SNMP in ACI

Overview, Configuration, Troubleshooting, and Caveats\Issues

Created by Tomas de Leon (ACI Solutions Delivery Team)

Version 4, updated January 17, 2017

Cisco Technote for Public Use

Table of Contents

Table of Contents (cont.)

ACI SNMP Overview

SNMP is Simple Network Management Protocol which is UDP based network protocol. The SNMP protocol governs the network management and monitoring of your network devices. *Cisco ACI provides SNMPv1, v2c, and v3 support, including Management Information Bases (MIBs) and notifications (traps). The SNMP standard allows any third-party applications that support the different MIBs to manage and monitor the ACI leaf & spine switches and APIC controllers.*

ACI SNMP Overview

- SNMP Basic components are:
  - **Managed Device**: the hardware device to be monitored
  - **Agent** (SNMP software running on the Managed Device)
  - **Network Management System**: the monitoring system which has SNMP Client to communicate with the Agent running on the Managed Device
- SNMP is widely supported across several Cisco platforms.
  - **For more information on SNMP, please refer to the Cisco presentation "INTRODUCTION TO SNMP AND MIB SESSION NMS-1N02" http://www.cisco.com/networkers/nw04/presos/docs/NMS-1N02.pdf**

ACI SNMP Overview (cont.)

- SNMP support in ACI is as follows:
  - SNMP read queries (Get, Next, Bulk, Walk) are supported by leaf and spine switches and by APIC.
  - SNMP write commands (Set) are **NOT** supported by leaf and spine switches or by APIC.
  - SNMP traps (v1, v2c, and v3) are supported by leaf and spine switches and by APIC.
  - SNMPv3 is supported by leaf and spine switches and by APIC.
  - SNMP is supported for IPV4 only. SNMP over IPV6 will be supported in Brazos-Maintenance Release.

ACI SNMP Overview (cont.)

- For more information about using SNMP, see the Cisco ACI MIB Quick Reference Guide. **[1]**

Note: The SNMP policy is applied & run ***independently*** on the leaf & spine switches and to APIC controllers. Since each ACI devices is it's own SNMP entity, ***Multiple APICs in an APIC Cluster must be monitored separately*** for SNMP MIBs. Each APIC provides MIB Objects local to it. Similarly, ***each switch must be queried independently*** to provide the monitoring information. However, the SNMP policy source is created as a monitoring policy for the entire ACI fabric. SNMP support for the APIC controllers was added in ACI version 1.2(xx) or later.

SNMP Support on the APIC

- MIBs supported on APIC

MIB	Supported Tables
Entity MIB	PhysicalTable
Cisco Entity Ext MIB	PhysicalProcessorTable, LEDTable
Cisco Entity FRU Control MIB	PowerSupplyGroupTable, PowerStatusTable, FanTrayStatusTable, PhysicalTable
Cisco Entity Censor MIB	SensorValueTable, SensorThresholdTable
Cisco Process MIB	CPUTotalTable, ProcessTable, ProcessExtRevTable,ThreadTable

SNMP Support on the APIC (cont.)

- SNMP Traps Supported on APIC:
  - cefcFRUInserted, cefcFRURemoved
  - cefcFanTrayStatusChange, cefcModuleStatusChange
  - entSensorThresholdNotification
  - cefcPowerStatusChange
  - cpmCPURisingThreshold, cpmCPUFallingThreshold

SNMP Support on the APIC (cont.)

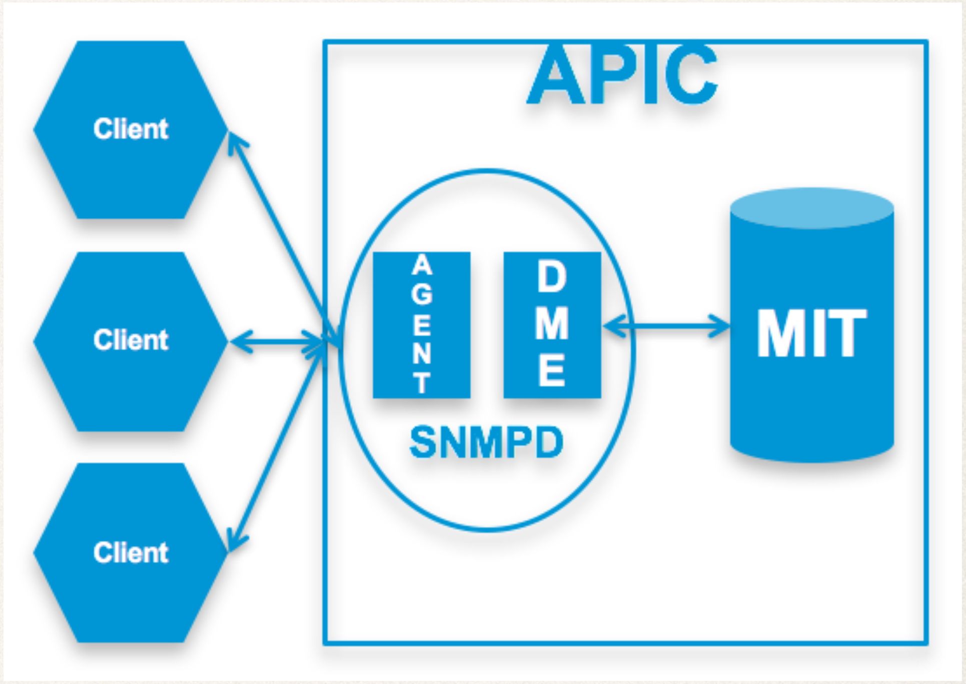
- SNMP on APIC and Policy Model
  - The APIC is build on a Policy Model. The policies framework defines Configuration Objects and Operational Objects (also known as **Managed Objects or Mos**).
  - All entities in the ACI fabric are represented as instances of Managed Objects (Mos). The collection of MOs is represented as Management Information Tree (MIT).
  - Each MO has a set of properties and the changes in properties of the MO can trigger "**events**" and "**faults**".
  - **The SNMP on APIC defines the MIB Objects to ACI MOs and provides a translation of MO information to the SNMP Objects.**
  - **SNMP Traps** are generated based on the "**events**" or "**faults**" on the MOs.

SNMP Support on the APIC (cont.)

SNMPD Architecture

The SNMPD process on APIC has two components:

- **Agent:** The SNMP Agent is open-source net-snmp agent (version: 5.7.6). The SNMP agent handles SNMP sessions from the snmp clients. It handles the SNMP protocol processing.
- **DME:** The SNMP DME handles the MIT interface to read the Managed Objects (MOs) and translate the information into SNMP Object Format.



SNMP Support on the APIC (cont.)

- Management Contracts required for SNMP
  - SNMP on APIC using OOB management EPG **requires** an explicit "**Out-Of-Band Contract**" on the APIC for enabling the SNMP port (**UDP:161**). *Note: In earlier versions of ACI firmware, certain ports were always open and a contract was not needed for SNMP support on the Leaf and Spine nodes.*
  - SNMP on APIC using INB management EPG **requires** an explicit "**In-Band Contract**" on the APIC for enabling the SNMP port (**UDP:161**).
  - The SNMP packets will be dropped by the APIC unless the contract is created. ***This is different from enabling/disabling the SNMP protocol.***

About this Technote on SNMP in ACI

The following document will use examples from using a "SNMP" utility or CLI commands to gather information about the Cisco ACI fabric system. The "SNMP" utility will also receive SNMP traps sent by the individual leaf & spine switches and APIC controllers. Not all SNMP Traps indicate problems with your system. Some messages are purely informational, while others may help diagnose problems with communications lines, internal hardware, or the system software. This document will not cover configuring 3rd Party SNMP monitoring utilities. *Just make sure the SNMP Utility has the ACI nodes IP addresses configured as SNMP Agents, correct UDP port for SNMP Traps, & community string used in the ACI Fabric.*

In this technote, I will show examples of configuring SNMP **utilizing the APIC Admin GUI**. In ACI version 1.2(xx) or later, there are two modes for the APIC Admin GUI. For this document, **I will use examples from the "ADVANCED" GUI Mode**. In addition to the APIC Admin GUI, SNMP can be configured using the APIC iNXOS CLI Mode and by using a REST API client. **[2] [3]**

About this Technote on SNMP in ACI

Note: At the time of writing this document, configuring SNMP using the APIC iNXOS CLI Mode was incomplete. Due to this incompleteness, parts of the SNMP configuration will still need to be configured via the GUI or the Rest API. In regards to the REST API, you can open the API inspector console from the APIC GUI. The API inspector displays the Rest API POST requests used for the tasks performed. The "Post" Requests in the API inspector can be used for sending requests to APIC controllers.

For Rest API examples listed in this document, there is an assumption made that you have a REST CLIENT (like POSTMAN) installed on your workstation. This is a sample tool that can be used for executing REST API requests to an APIC Controller.

ACI SNMP Configuration

In this technote, I will show examples of configuring SNMP **utilizing the APIC Admin GUI**. In ACI version 1.2(xx) or later, there are two modes for the APIC Admin GUI. For this document, **I will use examples from the "ADVANCED" GUI Mode**. In addition to the APIC Admin GUI, SNMP can be configured using the APIC iNXOS CLI Mode and by using a REST API client. **[2] [3]**

TASK 1:
Configure a SNMP Policy for the ACI Fabric

- For this configuration task, we will use the existing "**default**" policies. *(Note: perform the same tasks for configuring custom policies)*

- Configuration Steps:
  1. Access the APIC Admin GUI.
  2. Select **FABRIC -> FABRIC POLICIES**.
  3. In the policies navigation panel on the left, select and expand the **POD POLICIES -> POLICIES**.
  4. Expand SNMP and Select the "**default**" SNMP Policy.
  5. In the SNMP **Policy-default** configuration panel, perform the following actions:
     - Add a description (SNMP Policy for the RTP2 Fabric)
     - Select Admin State (Enabled)
     - Enter Contact (Sir deadbeef)
     - Enter Location (Cisco Systems, North Carolina)

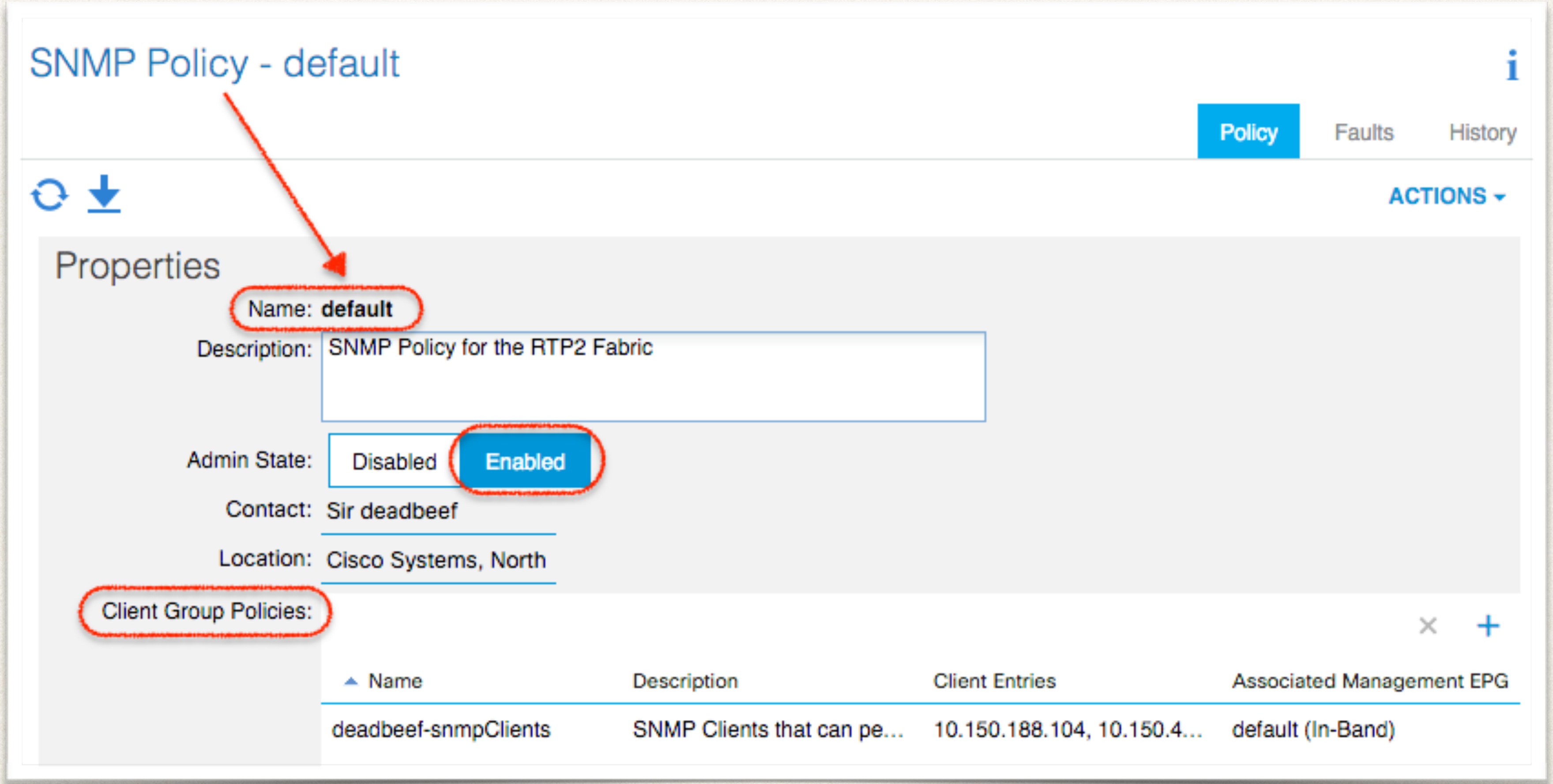
TASK 1: (cont.)
Configure a SNMP Policy for the ACI Fabric

- Configuration Steps:
  5. In the SNMP **Policy-default** configuration panel, perform the following actions: (cont.)
     - **Click** on the " + " sign to **CREATE SNMP CLIENT GROUP POLICIES.** *Note: The Client Group Policies is like an ACL for SNMP Clients that can perform SNMPGET and SNMPWALK requests.* In the Client Group Profile dialog box, perform the following actions:
       - **Enter Name** (deadbeef-snmpClients)
       - **Add a description** (SNMP Clients that can perform SNMPGET and SNMPWALK requests)
       - **Select Associated Management EPG** (default (In-Band))
       - Click on the " + " sign to **ADD CLIENT ENTRIES**. In the Client Entries Table, perform the following actions:
         - **Enter Name** (deadbeef-osx1)
         - **Enter IP Address** (10.150.188.104)
         - **Click UPDATE**

           Note: Repeat ADD CLIENT ENTRIES tasks to add additional SNMP CLIENT ENTRIES.
         - **Click SUBMIT** to complete "*Create SNMP Client Group Profile*" tasks.

TASK 1: (cont.)
Configure a SNMP Policy for the ACI Fabric

- Sample Screenshot:



Name: default

Description: SNMP Policy for the RTP2 Fabric

Admin State: Disabled [✓ Enabled]

Contact: Sir deadbeef

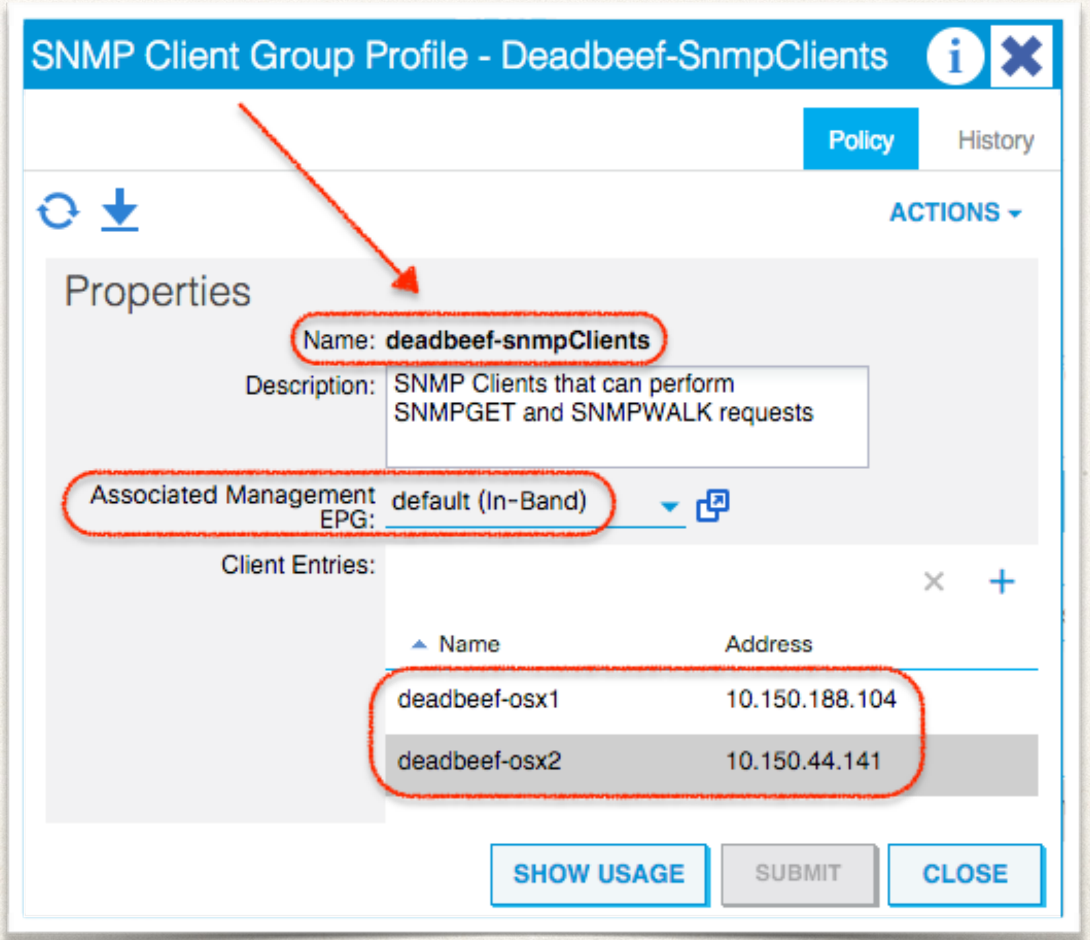
Location: Cisco Systems, North

Client Group Policies:

Name	Description	Client Entries	Associated Management EPG
deadbeef-snmpClients	SNMP Clients that can pe...	10.150.188.104, 10.150.4...	default (In-Band)

TASK 1: (cont.)
Configure a SNMP Policy for the ACI Fabric

- Sample Screenshot:



Name: deadbeef-snmpClients

Description: SNMP Clients that can perform SNMPGET and SNMPWALK requests

Associated Management EPG: default (In-Band)

Client Entries:

Name	Address
deadbeef-osx1	10.150.188.104
deadbeef-osx2	10.150.44.141

TASK 1: (cont.)
Configure a SNMP Policy for the ACI Fabric

- CAVEAT for CLIENT GROUPS when using SNMP Version 3:

  With the introduction of SNMP support on the APIC, contracts and client groups are needed to allow for SNMP GETs and Walks to the APIC. The SNMP client groups configure the iptables for SNMP and are needed to work for SNMP **v2c**.

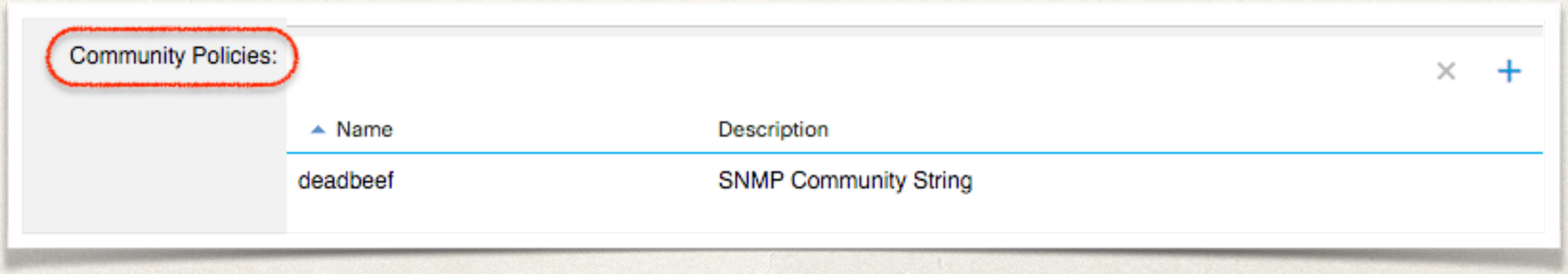
  For the APIC, there is a caveat with the use of "Client Groups" in regards to SNMP Gets & Walks when using SNMP version 3.   ***At this time, "Client Groups" are NOT used for the APIC when using SNMP version 3 so all SNMP Gets & Walks to the APIC(s) will be allowed.*** In regards to the Leaf & Spine switches, the "Client Groups" behavior is the same when using SNMP versions v2c & v3.

  A future release of ACI firmware will address this Caveat with Client Groups & SNMP version 3 on the APIC controllers.

TASK 1: (cont.) Configure a SNMP Policy for the ACI Fabric

- Configuration Steps:
  5. In the SNMP **Policy-default** configuration panel, perform the following actions: (cont.)
     - **Click** on the " **+** " sign to **ADD COMMUNITY POLICIES**. In the Community Policies Table, perform the following actions:
       - **Enter Name** (deadbeef)
       - **Add a description** (SNMP Community String)
       - **Click UPDATE**
     - **Click SUBMIT** to complete "*Configure SNMP Policy*" tasks.

- Sample Screenshot:

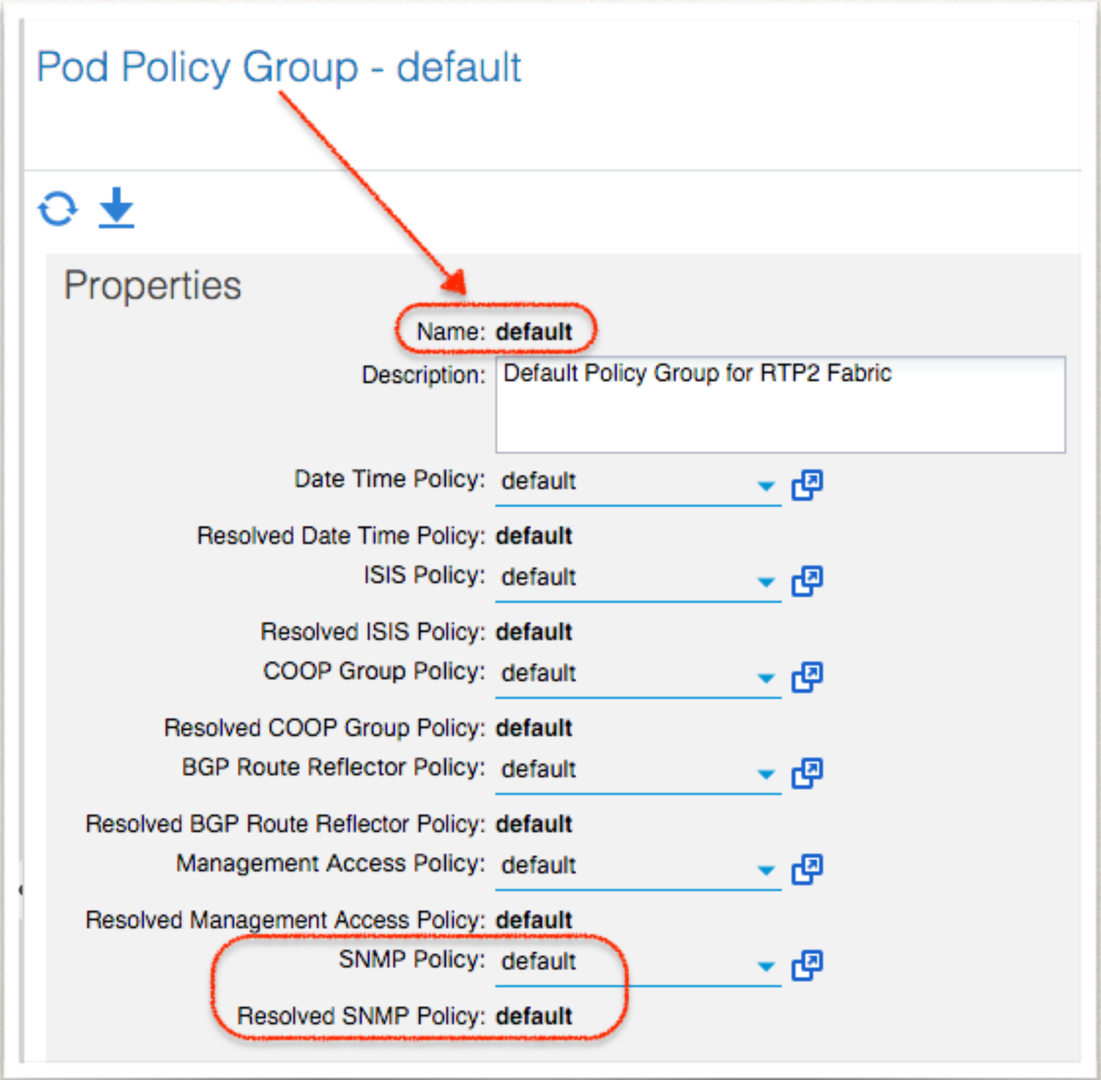

Community Policies:

Name	Description
deadbeef	SNMP Community String

TASK 1: (cont.) Configure a SNMP Policy for the ACI Fabric

- <u>Configuration Steps:</u>
  Note: For this example we are using SNMP v2c, If you were using SNMP v3, you would configure the SNMP User information for this policy also.

