperStatus  CpwOperStatus,
    cpwVcOutboundOperStatus CpwOperStatus,
    cpwVcTimeElapsed      Integer32,
    cpwVcValidIntervals   Integer32,
    cpwVcRowStatus        RowStatus,
    cpwVcStorageType      StorageType
  }
```

PW endpoint table from the CISCO-IETF-PW-TDM-MIB

```
cpwCTDMCfgTable OBJECT-TYPE
   SYNTAX       SEQUENCE OF CpwCTDMCfgEntry
   MAX-ACCESS   not-accessible
   STATUS       current
   DESCRIPTION
      "This table contains a set of parameters that may be
      referenced by one or more TDM PWs in cpwCTDMTable."
   ::= { cpwCTDMObjects 3 }

cpwCTDMCfgEntry OBJECT-TYPE
   SYNTAX       CpwCTDMCfgEntry
   MAX-ACCESS   not-accessible
   STATUS       current
   DESCRIPTION
      "These parameters define the characteristics of a
      TDM PW. They are grouped here to ease NMS burden.
      Once an entry is created here it may be re-used
      by many PWs."
   INDEX        { cpwCTDMCfgIndex }
   ::= { cpwCTDMCfgTable 1 }

CpwCTDMCfgEntry ::= SEQUENCE {
      cpwCTDMCfgIndex               CpwTDMCfgIndex,
      cpwCTDMCfgConfErr             BITS,
      cpwCTDMCfgPayloadSize         Unsigned32,
      cpwCTDMCfgPktReorder          TruthValue,
      cpwCTDMCfgRtpHdrUsed          TruthValue,
      cpwCTDMCfgJtrBfrDepth         Unsigned32,
      cpwCTDMCfgPayloadSuppression  INTEGER,
      cpwCTDMCfgConsecPktsInSynch   Unsigned32,
      cpwCTDMCfgConsecMissPktsOutSynch  Unsigned32,
      cpwCTDMCfgSetUp2SynchTimeOut  Unsigned32,
      cpwCTDMCfgPktReplacePolicy    INTEGER,
      cpwCTDMCfgAvePktLossTimeWindow  Integer32,
      cpwCTDMCfgExcessivePktLossThreshold Unsigned32,
      cpwCTDMCfgAlarmThreshold      Unsigned32,
      cpwCTDMCfgClearAlarmThreshold Unsigned32,
      cpwCTDMCfgMissingPktsToSes    Unsigned32,
      cpwCTDMCfgTimestampMode       INTEGER,
      cpwCTDMCfgStorageType         StorageType,
```

cpwCTDMCfgRowStatus *RowStatus*
}

Column responses for reference below

cpwVcName

.1.3.6.1.4.1.9.10.106.1.2.1.21.1 = STRING: "CEM0/1/**0**"
.1.3.6.1.4.1.9.10.106.1.2.1.21.2 = STRING: "CEM0/1/**1**"
.1.3.6.1.4.1.9.10.106.1.2.1.21.3 = STRING: "CEM0/1/**2**"

We have relation between the sourceName and the index of the table cpwCTDMCfgTable. One of the column responses of this table to identify the relation.

cpwCTDMCfgPayloadSize

.1.3.6.1.4.1.9.10.131.1.3.1.3.**0** = Gauge32: 192
.1.3.6.1.4.1.9.10.131.1.3.1.3.**1** = Gauge32: 192
.1.3.6.1.4.1.9.10.131.1.3.1.3.**2** = Gauge32: 192

The values marked in color match the value of the index in the second table. Screenshot from UAA endpoints page showing parent property for reference below -

Parent	Source	Source Type	Source Id	Source Index	End Point Template
CEM0/1/2	4	TDM PW	1.3.6.1.4.1.9.10.131.1.3.1.3.2	2	FLT/PM
CEM0/1/0	4	TDM PW	1.3.6.1.4.1.9.10.131.1.3.1.3.0	0	FLT/PM
CEM0/1/1	4	TDM PW	1.3.6.1.4.1.9.10.131.1.3.1.3.1	1	FLT/PM
	CEM0/1/0	virtual connection	1.3.6.1.4.1.9.10.106.1.2.1.26.1	1	FLT/PM
	CEM0/1/1	virtual connection	1.3.6.1.4.1.9.10.106.1.2.1.26.2	2	FLT/PM
	CEM0/1/2	virtual connection	1.3.6.1.4.1.9.10.106.1.2.1.26.3	3	FLT/PM

The below example explains the logic to map the parent child relation between these endpoints -

