Monitoring Cisco AireOS WLC via SNMP using OIDs

Contents

Introduction
Prerequisites
Requirements
Components used
Configuring SNMP settings on WLC
Object names and object IDs (OIDs)
What are object names and OIDs?
MIBs and list of all Object names and IDs on Cisco WLCs
Using OIDs to monitor the state of WLC
Monitoring via snmpwalk
Monitoring via Python 3 and pysnmp library
Integration with third party software (Grafana/PRTG Network Monitor/SolarWinds)
Table of most commonly monitored OIDs

Introduction

This article will explain how to configure SNMP monitoring on Cisco 3504 Wireless LAN Controller (WLC), translate object names into object identifiers and vice versa, as well as provide the list of OIDs most commonly used by Cisco customers.

Prerequisites

Requirements

Having a default snmp tool on your operating system or installing one.

Components used

All tests were performed on a 3504 WLC running image version 8.9 and MacOS 10.14. OIDs in this article are also valid on older AireOS releases and other AireOS based wireless controllers (8540/5508/5520/2504).

Configuring SNMP settings on WLC

SNMPv2c is a community-based version of SNMP and all communication between the devices is in clear text. SNMPv3 is the most secure version which offers message integrity checks, authentication and encryption of the packets. SNMPv1 is extremely outdated, but still exists to provide legacy software compatibility.
Important: SNMPv2c is enabled by default with community “private” having read+write privileges and community “public” having read only privileges. Its recommended to remove them and create a new community with a different name.

In this article, only SNMPv2c and SNMPv3 will be used. Log into the web interface of the controller. Under Management->SNMP->General make sure to enable the desired version of protocol.

<table>
<thead>
<tr>
<th>Management</th>
<th>SNMP System Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td>Contact</td>
</tr>
<tr>
<td></td>
<td>System Description</td>
</tr>
<tr>
<td></td>
<td>System Object ID</td>
</tr>
<tr>
<td></td>
<td>SNMP Port Number</td>
</tr>
<tr>
<td></td>
<td>Trap Port Number</td>
</tr>
<tr>
<td></td>
<td>SNMP v1 Mode</td>
</tr>
<tr>
<td></td>
<td>SNMP v2c Mode</td>
</tr>
<tr>
<td></td>
<td>SNMP v3 Mode</td>
</tr>
</tbody>
</table>

Under communities menu, all currently created communities will be displayed.

<table>
<thead>
<tr>
<th>Management</th>
<th>SNMP v1 / v2c Community</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Community Name</td>
</tr>
<tr>
<td></td>
<td>public</td>
</tr>
<tr>
<td></td>
<td>********</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Management</th>
<th>IPSec Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IPSec</td>
</tr>
</tbody>
</table>

It is best practice to remove default pre-configured communities and create a new one. IP address and netmask will behave like an access list. By default, both will be set to 0.0.0.0, meaning all IP addresses will be allowed to make SNMP queries for this community. Access mode field will be left as "Read Only" as we want this community to be used only for monitoring, and not configuration of the WLC.

Important: All versions lower than 8.7.1.135 will be affected with a bug CSCvg61933 where netmask will not be allowed to be set to 255.255.255.255. Either upgrade the controller to the latest recommended release higher than 8.7.1.135 or use the following command in the CLI to create a new community: `config snmp community ipaddr <ip_address> <netmask> <community_name>`
Under the SNMP V3 Users menu, you can see all the configured users, their privileges and protocols used for authentication and encryption. Button **New** allows a creation of a new user. Selecting HMAC-SHA as authentication protocol and CFB-AES-128 as privacy protocol is recommended. We will create a user named 'admin' with authentication and privacy password set to "Cisco123Cisco123":

**Object names and object IDs (OIDs)**

**What are object names and OIDs?**

Object IDs, or OIDs for short, are unique identifiers that represent a certain variable or object. For example, the current CPU usage is considered variable which values can be retrieved by calling upon their object ID. Each OID is unique and no two should be the same across the world, quite similar to a MAC address. These identifiers follow a tree hierarchy, and each OID can be tracked down back to its root. Each vendor will have it's own branch after a common root.

An analogy could be a home address, where the root would be the country or state, followed by a city zip code, the street, and finally the home number.

The numbers followed by a dot represent each step it takes to get to a certain point in that tree or branch.
All these values are stored in a Management Information Base, or MIB for short, in each network device. Each identifier has name and definition (range of possible values, type, etc...).

Loading MIBs on your snmp tool is not required in order to use SNMP and to query a device, as longs as a valid OID is known the device will respond with the value that is stored in the variable that the OID represents. For example, in the image below, the SNMP manager is querying the SNMP agent of a device for its system description using the OID 1.3.6.1.2.1.1.1.0.

However, loading the MIB on your query tool allows to benefit from translating the OID numbers to names and knowing their definition.

**MIBs and list of all Object names and IDs on Cisco WLCs**

As of May 2019, a simple, user-friendly table containing every single available object name and their respective OIDs for Wireless LAN Controllers does not exist. As an alternative, Cisco offers Management Information Base (MIBs), which might not be easily readable, but contains all available object names and their description. Cisco 3504 WLC MIB can be downloaded [HERE](#).
Downloaded archive file contains multiple .my text files that can either be imported into any third-party SNMP monitoring server or simply opened with a regular text editor. In order to find the OID of a specific object name, you first need to locate the exact file that contains it.