  6. Expand Policy Groups and Select the "**default**" Policy Group.
     - Make sure the "**default**" **SNMP Policy** (*or Custom SNMP Policy*) is **selected** and **Resolved.**


Pod Policy Group - default

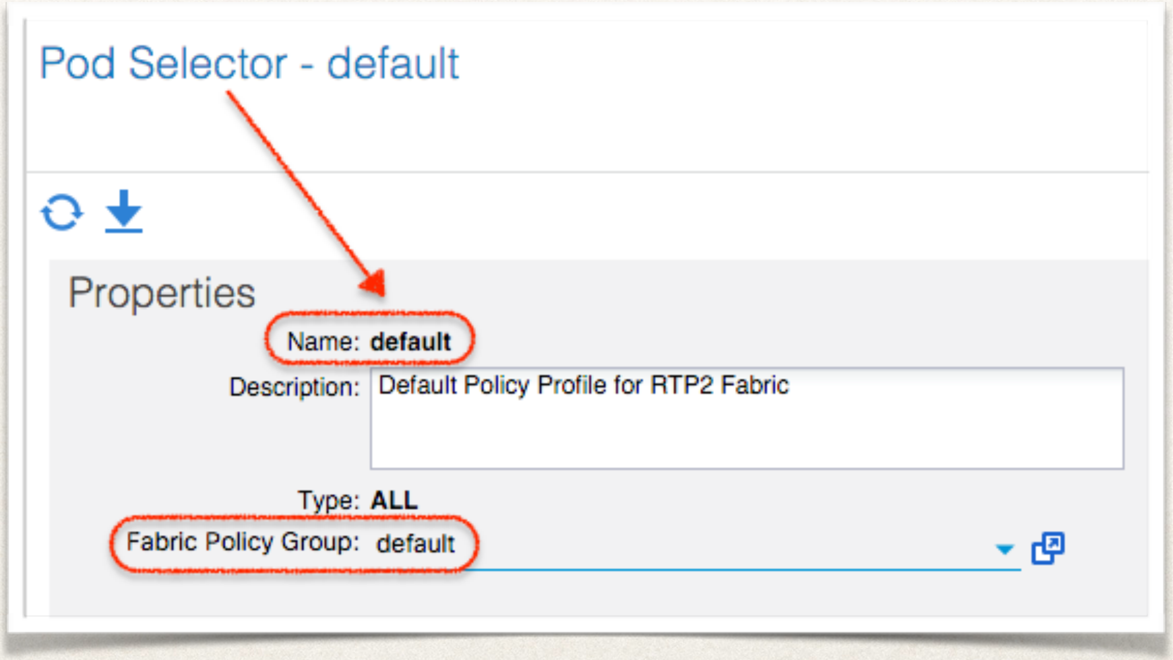
Properties

Name:	**default**
Description:	Default Policy Group for RTP2 Fabric
Date Time Policy:	default
Resolved Date Time Policy:	**default**
ISIS Policy:	default
Resolved ISIS Policy:	**default**
COOP Group Policy:	default
Resolved COOP Group Policy:	**default**
BGP Route Reflector Policy:	default
Resolved BGP Route Reflector Policy:	**default**
Management Access Policy:	default
Resolved Management Access Policy:	**default**
SNMP Policy:	default
Resolved SNMP Policy:	**default**

TASK 1: (cont.) Configure a SNMP Policy for the ACI Fabric

- Configuration Steps:
  7. Expand **Profiles**, expand **Pod Profile default**, and Select the "**default**" Pod Profile.
     - Make sure the "**default**" Fabric Policy Group is selected.

- Sample Screenshot:



Pod Selector - default

Properties

Name: default

Description: Default Policy Profile for RTP2 Fabric

Type: ALL

Fabric Policy Group: default

TASK 1: (cont.) Configure a SNMP Policy for the ACI Fabric

- Example of POSTs from the API Inspector:

method: POST
url: http://172.18.242.111/api/node/mo/uni/fabric/snmppol-default.json

payload{"snmpPol":{"attributes":{"dn":"uni/fabric/snmppol-default","descr":"SNMP Policy for the RTP2 Fabric","adminSt":"enabled","contact":"Sir deadbeef","loc":"Cisco Systems, North Carolina"},"children":[]}}

method: POST
url: http://172.18.242.111/api/node/mo/uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients.json

payload{"snmpClientGrpP":{"attributes":{"dn":"uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients","name":"deadbeef-snmpClients","descr":"SNMP Clients that can perform SNMPGET and SNMPWALK requests","rn":"clgrp-deadbeef-snmpClients","status":"created"},"children":[{"snmpClientP":{"attributes":{"dn":"uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients/client-[10.150.188.104]","name":"deadbeef-osx1","addr":"10.150.188.104","rn":"client-[10.150.188.104]","status":"created"},"children":[]}},{"snmpClientP":{"attributes":{"dn":"uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients/client-[10.150.44.141]","name":"deadbeef-osx2","addr":"10.150.44.141","rn":"client-[10.150.44.141]","status":"created"},"children":[]}},{"snmpRsEpg":{"attributes":{"tDn":"uni/tn-mgmt/mgmtp-default/inb-default","status":"created"},"children":[]}}]}}

TASK 1: (cont.)
Configure a SNMP Policy for the ACI Fabric

- <u>Example of POSTs from the API Inspector: (cont.)</u>

  method: POST
  url: http://172.18.242.111/api/node/mo/uni/fabric/snmppol-default/community-deadbeef.json

  payload{"snmpCommunityP":{"attributes":{"dn":"uni/fabric/snmppol-default/community-deadbeef","name":"deadbeef","status":"created","descr":"SNMP Community String","rn":"community-deadbeef"},"children":[]}}

TASK 2: Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

In "Brazos" & previous ACI releases, the leaf\spine node switches did **NOT** require a OOB or INB contract to allow SNMP Get Requests using UDP DestPort 161: for SNMP. *These requests cannot be blocked through contracts.* Creating a SNMP ClientGroup in the SNMP policy with a list of Client-IP Addresses restricts SNMP access to only the configured Client-IP Addresses. If no Client-IP address is configured, SNMP packets are allowed from anywhere.

In "Brazos", Cisco added SNMP support for the APIC(s). *The behavior for default allowed ports for the APIC it is "Different"*. Unlike the Switches, **a CONTRACT is needed for the APIC to allow SNMP.** This is "NEW" with brazos. In your OOB Contract defined for your External Management Network Instance Profile. Once you add Ports 161 & 162 to the filter of the OOB Contract, your SNMP Gets should work as expected.

Also in addition to contracts being needed, **Node Management Address(s) in the Tenant mgmt need to be configured for the APIC(s).** Verify that the APIC Node management address(s) are configured also.

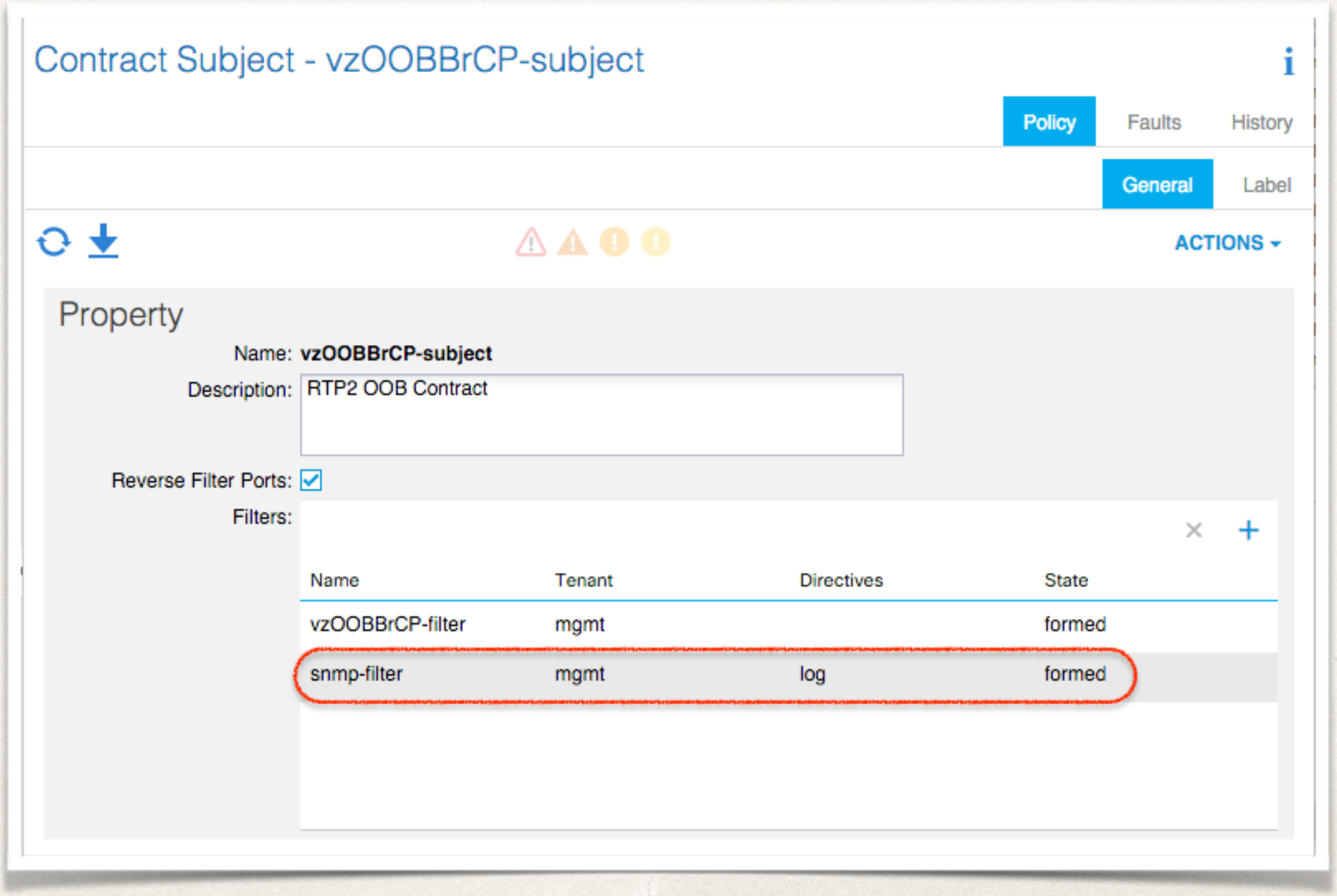
TASK 2:
Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

- If Out-Of-Band or In-Band Contract(s) already exist, verify that UDP Port 161 is configured for SNMP Requests. If SNMP ports are not in filters, add UDP Port 161 to existing filters & contracts. Create the Required Contracts & filters with the appropriate SNMP Ports.

- Configuration Steps:
  1. Access the APIC Admin GUI.
  2. Select **TENANTS -> ALL TENANTS**.
  3. In the tenants navigation panel on the left, double-click on the **MGMT Tenant**.
  4. In the Navigation pane, expand **Security Policies**:
     - Expand **Out-Of-Band Contracts**
       - Expand existing OOB Contract
       - Select **OOB Subject**
       - In the OOB Subject Panel, double-click on **OOB filter**(s)
       - Review filter(s) to ensure **UDP Port 161** is configured.
     - If the Fabric is also using **In-Band management** also, **verify the INB contract filter** is also configured for **UDP Port 161**.

TASK 2: Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

- Verify Out-Of-Band Contract:

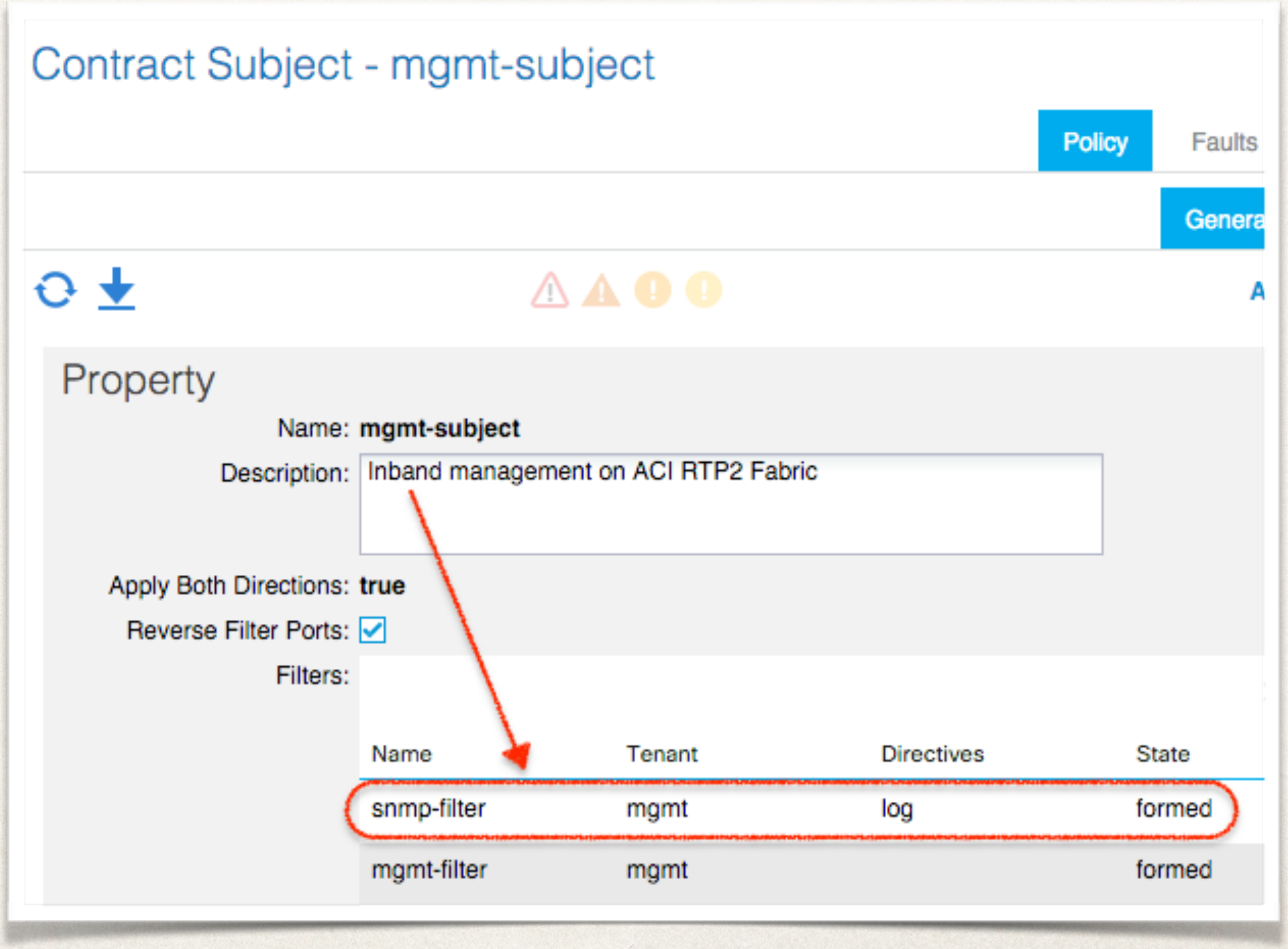


Name: **vzOOBBrCP-subject**
Description: RTP2 OOB Contract
Reverse Filter Ports: ✓
Filters:

Name	Tenant	Directives	State
vzOOBBrCP-filter	mgmt		formed
snmp-filter	mgmt	log	formed

TASK 2:
Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

- <u>Verify In-Band Contract:</u>

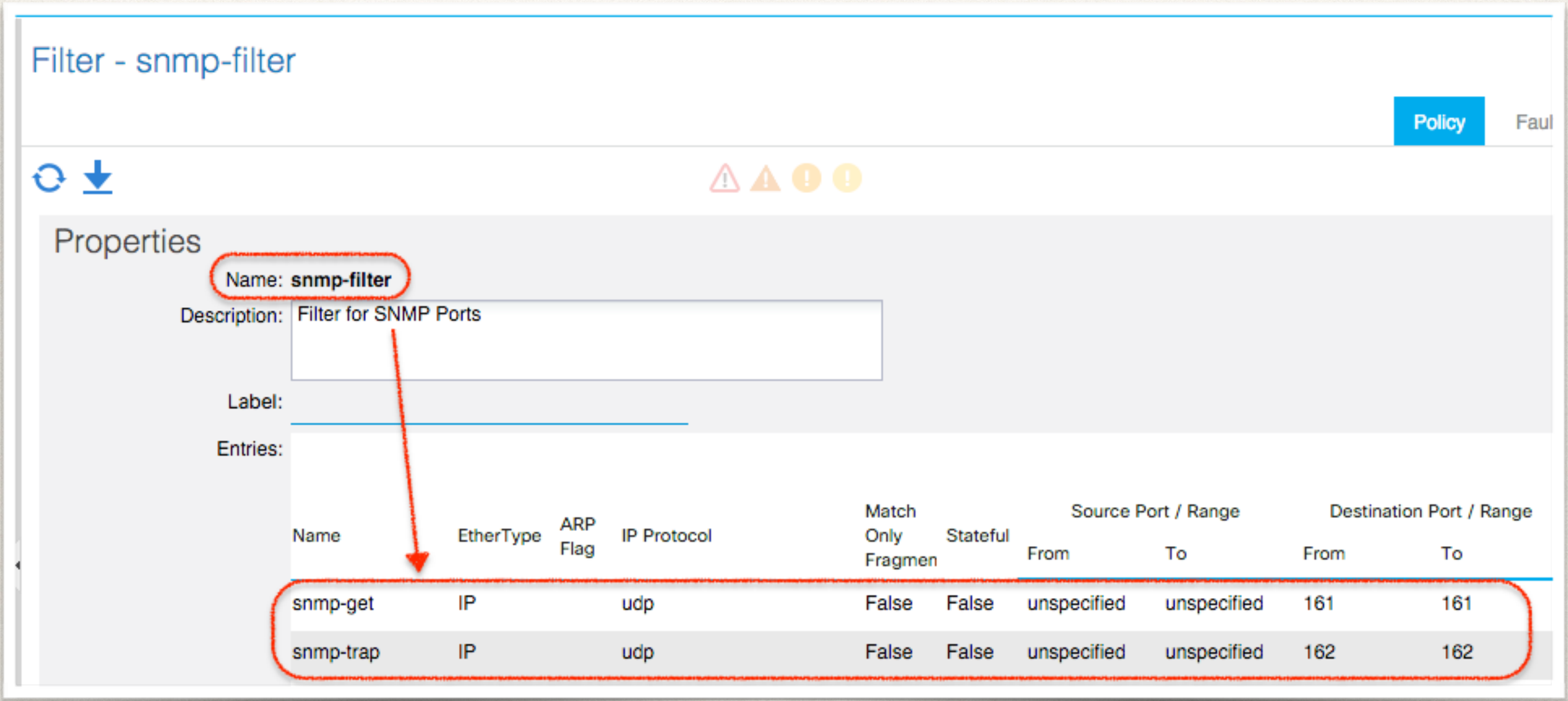


Name: **mgmt-subject**
Description: Inband management on ACI RTP2 Fabric
Apply Both Directions: **true**
Reverse Filter Ports: ☑
Filters:

Name	Tenant	Directives	State
snmp-filter	mgmt	log	formed
mgmt-filter	mgmt		formed

TASK 2:
Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

- Verify Contract Filter:



Filter - snmp-filter

Policy Faul

Properties

Name: **snmp-filter**

Description: Filter for SNMP Ports

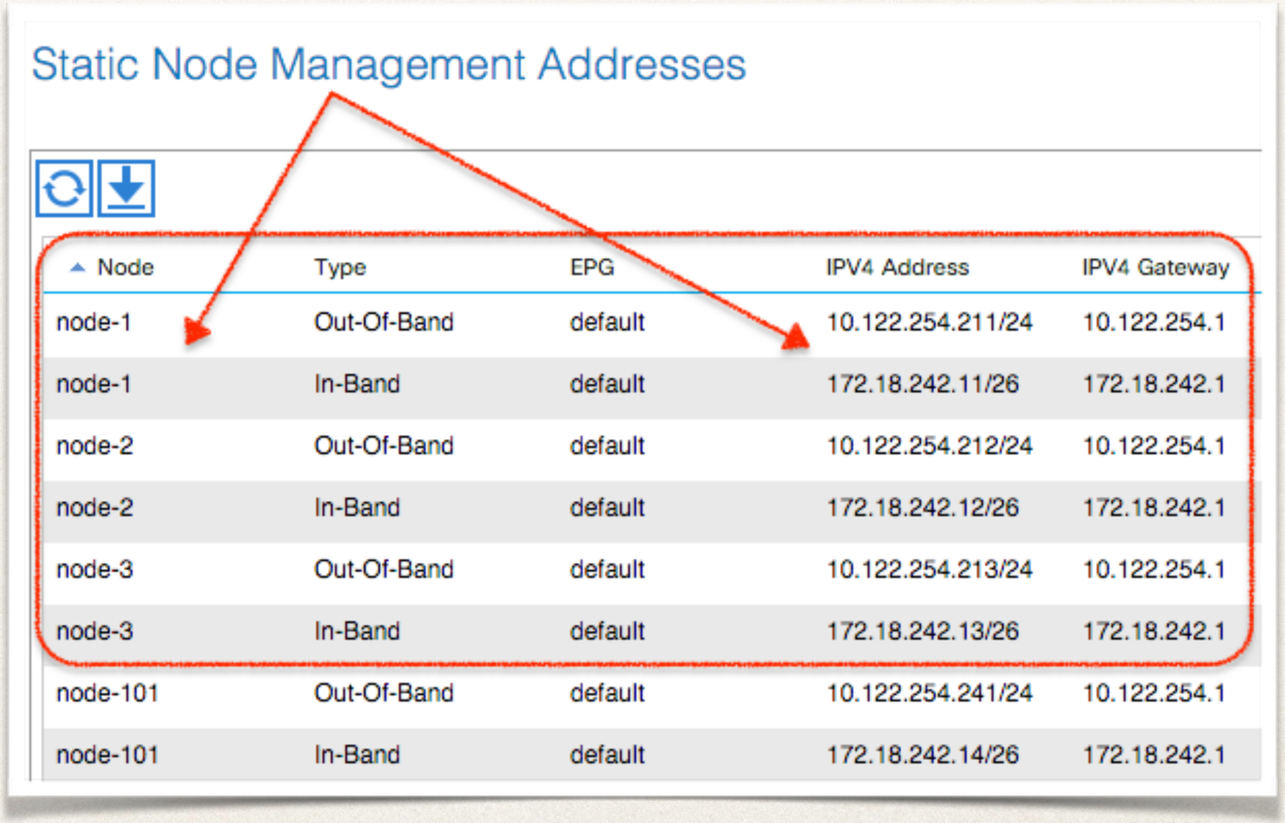
Label:

Entries:

Name	EtherType	ARP Flag	IP Protocol	Match Only Fragmen	Stateful	Source Port / Range		Destination Port / Range	
						From	To	From	To
snmp-get	IP		udp	False	False	unspecified	unspecified	161	161
snmp-trap	IP		udp	False	False	unspecified	unspecified	162	162

TASK 2:
Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

- Verify Node Management Addresses Configured:



Static Node Management Addresses

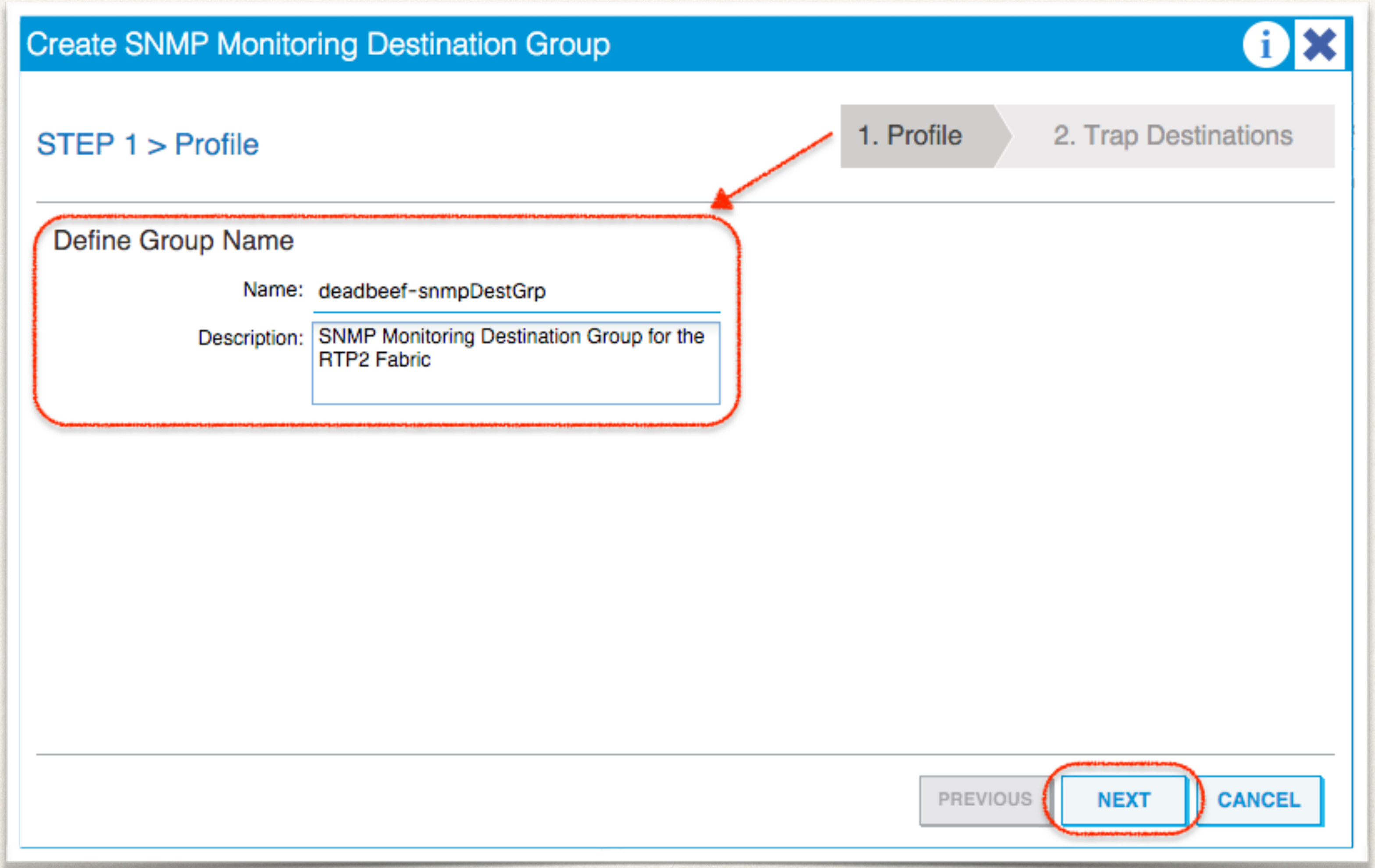
Node	Type	EPG	IPV4 Address	IPV4 Gateway
node-1	Out-Of-Band	default	10.122.254.211/24	10.122.254.1
node-1	In-Band	default	172.18.242.11/26	172.18.242.1
node-2	Out-Of-Band	default	10.122.254.212/24	10.122.254.1
node-2	In-Band	default	172.18.242.12/26	172.18.242.1
node-3	Out-Of-Band	default	10.122.254.213/24	10.122.254.1
node-3	In-Band	default	172.18.242.13/26	172.18.242.1
node-101	Out-Of-Band	default	10.122.254.241/24	10.122.254.1
node-101	In-Band	default	172.18.242.14/26	172.18.242.1

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- The first step is to create an External Data Collector source group for SNMP.
- Configuration Steps:
  1. Access the APIC Admin GUI.
  2. Select **ADMIN -> EXTERNAL DATA COLLECTORS.**
  3. In the External Data Collectors navigation panel on the left, select and expand the **MONITORING DESTINATIONS**.
     - Select **SNMP** and **Right-Click** to "**Create SNMP Monitoring Destination Group**".
     - In the "Create SNMP Monitoring Destination Group" configuration panel, perform the following actions:
       - **STEP 1 > Define a Group Name**
         - Enter Group **Name** (deadbeef-snmpDestGrp)
         - Add a **description** (SNMP Monitoring Destination Group for the RTP2 Fabric)
         - Click **NEXT**
       - **STEP 2 > Trap Destinations**
         - Click on the " + " sign to **CREATE SNMP TRAP DESTINATION**. In the "Create SNMP Trap Destination" configuration panel, perform the following actions:
           - Add HOSTNAME or IP (10.117.67.20)
           - Use DEFAULT UDP Port 162 or Define your desired UDP PORT for SNMP Traps
           - Select SNMP Version to be used for this SNMP Trap Destination (v2c)
           - Add COMMUNITY NAME (deadbeef)
           - Select MANAGEMENT EPG (default (In-Band))
           - Click OK
       - **Repeat STEP 2** to create additional SNMP Trap Destinations.
       - Click **FINISH**

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Sample Screenshots:

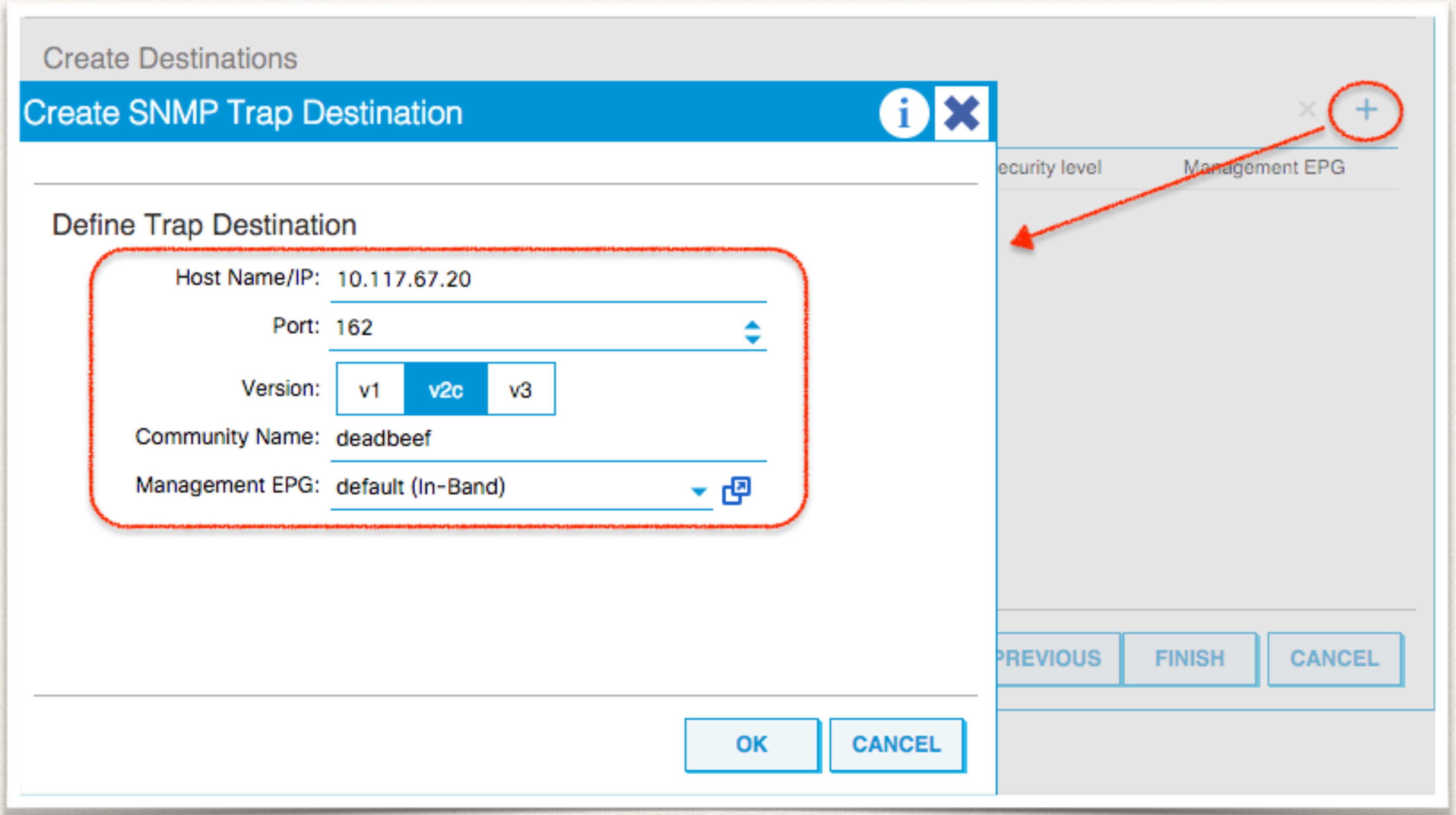


Name: deadbeef-snmpDestGrp

Description: SNMP Monitoring Destination Group for the RTP2 Fabric

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Sample Screenshots:

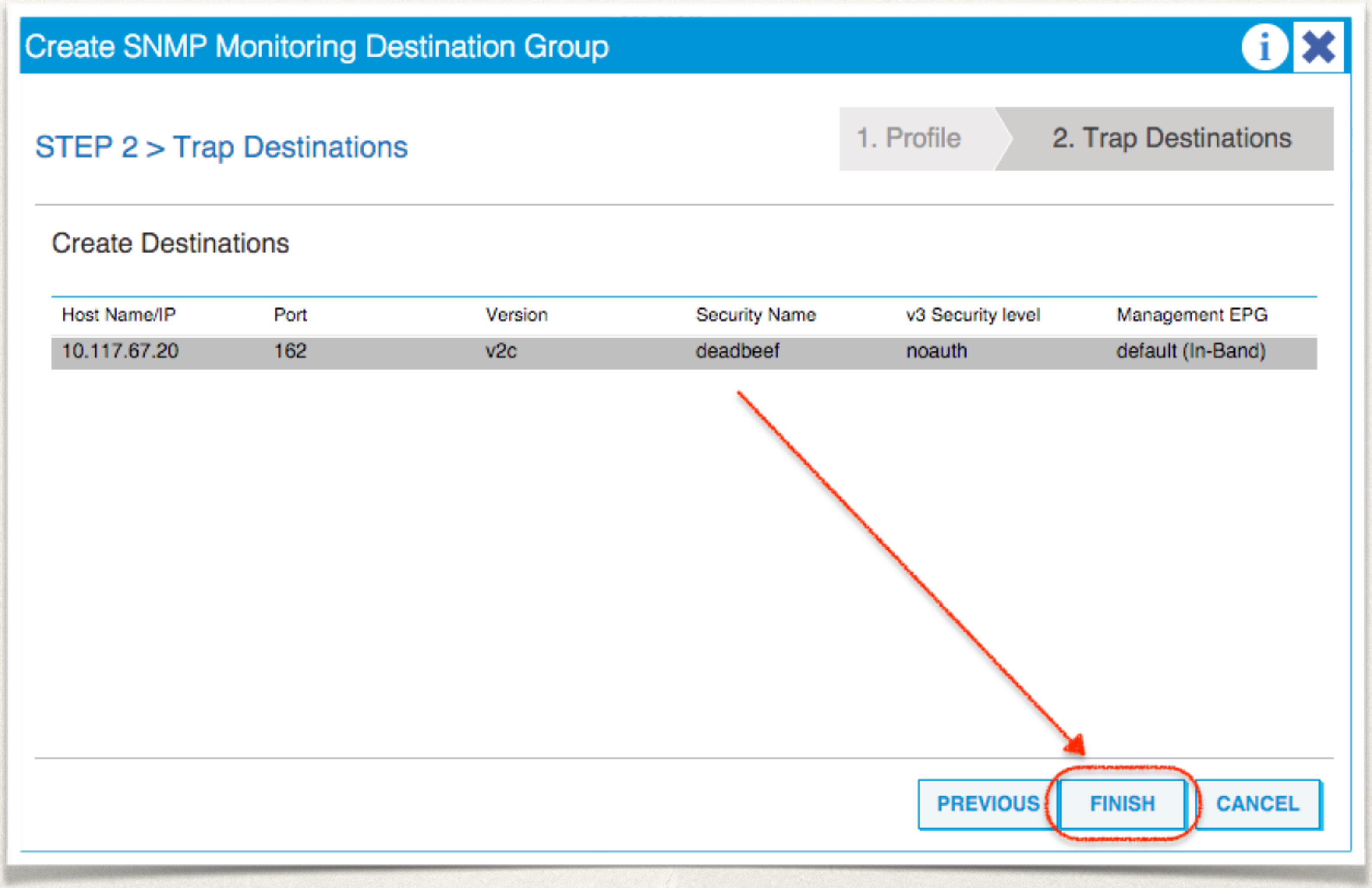


Define Trap Destination

Host Name/IP: 10.117.67.20
Port: 162
Version: v1 [✓ v2c] v3
Community Name: deadbeef
Management EPG: default (In-Band)

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Sample Screenshots:



Create SNMP Monitoring Destination Group

STEP 2 > Trap Destinations

Create Destinations

Host Name/IP	Port	Version	Security Name	v3 Security level	Management EPG
10.117.67.20	162	v2c	deadbeef	noauth	default (In-Band)

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Example of POST from the API Inspector:

  method: POST
  url: http://172.18.242.111/api/node/mo/uni/fabric/snmpgroup-deadbeef-snmpDestGrp.json

  payload{"snmpGroup":{"attributes":{"dn":"uni/fabric/snmpgroup-deadbeef-snmpDestGrp","name":"deadbeef-snmpDestGrp","descr":"SNMP Monitoring Destination Group for the RTP2 Fabric","rn":"snmpgroup-deadbeef-snmpDestGrp","status":"created"},"children":[{"snmpTrapDest":{"attributes":{"dn":"uni/fabric/snmpgroup-deadbeef-snmpDestGrp/trapdest-10.117.67.20-port-162","host":"10.117.67.20","secName":"deadbeef","rn":"trapdest-10.117.67.20-port-162","status":"created"},"children":[{"fileRsARemoteHostToEpg":{"attributes":{"tDn":"uni/tn-mgmt/mgmtp-default/inb-default","status":"created"},"children":[]}}]}}]}}

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- After you have created the ACI Fabric's SNMP Monitoring Destination Group with SNMP Trap Destinations, you will need to configure Fabric "Monitoring Sources" to use this SNMP Monitoring Destination Group. There are 3 main Monitoring Sources that can be configured. I give examples of configuring each of the monitoring sources. In later ACI firmware releases, you can create a monitoring source in the Tenant scope.
- "Which monitoring sources do I need to configure?" "Do I need to configure all 3 monitoring sources?" are common questions that we get from customers. Take a look at the online documentation at: http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/aci-fundamentals/b_ACI-Fundamentals/b_ACI-Fundamentals_chapter_01110.html

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

Configuring Monitoring Policies

Administrators can create monitoring policies with the following four broad scopes:

- **Fabric Wide:** includes both fabric and access objects
- **Access (also known as infrastructure):** access ports, FEX, VM controllers, and so on
- **Fabric:** fabric ports, cards, chassis, fans, and so on
- **Tenant:** EPGs , application profiles, services, and so on

The APIC includes the following four classes of default monitoring policies:

monCommonPol (uni/fabric/moncommon): *applies to both fabric and access infrastructure hierarchies*

monFabricPol (uni/fabric/monfab-default): applies to **fabric** hierarchies

monInfraPol (uni/infra/monifra-default): applies to the **access infrastructure** hierarchy

monEPGPol (uni/tn-common/monepg-default): applies to **tenant** hierarchies

In each of the four classes of monitoring policies, *the default policy can be overridden by a specific policy*. For example, a monitoring policy applied to the deadbeef tenant (tn-deadbeef) would override the default one for the deadbeef tenant while other tenants would still be monitored by the default policy.

TASK 3: Configure the ACI Fabric to send SNMP TRAPS

- After you have created the ACI Fabric's SNMP Monitoring Destination Group with SNMP Trap Destinations, you will need to configure Fabric "Monitoring Sources" to use this SNMP Monitoring Destination Group.

- Configuration Steps:
  1. Access the APIC Admin GUI.
  2. Select **FABRIC -> FABRIC POLICIES.**
  3. In the Policies navigation panel on the left, select and expand the **MONITORING POLICIES**.
     - Expand **DEFAULT** and **Select** "**CALLHOME/SNMP/SYSLOG**".
     - In the "Callhome/SNMP/Syslog" configuration panel, **Select SNMP** as the "Source Type" and Click on the " + " sign to **CREATE SNMP SOURCE**.
     - In the "Create SNMP Source" configuration panel, perform the following actions:
       - **Enter Source Name** (deadbeef-snmpSrc)
       - **Select** the **SNMP Monitoring Destination Group** that was created in a previous task (deadbeef-snmpDestGrp)
       - **Click Submit**

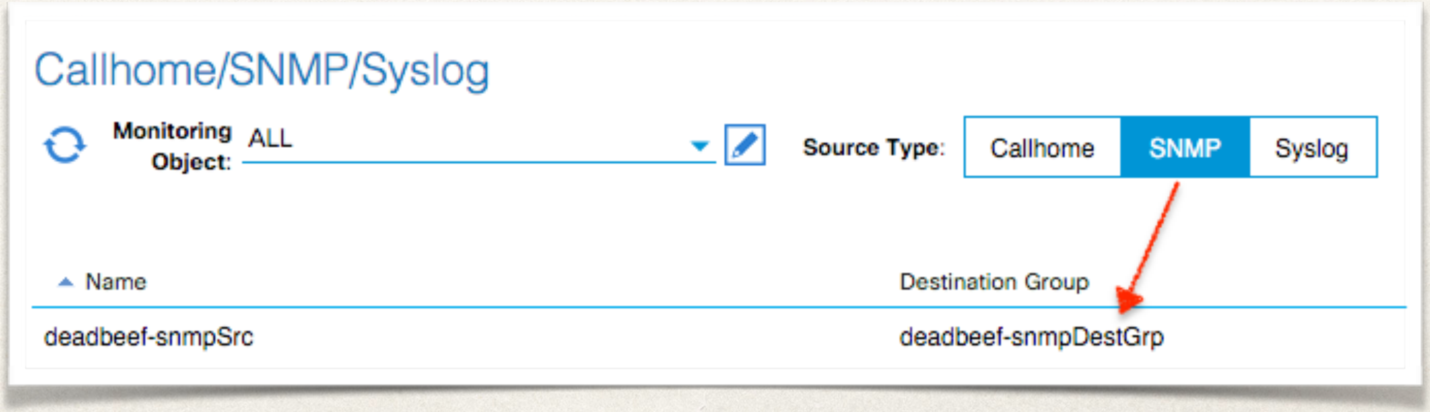
TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Example of POST from the API Inspector:

Fabric Policies - default (Callhome/SNMP/Syslog)

method: POST
url: http://172.18.242.111/api/node/mo/uni/fabric/monfab-default/snmpsrc-deadbeef-snmpSrc.json

payload{"snmpSrc":{"attributes":{"dn":"uni/fabric/monfab-default/snmpsrc-deadbeef-snmpSrc","incl":"audits,events,faults","name":"deadbeef-snmpSrc","rn":"snmpsrc-deadbeef-snmpSrc","status":"created"},"children":[{"snmpRsDestGroup":{"attributes":{"tDn":"uni/fabric/snmpgroup-deadbeef-snmpDestGrp","status":"created"},"children":[]}}]}}



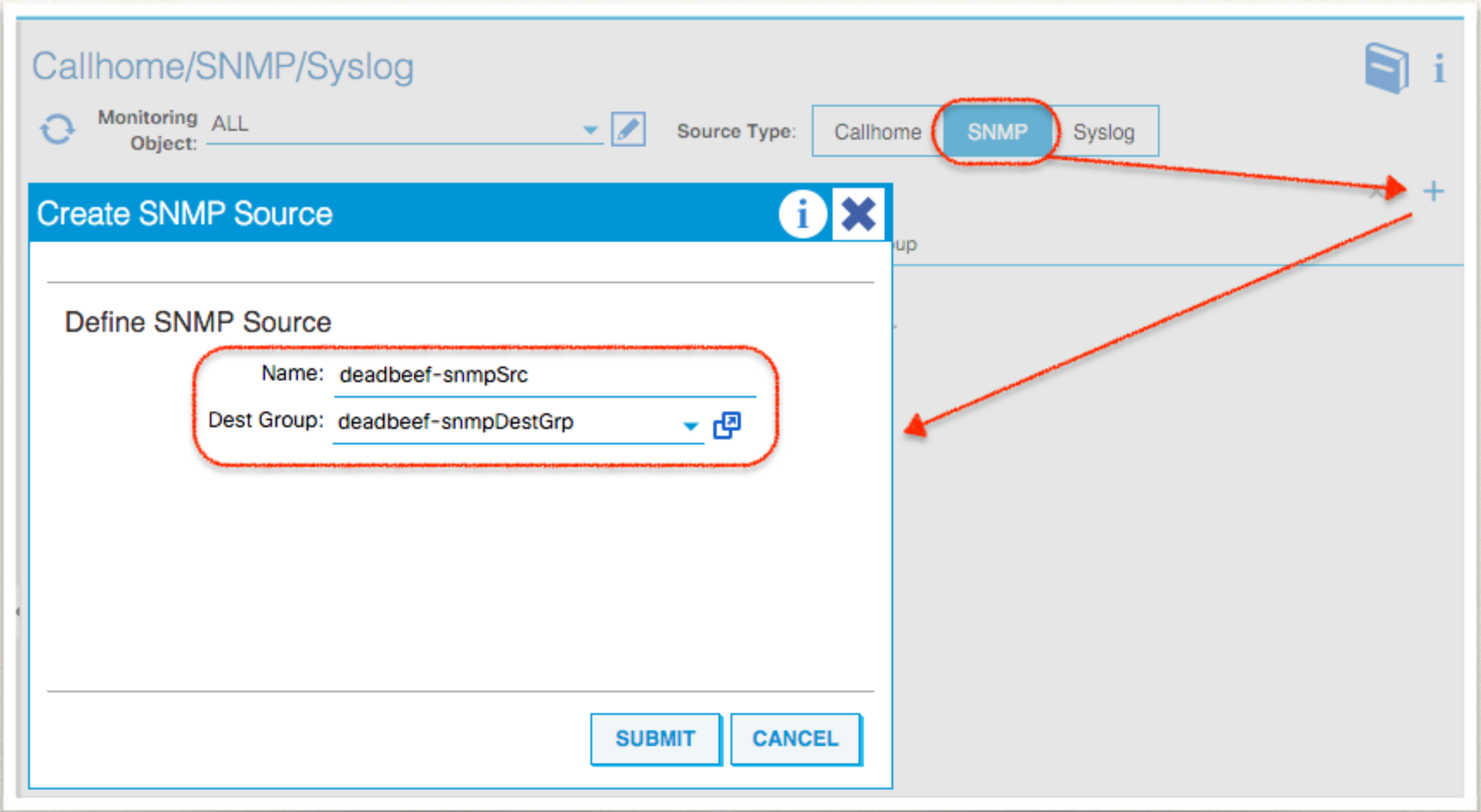
Monitoring Object: ALL

Source Type: SNMP

Name	Destination Group
deadbeef-snmpSrc	deadbeef-snmpDestGrp

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Sample Screenshots: (Fabric Policies - default (Callhome/SNMP/Syslog))



Define SNMP Source

Name: deadbeef-snmpSrc

Dest Group: deadbeef-snmpDestGrp

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- After you have created the ACI Fabric's SNMP Source in the Fabric Policies "Monitoring Sources" for Fabric Policies - DEFAULT, configure the SNMP Source in Fabric Policies - COMMON POLICY.