```
 for( EndPoint ep: epList )// Loop to create a map of required endpoints
{
    try
    {
        // filter the endpoints using source type
        if ("43473".equalsIgnoreCase(ep.getSourceTypeId()))
        {
            if(ep.getSourceName() != null)
            {
                String vcName = null;
/**

Logic to get the last value from the sourceName
        CEM0/1/0 ( 0
**/
                vcName = ep.getSourceName().split("/")[2];
// Map to store the index as key and the endpoint object as value
                virtualConnection.put(vcName, ep);
            }
        }

    }
    catch (Exception ex)
    {
        log.error("Unable to get the required sourcename for virtual
connection endpoint " +ex );
    }
}
```

```
// setting parent-child for virtual connection and TDM PW
// Iterating through the whole list of endpoints again to set the parent
property
for( EndPoint ep: epList )
{
    try
    {
        // Setting parent to the 44691 endpoints
        if("44691".equalsIgnoreCase(ep.getSourceTypeId()))
        {
           if(ep.getSourceIndex()!= null)
           {
               // Setting the parent by passing the parent endpoint object
                ep.setParent(virtualConnection.get(ep.getSourceIndex()));
            }
        }
    }
    catch (Exception ex)
    {
        log.error("Unable to set the parent for TDM PW sourcetype ep "+
ep.getSourceName() + "," + ex );
    }
}
```

LAG association

Endpoints have the following properties to support the LAG feature. This has to be set in the getProcessedEp method.

- associationType
  Property to set the name of association. Example names below
  - LinkAggregationGroup
  - MAC Domain
  - L2-L3

```
higerEp = <Endpoint object to which the association is to created>
ChildEp = <Endpoint which is part of the higher Endpoint>

higerEp.setProperty("associationType", "LinkAggregationGroup");
```

- associationName
  Property to identify the endpoints of an association

```
higerEp = <Endpoint object to which the association is to created>
ChildEp = <Endpoint which is part of the higher Endpoint>

higerEp.setProperty("associationName", ChildEp);
```

Screenshot from UAA endpoints page for reference below

Association Name	Association Type	Source	Source Id	Source Type	Source Index	Parent
Bundle-Ether2	LinkAggregationGroup	HundredGigE0/0/1/0	1.3.6.1.2.1.2.2.1.1.72	ethernet-csmacd(6)	72	0/RP0-Slice 0 Qumran Port Module #32

Layer2 services

The layer 2 service objects are created in the toGenericBeans method. This method is called once for each table present in the layer2 XMLs and a corresponding object as mentioned in the XML will be created and returned to UAA.

This method can be used directly to create the Layer 2 services instead of relying on the XMLs from the RA.

Method Arguments explained -

```
public ArrayList<GenericBean> toGenericBeans(
            SNMPLayer2Table layer2Table, NetworkElement ne, Target
remoteTarget, Snmp snmp,
            SnmpMessageDispatcher dispatcher, SnmpMIBRepository mibRepository,
            CommunicationSettings commSettings)
```

All the arguments are similar to the getProcessedEpList method so these can be used to query the device. This method has an argument which gives the information of SNMPLayer2Table which is created in XML. This method is called once for each table from XML during the topology/EVC discovery to create the services.

The layer2Table argument can be ignored when implementing a custom logic as this is used to create service Object from XMLs by RA

The layer2 Objects and their respective properties are explained in the Layer 2 adjacencies section. This method should create an instance of required service object and map the properties as required and return the list of all service object instances in a list.

Example: Creating EVCInfo object

