SNMP in ACI
Overview, Configuration, Troubleshooting, and Caveats\Issues
Created by Tomas de Leon (ACI Solutions Delivery Team)

April 13, 2016
Table of Contents

✧ **ACI SNMP Overview**
  - SNMP Basic Components
  - SNMP Support in ACI
  - SNMP Support on APIC

✧ **ACI SNMP Configuration**
  - Configuring the SNMP Feature using the APIC Admin GUI "Advanced Mode"
  - Configuring the ACI Fabric Nodes to send SNMP Traps using the APIC Admin GUI "Advanced Mode"

✧ **Troubleshooting ACI SNMP Configuration**
  - Verify ACI SNMP Configuration using “CLI Show Commands”
  - Verify ACI SNMP Configuration using “moquery”
## Troubleshooting ACI SNMP Configuration (cont.)
- Verify ACI SNMP Configuration using “VISORE”
- Verify ACI SNMP Configuration checking the “Logical Model”
- Verify SNMP GET/WALKS work on the LEAF/SPINE/APIC
- Verify SNMP TRAPS are being sent by the LEAF/SPINE/APIC
- Troubleshooting the ACI SNMP Configuration on the APIC
- Troubleshooting the ACI SNMP Configuration on the LEAF & SPINE nodes

## ACI SNMP Configuration Caveats - Issues

## References & Resources
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](http://www.cisco.com/networkers/nw04/presos/docs/NMS-1N02.pdf)
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.
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 its 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**

<table>
<thead>
<tr>
<th>MIB</th>
<th>Supported Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity MIB</td>
<td>PhysicalTable</td>
</tr>
<tr>
<td>Cisco Entity Ext MIB</td>
<td>PhysicalProcessorTable, LEDTable</td>
</tr>
<tr>
<td>Cisco Entity FRU Control MIB</td>
<td>PowerSupplyGroupTable, PowerStatusTable, FanTrayStatusTable, PhysicalTable</td>
</tr>
<tr>
<td>Cisco Entity Censor MIB</td>
<td>SensorValueTable, SensorThresholdTable</td>
</tr>
<tr>
<td>Cisco Process MIB</td>
<td>CPUTotalTable, ProcessTable, ProcessExtRevTable, ThreadTable</td>
</tr>
</tbody>
</table>
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 built 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:
TASK 1: (cont.)
Configure a SNMP Policy for the ACI Fabric

Sample Screenshot:
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:
TASK 1: (cont.)

Configure a SNMP Policy for the ACI Fabric

Configuration Steps:
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.

   - Make sure the "default" SNMP Policy (or Custom SNMP Policy) is selected and Resolved.
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:
TASK 1: (cont.)
Configure a SNMP Policy for the ACI Fabric

Example of POSTs from the API Inspector:

method: POST

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

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

Example of POSTs from the API Inspector: (cont.)

- method: POST

  payload:

  ```json
  {"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:
TASK 2:
Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

- Verify In-Band Contract:
TASK 2: Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

- Verify Contract Filter:
TASK 2:
Configure MGMT Contracts to allow UDP Port 161 for SNMP Requests

- Verify Node Management Addresses Configured:
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:
TASK 3:
Configure the ACI Fabric to send SNMP TRAPS

Sample Screenshots:
TASK 3: Configure the ACI Fabric to send SNMP TRAPS

Sample Screenshots:
TASK 3: Configure the ACI Fabric to send SNMP TRAPS

- **Example of POST from the API Inspector:**

  method: POST

  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 Monitoring Sources that need to be configured.

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

TASK 3: Configure the ACI Fabric to send SNMP TRAPS

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

TASK 3: Configure the ACI Fabric to send SNMP TRAPS

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

<table>
<thead>
<tr>
<th>APIC CLI COMMANDS</th>
<th>LEAF\SPINE CLI COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>show snmp</td>
<td>show snmp</td>
</tr>
<tr>
<td>show snmp summary</td>
<td>show snmp summary</td>
</tr>
<tr>
<td>show snmp clientgroups</td>
<td>show snmp community</td>
</tr>
<tr>
<td>show snmp community</td>
<td>show snmp community</td>
</tr>
<tr>
<td>show snmp hosts</td>
<td>show snmp host</td>
</tr>
</tbody>
</table>
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] 0x8000000980e2b692088976c7560000000

<table>
<thead>
<tr>
<th>Community</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deadbeef</td>
<td>SNMP Community String</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User</th>
<th>Authentication</th>
<th>Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>deadbeef-snmpClients</td>
<td>default (In-Band)</td>
<td>10.150.44.141, 10.117.67.20, 10.150.188.104</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Host</th>
<th>Port</th>
<th>Version</th>
<th>Level</th>
<th>SecName</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.117.67.20</td>
<td>162</td>
<td>v2c</td>
<td>noauth</td>
<td>deadbeef</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Community</th>
<th>Context</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>deadbeef</td>
<td></td>
<td>ok</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Client</th>
<th>VRF</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.150.188.104</td>
<td>mgmt:inb</td>
<td>ok</td>
</tr>
<tr>
<td>10.150.44.141</td>
<td>mgmt:inb</td>
<td>ok</td>
</tr>
<tr>
<td>10.117.67.20</td>
<td>mgmt:inb</td>
<td>ok</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Host</th>
<th>Port</th>
<th>Ver</th>
<th>Level</th>
<th>SecName</th>
<th>VRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.117.67.20</td>
<td>162</td>
<td>v2c</td>
<td>noauth</td>
<td>deadbeef</td>
<td>mgmt:inb</td>
</tr>
</tbody>
</table>
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
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).