- Configuration Steps:
  1. Access the APIC Admin GUI.
  2. Select **FABRIC -> FABRIC POLICIES.**
  3. In the Policies navigation panel on the left, select and expand the **MONITORING POLICIES**.
     - **Select COMMON POLICY** and **Right-Click** and **Select "Create SNMP Source"**.
     - In the "Create SNMP Source" configuration panel, perform the following actions:
       - **Enter Source Name** (deadbeef-snmpSrc)
       - **Select** the **SNMP Monitoring Destination Group** that was created in a previous task (deadbeef-snmpDestGrp)
       - **Click Submit**

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Example of POST from the API Inspector:

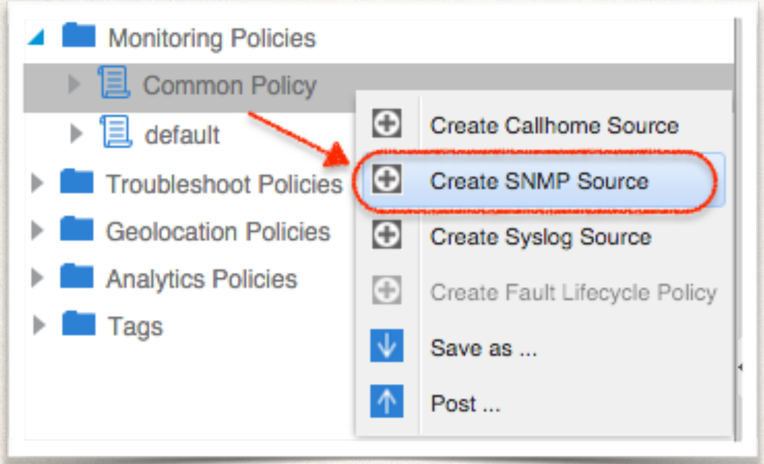
Fabric Policies - common (Callhome/SNMP/Syslog)

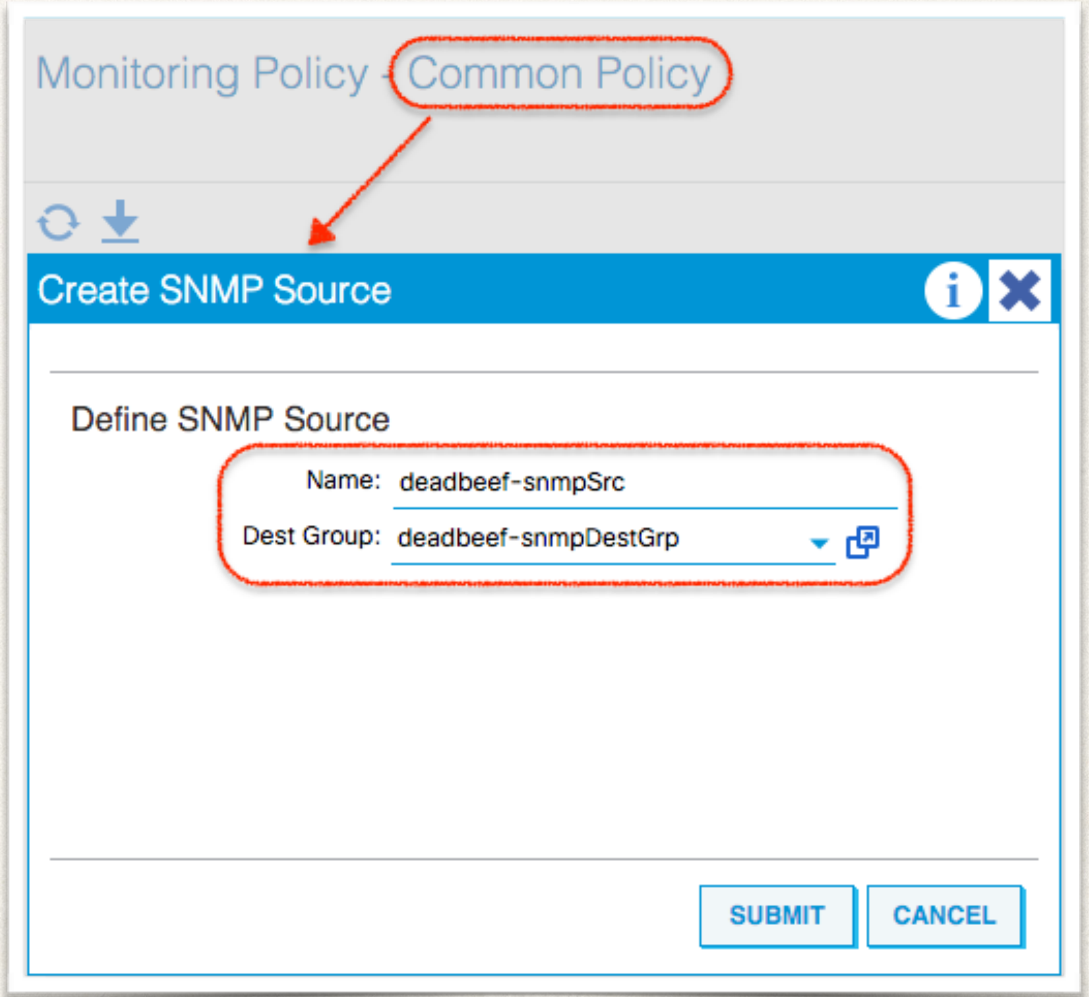
method: POST
url: http://172.18.242.111/api/node/mo/uni/fabric/moncommon/snmpsrc-deadbeef-snmpSrc.json

payload{"snmpSrc":{"attributes":{"dn":"uni/fabric/moncommon/snmpsrc-deadbeef-snmpSrc","incl":"audits,events,faults","name":"deadbeef-snmpSrc","rn":"snmpsrc-deadbeef-snmpSrc","status":"created"},"children":[{"snmpRsDestGroup":{"attributes":{"tDn":"uni/fabric/snmpgroup-deadbeef-snmpDestGrp","status":"created"},"children":[]}}]}}

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Sample Screenshots: (Fabric Policies - common (Callhome/SNMP/Syslog))





Define SNMP Source

Name: deadbeef-snmpSrc

Dest Group: deadbeef-snmpDestGrp

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- After you have created the ACI Fabric's SNMP Source in the Fabric Policies "Monitoring Sources" for Fabric Policies - DEFAULT & COMMON, configure the SNMP Source in Access Policies - DEFAULT.

- Configuration Steps:
  1. Access the APIC Admin GUI.
  2. Select **FABRIC -> ACCESS POLICIES.**
  3. In the Policies navigation panel on the left, select and expand the **MONITORING POLICIES**.
     - **Select DEFAULT** and **Right-Click** and **Select "Create SNMP Source"**.
     - In the "Create SNMP Source" configuration panel, perform the following actions:
       - **Enter Source Name** (deadbeef-snmpSrc)
       - **Select** the **SNMP Monitoring Destination Group** that was created in a previous task (deadbeef-snmpDestGrp)
       - **Click Submit**

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Example of POST from the API Inspector:

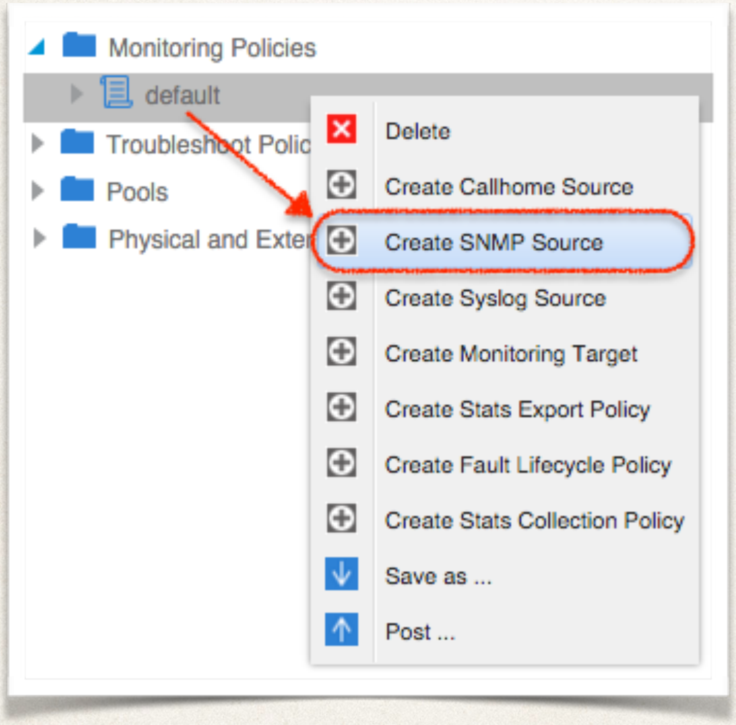
Access Policies - default (Callhome/SNMP/Syslog)

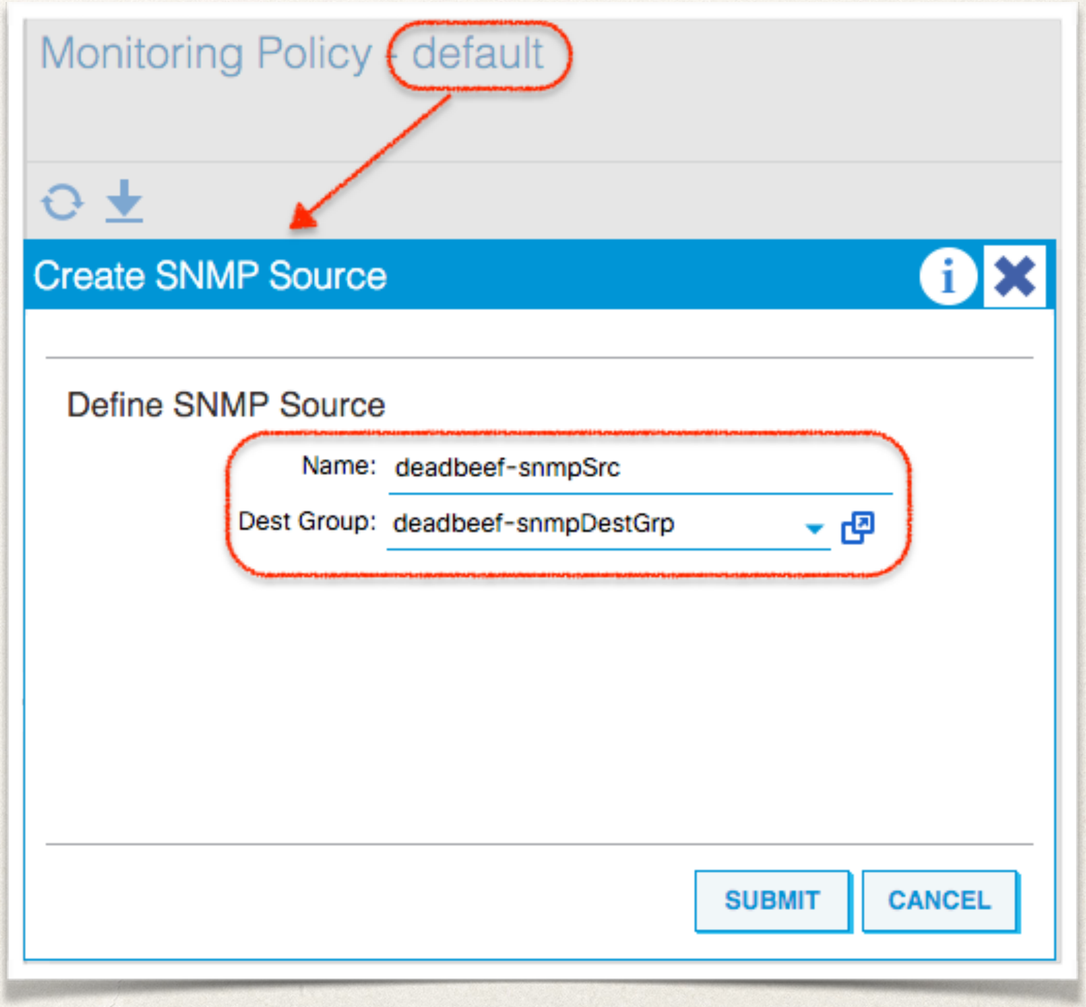
method: POST
url: http://172.18.242.111/api/node/mo/uni/infra/moninfra-default/snmpsrc-deadbeef-snmpSrc.json

payload{"snmpSrc":{"attributes":{"dn":"uni/infra/moninfra-default/snmpsrc-deadbeef-snmpSrc","incl":"audits,events,faults","name":"deadbeef-snmpSrc","rn":"snmpsrc-deadbeef-snmpSrc","status":"created"},"children":[{"snmpRsDestGroup":{"attributes":{"tDn":"uni/fabric/snmpgroup-deadbeef-snmpDestGrp","status":"created"},"children":[]}}]}}

TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

- Sample Screenshots: (Access Policies - default (Callhome/SNMP/Syslog))





Define SNMP Source

Name: deadbeef-snmpSrc

Dest Group: deadbeef-snmpDestGrp

Troubleshooting ACI SNMP Configuration

This section will provide an overview on generic troubleshooting SNMP policies in the ACI Fabric. Once SNMP policies are configured for SNMP GET\WALK Requests and SNMP TRAPS, verify that the configuration is pushed to the LEAF\SPINE\APIC nodes. Use the available CLI commands to verify configuration is enabled and applied. If needed, use of external tools and apps may be necessary.

Verify ACI SNMP Configuration (cont.)

“show commands”

- **After completing the configuration of SNMP policies, verify configuration on Leaf\Spine\APIC Nodes.** *Note: SNMP support for the APIC controllers was added in ACI version 1.2(xx) or later so the APIC related information only pertains to fabrics running ACI version 1.2(xx) or later.*

1. SSH to a Fabric APIC. Use the “*attach node-name*” command to connect to the desired Leaf\Spine Nodes.
2. Use the following ACI CLI **SHOW** commands to verify the configuration on the Leaf\Spine\APIC nodes:

APIC CLI COMMANDS	LEAF\SPINE CLI COMMANDS
show snmp show snmp summary show snmp clientgroups show snmp community show snmp hosts	show snmp show snmp summary show snmp community show snmp host

Verify APIC SNMP Configuration

“show commands”

- **Use the output from the "*show snmp summary*" command to retrieve summary information on the APIC SNMP configuration.**

```
rtp2-apic1# show snmp summary

Active Policy: default, Admin State: enabled

Local SNMP engineID: [Hex] 0x8000000980e2b692088976c75600000000

----------------------------------------
Community                Description
----------------------------------------
deadbeef                 SNMP Community String

------------------------------------------------------------
User                     Authentication        Privacy
------------------------------------------------------------

------------------------------------------------------------
Client-Group             Mgmt-Epg                    Clients
------------------------------------------------------------
deadbeef-snmpClients default (In-Band)           10.150.44.141,10.117.67.20,10.150.188.104

------------------------------------------------------------
Host                     Port   Version   Level       SecName
------------------------------------------------------------
10.117.67.20             162    v2c       noauth      deadbeef
```

Verify LEAF\SPINE SNMP Configuration

“show commands”

- **Use the output from the "*show snmp summary*" command to retrieve summary information on the Leaf/Spine SNMP configuration.**

```
rtp2-leaf1# show snmp summary

Admin State : enabled, running (pid:7808)

Local SNMP engineID: [Hex] 80000009037C69F6105BF9
                     [Dec] 128:000:000:009:003:124:105:246:016:091:249

----------------------------------------------------------------------
Community                         Context                   Status
----------------------------------------------------------------------
deadbeef                                                    ok

----------------------------------------------------------------------
Client                                  VRF                 Status
----------------------------------------------------------------------
10.150.188.104                          mgmt:inb            ok
10.150.44.141                           mgmt:inb            ok
10.117.67.20                            mgmt:inb            ok

-------------------------------------------------------------------------------
Host                                    Port Ver  Level   SecName        VRF
-------------------------------------------------------------------------------
10.117.67.20                            162  v2c  noauth  deadbeef       mgmt:inb
```

Verify ACI SNMP Configuration (cont.)

“show commands”

- **What to look for in the output from the "*show snmp summary*" command ?**

✓ The “**Admin State**” should be “**Enabled**”.

- LEAF = Admin State : **enabled, running (pid:7808)**
- APIC = Admin State: **enabled**

✓ The “**SNMP Community**” configured.

- LEAF = **deadbeef**
- APIC = **deadbeef**

✓ The “**Client Group**” Hosts & VRF allowed to send SNMP Get\Walk Requests (UDP PORT 161).

- LEAF = **10.150.188.104, 10.150.44.141, 10.117.67.20, mgmt:inb**
- APIC = **10.150.44.141, 10.117.67.20, 10.150.188.104, default (In-Band)**

✓ The **Destination Host(s), PORT# & VRF** for sending SNMP Traps (UDP PORT 162 or Custom Port).

- LEAF = **10.117.67.20, 162, mgmt:inb**
- APIC = **10.117.67.20, 162**

Verify ACI SNMP Configuration (cont.)

"moquery"

- **Managed Object(MO) Queries is another way to verify configuration of SNMP Policies. On each Leaf\Spine\APIC with SNMP configured, run "*moquery -c [object class]*"**
  ie. (snmpPol, snmpClientGrpP, snmpClientP, snmpCommunityP, snmpTrapDest, snmpSrc).

snmpPol

```
apic1# moquery -c snmpPol
Total Objects shown: 1

# snmp.Pol
name             : default
adminSt          : enabled
childAction      :
contact          : Sir deadbeef
descr            : SNMP Policy for the RTP2 Fabric
dn               : uni/fabric/snmppol-default
lcOwn            : local
loc              : Cisco Systems, North Carolina
modTs            : 2016-03-07T15:43:06.559+00:00
monPolDn         : uni/fabric/monfab-default
ownerKey         :
ownerTag         :
rn               : snmppol-default
status           :
uid              : 0
```

Note: Repeat the "moquery -c snmpPol" command on each Leaf \Spine\APIC node configured for SNMP.

Verify ACI SNMP Configuration (cont.)

"moquery"

- **Managed Object(MO) Queries is another way to verify configuration of SNMP Policies. On each Leaf\Spine\APIC with SNMP configured, run "*moquery -c [object class]*"**
  ie. (snmpPol, snmpClientGrpP, snmpClientP, snmpCommunityP, snmpTrapDest, snmpSrc).

snmpClientGrpP

```
leaf1# moquery -c snmpClientGrpP -x query-target=children
Total Objects shown: 4

# snmp.ClientP
addr         : 10.150.188.104
dn           : uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients/client-[10.150.188.104]
name         : deadbeef-osx1
rn           : client-[10.150.188.104]

# snmp.ClientP
addr         : 10.150.44.141
dn           : uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients/client-[10.150.44.141]
name         : deadbeef-osx2
rn           : client-[10.150.44.141]

# snmp.ClientP
addr         : 10.117.67.20
dn           : uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients/client-[10.117.67.20]
name         : deadbeef-osx3
rn           : client-[10.117.67.20]
```

Note: Repeat the "moquery -c snmpClientGrpP -x query-target=children" command on each Leaf\Spine\APIC node configured for SNMP.

Verify ACI SNMP Configuration (cont.)
“moquery”

- **Managed Object(MO) Queries is another way to verify configuration of SNMP Policies. On each Leaf\Spine\APIC with SNMP configured, run “*moquery -c [object class]*” ie. (snmpPol, snmpClientGrpP, snmpClientP, snmpCommunityP, snmpTrapDest, snmpSrc).**

snmpCommunityP

```
apic1# moquery -c snmpCommunityP
Total Objects shown: 1

# snmp.CommunityP
name          : deadbeef
childAction   :
descr         : SNMP Community String
dn            : uni/fabric/snmppol-default/community-deadbeef
lcOwn         : local
modTs         : 2016-03-07T15:45:50.186+00:00
rn            : community-deadbeef
status        :
uid           : 15374
```

Note: Repeat the “moquery -c snmpCommunityP” command on each Leaf \Spine\APIC node configured for SNMP.

Verify ACI SNMP Configuration (cont.)

“moquery”

- **Managed Object(MO) Queries is another way to verify configuration of SNMP Policies. On each Leaf\Spine\APIC with SNMP configured, run “*moquery -c [object class]*” ie. (snmpPol, snmpClientGrpP, snmpClientP, snmpCommunityP, snmpTrapDest, snmpSrc).**

snmpTrapDest

```
leaf1# moquery -c snmpTrapDest
Total Objects shown: 1

# snmp.TrapDest
host          : 10.117.67.20
port          : 162
childAction   :
descr         :
dn            : uni/fabric/snmpgroup-deadbeef-snmpDestGrp/trapdest-10.117.67.20-port-162
epgDn         : uni/tn-mgmt/mgmtp-default/inb-default
lcOwn         : policy
modTs         : 2016-03-08T19:27:25.464+00:00
monPolDn      : uni/fabric/monfab-default
name          :
notifT        : traps
rn            : trapdest-10.117.67.20-port-162
secName       : deadbeef
status        :
uid           : 15374
v3SecLvl      : noauth
ver           : v2c
vrfName       : mgmt:inb
```

Note: Repeat the “moquery -c snmpTrapDest -x query-target=children” command on each Leaf \Spine\APIC node configured for SNMP.

Verify ACI SNMP Configuration (cont.)

"moquery"

- **Managed Object(MO) Queries is another way to verify configuration of SNMP Policies. On each Leaf\Spine\APIC with SNMP configured, run "*moquery -c [object class]*"**
  ie. (snmpPol, snmpClientGrpP, snmpClientP, snmpCommunityP, snmpTrapDest, snmpSrc).

snmpSrc

```
apic1# moquery -c snmpSrc | egrep "snmp.Src|name|dn|incl|minSev|monPolDn"
# snmp.Src
name            : deadbeef-snmpSrc
dn              : uni/infra/moninfra-default/snmpsrc-deadbeef-snmpSrc
incl            : events,faults
minSev          : info
monPolDn        : uni/infra/moninfra-default

# snmp.Src
name            : deadbeef-snmpSrc
dn              : uni/fabric/monfab-default/snmpsrc-deadbeef-snmpSrc
incl            : events,faults
minSev          : info
monPolDn        : uni/fabric/monfab-default

# snmp.Src
name            : deadbeef-snmpSrc
dn              : uni/fabric/moncommon/snmpsrc-deadbeef-snmpSrc
incl            : events,faults
minSev          : info
monPolDn        : uni/fabric/moncommon
```

Note: Repeat the "moquery -c snmpSrc | egrep "snmp.Src|name|dn|incl|minSev|monPolDn" " command on each Leaf \Spine\APIC node configured for SNMP.

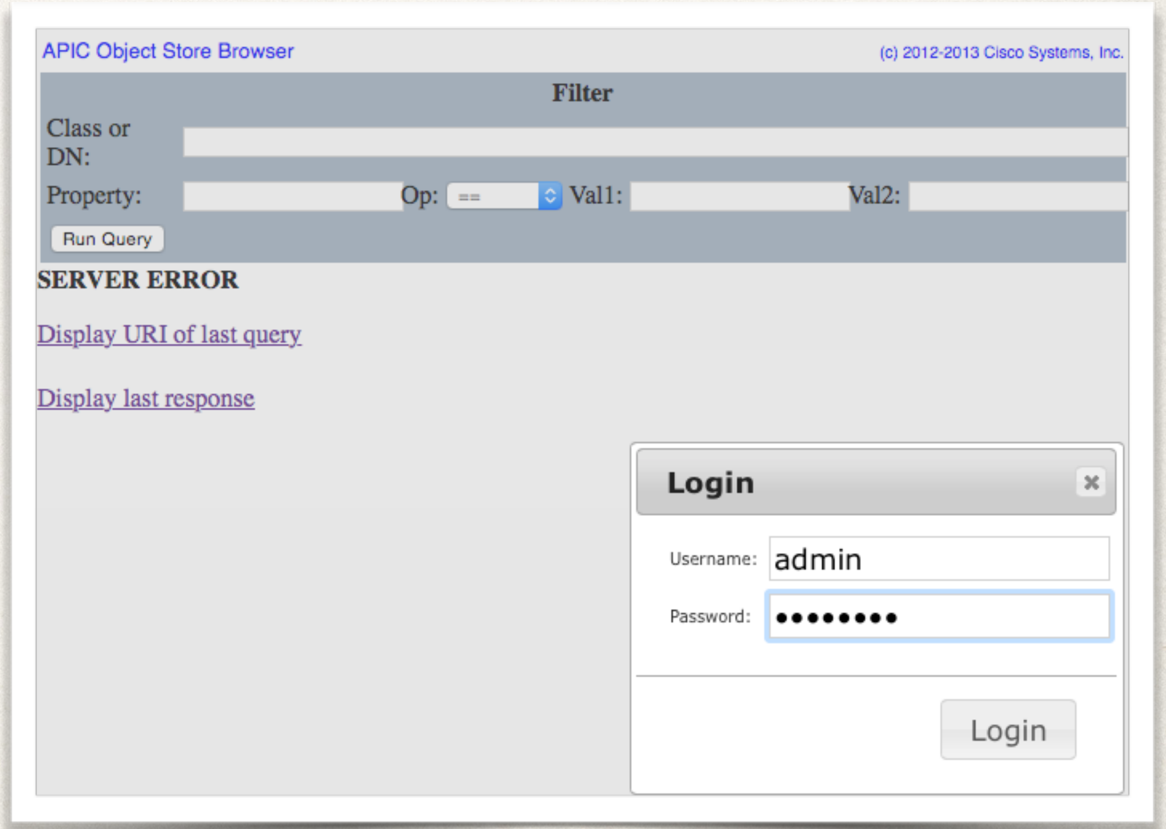
Verify ACI SNMP Configuration (cont.)
"VISORE"

- **Another tool to verify SNMP configuration is VISORE. Enclosed are some samples of the VISORE information related to the SNMP configuration.**
  (snmpPol, snmpClientGrpP, snmpClientP, snmpCommunityP, snmpTrapDest, snmpSrc)

- **To access VISORE, use a browser using the following address:**

  https://<APIC_IP_address>/visore.html

  note: use your APIC Admin Credentials to login to VISORE


APIC Object Store Browser (c) 2012-2013 Cisco Systems, Inc.

Filter

Class or DN:

Property: Op: == Val1: Val2:

Run Query

SERVER ERROR

Display URI of last query

Display last response

Login

Username: admin

Password: ••••••••

Login

Verify ACI SNMP Configuration (cont.)

“VISORE”

- **Managed Object(MO) Classes for SNMP Policy configuration in ACI**

snmpGroup - The SNMP destination group, which contains information needed to send traps or informs to a set of destinations.. SNMP is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language used for the monitoring and management of devices in a network.

snmpTrapDest - A destination to which traps and informs are sent.

snmpRtDestGroup - A target relation to SNMP destination group. This group contains information needed to send traps or informs to a set of destinations

snmpPol - The SNMP policy, which enables you to monitor client group, v3 user, and/or community SNMP policies. SNMP is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language used for the monitoring and management of devices in a network.

snmpClientGrpP - A client group, which is a group of client IP addresses that allows SNMP access to routers or switches.

Verify ACI SNMP Configuration (cont.)

"VISORE"

- **Managed Object(MO) Classes for SNMP Policy configuration in ACI (cont.)**

snmpCommunityP - The SNMP community profile, which enables access to the router or switch statistics for monitoring. SNMP is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language used for the monitoring and management of devices in a network.

snmpRtSnmpPol - A target relation to an SNMP policy that contains site information and general protocol configuration parameters. Note that this relation is an internal object.

snmpClientP - The client profile information.

snmpRsEpg - A source relation to the endpoint group VRF through which the clients can connect. The VRF is an in-band or out-of-band management endpoint.