```
import centina.sa.model.profile.layer2.EvcInfo;
EvcInfo evc = new EvcInfo();
evc.setEvcId( // Todo set value);
evc.setLogicalId(// Todo set value);
evc.setName(// Todo set value);
```

All the service objects have setter methods to set the required properties. The device can be queried to fetch the data from required tables to use in creating the service

Example: Creating the EVCLink

```
import centina.sa.model.profile.layer2.EvcLink;
EvcLink evcLink = new EvcLink();
evcLink.setADeviceIdPropName( "IP");
evcLink.setADeviceId( ne.getIp() );
evcLink.setAPortPropName( "sourceId" );
evcLink.setAPort( "1.3.6.1.4.1." + //Todo fill the OID using custom logic );
evcLink.setLogicalId( //Todo Set the logicalId);
```

Example: Creating a Layer 2 adjacency

```
import centina.sa.model.profile.layer2.Link;
Link link = new Link();
link.setADeviceId(ne.getBaseBridgeId())// Todo set the device ID Ex: device
IP/basebridgeID);
link.setADeviceIdPropName("MAC");
link.setAPort(// Todo set the endpoint identifier value);
```

```
link.setAPortPropName("sourceName");
link.setZDeviceId((// Todo set the device ID Ex: device IP/basebridgeID);
link.setZDeviceIdPropName("MAC");
link.setZPort((// Todo set the endpoint identifier value);
link.setZPortPropName("sourceName");
```

Example: Creating a vLAN info object

```
Vlan vl = new Vlan();
vl.setName(// Todo set the vlan name);
vl.setDeviceIdProp("MAC");
vl.setVlanId(// set the vlanId value);
vl.setVlanIndex(// set the vlanIndex value);
```

Example: Creating vlanports Object