```
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
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.
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).

```
moquery -c snmpCommunityP
```

```
Total Objects shown: 1

# snmp.CommunityP
ame : 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.
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).

```
leaf1# moquery -c snmpTrapDest
# 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"
```

```
  name : deadbeef-snmpSrc
  dn   : uni/infra/moninfra-default/snmpsrc-deadbeef-snmpSrc
  incl : events,faults
  minSev : info
  monPolDn : uni/infra/moninfra-default
```

```
  name : deadbeef-snmpSrc
  dn   : uni/fabric/monfab-default/snmpsrc-deadbeef-snmpSrc
  incl : events,faults
  minSev : info
  monPolDn : uni/fabric/monfab-default
```

```
  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.
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
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.

**snmpClientGrp** - 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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminSt</td>
<td>enabled</td>
</tr>
<tr>
<td>childAction</td>
<td></td>
</tr>
<tr>
<td>contact</td>
<td>Sir deadbeef</td>
</tr>
<tr>
<td>descr</td>
<td>SNMP Policy for the RTP2 Fabric</td>
</tr>
<tr>
<td>dn</td>
<td>uni/fabric/snmppol-default</td>
</tr>
<tr>
<td>lcOwn</td>
<td>local</td>
</tr>
<tr>
<td>loc</td>
<td>Cisco Systems, North Carolina</td>
</tr>
<tr>
<td>modTs</td>
<td>2016-03-07T15:43:06.559+00:00</td>
</tr>
<tr>
<td>monPolDn</td>
<td>uni/fabric/monfab-default</td>
</tr>
<tr>
<td>name</td>
<td>default</td>
</tr>
</tbody>
</table>
Verify ACI SNMP Configuration (cont.)

“VISORE”

<table>
<thead>
<tr>
<th><strong>Class or DN:</strong></th>
<th>snmpClientGrpP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Op:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Val1:</strong></td>
<td></td>
</tr>
</tbody>
</table>

**APIC Object Store Browser**

<table>
<thead>
<tr>
<th><strong>childAction</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>descr</strong></td>
<td>SNMP Clients that can perform SNMPGET and SNMPWALK requests</td>
</tr>
<tr>
<td><strong>dn</strong></td>
<td>uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients</td>
</tr>
<tr>
<td><strong>epgDn</strong></td>
<td>uni/tn-mgmt/mgmtp-default/inb-default</td>
</tr>
<tr>
<td><strong>lcOwn</strong></td>
<td>local</td>
</tr>
<tr>
<td><strong>modTs</strong></td>
<td>2016-03-07T15:43:06.595+00:00</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>deadbeef-snmpClients</td>
</tr>
</tbody>
</table>
## Verify ACI SNMP Configuration (cont.)

### “VISORE”

<table>
<thead>
<tr>
<th>Class or DN:</th>
<th>snmpClientP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property:</td>
<td></td>
</tr>
<tr>
<td>Op:</td>
<td>==</td>
</tr>
<tr>
<td>Val1:</td>
<td></td>
</tr>
</tbody>
</table>

### snmpClientP

<table>
<thead>
<tr>
<th>addr</th>
<th>10.150.188.104</th>
</tr>
</thead>
<tbody>
<tr>
<td>dn</td>
<td>10.150.44.141</td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
</tbody>
</table>

### snmpClientP

<table>
<thead>
<tr>
<th>addr</th>
<th>10.117.67.20</th>
</tr>
</thead>
<tbody>
<tr>
<td>dn</td>
<td>uni/fabric/snmppol-default/clgrp-deadbeef-snmpClients/client-[10.117.67.20]</td>
</tr>
<tr>
<td>lcOwn</td>
<td>local</td>
</tr>
<tr>
<td>modTs</td>
<td>2016-03-08T15:08:16.481+00:00</td>
</tr>
<tr>
<td>name</td>
<td>deadbeef-osx3</td>
</tr>
</tbody>
</table>
Verify ACI SNMP Configuration (cont.)