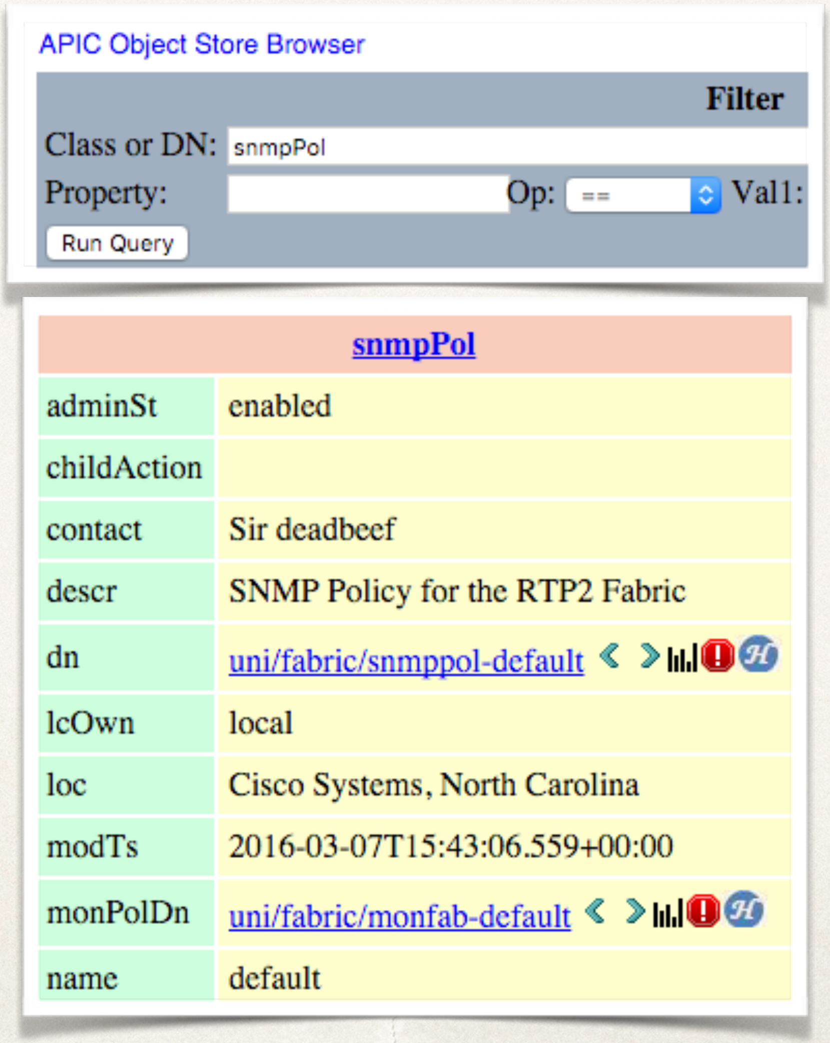
snmpSrc - The SNMP source profile, which determines the fault information, severity level, and destination for sending messages to the SNMP destination. SNMP is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language used for the monitoring and management of devices in a network .

snmpCtxP - The SNMP context profile, which enables you to specify a context to monitor with a community profile. SNMP is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language used for the monitoring and management of devices in a network.

Verify ACI SNMP Configuration (cont.)

“VISORE”

snmpPol



Class or DN: snmpPol

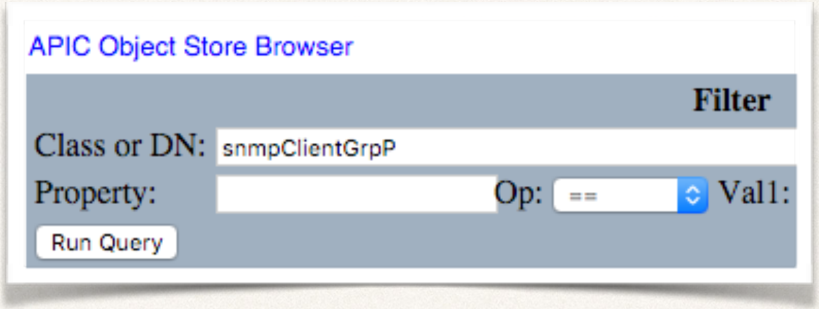
Property: Op: == Val1:

snmpPol	
adminSt	enabled
childAction	
contact	Sir deadbeef
descr	SNMP Policy for the RTP2 Fabric
dn	uni/fabric/snmppol-default
lcOwn	local
loc	Cisco Systems, North Carolina
modTs	2016-03-07T15:43:06.559+00:00
monPolDn	uni/fabric/monfab-default
name	default

Verify ACI SNMP Configuration (cont.)

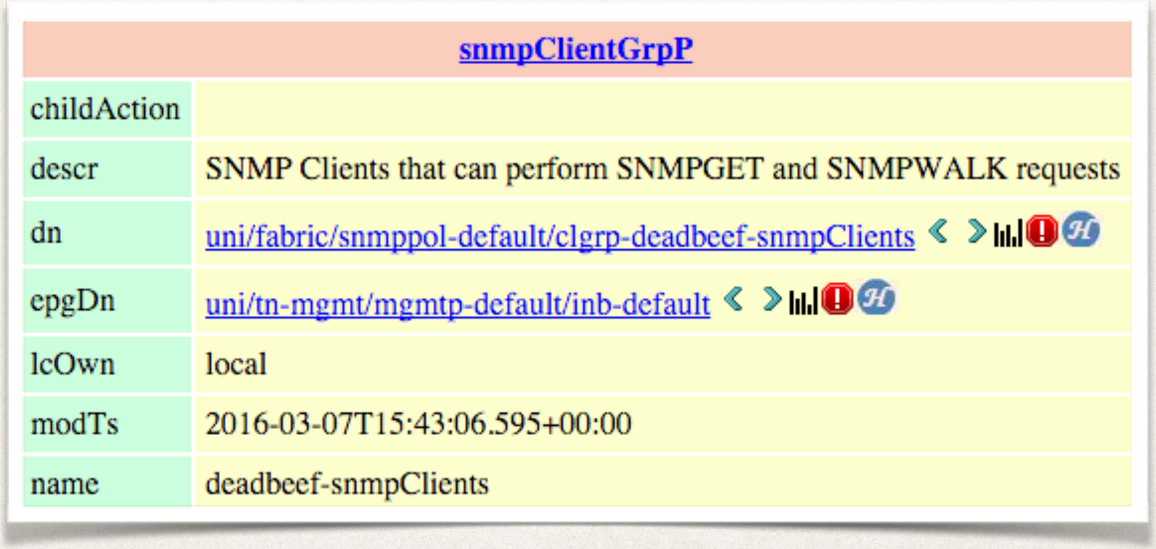
"VISORE"

snmpClientGrpP



APIC Object Store Browser

	Filter
Class or DN:	snmpClientGrpP
Property:	Op: == Val1:

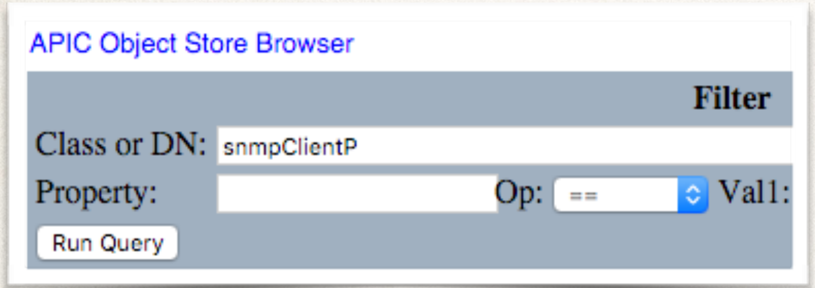


snmpClientGrpP

childAction	
descr	SNMP Clients that can perform SNMPGET and SNMPWALK requests
dn	uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients
epgDn	uni/tn-mgmt/mgmtp-default/inb-default
lcOwn	local
modTs	2016-03-07T15:43:06.595+00:00
name	deadbeef-snmpClients

Verify ACI SNMP Configuration (cont.)

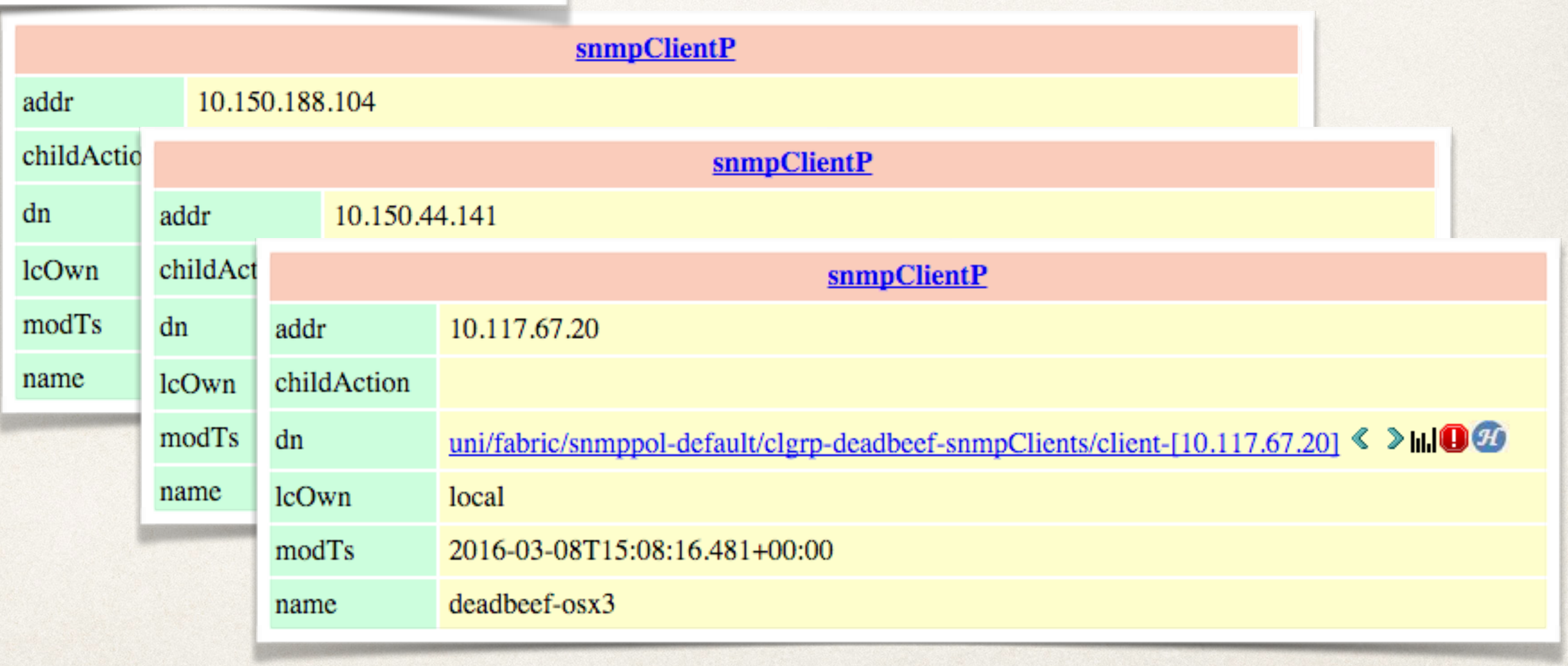
"VISORE"



APIC Object Store Browser

Filter			
Class or DN:	snmpClientP		
Property:		Op: ==	Val1:

snmpClientP



snmpClientP

addr	10.150.188.104
childActio	
dn	
lcOwn	
modTs	
name	

snmpClientP

addr	10.150.44.141
childAct	
dn	
lcOwn	
modTs	
name	

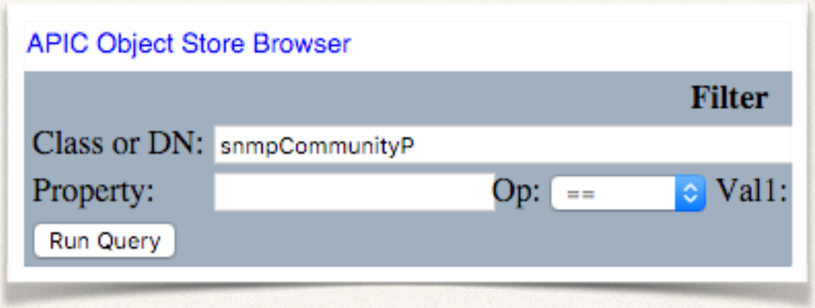
snmpClientP

addr	10.117.67.20
childAction	
dn	uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients/client-[10.117.67.20]
lcOwn	local
modTs	2016-03-08T15:08:16.481+00:00
name	deadbeef-osx3

Verify ACI SNMP Configuration (cont.)

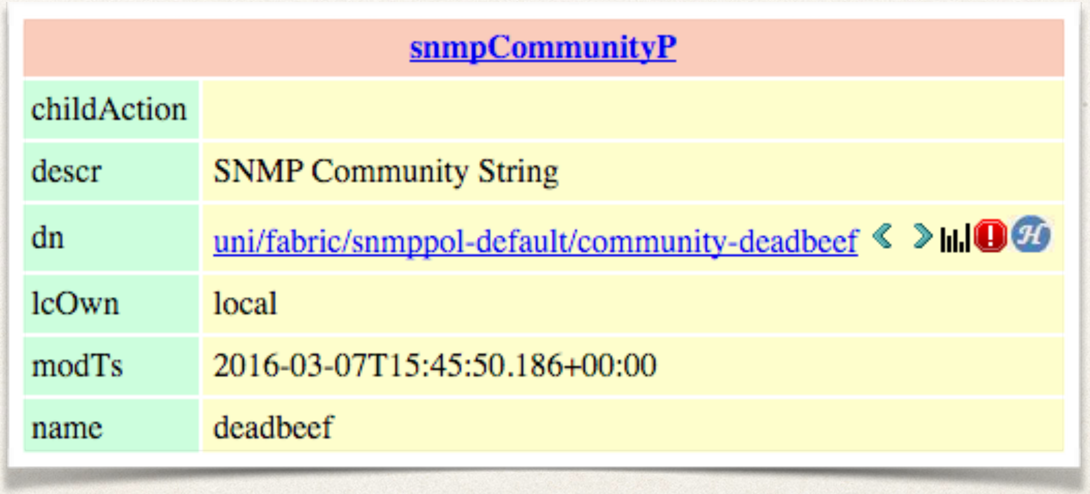
“VISORE”

snmpCommunityP


APIC Object Store Browser

Filter

Class or DN:	snmpCommunityP
Property:	
Op:	==
Val1:	

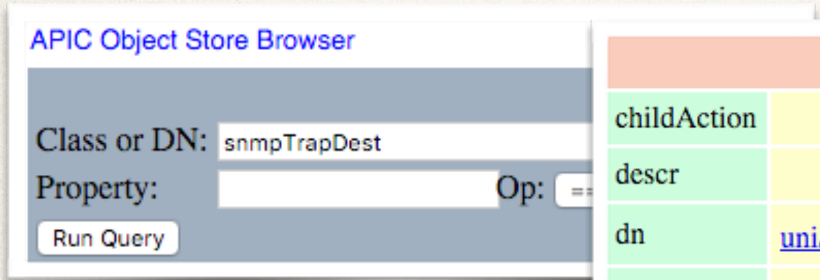

snmpCommunityP

childAction	
descr	SNMP Community String
dn	uni/fabric/snmppol-default/community-deadbeef
lcOwn	local
modTs	2016-03-07T15:45:50.186+00:00
name	deadbeef

Verify ACI SNMP Configuration (cont.)

“VISORE”

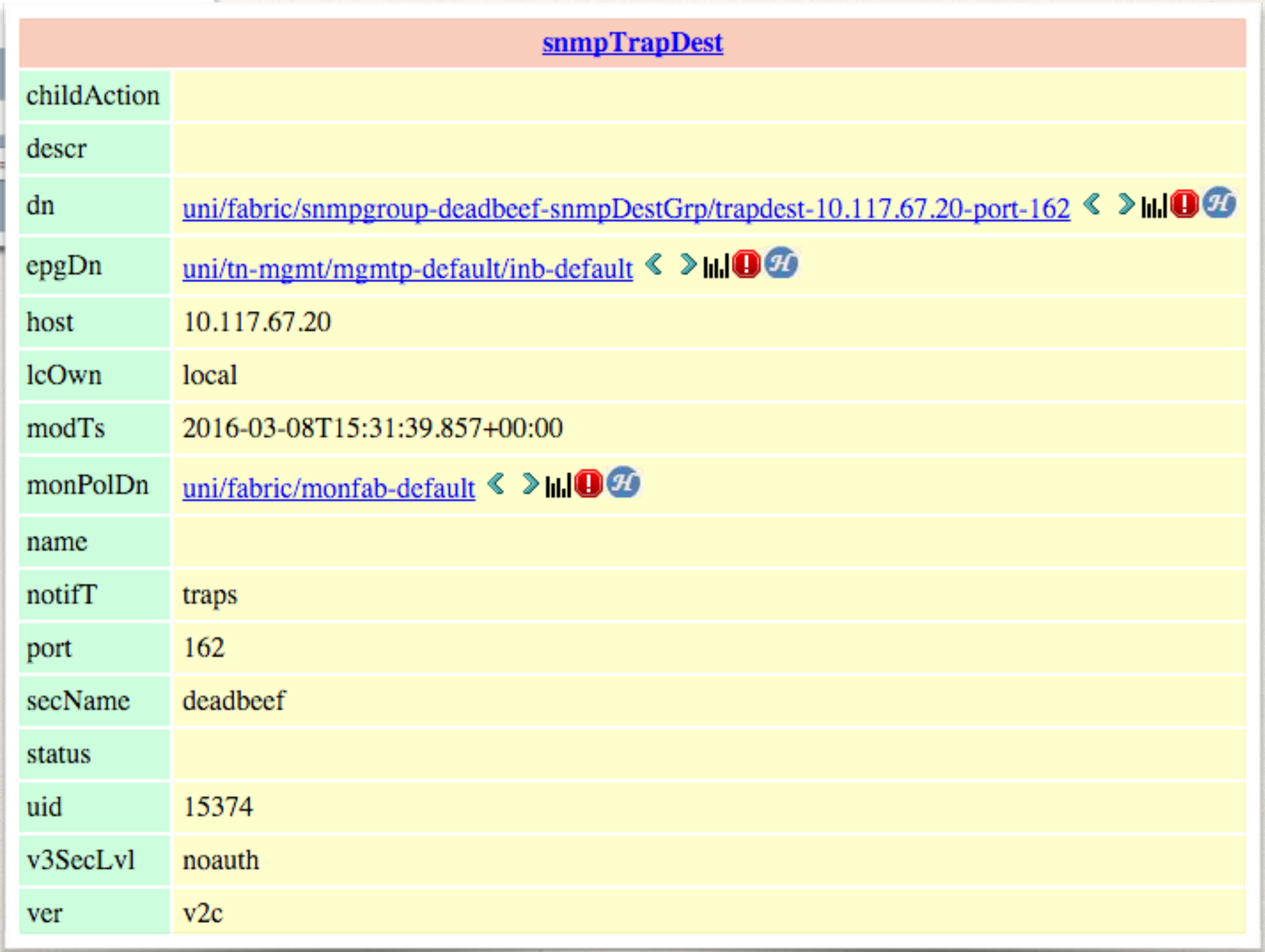
snmpTrapDest



APIC Object Store Browser

Class or DN:	snmpTrapDest
Property:	
Op:	==

Run Query

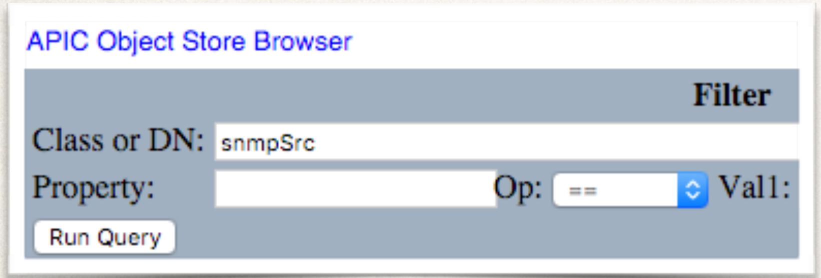


snmpTrapDest	
childAction	
descr	
dn	uni/fabric/snmpgroup-deadbeef-snmpDestGrp/trapdest-10.117.67.20-port-162
epgDn	uni/tn-mgmt/mgmtp-default/inb-default
host	10.117.67.20
lcOwn	local
modTs	2016-03-08T15:31:39.857+00:00
monPolDn	uni/fabric/monfab-default
name	
notifT	traps
port	162
secName	deadbeef
status	
uid	15374
v3SecLvl	noauth
ver	v2c

Verify ACI SNMP Configuration (cont.)

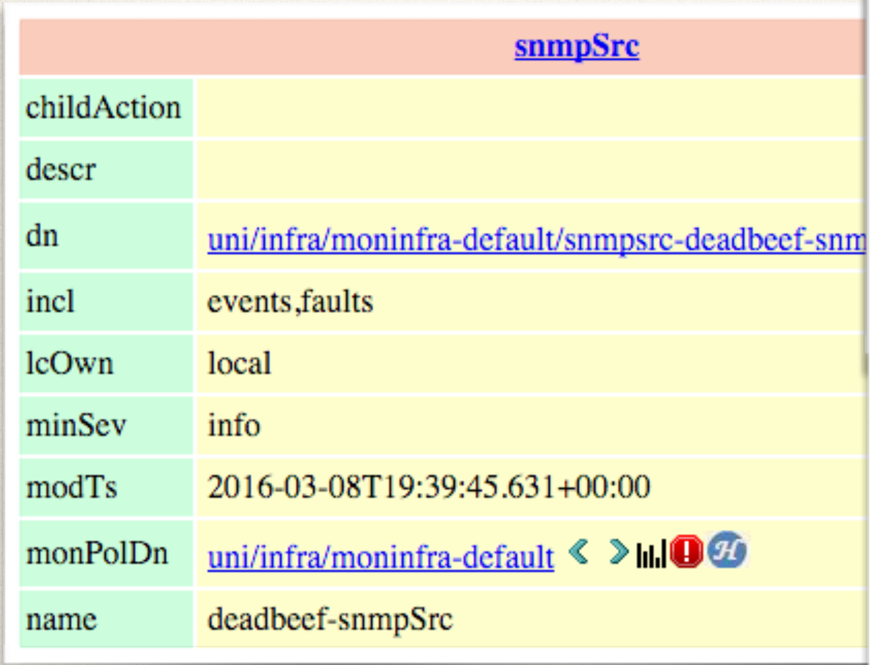
"VISORE"

snmpSrc

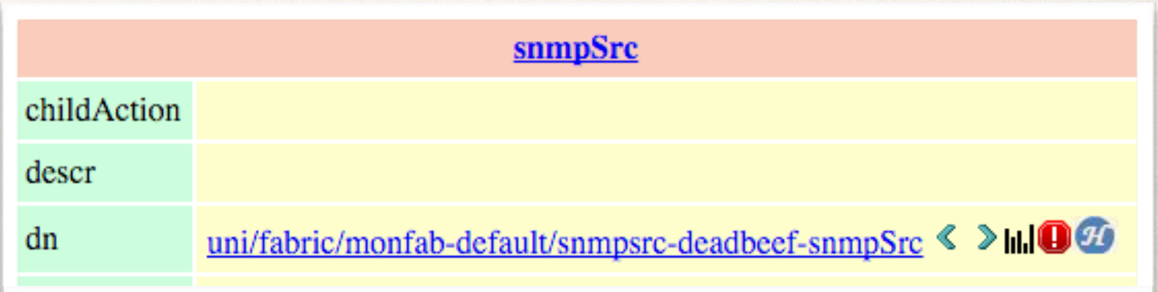


APIC Object Store Browser

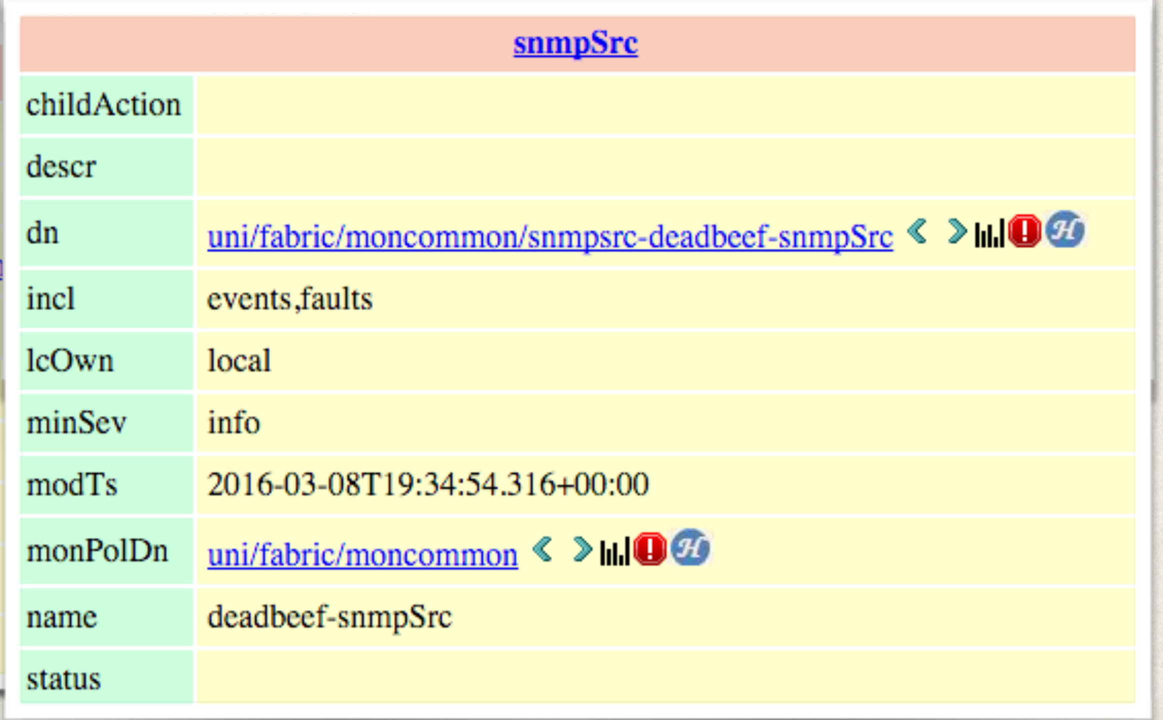
	Filter
Class or DN:	snmpSrc
Property:	Op: == Val1:



snmpSrc	
childAction	
descr	
dn	uni/infra/moninfra-default/snmpsrc-deadbeef-snm
incl	events,faults
lcOwn	local
minSev	info
modTs	2016-03-08T19:39:45.631+00:00
monPolDn	uni/infra/moninfra-default
name	deadbeef-snmpSrc



snmpSrc	
childAction	
descr	
dn	uni/fabric/monfab-default/snmpsrc-deadbeef-snmpSrc



snmpSrc	
childAction	
descr	
dn	uni/fabric/moncommon/snmpsrc-deadbeef-snmpSrc
incl	events,faults
lcOwn	local
minSev	info
modTs	2016-03-08T19:34:54.316+00:00
monPolDn	uni/fabric/moncommon
name	deadbeef-snmpSrc
status	

Verify ACI SNMP Configuration (cont.)