```
import centina.sa.model.profile.layer2.vlanports
VlanPorts vlanports = new VlanPorts();
vlanports.setDeviceId(ne.getBaseBridgeId());
vlanports.setVlanIndex(// Todo set the vlanindex);
vlanports.setUntaggedPorts(List<String>// Todo set the list of untaggedports
index's);
/**
        when the sourceId is set as property, the ports property should be
list of interface index's
**/
vlanports.setTaggedPortsProp(EndPoint._SOURCE_ID);
vlanports.setTaggedPorts(List<String> // Todo set the list of tagged port
index's);
vlanports.setUntaggedPortsProp(EndPoint._SOURCE_ID);
```

PM Automation Script

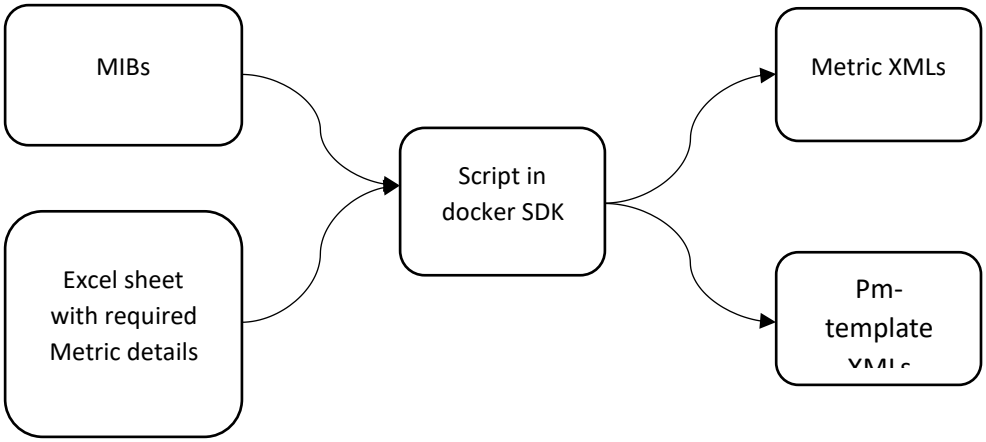
SDK Docker image provides a script to create the performance metrics and pm-template files by reading the data from an excel sheet. This can be used as a starting point to create the basic version of metrics and the templates.

Script gets the OID, Description and syntax related details to fill in the metric tag in the XMLs from the MIB or mib-repository.

All the files related to this script are present in MetricxmlGenerator folder inside the project folder and the excel sheet should also be created in this folder.

pre-requisites

- MIBs should be error free
- MIB name should be with the MIB definition name
    - **Example:** if the MIB definition is **EXAMPLEDefinition** then the mib name should be **EXAMPLEDefinition.mib**
- Profile template generation should be done successfully as this script uses mib-repository to calculate the OID of the parameters



Functional steps

1. Create an excel sheet as per the required template in the location ProjectFolder -> MetricxmlGenerator
2. Fill the metrics and template data required in the above created sheet
3. Run the docker container and provide the RA name and the excel sheet name as input
4. The script will generate the XMLs and the templates to the location ProjectFolder -> MetricxmlGenerator -> xmls

NOTE

Script converts the MIBs to json format to fetch information of OID, Description and syntax related information.

Json conversion can fail if there are errors in the MIB, In this case script uses the MIB repository to calculate the OID.

The script uses a template xml to create metrics, this can be modified to meet various requirements.

Creating and Filling the data in Excel sheet

The script does a few functions based on the below column names -

MIB

This column name is used to get the MIB from the RA mibs folder. The json converted MIBs are created into the RA MIBs folder itself.

GroupName

This column is to be filled with the required metric Group name

parameter

This column is to be filled with the MIB object name to which metric is to be created

MetricName

This column is to be filled with the metric name

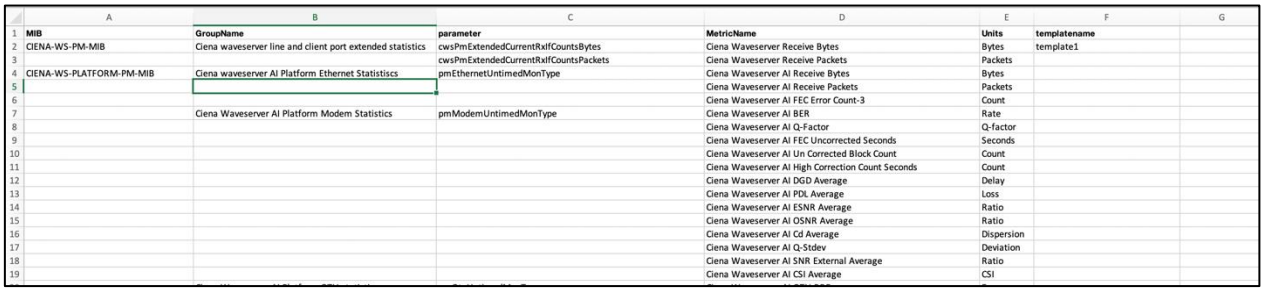
Units

The columns is to be filled with the units of the metric

templatename

This column is to be filled with the desired pm template name for the metric group

A screenshot of a sample excel sheet below



	A	B	C	D	E	F	G
1	**MIB**	**GroupName**	**parameter**	**MetricName**	**Units**	**templatename**	
2	CIENA-WS-PM-MIB	Ciena waveserver line and client port extended statistics	cwsPmExtendedCurrentRxIfCountsBytes	Ciena Waveserver Receive Bytes	Bytes	template1	
3			cwsPmExtendedCurrentRxIfCountsPackets	Ciena Waveserver Receive Packets	Packets		
4	CIENA-WS-PLATFORM-PM-MIB	Ciena waveserver AI Platform Ethernet Statistiscs	pmEthernetUntimedMonType	Ciena Waveserver AI Receive Bytes	Bytes		
5				Ciena Waveserver AI Receive Packets	Packets		
6				Ciena Waveserver AI FEC Error Count-3	Count		
7		Ciena Waveserver AI Platform Modem Statistics	pmModemUntimedMonType	Ciena Waveserver AI BER	Rate		
8				Ciena Waveserver AI Q-Factor	Q-factor		
9				Ciena Waveserver AI FEC Uncorrected Seconds	Seconds		
10				Ciena Waveserver AI Un Corrected Block Count	Count		
11				Ciena Waveserver AI High Correction Count Seconds	Count		
12				Ciena Waveserver AI DGD Average	Delay		
13				Ciena Waveserver AI PDL Average	Loss		
14				Ciena Waveserver AI ESNR Average	Ratio		
15				Ciena Waveserver AI OSNR Average	Ratio		
16				Ciena Waveserver AI Cd Average	Dispersion		
17				Ciena Waveserver AI Q-Stdev	Deviation		
18				Ciena Waveserver AI SNR External Average	Ratio		
19				Ciena Waveserver AI CSI Average	CSI		

Forward filling in Excel sheet

The script fills the value into the empty cells using the following strategy. The value for an empty cell is filled with a value from its first higher cell which has a value.

Example:

	MIB	GroupName
2	CIENA-WS-PM-MIB	Ciena waveserver line and client port extended statistics
3		
4	CIENA-WS-PLATFORM-PM-MIB	Ciena waveserver AI Platform Ethernet Statistiscs
5		
6		
7		Ciena Waveserver AI Platform Modem Statistics
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		

In above case the value in the cell A3 will be filled with value in A2. Similarly, the values for the cells from A5 till the next empty cell will be filled with the value in A4.

Any column in the sheet will be forward filled as shown in the above example.

NOTE Custom column names can be used in metric creation by doing corresponding changes in template.xml file

Metric template

The default template.xml file is present in MetricxmlGenerator folder inside project folder. The template xml has a default metric template set and contains the placeholders for columns used in the excel sheet.