“VISORE”

**snmpCommunityP**

![APIC Object Store Browser](image)

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>descr</td>
<td>SNMP Community String</td>
</tr>
<tr>
<td>dn</td>
<td>uni/fabric/snmppol-default/community-deadbeef</td>
</tr>
<tr>
<td>lcOwn</td>
<td>local</td>
</tr>
<tr>
<td>modTs</td>
<td>2016-03-07T15:45:50.186+00:00</td>
</tr>
<tr>
<td>name</td>
<td>deadbeef</td>
</tr>
</tbody>
</table>
### snmpTrapDest

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>childAction</td>
<td></td>
</tr>
<tr>
<td>descr</td>
<td></td>
</tr>
<tr>
<td>dn</td>
<td>uni/fabric/snmpgroup-deadbeef-snmpDestGrp/trapdest-10.117.67.20-port-162</td>
</tr>
<tr>
<td>cpgDn</td>
<td>uni/ml-mgmt/mgmtp-default/intl-default</td>
</tr>
<tr>
<td>host</td>
<td>10.117.67.20</td>
</tr>
<tr>
<td>leOwn</td>
<td>local</td>
</tr>
<tr>
<td>modTs</td>
<td>2016-03-08T15:31:39.857+00:00</td>
</tr>
<tr>
<td>monPolDn</td>
<td>uni/fabric/morfab-default</td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>notifT</td>
<td>traps</td>
</tr>
<tr>
<td>port</td>
<td>162</td>
</tr>
<tr>
<td>secName</td>
<td>deadbeef</td>
</tr>
<tr>
<td>status</td>
<td></td>
</tr>
<tr>
<td>uid</td>
<td>15374</td>
</tr>
<tr>
<td>v3SecLvl</td>
<td>noauth</td>
</tr>
<tr>
<td>ver</td>
<td>v2c</td>
</tr>
</tbody>
</table>
Verify ACI SNMP Configuration (cont.)

“VISORE”

<table>
<thead>
<tr>
<th>Class or DN:</th>
<th>snmpSrc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property:</td>
<td></td>
</tr>
<tr>
<td>Op:</td>
<td>==</td>
</tr>
<tr>
<td>Val1:</td>
<td></td>
</tr>
</tbody>
</table>

**snmpSrc**

<table>
<thead>
<tr>
<th>childAction</th>
<th>descr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dn</th>
<th>uni/fabric/monfab-default/snmpsrec-deadbeef-snmpSrc</th>
</tr>
</thead>
</table>

**snmpSrc**

<table>
<thead>
<tr>
<th>childAction</th>
<th>descr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dn</th>
<th>uni/fabric/moncommon/snmpsrec-deadbeef-snmpSrc</th>
</tr>
</thead>
</table>

**incl**

<table>
<thead>
<tr>
<th>events,faults</th>
</tr>
</thead>
</table>

**lcOwn**

<table>
<thead>
<tr>
<th>local</th>
</tr>
</thead>
</table>

**minSev**

<table>
<thead>
<tr>
<th>info</th>
</tr>
</thead>
</table>

**modTs**

<table>
<thead>
<tr>
<th>2016-03-08T19:39:45.631+00:00</th>
</tr>
</thead>
</table>

**monPolDn**

<table>
<thead>
<tr>
<th>uni/infra/moninfra-default</th>
</tr>
</thead>
</table>

**name**

<table>
<thead>
<tr>
<th>deadbeef-snmpSrc</th>
</tr>
</thead>
</table>

**status**

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
"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/fabric/fabric-policies/pod-policies/policy-groups/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:

- Get ROOT Password from the INSBU Tool.
  - 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 INSBU Tool
    http://git.insieme.local/cgi-bin/generateRootPassword.py
  - access root (i.e. ssh root@localhost). Use the root password string from the tool for the password.