"Logical Model"

- **Checking the Logical Model on the APIC is another way to verify configuration of SNMP Policies. On an APIC , run " *Cat ..../summary* " on the key components of the SNMP configuration for the ACI Fabric. The following is a list of SUMMARY files to use to verify the SNMP configuration.**
  - cat /aci/tenants/mgmt/security-policies/out-of-band-contracts/summary
  - cat /aci/tenants/mgmt/security-policies/filters/summary
  - cat /aci/tenants/mgmt/node-management-epgs/default/out-of-band/default/summary
  - cat /aci/admin/external-data-collectors/monitoring-destinations/snmp/*/snmp-trap-destinations/summary
  - cat /aci/fabric/fabric-policies/pod-policies/policies/snmp/summary
  - cat /aci/fabric/fabric-policies/pod-policies/policies/snmp/*/summary
  - cat /aci/fabric/fabric-policies/pod-policies/policies/snmp/*/client-group-policies/*/*/summary
  - cat /aci/fabric/fabric-policies/pod-policies/policy-groups/summary
  - cat /aci/fabric/fabric-policies/pod-policies/pod-selector-default-all/summary
  - cat /aci/fabric/fabric-policies/monitoring-policies/monitoring-policy-default/callhome-snmp-syslog/all/snmp*/summary
  - cat /aci/fabric/fabric-policies/monitoring-policies/common-policy/callhome-snmp-syslog/snmp/*/summary
  - cat /aci/fabric/access-policies/monitoring-policies/default/callhome-snmp-syslog/all/snmp*/summary

Debugging SNMP on LEAF\SPINE Nodes

In addition to the “Show” commands that listed earlier to verify the SNMP configuration on Leaf\Spine Nodes, you can use some additional commands to gather more information in regards to SNMP. Some of the following commands may require ROOT access. Temporary “Root” access requires assistance from a Cisco ACI TAC Engineer.

- Additional Commands to run on the leaf or spine prior to accessing ROOT:
  - **show vrf**

    used to get the “VRF-ID” for “management” & “mgmt:inb”. The VRF-IDs are used in reading the iptables.
  - **show ip route vrf management**
  - **show ip route vrf mgmt:inb**

    “show ip route vrf” commands are used to verify routes in the management VRFs.

For Example:

```
rtp1-leaf1# show vrf
 VRF-Name                           VRF-ID State    Reason
 management                              2 Up       --
 mgmt:inb                                9 Up       --
```

Debugging SNMP on LEAF\SPINE (cont.)

- On each Leaf or Spine that you are troubleshooting SNMP issues access ROOT user before moving on to each of the sections to follow. As mentioned earlier temporary ROOT access is granted by the Cisco ACI TAC Engineer. And you can work with the Cisco ACI CSE running the following troubleshooting commands. We are including sample output so that you can see the different information that you can gather.

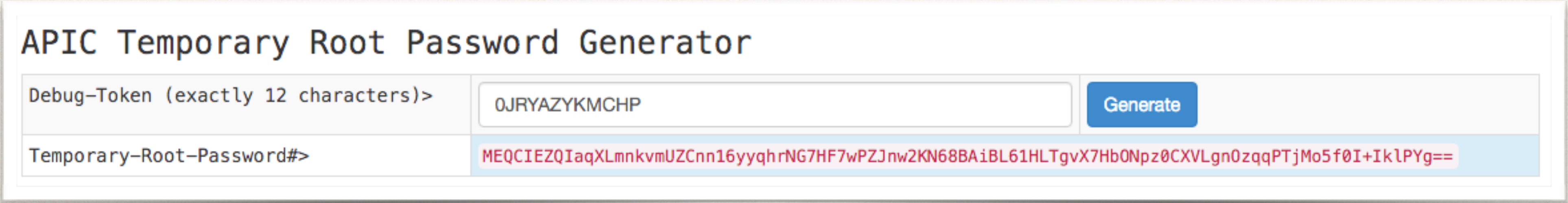
  - Get ROOT Password from the Cisco TAC Support Team.
    - **ssh to leaf\spine as admin user (***i.e. ssh admin@a.b.c.d***)**
    - **acidiag dbgtoken**

      used to generate a temporary password token to be used with ROOT password tool. Get ROOT Password from the Cisco ACI TAC Engineer
    - **access root (***i.e. ssh root@localhost***).** *Use the root password string provided by the Cisco ACI TAC Engineer*

For Example:

(note: some output has been abbreviated for display purposes)

```
rtp1-leaf1# acidiag dbgtoken
0JRYAZYKMCHP
```



APIC Temporary Root Password Generator

Debug-Token (exactly 12 characters)>	0JRYAZYKMCHP	Generate
Temporary-Root-Password#>	MEQCIEZQIaqXLmnkvmUZCnn16yyqhrNG7HF7wPZJnw2KN68BAiBL61HLTgvX7HbONpz0CXVLgnOzqqPTjMo5f0I+IklPYg==	

```
rtp1-leaf1# ssh root@localhost
Welcome to RTP Fabric 1!
Password: MEQCIEZQIaqXLmnkvmUZCnn16yyqhrNG7HF7wPZJnw2KN68BAiBL61HLTgvX7HbONpz0CXVLgnOzqqPTjMo5f0I+IklPYg==

rtp1-leaf1# whoami
root
```

Debugging SNMP on LEAF\SPINE (cont.)

- On each Leaf or Spine, verify the "snmpd" process is running. Record the process ID (pid) for "snmpd". You can use one or both of the following commands:
  - ps aux | grep snmp
  - pidof snmpd

For Example:
(note: some output has been abbreviated for display purposes)

```
rtp1-leaf1# ps aux | grep snmp
root      5881 16.2  2.5 1907404 411444 ?      Ssl  Apr05 496:35 /isan/bin/snmpd -f -s
-d udp:161 udp6:161 tcp:161

rtp1-leaf1# pidof snmpd
5881
```

Note: Repeat on each Leaf or Spine node having issues with the SNMP feature.

Debugging SNMP on LEAF\SPINE

"netstat"

- On each Leaf or Spine, gather some network statistics in relation to "snmp" and "snmp ports". You use the output to verify the management interfaces are transmitting & recieving packets. You can also verify that the Leaf or Spine node is listening on the SNMP ports. You can use the following commands to gather network status:
    - netstat -ai | grep eth0
    - netstat -ai | grep kpm_inb
    - netstat -nr
    - netstat -uta | grep snmp
    - netstat -lutn | grep 161

For Example:
(note: some output has been abbreviated for display purposes)

```
rtp1-leaf1# netstat -ai | grep eth0
Kernel Interface table
Iface    MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500 0    501277      0      0 0        633546      0      0      0 BMRU
```

Note: Repeat on each Leaf or Spine node having issues with the SNMP feature.

Debugging SNMP on LEAF\SPINE

"netstat"

For Example: (cont.)

```
rtp1-leaf1# netstat -ai | grep kpm_inb
Kernel Interface table
Iface    MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
kpm_inb    9300 0  10361421      0      0 0       8958506      0    126      0 BMRU

rtp1-leaf1# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.122.254.1    0.0.0.0         UG        0 0          0 eth0
10.122.254.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
127.1.0.0       0.0.0.0         255.255.0.0     U         0 0          0 kpm_inb

rtp1-leaf1# netstat -uta | grep snmp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:snmp                  *:*                     LISTEN
udp        0      0 *:snmp                  *:*
udp6       0      0 [::]:snmp               [::]:*

rtp1-leaf1# netstat -lutn | grep 161
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:161             0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:161             0.0.0.0:*
udp6       0      0 :::161                  :::*
```

Note: Repeat on each Leaf or Spine node having issues with the SNMP feature.

Debugging SNMP on LEAF\SPINE

“iptables”

- On each Leaf or Spine, check the “**iptables**” to see what rules are programmed for SNMP . The programming of “iptable” rules is crucial to the success of the SNMP configuration and deployment to Leaf & Spine nodes. You can use the following commands to check the “iptable” rules:
  - iptables --list | grep snmp
  - iptables -S | grep -i snmp
  - iptables -nvL

Note: Refer to the “show vrf” commands mentioned earlier and repeat on each Leaf or Spine node having issues with the SNMP feature.

For Example:

(note: some output has been abbreviated for display purposes)

```
rtp1-leaf1# show vrf
 VRF-Name                          VRF-ID State    Reason
 management                             2 Up       --
 mgmt:inb                               9 Up       --
```

Debugging SNMP on LEAF\SPINE

“iptables”

For Example: (cont.)
(note: some output has been abbreviated for display purposes)

```
rtp1-leaf1# iptables --list | grep snmp
snmp_rules  udp  --  anywhere             anywhere             udp dpt:snmp
Chain snmp_rules (1 references)
vrf_2_snmp_rules  all  --  anywhere             anywhere             vrf  2
vrf_9_snmp_rules  all  --  anywhere             anywhere             vrf  9
ACCEPT     udp  --  anywhere             anywhere             src-class-id  49153  udp dpt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere             src-class-id  49153  udp dpt:snmp
ACCEPT     udp  --  anywhere             anywhere             src-class-id  49153  udp spt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere             src-class-id  49153  udp spt:snmp
Chain vrf_2_snmp_rules (1 references)
ACCEPT     udp  --  anywhere             anywhere             src-class-id  32771  udp dpt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere             src-class-id  32771  udp dpt:snmp
ACCEPT     udp  --  anywhere             anywhere             src-class-id  49153  udp dpt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere             src-class-id  49153  udp dpt:snmp
ACCEPT     udp  --  anywhere             anywhere             src-class-id  49153  udp spt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere             src-class-id  49153  udp spt:snmp
ACCEPT     udp  --  anywhere             anywhere             src-class-id  32771  udp spt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere             src-class-id  32771  udp spt:snmp
ACCEPT     udp  --  anywhere             anywhere              udp dpt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere              udp dpt:snmp
ACCEPT     udp  --  anywhere             anywhere              udp spt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere              udp spt:snmp
Chain vrf_9_snmp_rules (1 references)
```

Debugging SNMP on LEAF\SPINE

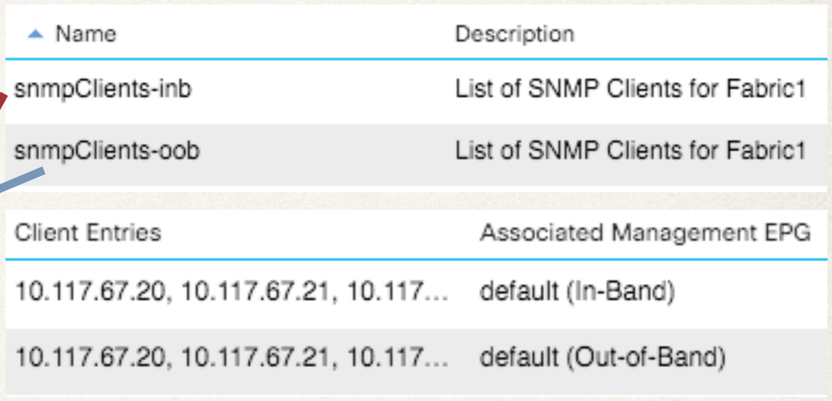
"iptables"

For Example: (cont.)

(note: some output has been abbreviated for display purposes)

```
rtp1-leaf1# iptables -S | grep -i snmp
-N snmp_rules
-N vrf_2_snmp_rules
-N vrf_9_snmp_rules
-A INPUT -p udp -m udp --dport 161 -j snmp_rules
-A snmp_rules -m vrf --vrf 2 -j vrf_2_snmp_rules
-A snmp_rules -m vrf --vrf 9 -j vrf_9_snmp_rules
-A snmp_rules -j DROP
-A vrf_2_snmp_rules -s 10.150.45.175/32 -j ACCEPT
-A vrf_2_snmp_rules -s 10.150.188.139/32 -j ACCEPT
-A vrf_2_snmp_rules -s 10.117.67.23/32 -j ACCEPT
-A vrf_2_snmp_rules -s 10.117.67.21/32 -j ACCEPT
-A vrf_2_snmp_rules -s 10.122.254.129/32 -j ACCEPT
-A vrf_2_snmp_rules -s 10.117.67.20/32 -j ACCEPT
-A vrf_2_snmp_rules -s 10.117.67.25/32 -j ACCEPT
-A vrf_2_snmp_rules -s 10.117.67.24/32 -j ACCEPT
-A vrf_2_snmp_rules -s 10.117.67.22/32 -j ACCEPT
-A vrf_2_snmp_rules -j DROP
-A vrf_9_snmp_rules -s 10.150.45.175/32 -j ACCEPT
-A vrf_9_snmp_rules -s 10.150.188.139/32 -j ACCEPT
-A vrf_9_snmp_rules -s 10.117.67.23/32 -j ACCEPT
-A vrf_9_snmp_rules -s 10.117.67.21/32 -j ACCEPT
-A vrf_9_snmp_rules -s 10.117.67.20/32 -j ACCEPT
-A vrf_9_snmp_rules -s 10.117.67.25/32 -j ACCEPT
-A vrf_9_snmp_rules -s 10.117.67.24/32 -j ACCEPT
-A vrf_9_snmp_rules -s 10.117.67.22/32 -j ACCEPT
-A vrf_9_snmp_rules -j DROP
```

SNMP Policy - Client Group Policies



▲ Name	Description
snmpClients-inb	List of SNMP Clients for Fabric1
snmpClients-oob	List of SNMP Clients for Fabric1

Client Entries	Associated Management EPG
10.117.67.20, 10.117.67.21, 10.117...	default (In-Band)
10.117.67.20, 10.117.67.21, 10.117...	default (Out-of-Band)

Debugging SNMP on LEAF\SPINE

“iptables”

For Example: (cont.)

```
rtp1-leaf1# iptables -nvL
Chain INPUT (policy DROP 1806 packets, 156K bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    73 snmp_rules  udp  --  *      *       0.0.0.0/0            0.0.0.0/0            udp dpt:161
 794K  292M vrf_2_mrules  all  --  *      *       0.0.0.0/0            0.0.0.0/0            vrf  2
   24  4240 vrf_9_mrules  all  --  *      *       0.0.0.0/0            0.0.0.0/0            vrf  9

Chain snmp_rules (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 vrf_2_snmp_rules  all  --  *      *       0.0.0.0/0            0.0.0.0/0            vrf  2
    1    73 vrf_9_snmp_rules  all  --  *      *       0.0.0.0/0            0.0.0.0/0            vrf  9
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain vrf_2_mrules (1 references)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           src-class-id  49153  udp dpt:162
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           src-class-id  49153  udp dpt:161
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           src-class-id  49153  udp spt:162
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           src-class-id  49153  udp spt:161

Chain vrf_2_snmp_rules (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       10.150.45.175        0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.150.188.139       0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.23         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.21         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.122.254.129       0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.20         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.25         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.24         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.22         0.0.0.0/0
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Debugging SNMP on LEAF\SPINE

“iptables”

For Example: (cont.)

iptables -nvL (output continued)

```
Chain vrf_9_mrules (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            src-class-id  32771  udp dpt:162
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            src-class-id  32771  udp dpt:161
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            src-class-id  49153  udp dpt:162
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            src-class-id  49153  udp dpt:161
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            src-class-id  49153  udp spt:162
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            src-class-id  49153  udp spt:161
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            src-class-id  32771  udp spt:162
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            src-class-id  32771  udp spt:161
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0             udp dpt:162
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0             udp dpt:161
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0             udp spt:162
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0             udp spt:161

Chain vrf_9_snmp_rules (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       10.150.45.175        0.0.0.0/0
    1    73 ACCEPT     all  --  *      *       10.150.188.139       0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.23         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.21         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.20         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.25         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.24         0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       10.117.67.22         0.0.0.0/0
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Note: If SNMP is running and you are not seeing snmp in the IP Tables, use the recorded "snmpd" pid and kill the process. Use "kill -9 [snmp pid]". After killing the process, check the IP Tables again.

Debugging SNMP on LEAF\SPINE

Verify SNMP Traps using “tcpdump”

- Access the Leaf\Spine as "root" user and use "**tcpdump**" command to verify SNMP Traps are being sent. Use UDP port **162** or any other UDP Ports that are configured for the SNMP trap destinations in the ACI SNMP Monitoring Group. You can use the following "**tcpdump**" commands to check for SNMP Traps on Leaf\Spine Nodes:
    - tcpdump -i oobmgmt -f port 162 -vv
    - tcpdump -i eth0 -f port 162 -vv
    - tcpdump -i kpm_inb -f port 162 -vv

For Example:

LEAF (OOB)

```
rtp1-leaf1#  tcpdump -i eth0 -f port 162 -vv
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
17:21:49.810052 IP (tos 0x0, ttl 64, id 63116, offset 0, flags [none], proto UDP (17), length 218)

    rtp1-leaf1.cisco.com.35582 > kilimanjaro.cisco.com.snmp-trap: [bad udp cksum 5d96!]  { SNMPv2c C=deadbeef
{ V2Trap(171) R=253  system.sysUpTime.0=5888267 S:1.1.4.1.0=E:cisco.9.276.0.1
interfaces.ifTable.ifEntry.ifIndex.436224000=436224000 interfaces.ifTable.ifEntry.ifAdminStatus.436224000=2
interfaces.ifTable.ifEntry.ifOperStatus.436224000=2 31.1.1.1.1.436224000="eth1/5"
interfaces.ifTable.ifEntry.ifType.436224000=6 } }
17:21:49.911304 IP (tos 0x0, ttl 64, id 63121, offset 0, flags [none], proto UDP (17), length 218)

    rtp1-leaf1.cisco.com.35582 > kilimanjaro.cisco.com.snmp-trap: [bad udp cksum 5d8b!]  { SNMPv2c C=deadbeef
{ V2Trap(171) R=254  system.sysUpTime.0=5888277 S:1.1.4.1.0=E:cisco.9.276.0.1
interfaces.ifTable.ifEntry.ifIndex.436224000=436224000 interfaces.ifTable.ifEntry.ifAdminStatus.436224000=2
interfaces.ifTable.ifEntry.ifOperStatus.436224000=2 31.1.1.1.1.436224000="eth1/5"
interfaces.ifTable.ifEntry.ifType.436224000=6 } }
17:22:32.864114 IP (tos 0x0, ttl 64, id 63205, offset 0, flags [none], proto UDP (17), length 218)
```

Debugging SNMP on LEAF\SPINE

Verify SNMP Traps using "tcpdump" (cont.)

For Example:

```
LEAF (INB)

rtp1-leaf1#   tcpdump -i kpm_inb  -f port 162 -vv
tcpdump: listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes

    172.18.242.14.35944 > 10.150.188.139.snmp-trap: [udp sum ok]  { SNMPv2c C=deadbeef { V2Trap(172) R=760
system.sysUpTime.0=18531622 S:1.1.4.1.0=E:cisco.9.276.0.1 interfaces.ifTable.ifEntry.ifIndex.436224000=436224000
interfaces.ifTable.ifEntry.ifAdminStatus.436224000=2 interfaces.ifTable.ifEntry.ifOperStatus.436224000=2
31.1.1.1.1.436224000="eth1/5" interfaces.ifTable.ifEntry.ifType.436224000=6 } }
17:31:45.302886 IP (tos 0x0, ttl 65, id 7435, offset 0, flags [none], proto UDP (17), length 219)

    172.18.242.14.50066 > 10.150.45.175.snmp-trap: [udp sum ok]  { SNMPv2c C=deadbeef { V2Trap(172) R=760
system.sysUpTime.0=18531622 S:1.1.4.1.0=E:cisco.9.276.0.1 interfaces.ifTable.ifEntry.ifIndex.436224000=436224000
interfaces.ifTable.ifEntry.ifAdminStatus.436224000=2 interfaces.ifTable.ifEntry.ifOperStatus.436224000=2
31.1.1.1.1.436224000="eth1/5" interfaces.ifTable.ifEntry.ifType.436224000=6 } }
17:31:45.403462 IP (tos 0x0, ttl 65, id 7517, offset 0, flags [none], proto UDP (17), length 219)

    172.18.242.14.35944 > 10.150.188.139.snmp-trap: [udp sum ok]  { SNMPv2c C=deadbeef { V2Trap(172) R=761
system.sysUpTime.0=18531633 S:1.1.4.1.0=E:cisco.9.276.0.1 interfaces.ifTable.ifEntry.ifIndex.436224000=436224000
interfaces.ifTable.ifEntry.ifAdminStatus.436224000=2 interfaces.ifTable.ifEntry.ifOperStatus.436224000=2
31.1.1.1.1.436224000="eth1/5" interfaces.ifTable.ifEntry.ifType.436224000=6 } }
17:31:45.404221 IP (tos 0x0, ttl 65, id 7523, offset 0, flags [none], proto UDP (17), length 219)

    172.18.242.14.50066 > 10.150.45.175.snmp-trap: [udp sum ok]  { SNMPv2c C=deadbeef { V2Trap(172) R=761
system.sysUpTime.0=18531633 S:1.1.4.1.0=E:cisco.9.276.0.1 interfaces.ifTable.ifEntry.ifIndex.436224000=436224000
interfaces.ifTable.ifEntry.ifAdminStatus.436224000=2 interfaces.ifTable.ifEntry.ifOperStatus.436224000=2
31.1.1.1.1.436224000="eth1/5" interfaces.ifTable.ifEntry.ifType.436224000=6 } }
17:31:45.504683 IP (tos 0x0, ttl 65, id 7533, offset 0, flags [none], proto UDP (17), length 219)
```

Debugging SNMP on LEAF\SPINE

Verify SNMP GET & WALK requests using "tcpdump"

- Access the Leaf\Spine as "root" user and use "**tcpdump**" command to verify SNMP GET & WALK requests are being received and responded to. Use UDP port **161** to monitor for SNMP GET & WALK requests. You can use the following "**tcpdump**" commands to check for SNMP GET & WALK requests on Leaf\Spine Nodes:
  - tcpdump -i oobmgmt -f port 161 -vv
  - tcpdump -i eth0 -f port 161 -vv
  - tcpdump -i kpm_inb -f port 161 -vv

Note: Refer to the earlier "iptables" material to verify and check that SNMP Client Group Policies are configured and deployed correctly.

For Example:

```
Sample SNMP GET Requests:

snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::sysDescr.0
snmpget -v2c -c deadbeef w.x.y.z SNMPv2-MIB::sysDescr.0
```

Where "deadbeef" is the community string; and "a.b.c.d" is the IP address for Leaf\Spine OOB mgmt interface and "w.x.y.z" is the IP address for Leaf\Spine In-Band mgmt interface.