```
<metric
      id="{{id}}"
      name="{{MetricName}}"
      desc="{{description}}"
      protocol="{{Protocol}}"
      units="{{Units}}"
      conversion-function="{{ConversionFuntion}}"
      consolidation-function="{{ConsolidationFunction}}"
      displayType="{{Displaytype}}"
      displayColor="{{Displaycolour}}"
    >
      <parameter name="{{parameter}}" collector="SNMP" oid="{{oid}}" type="{{Type}}"
indexed="{{Indexed}}"/>
      <value parameter="{{parameter}}"/>
 </metric>
```

The values enclosed in “{{” are the column names in the excel sheet. The placeholder values shown in the template are filled by the script.

This template can be modified as required to include different metric format and new placeholders can be added by adding different columns in excel sheet so the script will replace the value in excel sheet to the metric XML.

Example:

To fill a custom description to the metrics, user should add a column with a name **customdescription** (user defined name) and the same column name is to be written added to the template.xml

```
<metric
      id="{{id}}"
      name="{{MetricName}}"
      desc="{{customdescription}}"
      protocol="{{Protocol}}"
      units="{{Units}}"
      conversion-function="{{ConversionFuntion}}"
      consolidation-function="{{ConsolidationFunction}}"

      displayType="{{Displaytype}}"
      displayColor="{{Displaycolour}}"
    >
      <parameter name="{{parameter}}" collector="SNMP" oid="{{oid}}" type="{{Type}}"
indexed="{{Indexed}}"/>
      <value parameter="{{parameter}}"/>
 </metric>
```

In short, each row in the excel sheet will get created as a metric in xml and the column names other than the default ones already used in template.xml can be used as place holders to create different types of metrics

SNMP discovery tree

UAA supports SNMP nodal discovery by using the logic in sdk-snmp-discoverytree profile. This one profile will be used to add logic to discovery all the RAs in the UAA.

This profile is named as sdk-snmp-discovery-tree and this should be created in the profiles folder of project folder.

The following xml has to be created once in the profiles folder (first time setup) and can be used to update the discovery rules for all the RAs as required.

sdk-snmp-discoverytree.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE module PUBLIC "profile.dtd" "profile.dtd">

<module id="sdk-snmp-discoverytree">
 <meta>
  <vendor name=""/>
  <protocol name="SNMP"/>
  <product name=""/>
  <version name=""/>
  <profile-version name="7.0.0"/>
  <sdk-id value=""/>
  <capabilities/>
 </meta>
 <dependencies>
  <file path="snmp/discoverytree.xml" version=" "/>
 </dependencies>
</module>
```