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

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

```
rtp1-leaf1# ssh root@localhost
Welcome to RTP Fabric 1!
Password: MEQCIEZQIaqXLMnkvUZCnn16yyqhrNG7HF7wPZJnw2KN68BAiBL61HLTqvX7Hb0Npz0CXVLgn0zqqPTjMo5f0I+Ik1Pyg==
rtp1-leaf1# whoami
root
```
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)

```
root  5881  2.5 1907404 411444 ?   5 Sl  Apr05 496:35 /isan/bin/snmpd -f -s
-d udp:161 udp6:161 tcp:161
```

```
rtp1-leaf1# ps aux | grep snmp
root  5881  2.5 1907404 411444 ?   5 Sl  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.
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 & receiving 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)

```
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 :::snmp *:* LISTEN
udp 0 0 *:snmp :::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 “iptables” 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 “iptables” 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)

```markdown
rtp1-leaf1# show vrf
  VRF-Name       VRF-ID State Reason
management      2       Up    --
mgmt:inb        9       Up    --
```
Debugging SNMP on LEAF/SPINE

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

rtpl-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  32771  udp dpt:snmp-trap
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 spt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere            src-class-id  49153  udp spt:snmp

Chain vrf_9_snmp_rules (1 references)
ACCEPT     udp  --  anywhere             anywhere            src-class-id  32771  udp spt:snmp-trap
ACCEPT     udp  --  anywhere             anywhere            src-class-id  49153  udp spt:snmp
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
```
### Debugging SNMP on LEAF/SPINE

#### “iptables”

For Example: (cont.)

```bash
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

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
   0     0 ACCEPT     all --  *      *       0.0.0.0/0

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
```
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 } } 17:31:45.504683 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**
```
deallelon$ 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**

```
deadeeef:~ 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  

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  

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\0\0\0\0\0\0\0\0\0\0\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\0\0\0\0\0\0\0\0\0\0\f\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 search 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_dbgrem.log*`
- `zgrep "snmpd" /var/log/dme/log/svc_ifc_dbgrem.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/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="0" lcOwn="local" moDn="uni/fabric/snmpgroup-deadbeef-snmpGrp/trapdest-10.117.67.23-port-162" modTs="never" priKey="1691:23053" rn="" status=""/>

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:

- Get ROOT Password from the INSBU Tool.
  - ssh to APIC 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 INSBU Tool
    http://git.insieme.local/cgi-bin/generateRootPassword.py
  - access root (i.e. ssh root@localhost). Use the root password string from the tool for the password.

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

rtp1-apic1# acidiag dbgtoken
0JRYAZYKMCHP

rtp1-apic1# ssh root@localhost
Welcome to RTP Fabric 1!
Password: MEQCIEZQIaqXLMnkvmUZCnn16yyqhrNG7HF7wPZJnw2KN68BAiBL61HLTgvX7Hb0Npz0CXVLgn0zqqPTjMo5f0I+IklPYg==
rtp1-apic1# whoami
root
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`

**For Example:**
(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
```

* 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 & receiving 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

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Flags</th>
<th>MSS</th>
<th>Window</th>
<th>irtt</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0</td>
<td>172.18.242.1</td>
<td>0.0.0.0</td>
<td>UG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>bond0.1100</td>
</tr>
<tr>
<td>0.0.0.0</td>
<td>10.122.254.1</td>
<td>0.0.0.0</td>
<td>UG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>oobmgmt</td>
</tr>
<tr>
<td>10.122.254.0</td>
<td>0.0.0.0</td>
<td>255.255.255.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>oobmgmt</td>
</tr>
<tr>
<td>172.18.242.0</td>
<td>0.0.0.0</td>
<td>255.255.255.192</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>bond0.1100</td>
</tr>
<tr>
<td>172.18.242.1</td>
<td>0.0.0.0</td>
<td>255.255.255.255</td>
<td>UH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>bond0.1100</td>
</tr>
</tbody>
</table>

root@rtp1-apic1:~# netstat -lutn | grep 161
Active Internet connections (only servers)

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>udp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0.0:161</td>
<td>0.0.0.0:*</td>
<td></td>
</tr>
<tr>
<td>udp</td>
<td>0</td>
<td>0</td>
<td>::::161</td>
<td>:::::*</td>
<td></td>
</tr>
</tbody>
</table>

rtp1-apic1# netstat -aon | grep ":161"
Active Internet connections (servers and established)