Debugging SNMP on LEAF\SPINE

Verify SNMP GET & WALK requests using "tcpdump"

For Example:

```
snmpget -v2c -c deadbeef 10.122.254.241 SNMPv2-MIB::sysDescr.0

LEAF (OOB)
rtp1-leaf1# tcpdump -i eth0 -f port 161 -vv
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
17:26:08.548149 IP (tos 0x0, ttl 54, id 45467, offset 0, flags [none], proto UDP (17),
length 73)
    10.150.188.139.64245 > rtp1-leaf1.cisco.com.snmp: [udp sum ok]  { SNMPv2c C=deadbeef
{ GetRequest(28) R=949769396  system.sysDescr.0 } }
17:26:08.552290 IP (tos 0x0, ttl 64, id 30954, offset 0, flags [none], proto UDP (17),
length 238)
    rtp1-leaf1.cisco.com.snmp > 10.150.188.139.64245: [bad udp cksum 3c36!]  { SNMPv2c
C=deadbeef { GetResponse(191) R=949769396  system.sysDescr.0="Cisco NX-OS(tm) aci,
Software (aci-n9000-system), Version 11.2(2h), RELEASE SOFTWARE Copyright (c) 2002-2015
by Cisco Systems, Inc. Compiled 2016/02/24 07:51:47" } }


SNMP Agent - CLIENT
deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 10.122.254.241 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco NX-OS(tm) aci, Software (aci-n9000-system),
Version 11.2(2h), RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc.
Compiled 2016/02/24 07:51:47
```

Debugging SNMP on LEAF\SPINE

Verify SNMP GET & WALK requests using "tcpdump"

For Example:

```
snmpget -v2c -c deadbeef 172.18.242.14 SNMPv2-MIB::sysDescr.0

LEAF (INB)
rtp1-leaf1# tcpdump -i kpm_inb -f port 161 -vv
tcpdump: listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
17:27:38.527229 IP (tos 0x0, ttl 53, id 14415, offset 0, flags [none], proto UDP (17),
length 73)
    10.150.188.139.63531 > 172.18.242.14.snmp: [udp sum ok]  { SNMPv2c C=deadbeef
{ GetRequest(28) R=799765014  system.sysDescr.0 } }
17:27:38.542411 IP (tos 0x0, ttl 65, id 58277, offset 0, flags [none], proto UDP (17),
length 238)
    172.18.242.14.snmp > 10.150.188.139.63531: [udp sum ok]  { SNMPv2c C=deadbeef
{ GetResponse(191) R=799765014  system.sysDescr.0="Cisco NX-OS(tm) aci, Software (aci-
n9000-system), Version 11.2(2h), RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco
Systems, Inc. Compiled 2016/02/24 07:51:47" } }


SNMP Agent - CLIENT
deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.14 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco NX-OS(tm) aci, Software (aci-n9000-system),
Version 11.2(2h), RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc.
Compiled 2016/02/24 07:51:47
```

Debugging SNMP on LEAF\SPINE

Sample SNMP requests (OSX, Windows, & linux)

From MAC OSX Client:

```
(to leaf)  snmpget -v2c -c deadbeef 172.18.242.14 SNMPv2-MIB::sysDescr.0
(to spine) snmpget -v2c -c deadbeef 172.18.242.18 SNMPv2-MIB::sysDescr.0

deadbeef(osx):~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.14 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco NX-OS(tm) aci, Software (aci-n9000-system), Version 11.2(2h),
RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc. Compiled 2016/02/24 07:51:47

deadbeef(osx):~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.18 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco NX-OS(tm) aci, Software (aci-n9000-system), Version 11.2(2h),
RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc. Compiled 2016/02/24 07:51:47
```

From linux CentOS Client:

```
(to leaf)  snmpget -v2c -c deadbeef 172.18.242.14 SNMPv2-MIB::sysDescr.0
(to spine) snmpget -v2c -c deadbeef 172.18.242.18 SNMPv2-MIB::sysDescr.0

[root@deadbeef(linux)]# snmpget -v2c -c deadbeef 172.18.242.14 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco NX-OS(tm) aci, Software (aci-n9000-system), Version 11.2(2h),
RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc. Compiled 2016/02/24 07:51:47

[root@deadbeef(linux)]# snmpget -v2c -c deadbeef 172.18.242.18 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco NX-OS(tm) aci, Software (aci-n9000-system), Version 11.2(2h),
RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc. Compiled 2016/02/24 07:51:47
```

Debugging SNMP on LEAF\SPINE

Sample SNMP requests (OSX, Windows, & linux)

From Windows Client:

```
https://www.snmpsoft.com/cmd-tools/

(to leaf)  SnmpGet.exe -r:172.18.242.14 -v:2c -c:"deadbeef" -o:.1.3.6.1.2.1.1.1.0
(to spine) SnmpGet.exe -r:172.18.242.18 -v:2c -c:"deadbeef" -o:.1.3.6.1.2.1.1.1.0


C:\snmp-cli>SnmpGet.exe -r:172.18.242.14 -v:2c -c:"deadbeef" -o:.1.3.6.1.2.1.1.1.0
SnmpGet v1.01 - Copyright (C) 2009 SnmpSoft Company
[ More useful network tools on http://www.snmpsoft.com ]

OID=.1.3.6.1.2.1.1.1.0
Type=OctetString
Value=Cisco NX-OS(tm) aci, Software (aci-n9000-system), Version 11.2(2h),
RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc. Compiled 2016/02/24 07:51:47


C:\snmp-cli>SnmpGet.exe -r:172.18.242.18 -v:2c -c:"deadbeef" -o:.1.3.6.1.2.1.1.1.0
SnmpGet v1.01 - Copyright (C) 2009 SnmpSoft Company
[ More useful network tools on http://www.snmpsoft.com ]

OID=.1.3.6.1.2.1.1.1.0
Type=OctetString
Value=Cisco NX-OS(tm) aci, Software (aci-n9000-system), Version 11.2(2h),
RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc. Compiled 2016/02/24 07:51:47
```

Debugging SNMP on LEAF\SPINE

Verify SNMP Requests & Traps using “strace”

- Access the Leaf\Spine as "root" user and use "**strace**" command to trace the SNMP process for SNMP Requests & SNMP Traps on the Leaf\Spine Nodes. You can use the following "**strace**" commands to trace the SNMP process on Leaf\Spine Nodes:
    - strace -p 5881 -f -e trace=network -s 10000
    - strace -p 5881 -o snmpd_trace.txt

      where 5881 is PID of SNMP process

For Example:

```
LEAF (PROCESS)

rtp1-leaf1# strace -p 5881 -f -e trace=network -s 10000
Process 5881 attached with 3 threads

[pid  5881] recvmsg(25, {msg_name(16)={sa_family=AF_INET, sin_port=htons(55400),
sin_addr=inet_addr("10.150.188.139")}, msg_iov(1)=[{"0+\2\1\1\4\10deadbeef\240\34\2\4\22\\
\10:\2\1\0\2\1\0000\0160\f\6\10+\6\1\2\1\1\1\0\5\0", 65536}], msg_controllen=24, {cmsg_len=24,
cmsg_level=SOL_IP, cmsg_type=, ...}, msg_flags=0}, 0) = 45

[pid  5881] sendmsg(25, {msg_name(16)={sa_family=AF_INET, sin_port=htons(55400),
sin_addr=inet_addr("10.150.188.139")}, msg_iov(1)=[{"0\201\317\2\1\1\4\10deadbeef\242\201\277\2\4\22\\
\10:\2\1\0\2\1\0000\201\2600\201\255\6\10+\6\1\2\1\1\1\0\4\201\240Cisco NX-OS(tm) aci, Software (aci-
n9000-system), Version 11.2(2h), RELEASE SOFTWARE Copyright (c) 2002-2015 by Cisco Systems, Inc.
Compiled 2016/02/24 07:51:47", 210}], msg_controllen=24, {cmsg_len=24, cmsg_level=SOL_IP,
cmsg_type=, ...}, msg_flags=0}, 0) = 210
```

Debugging SNMP on LEAF\SPINE

Some Log Files to search when Troubleshooting

- Access the Leaf\Spine as "admin" user and seach some of the following logs when troubleshooting SNMP Requests Leaf\Spine Nodes:
  - zgrep "snmp" /var/log/dme/log/*
  - zgrep "snmp" /var/log/dme/log/svc_ifc_dbgrelem.log*
  - zgrep "snmpd" /var/log/dme/log/svc_ifc_dbgrelem.log*
  - zgrep "snmpd_log" /var/log/dme/log/*

Note: Some of the above commands may or may not produce output when performed on a Leaf or Spine node. These are just some examples which may point you in the right direction.

```
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="0" lcOwn="local" moDn="sys/snmp" modTs="never"
priKey="2533:10078" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="0" lcOwn="local" moDn="uni/fabric/monfab-default/
snmpsrc-deadbeef-snmpSrc" modTs="never" priKey="1688:22618" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="0" lcOwn="local" moDn="uni/fabric/monfab-default/
snmpsrc-deadbeef-snmpSrc/monobjdn-[uni/fabric/snmpgroup-deadbeef-snmpGrp]" modTs="never" priKey="7876:22619" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="4" lcOwn="local" moDn="uni/fabric/moncommon/snmpsrc-
deadbeef-snmpSrc/monobjdn-[uni/fabric/snmpgroup-deadbeef-snmpGrp]" modTs="never" priKey="7876:22587" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="0" lcOwn="local" moDn="uni/infra/moninfra-default/
snmpsrc-deadbeef-snmpSrc" modTs="never" priKey="1688:22944" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="2" lcOwn="local" moDn="uni/infra/moninfra-default/
snmpsrc-deadbeef-snmpSrc/monobjdn-[uni/fabric/snmpgroup-deadbeef-snmpGrp]" modTs="never" priKey="7876:22945" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="0" lcOwn="local" moDn="uni/fabric/snmpgroup-deadbeef-
snmpGrp" modTs="never" priKey="1692:23052" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="0" lcOwn="local" moDn="uni/fabric/snmpgroup-deadbeef-
snmpGrp/trapdest-10.117.67.23-port-162" modTs="never" priKey="1691:23053" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:<moUpdateInfo chgBmp="" childAction="" dn="" index="4" lcOwn="local" moDn="uni/fabric/snmpgroup-deadbeef-
snmpGrp/trapdest-10.122.254.129-port-162" modTs="never" priKey="1691:23061" rn="" status=""/>
/var/log/dme/log/svc_ifc_eventmgr.log.3810.gz:6237||16-04-10 21:02:45.764-04:00||snmp||DBG4||co=doer:0:0:0x11cb:
1,dn="DmKUDAKIAAACpBQAAEQBkZWFkYmVlZi1zbm1wR3JwAA0AMTAuMTE3LjY3LjIzAA==",fn=[static bool trap::Notifier::notifyHelper(const snmp::GroupMo*,
const condition::InfoMo*, const mo::Mo*, double, double)]||Could not find the encoder function||../include/isan/objstoredefs/snmp/TrapNotifier.h||62
```

Debugging SNMP on the APIC

In addition to the “Show” commands that listed earlier to verify the SNMP configuration on APIC Controllers, you can use some additional commands to gather more information in regards to SNMP. Some of the following commands may require ROOT access. Temporary “Root” access requires assistance from a Cisco ACI TAC Engineer.

In "Brazos" & previous ACI releases, the leaf\spine node switches did NOT require a OOB or INB contract to allow SNMP Get Requests using UDP DestPort 161: for SNMP. These requests cannot be blocked through contracts. Creating a SNMP ClientGroup in the SNMP policy with a list of Client-IP Addresses restricts SNMP access to only the configured Client-IP Addresses. If no Client-IP address is configured, SNMP packets are allowed from anywhere.

In "Brazos", Cisco added SNMP support for the APIC(s). The behavior for default allowed ports for the APIC it is “Different". Unlike the Switches, a CONTRACT is needed for the APIC to allow SNMP. This is “NEW” with brazos. In your OOB Contract defined for your External Management Network Instance Profile. Once you add Ports 161 & 162 to the filter of the OOB Contract, your SNMP Gets should work as expected.

Also in addition to contracts being needed, Node Management Address(s) in the Tenant mgmt need to be configured for the APIC(s). Verify that the APIC Node management address(s) are configured also.

Debugging SNMP on the APIC (cont.)

- On each APIC that you are troubleshooting SNMP issues access ROOT user before moving on to each of the sections to follow. As mentioned earlier temporary ROOT access is granted by the Cisco ACI TAC Engineer. And you can work with the Cisco ACI CSE running the following troubleshooting commands. We are including sample output so that you can see the different information that you can gather.

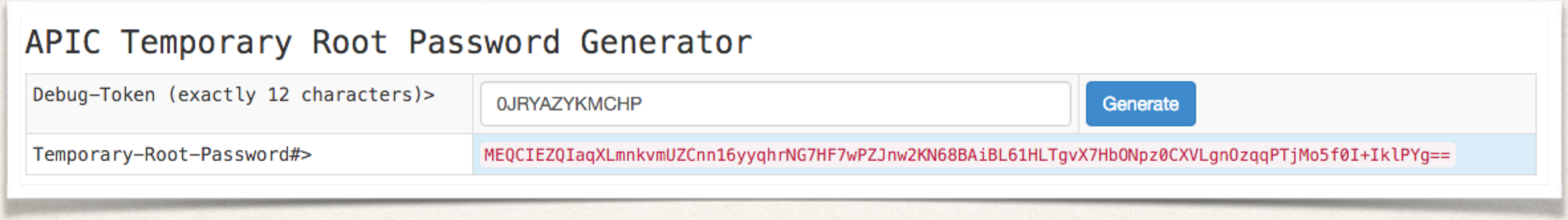
  - Get ROOT Password from the Cisco TAC Support Team.
    - **ssh to APIC(s) as admin user (***i.e. ssh admin@a.b.c.d***)**
    - **acidiag dbgtoken**

      used to generate a temporary password token to be used with ROOT password tool. Get ROOT Password from the Cisco ACI TAC Engineer
    - **access root (***i.e. ssh root@localhost***).** *Use the root password string provided by the Cisco ACI TAC Engineer*

For Example:

(note: some output has been abbreviated for display purposes)

```
rtp1-apic1# acidiag dbgtoken
0JRYAZYKMCHP
```



APIC Temporary Root Password Generator

Debug-Token (exactly 12 characters)>	0JRYAZYKMCHP	Generate
Temporary-Root-Password#>	MEQCIEZQIaqXLmnkvmUZCnn16yyqhrNG7HF7wPZJnw2KN68BAiBL61HLTgvX7HbONpz0CXVLgnOzqqPTjMo5f0I+IklPYg==	

```
rtp1-apic1# ssh root@localhost
Welcome to RTP Fabric 1!
Password: MEQCIEZQIaqXLmnkvmUZCnn16yyqhrNG7HF7wPZJnw2KN68BAiBL61HLTgvX7HbONpz0CXVLgnOzqqPTjMo5f0I+IklPYg==

rtp1-apic1# whoami
root
```

Debugging SNMP on APIC (cont.)

- On each APIC, verify the "snmpd" process is running. Record the process ID (pid) for "snmpd". You can use one or both of the following commands:
  - ps aux | grep snmp

<u>For Example:</u>
(note: some output has been abbreviated for display purposes)

```
root@rtp1-apic1:~# ps aux | grep snmp
ifc      32182  1.4  0.1 641196 239716 ?       Ssl  Apr10  54:47 /mgmt//bin/snmpd.bin
-f -p /tmp/snmpd2.pid -a -A -LE 0-2 -c /data//snmp/snmpd.conf

* snmpd PID = 32182
```

Note: Repeat on each APIC node having issues with the SNMP feature.

Debugging SNMP on APIC

“netstat”

- On each APIC, gather some network statistics in relation to “snmp” and “snmp ports”. You use the output to verify the management interfaces are transmitting & recieving packets. You can also verify that the APIC node is listening on the SNMP ports. You can use the following commands to gather network status:
  - netstat -ai | egrep "Iface|bond0.1100"
  - netstat -ai | egrep "Iface|bond0.1100|oobmgmt"
  - netstat -nr
  - netstat -lutn | grep 161
  - netstat -aon | grep ":161"

Note: “bond0.1100” is the vlan encap configured on the INB mgmt EPG for APIC. Replace “1100” for your configured vlan encap.

For Example:
(note: some output has been abbreviated for display purposes)

```
root@rtp1-apic1:~# netstat -ai | egrep "Iface|bond0.1100"
Iface        MTU    Met RX-OK   RX-ERR RX-DRP RX-OVR TX-OK   TX-ERR TX-DRP TX-OVR Flg
bond0.1100   1496   0   364145  0      153    0      304014   0     0      0      BMRU

root@rtp1-apic1:~# netstat -ai | egrep "Iface|bond0.1100|oobmgmt"
Iface        MTU    Met RX-OK   RX-ERR RX-DRP RX-OVR TX-OK   TX-ERR TX-DRP TX-OVR Flg
bond0.1100   1496   0   364174  0      153       0   304035  0      0      0      BMRU
oobmgmt      1500   0   2698447 0      0         0   1977753 0      0      0      BMRU
```

Note: Repeat on each APIC node having issues with the SNMP feature.

Debugging SNMP on APIC

“netstat”

For Example: (cont.)

```
root@rtp1-apic1:~# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         172.18.242.1    0.0.0.0         UG        0 0          0 bond0.1100
0.0.0.0         10.122.254.1    0.0.0.0         UG        0 0          0 oobmgmt
10.122.254.0    0.0.0.0         255.255.255.0   U         0 0          0 oobmgmt
172.18.242.0    0.0.0.0         255.255.255.192 U         0 0          0 bond0.1100
172.18.242.1    0.0.0.0         255.255.255.255 UH        0 0          0 bond0.1100

root@rtp1-apic1:~# netstat -lutn | grep 161
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
udp        0      0 0.0.0.0:161                 0.0.0.0:*
udp        0      0 :::161                      :::*

rtp1-apic1# netstat -aon | grep ":161"
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State       Timer
udp        0      0 0.0.0.0:161                 0.0.0.0:*                               off (0.00/0/0)
udp        0      0 :::161                      :::*                                    off (0.00/0/0)
```

Note: Repeat on each APIC node having issues with the SNMP feature.

Debugging SNMP on APIC
"iptables"

- On each APIC, check the "**iptables**" to see what rules are programmed for SNMP . The programming of "iptable" rules is crucial to the success of the SNMP configuration and deployment to APICs. You can use the following commands to check the "iptable" rules:
  - Use the output of "show snmp clientgroups" and "show snmp hosts" and compare with "iptables"
  - iptables -S | grep 161
  - iptables -S | grep 162
  - iptables --list | grep snmp
  - iptables --list -v | grep snmp
  - iptables --list -v
  - iptables -nvL

For Example:

```
rtp1-apic1# show snmp hosts
 IP-Address             Version     Security Level  Community
 --------------------   ----------  ----------      --------------------
 10.150.188.139         v2c         noauth          deadbeef
 10.150.45.175          v2c         noauth          deadbeef
 10.117.67.20           v2c         noauth          deadbeef
 10.122.254.129         v2c         noauth          deadbeef
 10.117.67.20           v2c         noauth          deadbeef
 10.117.67.23           v2c         noauth          deadbeef
```

Debugging SNMP on APIC

“iptables”

For Example: (cont.)

(note: some output has been abbreviated for display purposes)

```
rtp1-apic1# show snmp clientgroups
 SNMP Policy           Name                  Description           Client Entries        Associated Management EPG
 --------------------  --------------------  --------------------  --------------------  --------------------
 default               snmpClients-oob       List of SNMP Clients  10.117.67.21,10.117.  default (Out-Of-Band)
                                             for Fabric1           67.20,10.117.67.23,1
                                                                   0.117.67.22,10.117.6
                                                                   7.25,10.117.67.24,10
                                                                   .150.46.242,10.122.2
                                                                   54.129,10.150.188.85
 default               snmpClients-inb       List of SNMP Clients  10.117.67.21,10.117.  default (In-Band)
                                             for Fabric1           67.20,10.117.67.23,1
                                                                   0.117.67.22,10.117.6
                                                                   7.25,10.117.67.24,10
                                                                   .150.46.242,10.150.1
                                                                   88.85

rtp1-apic1# iptables -S | grep 161
-A fp-137 -s 10.0.0.0/8 -p udp -m udp --dport 161 -j ACCEPT
-A fp-137 -s 172.18.217.0/24 -p udp -m udp --dport 161 -j ACCEPT
-A fp-137 -s 172.18.242.0/24 -p udp -m udp --dport 161 -j ACCEPT

rtp1-apic1# iptables -S | grep 162
-A fp-137 -s 10.0.0.0/8 -p udp -m udp --dport 1162 -j ACCEPT
-A fp-137 -s 172.18.217.0/24 -p udp -m udp --dport 1162 -j ACCEPT
-A fp-137 -s 172.18.242.0/24 -p udp -m udp --dport 1162 -j ACCEPT
-A fp-137 -s 10.0.0.0/8 -p udp -m udp --dport 162 -j ACCEPT
-A fp-137 -s 172.18.217.0/24 -p udp -m udp --dport 162 -j ACCEPT
-A fp-137 -s 172.18.242.0/24 -p udp -m udp --dport 162 -j ACCEPT
```

Debugging SNMP on APIC

"iptables"

For Example: (cont.)

(note: some output has been abbreviated for display purposes)

```
root@rtp1-apic1:~# iptables --list | grep snmp
ACCEPT     udp  --  10.0.0.0/8           anywhere             udp dpt:snmp
ACCEPT     udp  --  172.18.217.0/24      anywhere             udp dpt:snmp
ACCEPT     udp  --  172.18.242.0/24      anywhere             udp dpt:snmp
ACCEPT     udp  --  10.0.0.0/8           anywhere             udp dpt:snmptrap
ACCEPT     udp  --  172.18.217.0/24      anywhere             udp dpt:snmptrap
ACCEPT     udp  --  172.18.242.0/24      anywhere             udp dpt:snmptrap

root@rtp1-apic1:~# iptables --list -v | grep snmp
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  any    any     10.0.0.0/8           anywhere             udp dpt:snmp
    0     0 ACCEPT     udp  --  any    any     172.18.217.0/24      anywhere             udp dpt:snmp
    0     0 ACCEPT     udp  --  any    any     172.18.242.0/24      anywhere             udp dpt:snmp
    0     0 ACCEPT     udp  --  any    any     10.0.0.0/8           anywhere             udp dpt:snmptrap
    0     0 ACCEPT     udp  --  any    any     172.18.217.0/24      anywhere             udp dpt:snmptrap
    0     0 ACCEPT     udp  --  any    any     172.18.242.0/24      anywhere             udp dpt:snmptrap

root@rtp1-apic1:~# iptables --list -v
Chain INPUT (policy DROP 53 packets, 7093 bytes)
 pkts bytes target             prot opt in     out     source               destination
10557 2595K apic-default-drop  all  --  any    any     anywhere             anywhere
10557 2595K apic-default-allow all  --  any    any     anywhere             anywhere
   65  7717 fp-137             all  --  any    any     anywhere             anywhere
   53  7093 fp-138             all  --  any    any     anywhere             anywhere
   43  3813 apic-default       all  --  any    any     10.122.254.0/24      anywhere
```

Note: the "fp-137 & fp-138" listed above is the OOB contract & filters. INB contracts & filters are not programmed since the filtering is applied at the border or services leaf.

Debugging SNMP on APIC

“iptables”

For Example: (cont.)

(note: some output has been abbreviated for display purposes)

```
root@rtp1-apic1:~# iptables -nvL
Chain INPUT (policy DROP 304 packets, 31953 bytes)
 pkts bytes target              prot opt in     out     source               destination
64774   12M apic-default-drop   all  --  *      *       0.0.0.0/0            0.0.0.0/0
64774   12M apic-default-allow  all  --  *      *       0.0.0.0/0            0.0.0.0/0
  378 35801 fp-137              all  --  *      *       0.0.0.0/0            0.0.0.0/0
  304 31953 fp-138              all  --  *      *       0.0.0.0/0            0.0.0.0/0
  282 24737 apic-default        all  --  *      *       10.122.254.0/24      0.0.0.0/0

Chain fp-137 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  *      *       10.0.0.0/8           0.0.0.0/0            udp dpt:161
    0     0 ACCEPT     udp  --  *      *       172.18.217.0/24      0.0.0.0/0            udp dpt:161
    0     0 ACCEPT     udp  --  *      *       172.18.242.0/24      0.0.0.0/0            udp dpt:161
    0     0 ACCEPT     udp  --  *      *       10.0.0.0/8           0.0.0.0/0            udp dpt:162
    0     0 ACCEPT     udp  --  *      *       172.18.217.0/24      0.0.0.0/0            udp dpt:162
    0     0 ACCEPT     udp  --  *      *       172.18.242.0/24      0.0.0.0/0            udp dpt:162
```

Note: the “fp-137 & fp-138” listed above is the OOB contract & filters. INB contracts & filters are not programmed since the filtering is applied at the border or services leaf.