NOTE

The above file is similar to profile xml and no changes are required to this file

The dependency file snmp/discoverytree.xml has to be updated with the logic/rule to discover any RA

This sdk-snmp-discoverytree can be built as a regular RA and should be deployed as any other RA

Updating the discovery.xml

The discoverytree xml is the collection of all the rules to discover an RA in UAA. The software loops through all the rules present in the xml in top-down approach and assigns an RA which ever rules is satisfied first.

discoverytree

This is the root tag of the xml which will contain all the rulelists required to discover RAs in UAA.

Child tags

- rulelist

Example:

```
<discoverytree>
</discoverytree>
```

rulelist

This tag contains the list of rules to discover the RA.

Child tags

- rule

Attributes

Attribute	Required	Description
deviceid	Yes	RA name – This RA will be assigned to the node when the rules under this tag are satisfied
comment	Yes	Any comment

Example:

```
<discoverytree>

  <!-- TODO Set comment and regexp pattern to match sysobjectid for RA -->
  <rulelist deviceid="sdk-<RA Name>" comment="Sample Comment">

</rulelist>

  <rulelist deviceid="generic-snmp" comment="Generic SNMP Device"/>
</discoverytree>
```

NOTE The rulelist generic-snmp is not to be removed and it should always be present as a last tag in the discoverytree.xml

rule

child tags

- property

Attributes

Attribute	Required	Description
object	Yes	centina.common.snmp.SnmpRuleOIDRegExp

Example:

```
<rule object="centina.common.snmp.SnmpRuleOIDRegExp">
</rule>
```

property

Attributes

Attribute	Required	Description
name	Yes	Possible values • **oid** – OID to poll the device • **regex** – Value to match or to identify the device to assign RA
value	Yes	Contains the value of OID or regex to poll/match

Example:

```
<property name="oid" value="1.3.6.1.2.1.1.2.0"/> <!-- device's sysOId -->
<property name="regexp" value="1\.3\.6\.1\.4\.1\.6527\.6\.([2|3|4|5])"/> <!—Matching sysOID pattern to assign the RA-->
```

In the above example, UAA poll the oid present in the value tag of the property with name=”oid” and matches with the pattern specified in value attribute of property with name=”regex” if the pattern matches then he RA name give in the deviceid attribute is assigned to the node

Complete Example

```
<?xml version="1.0" encoding="UTF-8"?>

<discoverytree>


   <!-- TODO Set comment and regexp pattern to match sysobjectid for RA -->
   <rulelist deviceid="sdk-RAName" comment="Sample Comment">
      <rule object="centina.common.snmp.SnmpRuleOIDRegExp">
         <property name="oid" value="1.3.6.1.2.1.1.2.0"/>
         <property name="regexp"
value="1\.3\.6\.1\.4\.1\.6527\.6\.([2|3|4|5])"/>
      </rule>
   </rulelist>

   <rulelist deviceid="generic-snmp" comment="Generic SNMP Device"/>
</discoverytree>
```

Contacting Blue Planet

Blue Planet Division Headquarters	7035 Ridge Road Hanover, MD 21076 +1 800-921-1144
Blue Planet Support	https://www.blueplanet.com/support
Sales and General Information	https://www.blueplanet.com/contact
Training	https://www.blueplanet.com/learning

For additional information, please visit https://www.blueplanet.com.

LEGAL NOTICES

Security

Ciena® cannot be responsible for unauthorized use of equipment and will not make allowance or credit for unauthorized use or access.