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
<th>Timer</th>
</tr>
</thead>
<tbody>
<tr>
<td>udp</td>
<td>0</td>
<td>0</td>
<td>0.0.0.0.0:161</td>
<td>0.0.0.0:*</td>
<td></td>
<td>off (0.00/0/0)</td>
</tr>
<tr>
<td>udp</td>
<td>0</td>
<td>0</td>
<td>::::161</td>
<td>:::::*</td>
<td></td>
<td>off (0.00/0/0)</td>
</tr>
</tbody>
</table>

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 “iptables” rules is crucial to the success of the SNMP configuration and deployment to APICs. You can use the following commands to check the “iptables” 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)

<table>
<thead>
<tr>
<th>rtp1-apic1# show snmp clientgroups</th>
<th>Description</th>
<th>Client Entries</th>
<th>Associated Management EPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP Policy</td>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>snmpClients-oob</td>
<td>List of SNMP Clients for Fabric1</td>
<td>10.117.67.21,10.117.67.20,10.117.67.23,10.117.67.22,10.117.67.24,10.150.46.242,10.150.188.85</td>
</tr>
<tr>
<td>default</td>
<td>snmpClients-inb</td>
<td>List of SNMP Clients for Fabric1</td>
<td>10.117.67.21,10.117.67.20,10.117.67.23,10.117.67.22,10.117.67.24,10.150.46.242,10.150.188.85</td>
</tr>
</tbody>
</table>

```
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
```
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     10.122.254.0/24      anywhere
53    7093 fp-138             all  --  any    any     10.122.254.0/24      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
     0     0 ACCEPT     udp  -- *      *       172.18.217.0/24      0.0.0.0/0
     0     0 ACCEPT     udp  -- *      *       172.18.242.0/24      0.0.0.0/0
     0     0 ACCEPT     udp  -- *      *       10.0.0.0/8           0.0.0.0/0
     0     0 ACCEPT     udp  -- *      *       172.18.217.0/24      0.0.0.0/0
     0     0 ACCEPT     udp  -- *      *       172.18.242.0/24      0.0.0.0/0

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)**

```
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
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 }

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=123456789012 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 }

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 } }
Debugging SNMP on APIC

Verify SNMP GET & WALK requests using “tcpdump”

 Fach 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 -v -vvxi oobmgmt udp port 161
- tcpdump -v -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.

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 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 } }
0 x 0 0 0 : 4500 0049 a196 0000 3311 8104 0a96 bc55
0 x 0 0 10 : ac12 f20b f787 00a1 0035 4ff8 302b 0201
0 x 0 0 20 : 0104 0064 6561 6462 6565 66a0 1c02 0401
0 x 0 0 30 : ef5b 4302 0100 0201 0030 0e30 0c06 082b
0 x 0 0 40 : 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" } }
0 x 0 0 00 : 4500 0085 0000 0000 4011 4011 d55e ac12 f20b
0 x 0 0 10 : 0a96 bc55 00a1 f787 0071 658c 3067 0201
0 x 0 0 20 : 0104 0064 6561 6462 6565 66a0 5802 0401
0 x 0 0 30 : ef5b 4302 0100 0201 0030 4a30 4806 082b
0 x 0 0 40 : 0601 0201 0101 0004 3c41 5049 4320 5645
0 x 0 0 50 : 5253 494f 4e20 312e 3228 3363 293b 2050
0 x 0 0 60 : 4944 2041 5049 432d 5345 5256 4552 2d4c
0 x 0 0 70 : 313b 2053 6972 6961 6c20 4643 4831 3734
0 x 0 0 80 : 3555 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 0064 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 0000 4011 6a2f 0a7a fed3
0x0010: 0a96 bc55 00a1 ffcd 0071 d0bb 3067 0201
0x0020: 0104 0064 6561 6462 6565 66a2 5802 040e
0x0030: 46aa 7a02 0100 0201 0030 3c41 5049 4320 5645
0x0040: 5253 494f 4e20 312e 3228 3363 293b 2050
0x0050: 4944 2041 5049 432d 5345 5256 4552 2d4c
0x0060: 313b 2053 6572 6961 6c20 4643 4831 3734
0x0070: 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

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.

```
deadeef:~ 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

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

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

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

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

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

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

deadeef:~ 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\f6\10\6\1\1\1\0\5\0\0\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 search 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:255:127:0xff000000002b908:1,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:281:0xe0000000010ce86fe:1,dn=uni/fabric/moncommon/snmpsrec-deadbeef-snmpSrc,fn=[findMonObjDnMo] | No MonObjDnMo found, dn class: snmpGroup(1692) | ../common/src/events/common/MonObjDnMoHelper.h|32
```
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


[3] Cisco APIC NX-OS Style CLI Command Reference


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