Debugging SNMP on APIC

Verify SNMP Traps using "tcpdump"

- Access the APIC as "root" user and use "**tcpdump**" command to verify SNMP Traps are being sent. Use UDP port **162** or any other UDP Ports that are configured for the SNMP trap destinations in the ACI SNMP Monitoring Group. You can use the following "**tcpdump**" commands to check for SNMP Traps on APIC Nodes:
  - tcpdump -i oobmgmt -f port 162
  - tcpdump -i bond0.1100 -f port 162
  - tcpdump -vvxi oobmgmt udp port 162
  - tcpdump -vvxi bond0.1100 udp port 162

For Example:

APIC (INB)

```
root@rtp1-apic1:~# tcpdump -i bond0.1100 -f port 162
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on bond0.1100, link-type EN10MB (Ethernet), capture size 65535 bytes
20:01:08.453473 IP rtp1-apic1-inb.cisco.com.59417 > 10.117.67.23.snmptrap:  C=deadbeef V2Trap(85)  S:
1.1.4.1.0=E:cisco.9.117.2.0.2 E:cisco.9.117.1.1.2.1.1.10548=1 E:cisco.9.117.1.1.2.1.2.10548=2

20:01:08.459621 IP rtp1-apic1-inb.cisco.com.41098 > 10.117.67.20.snmptrap:  C=deadbeef V2Trap(85)  S:
1.1.4.1.0=E:cisco.9.117.2.0.2 E:cisco.9.117.1.1.2.1.1.10548=1 E:cisco.9.117.1.1.2.1.2.10548=2

20:01:08.483182 IP rtp1-apic1-inb.cisco.com.37798 > 10.150.188.85.snmptrap:  C=deadbeef V2Trap(85)
S:1.1.4.1.0=E:cisco.9.117.2.0.2 E:cisco.9.117.1.1.2.1.1.10548=1 E:cisco.9.117.1.1.2.1.2.10548=2
```

Debugging SNMP on APIC

Verify SNMP Traps using “tcpdump”

For Example:

APIC (INB)

```
root@rtp1-apic1:~# tcpdump -vvxi bond0.1100 udp port 162
tcpdump: listening on bond0.1100, link-type EN10MB (Ethernet), capture size 65535 bytes
20:26:11.756265 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 130)
    rtp1-apic1-inb.cisco.com.58168 > rtp-tdeleon-8816.cisco.com.snmptrap: [bad udp cksum a32a!]  { SNMPv2c C=deadbeef
{ V2Trap(85) R=1203041889  S:1.1.4.1.0=E:cisco.9.117.2.0.2 E:cisco.9.117.1.1.2.1.1.10608=1 E:cisco.
9.117.1.1.2.1.2.10608=2 } }
    0x0000:  4500 0082 0000 4000 4011 4ec1 ac12 f20b
    0x0010:  0a75 4317 e338 00a2 006e ec29 3064 0201
    0x0020:  0104 0864 6561 6462 6565 66a7 5502 0447
    0x0030:  b4f6 6102 0100 0201 0030 4730 1906 0a2b
    0x0040:  0601 0603 0101 0401 0006 0b2b 0601 0401
    0x0050:  0909 7502 0002 3014 060f 2b06 0104 0109
    0x0060:  0975 0101 0201 01d2 7002 0101 3014 060f
    0x0070:  2b06 0104 0109 0975 0101 0201 02d2 7002
    0x0080:  0102

20:26:11.786539 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 130)
    rtp1-apic1-inb.cisco.com.46108 > 10.150.188.85.snmptrap: [bad udp cksum d50!]  { SNMPv2c C=deadbeef { V2Trap(85)
R=1246750429  S:1.1.4.1.0=E:cisco.9.117.2.0.2 E:cisco.9.117.1.1.2.1.1.10608=1 E:cisco.9.117.1.1.2.1.2.10608=2 } }
    0x0000:  4500 0082 0000 4000 4011 d561 ac12 f20b
    0x0010:  0a96 bc55 b41c 00a2 006e 6589 3064 0201
    0x0020:  0104 0864 6561 6462 6565 66a7 5502 044a
    0x0030:  4fe6 dd02 0100 0201 0030 4730 1906 0a2b
    0x0040:  0601 0603 0101 0401 0006 0b2b 0601 0401
    0x0050:  0909 7502 0002 3014 060f 2b06 0104 0109
    0x0060:  0975 0101 0201 01d2 7002 0101 3014 060f
    0x0070:  2b06 0104 0109 0975 0101 0201 02d2 7002
    0x0080:  0102
```

Debugging SNMP on APIC

Verify SNMP Traps using “tcpdump”

For Example:

APIC (OOB)

```
root@rtp1-apic1:~# tcpdump -i oobmgmt -f port 162
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on oobmgmt, link-type EN10MB (Ethernet), capture size 65535 bytes
20:06:08.559990 IP apic1-fab-1.cisco.com.57694 > kilimanjaro.cisco.com.snmptrap:  C=deadbeef V2Trap(85)  S:
1.1.4.1.0=E:cisco.9.117.2.0.2 E:cisco.9.117.1.1.2.1.1.10608=1 E:cisco.9.117.1.1.2.1.2.10608=1

20:06:08.596196 IP apic1-fab-1.cisco.com.42450 > kilimanjaro.cisco.com.snmptrap:  C=deadbeef V2Trap(85)  S:
1.1.4.1.0=E:cisco.9.117.2.0.2 E:cisco.9.117.1.1.2.1.1.10608=1 E:cisco.9.117.1.1.2.1.2.10608=1
```

APIC (OOB)

```
root@rtp1-apic1:~# tcpdump -vvxi oobmgmt udp port 162
tcpdump: listening on oobmgmt, link-type EN10MB (Ethernet), capture size 65535 bytes
20:26:11.780472 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 130)
    apic1-fab-1.cisco.com.33444 > kilimanjaro.cisco.com.snmptrap: [bad udp cksum f3f8!]  { SNMPv2c C=deadbeef
{ V2Trap(85) R=2045233523  S:1.1.4.1.0=E:cisco.9.117.2.0.2 E:cisco.9.117.1.1.2.1.1.10608=1 E:cisco.
9.117.1.1.2.1.2.10608=2 } }
    0x0000:  4500 0082 0000 4000 4011 2822 0a7a fed3
    0x0010:  0a7a fe81 82a4 00a2 006e 12c9 3064 0201
    0x0020:  0104 0864 6561 6462 6565 66a7 5502 0479
    0x0030:  e7c9 7302 0100 0201 0030 4730 1906 0a2b
    0x0040:  0601 0603 0101 0401 0006 0b2b 0601 0401
    0x0050:  0909 7502 0002 3014 060f 2b06 0104 0109
    0x0060:  0975 0101 0201 01d2 7002 0101 3014 060f
    0x0070:  2b06 0104 0109 0975 0101 0201 02d2 7002
    0x0080:  0102
```

Debugging SNMP on APIC

Verify SNMP GET & WALK requests using “tcpdump”

- Access the APIC as "root" user and use "**tcpdump**" command to verify SNMP GET & WALK requests are being received and responded to. Use UDP port **161** to monitor for SNMP GET & WALK requests. You can use the following "**tcpdump**" commands to check for SNMP GET & WALK requests on APICs:
  - tcpdump -i oobmgmt -f port 161
  - tcpdump -i bond0.1100 -f port 161
  - tcpdump -vvxi oobmgmt udp port 161
  - tcpdump -vvxi bond0.1100 udp port 161

Note: Refer to the earlier “iptables” material to verify and check that SNMP Client Group Policies are configured and deployed correctly.

<u>For Example:</u>

Sample SNMP GET Requests:

```
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::sysDescr.0
snmpget -v2c -c deadbeef w.x.y.z SNMPv2-MIB::sysDescr.0
```

Where “deadbeef” is the community string; and “a.b.c.d” is the IP address for APIC OOB mgmt interface and “w.x.y.z” is the IP address for APIC In-Band mgmt interface.

Debugging SNMP on APIC

Verify SNMP GET & WALK requests using “tcpdump”

For Example:

APIC (INB)

```
deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.11 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial
FCH1745V13S


root@rtp1-apic1:~# tcpdump -i bond0.1100 -f port 161
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on bond0.1100, link-type EN10MB (Ethernet), capture size 65535 bytes
20:35:51.012108 IP 10.150.188.85.57139 > rtp1-apic1-inb.cisco.com.snmp:  C=deadbeef
GetRequest(28)  system.sysDescr.0

20:35:51.012359 IP rtp1-apic1-inb.cisco.com.snmp > 10.150.188.85.57139:  C=deadbeef
GetResponse(88)  system.sysDescr.0="APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial
FCH1745V13S"
```

Debugging SNMP on APIC

Verify SNMP GET & WALK requests using “tcpdump”

For Example:

```
APIC (INB)

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.11 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial FCH1745V13S

root@rtp1-apic1:~# tcpdump -vvxi bond0.1100 udp port 161
tcpdump: listening on bond0.1100, link-type EN10MB (Ethernet), capture size 65535 bytes
20:37:38.846222 IP (tos 0x0, ttl 51, id 41366, offset 0, flags [none], proto UDP (17), length 73)
    10.150.188.85.63367 > rtp1-apic1-inb.cisco.com.snmp: [udp sum ok]  { SNMPv2c C=deadbeef { GetRequest(28)
R=32463683  system.sysDescr.0 } }
    0x0000:  4500 0049 a196 0000 3311 8104 0a96 bc55
    0x0010:  ac12 f20b f787 00a1 0035 4ff8 302b 0201
    0x0020:  0104 0864 6561 6462 6565 66a0 1c02 0401
    0x0030:  ef5b 4302 0100 0201 0030 0e30 0c06 082b
    0x0040:  0601 0201 0101 0005 00

20:37:38.846458 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 133)
    rtp1-apic1-inb.cisco.com.snmp > 10.150.188.85.63367: [bad udp cksum d324!]  { SNMPv2c C=deadbeef
{ GetResponse(88) R=32463683  system.sysDescr.0="APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial FCH1745V13S" } }
    0x0000:  4500 0085 0000 4000 4011 d55e ac12 f20b
    0x0010:  0a96 bc55 00a1 f787 0071 658c 3067 0201
    0x0020:  0104 0864 6561 6462 6565 66a2 5802 0401
    0x0030:  ef5b 4302 0100 0201 0030 4a30 4806 082b
    0x0040:  0601 0201 0101 0004 3c41 5049 4320 5645
    0x0050:  5253 494f 4e20 312e 3228 3363 293b 2050
    0x0060:  4944 2041 5049 432d 5345 5256 4552 2d4c
    0x0070:  313b 2053 6572 6961 6c20 4643 4831 3734
    0x0080:  3556 3133 53
```

Debugging SNMP on APIC

Verify SNMP GET & WALK requests using “tcpdump”

For Example:

APIC (OOB)

```
deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 10.122.254.211 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial
FCH1745V13S


root@rtp1-apic1:~# tcpdump -i oobmgmt -f port 161
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on oobmgmt, link-type EN10MB (Ethernet), capture size 65535 bytes
20:35:16.569148 IP 10.150.188.85.54795 > apic1-fab-1.cisco.com.snmp:  C=deadbeef
GetRequest(28)  system.sysDescr.0

20:35:16.569395 IP apic1-fab-1.cisco.com.snmp > 10.150.188.85.54795:  C=deadbeef
GetResponse(88)  system.sysDescr.0="APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial
FCH1745V13S"
```

Debugging SNMP on APIC

Verify SNMP GET & WALK requests using "tcpdump"

For Example:

```
APIC (OOB)

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 10.122.254.211 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial FCH1745V13S

root@rtp1-apic1:~# tcpdump -vvxi oobmgmt udp port 161
tcpdump: listening on oobmgmt, link-type EN10MB (Ethernet), capture size 65535 bytes
20:37:14.005150 IP (tos 0x0, ttl 54, id 28505, offset 0, flags [none], proto UDP (17), length 73)
    10.150.188.85.65485 > apic1-fab-1.cisco.com.snmp: [udp sum ok]  { SNMPv2c C=deadbeef { GetRequest(28)
R=239512186  system.sysDescr.0 } }
    0x0000:  4500 0049 6f59 0000 3611 4512 0a96 bc55
    0x0010:  0a7a fed3 ffcd 00a1 0035 4e27 302b 0201
    0x0020:  0104 0864 6561 6462 6565 66a0 1c02 040e
    0x0030:  46aa 7a02 0100 0201 0030 0e30 0c06 082b
    0x0040:  0601 0201 0101 0005 00

20:37:14.005334 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 133)
    apic1-fab-1.cisco.com.snmp > 10.150.188.85.65485: [bad udp cksum d2b7!]  { SNMPv2c C=deadbeef { GetResponse(88)
R=239512186  system.sysDescr.0="APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial FCH1745V13S" } }
    0x0000:  4500 0085 0000 4000 4011 6a2f 0a7a fed3
    0x0010:  0a96 bc55 00a1 ffcd 0071 d0bb 3067 0201
    0x0020:  0104 0864 6561 6462 6565 66a2 5802 040e
    0x0030:  46aa 7a02 0100 0201 0030 4a30 4806 082b
    0x0040:  0601 0201 0101 0004 3c41 5049 4320 5645
    0x0050:  5253 494f 4e20 312e 3228 3363 293b 2050
    0x0060:  4944 2041 5049 432d 5345 5256 4552 2d4c
    0x0070:  313b 2053 6572 6961 6c20 4643 4831 3734
    0x0080:  3556 3133 53
```

Debugging SNMP on APIC

Sample SNMP requests (OSX, Windows, & linux)

From MAC OSX Client:

```
deadbeef(osx):~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: APIC VERSION 1.2(2h); PID APIC-SERVER-L1; Serial FCH1806V0K2
```

From linux CentOS Client:

```
[root@deadbeef(linux)]# snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::sysDescr.0
No log handling enabled - turning on stderr logging
Created directory: /var/lib/net-snmp/mib_indexes
SNMPv2-MIB::sysDescr.0 = STRING: APIC VERSION 1.2(2h); PID APIC-SERVER-L1; Serial FCH1806V0K2
```

From Windows Client:
https://www.snmpsoft.com/cmd-tools/

```
C:\snmp-cli>SnmpGet.exe -r:172.18.242.13 -v:2c -c:"deadbeef" -o:.1.3.6.1.2.1.1.1.0
SnmpGet v1.01 - Copyright (C) 2009 SnmpSoft Company
[ More useful network tools on http://www.snmpsoft.com ]

OID=.1.3.6.1.2.1.1.1.0
Type=OctetString
Value=APIC VERSION 1.2(2h); PID APIC-SERVER-L1; Serial FCH1806V0K2
```

Debugging SNMP on APIC

Addition samples of useful SNMP requests (OSX)

From MAC OSX Client:

```
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::sysDescr.0
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::sysName.0
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::snmpOutTraps.0
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::snmpInGetRequests.0
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::snmpInGetNexts.0
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::snmpOutGetResponses.0
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::snmpInPkts.0
snmpget -v2c -c deadbeef a.b.c.d SNMPv2-MIB::snmpOutPkts.0
```

where a.b.c.d is the ip address of APIC.

```
deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: APIC VERSION 1.2(3c); PID APIC-SERVER-L1; Serial FCH1806V0K2

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::sysName.0
SNMPv2-MIB::sysName.0 = STRING: rtp1-apic3

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::snmpOutTraps.0
SNMPv2-MIB::snmpOutTraps.0 = Counter32: 0

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::snmpInGetRequests.0
SNMPv2-MIB::snmpInGetRequests.0 = Counter32: 4

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::snmpInGetNexts.0
SNMPv2-MIB::snmpInGetNexts.0 = Counter32: 0

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::snmpOutGetResponses.0
SNMPv2-MIB::snmpOutGetResponses.0 = Counter32: 5

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::snmpInPkts.0
SNMPv2-MIB::snmpInPkts.0 = Counter32: 7

deadbeef:~ tdeleon$ snmpget -v2c -c deadbeef 172.18.242.13 SNMPv2-MIB::snmpOutPkts.0
SNMPv2-MIB::snmpOutPkts.0 = Counter32: 7
```

Debugging SNMP on APIC

Verify SNMP Requests & Traps using "strace"

- Access the APIC as "root" user and use "**strace**" command to trace the SNMP process for SNMP Requests & SNMP Traps on the APICs. You can use the following "**strace**" commands to trace the SNMP process on APICs:
  - strace -p 32182 -f -e trace=network -s 10000
  - strace -p 32182 -o snmpd_trace.txt

    where 32182 is PID of SNMP process

For Example:

```
APIC (PROCESS)

root@rtp1-apic1:~# strace -p 32182 -f -e trace=network -s 10000
Process 32182 attached with 13 threads - interrupt to quit

[pid 32182] recvmsg(5, {msg_name(16)={sa_family=AF_INET, sin_port=htons(59923),
sin_addr=inet_addr("10.150.188.85")}, msg_iov(1)=[{"0+
\2\1\1\4\10deadbeef\240\34\2\4\30\223Z\f\2\1\0\2\1\0000\0160\f\6\10+\6\1\2\1\1\1\0\5\0\0\0\0\0\0"...,
65536}], msg_controllen=32, {cmsg_len=28, cmsg_level=SOL_IP, cmsg_type=, ...}, msg_flags=0}, MSG_DONTWAIT)
= 45

[pid 32182] getsockname(5, {sa_family=AF_INET, sin_port=htons(161), sin_addr=inet_addr("0.0.0.0")}, [16]) =
0

[pid 32182] sendmsg(5, {msg_name(16)={sa_family=AF_INET, sin_port=htons(59923),
sin_addr=inet_addr("10.150.188.85")},
msg_iov(1)=[{"0g\2\1\1\4\10deadbeef\242X\2\4\30\223Z\f\2\1\0\2\1\0000J0H\6\10+\6\1\2\1\1\1\0\4<APIC VERSION
1.2(3c); PID APIC-SERVER-L1; Serial FCH1745V13S", 105}], msg_controllen=32, {cmsg_len=28,
cmsg_level=SOL_IP, cmsg_type=, ...}, msg_flags=0}, MSG_DONTWAIT|MSG_NOSIGNAL) = 105
```

Debugging SNMP on APIC

Some Log Files to search when Troubleshooting

- Access the APIC as "admin" user and seach some of the following logs when troubleshooting SNMP Requests to APIC:
  - zgrep "snmptrap" /var/log/dme/log/svc_ifc_eventmgr.bin.log*
  - zgrep "snmp" /var/log/dme/log/svc_ifc_eventmgr.bin.log*

- Additional information can be looked from snmpd logs (**/var/log/dme/log/snmpd.bin.log***).
  The snmpd log-files provide information about MIT-lookups done for generating the snmp table entries.

Note: Some of the above commands may or may not produce output when performed on a APIC. These are just some examples which may point you in the right direction.

For Example:

```
rtp1-apic1# zgrep "snmptrap" /var/log/dme/log/svc_ifc_eventmgr.bin.log*

/var/log/dme/log/svc_ifc_eventmgr.bin.log.184.gz:526||16-04-13 20:01:08.445+00:00||snmp||DBG4||co=doer:
255:127:0xff0000000002b908:1,dn=uni/fabric/snmpgroup-deadbeef-snmpGrp/trapdest-10.117.67.23-port-162,fn=[notify]|| Invoking "snmptrap -v 2c
-c deadbeef 10.117.67.23:162 uptime 1.3.6.1.4.1.9.9.117.2.0.2 1.3.6.1.4.1.9.9.117.1.1.2.1.1.10548 i 1 1.3.6.1.4.1.9.9.117.1.1.2.1.2.10548 i 2" argc=14||../
common/src/events/local/./IfcSnmpTrapNotifier.cc||696

rtp1-apic1# zgrep "snmp" /var/log/dme/log/svc_ifc_eventmgr.bin.log*

/var/log/dme/log/svc_ifc_eventmgr.bin.log.114.gz:526||16-04-12 17:57:36.488+00:00||event_||DBG4||co=doer:28:1:0xe0000000010c86fe:
1,dn=uni/fabric/moncommon/snmpsrc-deadbeef-snmpSrc,fn=[findMonObjDnMo]|| No MonObjDnMo found, dn class: snmpGroup(1692)||../
common/src/events/common/MonObjDnMoHelper.h||32
```

ACI SNMP Caveats - Issues

This section will discuss some known caveats or issues with the SNMP feature in the ACI Solution. A few notable Caveats or Issues are:

ACI SNMP Caveats - Issues - Gotchas

When SNMP is configured correctly for SNMP Traps & SNMP GETs\WALKs, the SNMP feature works as expected. Most of the issues relate to misconfiguration or issues with software programming. The following are some common gotchas that we see and you can use the material in the technote to troubleshoot snmp issues in the ACI Fabric**.**

- In "Brazos", Cisco added SNMP support for the APIC(s). The behavior for default allowed ports for the APIC it is "Different". Unlike the Switches, a **CONTRACT** is needed for the APIC to allow SNMP. This is "NEW" with brazos. In your OOB Contract defined for your External Management Network Instance Profile. Once you add Ports **161 & 162** to the filter of the OOB Contract, your SNMP Gets should work as expected.
- If you are using SNMP ports other than ports **161 & 162,** make sure the non-standard ports are configured in your ACI SNMP configuration.
- Node Management Address(s) in the Tenant mgmt need to be configured for the APIC(s), Leaf(s), and Spine(s). Verify that the Node management address(s) are configured.
- The ACI Devices (APIC(s), Leaf(s), and Spine(s)) **IP addresses for OOB & INB** need to be added to your SNMP AGENT Monitoring Application.
- Check Firewall configuration on the SNMP AGENT Monitoring Application Server.
- "iptables" programming on the ACI devices
- Verify the correct MIBs loaded on the SNMP AGENT Monitoring Application Server.

References & Resources

References and Resources

Reference Links

- **[1] Cisco ACI MIB Quick Reference**
  http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/mib/guide/b_Cisco_ACI_MIB_Quick_Reference/b_Cisco_ACI_MIB_Quick_Reference_chapter_00.html
- **[2] Cisco APIC NX-OS Style Command-Line Interface Configuration Guide**
  http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/cli/nx/cfg/b_APIC_NXOS_CLI_User_Guide.html
- **[3] Cisco APIC NX-OS Style CLI Command Reference**
  http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/cli/nx/cr/b_APIC_NXOS_CLI_Cmd_Reference.html
- **[4] ACI MIB Support List**
  http://www.cisco.com/c/dam/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/mib/list/mib-support.html
- **[5] SNMP Object Navigator**
  http://snmp.cloudapps.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en

VISORE Class or DN

- (snmpPol, snmpClientGrpP, snmpCommunityP, snmpTrapDest, snmpSrc, snmpCtxP)
- (mgmtSubnet, mgmtRsOoBCons, vzOOBBrCP, vzEntry)

References and Resources (cont.)

APIC CLI "Show" Commands

- show snmp
- show snmp policy default
- show snmp summary
- show snmp clientgroups
- show snmp community
- show snmp hosts
- show snmp engineid

LEAF\SPINE CLI "Show" Commands

- show snmp
- show snmp | grep "SNMP packets"
- show snmp summary
- show snmp community
- show snmp host
- show snmp engineid
- show snmp context
- show snmp user
- show snmp internal dump-internal-log
- show snmp internal globals
- show snmp internal trace log

APIC CLI Commands

- show snmp
- show snmp policy default
- show snmp summary
- show snmp clientgroups
- show snmp community
- show snmp hosts
- show snmp engineid

LEAF\SPINE CLI Commands

- show snmp
- show snmp | grep "SNMP packets"
- show snmp summary
- show snmp community
- show snmp host
- show snmp engineid
- show snmp context
- show snmp user
- show snmp internal dump-internal-log
- show snmp internal globals
- show snmp internal trace log

Review Questions

1. Which of the following fuctions of SNMP are supported on the APIC and Leaf & Spine Nodes: (Choose all that apply)

a. SNMPv3
b. SNMP Traps (v1, v2c, and v3)
c. SNMP write commands (Set)
d. SNMP read queries (Get, Next, Bulk, Walk)
e. All of the above are functions of SNMP support for ACI

2. SNMP support was added to APIC controllers in ACI release 1.2 (Brazos). In order for SNMP read queries to work successfully to an APIC, what configuration requirements need to be configured: (Choose all that apply)

a. SNMP on APIC using OOB management EPG requires an explicit "Out-Of- Band Contract" on the APIC for enabling the SNMP port (UDP:161).
b. Configure a SNMP Policy for the ACI Fabric.
c. SNMP on APIC using INB management EPG requires an explicit "In-Band Contract" on the APIC for enabling the SNMP port (UDP:161).
d. Static node management address for the APIC management interfaces.
e. Create SNMP client group policies with SNMP clients.
f. All of the above need to be configured for SNMP read queries to work successfully to an APIC.

Review Questions

3. What SNMP configuration steps are required to be enable SNMP Traps for the APICs and Leaf & Spine nodes in the ACI Fabric: (Choose all that apply)

a. Create SNMP Monitoring Destination Group in the External Data Collectors.
b. Create an Access List for SNMP.
c. Create SNMP Trap Destination.
d. Create an Access List for SNMP Traps.
e. Create a Fabric Monitoring Source for SNMP.
f. Create a SNMP User to be used for the SNMP Agent Role.

4. Which single ACI Fabric Monitoring policy can be configured to use an SNMP Source that will be applied to both fabric and access infrastructure hierarchies:

a. Fabric Policies -> Default Policy "monFabricPol (uni/fabric/monfab-default)"
b. Access Policies -> Default Policy "monInfraPol (uni/infra/monifra-default)"
c. Fabric Policies -> Common Policy "monCommonPol (uni/fabric/moncommon)"
d. Tenant -> EPG Policy "monEPGPol (uni/tn-common/monepg-default)"
e. None of the above are correct

Review Questions

5. Which SNMP MIBs are supported on the APIC: (Choose all that apply)

a. Cisco CDP MIB
b. Cisco Entity Censor MIB
c. Entity MIB
d. Cisco NTP MIB
e. Cisco Entity FRU Control MIB
f. RFC1213 MIB
g. Cisco Process MIB

Review Questions (Answer Key)

1. a, b, d
2. f
3. a, c, e
4. c
5. b, c, e, g