For example, all objects related to monitoring the device’s physical state (like temperature and fan speed) will be located inside a MIB called CISCO-ENVMON-MIB.my. Here, “ciscoEnvMonFanState” is the object name which will be used to provide the state of the WLC fan. MIB files follow the syntax shown below. Information about the fan state object will look like this:

```plaintext
ciscoEnvMonFanState OBJECT-TYPE
   SYNTAX CiscoEnvMonState
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
       "The current state of the fan being instrumented."
   ::= { ciscoEnvMonFanStatusEntry 3 }
```

Most of the third-party monitoring software relies on OIDs, and not object names. Translation between object name and object ID can be done using Cisco’s SNMP object navigator tool. Enter the object name into the search bar. The output will provide the OID and a short description. Additionally, same tool can be used to find the corresponding object name of the OID.

### SNMP Object Navigator

- **Translate/Browse**: Translate OID into object name or object name into OID to receive object details.

Enter OID or object name: 1.3.6.1.4.1.9.9.618.1.4.1

**examples**

- OID: 1.3.6.1.4.1.9.9.27
  - Object Name: ifindex

**Object Information**

<table>
<thead>
<tr>
<th>Specific Object Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
</tr>
<tr>
<td>OID</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Permission</td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td>MIB</td>
</tr>
</tbody>
</table>

**Description**: This object represents the CPU usage string.

### Using OIDs to monitor the state of WLC

After acquiring the OID of the object that needs to be monitored, first SNMP query can be executed. Following examples will showcase how to acquire a WLC CPU usage per core (OID =
1.3.6.1.4.1.9.9.618.1.4.1) for SNMPv2 community **snmp_test** and SNMPv3 user **admin** with SHA Auth password **Cisco123Cisco123** and AES Privacy password set to “Cisco123Cisco123”.

Controller management interface is located on 10.48.39.164.

### Monitoring via snmpwalk

Snmpwalk is an SNMP application that uses SNMP GETNEXT requests to query a network entity for a tree of information. It is present by default on MacOS and most Linux distributions. For SNMPv2c, the command will follow the syntax:

```
snmpwalk -v2c -c <community_name> <WLC_management_interface_ip> <OID>
```

Example:

```
VAPEROVI-M-H1YM:- vaperovi$ snmpwalk -v2c -c snmp_test 10.48.39.164 1.3.6.1.4.1.9.9.618.1.4.1
SNMPv2-SMI::enterprises.9.9.618.1.4.1.0 = STRING: "0%/1%, 0%/1%, 0%/1%, 0%/1%"
```

If SNMPv3 is used, the command will follow the syntax:

```
snmpwalk -v3 -l authPriv -u <username> -a [MD5|SHA] -A <auth_password> -x [AES|DES] -X <priv_password> <WLC_management_interface_ip> <OID>
```

Select MD5/SHA and AES/DES based on how you created the SNMPv3 user on the controller.

Example:

```
VAPEROVI-M-H1YM:- vaperovi$ snmpwalk -v3 -l authPriv -u admin -a SHA -A Cisco123Cisco123 -x AES -X Cisco123Cisco123 10.48.39.164 1.3.6.1.4.1.9.9.618.1.4.1
SNMPv2-SMI::enterprises.9.9.618.1.4.1.0 = STRING: "0%/1%, 0%/1%, 0%/0%, 0%/1%"
```

### Monitoring via Python 3 and pysnmp library

The following code snippets are written in Python 3.7 and will utilize pysnmp module (pip install pysnmp) to make SNMP queries for CPU utilization of Cisco 3504 WLC. These examples will use same SNMPv2 community and SNMPv3 user created in one of the previous chapters. Simply replace the variable values and integrate the code with your own custom scripts.

#### SNMPv2c example:

```python
from pysnmp.hlapi import *

communityName = 'snmp_test'
ipAddress = '10.48.39.164'
OID = '1.3.6.1.4.1.14179.2.3.1.13.0'
errorIndication, errorStatus, errorIndex, varBinds = next( getCmd(SnmpEngine(), CommunityData(communityName), UdpTransportTarget((ipAddress, 161)), ContextData(), ObjectType(ObjectIdentity(OID))) if errorIndication: print(errorIndication) elif errorStatus: print('%s at %s' % (errorStatus.prettyPrint(), errorIndex and varBinds[int(errorIndex) - 1][0] or '?')) else: for varBind in varBinds: print(' = '.join([x.prettyPrint() for x in varBind]))
```

Output will print out:
SNMPv3 example:

```python
from pysnmp.hlapi import *

username = 'admin'
ipAddress = '10.48.39.164'
OID = '1.3.6.1.4.1.14179.2.3.1.13.0'
authKey = 'Cisco123Cisco123'
privKey = 'Cisco123Cisco123'

errorIndication, errorStatus, errorIndex, varBinds = next(
    getCmd(SnmpEngine(),
        UsmUserData(username, authKey, privKey,
            authProtocol=usmHMACSHAAuthProtocol,
            privProtocol=usmAesCfb128Protocol),
        UdpTransportTarget((ipAddress, 161)),
        ContextData(),
        ObjectType(ObjectIdentity(OID)))
)

if errorIndication:
    print(errorIndication)
elif errorStatus:
    print('%s at %s' % (errorStatus.prettyPrint(),
        errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
else:
    for varBind in varBinds:
        print(' = '.join([x.prettyPrint() for x in varBind]))
```

Integration with third party software (Grafana/PRTG Network Monitor/SolarWinds)

Cisco’s Prime Infrastructure offers the ability to easily monitor and configure multiple network devices, including wireless controllers. Prime Infrastructure comes preloaded with all the OIDs and integration with WLC simply consists of adding the WLC credentials to Prime. After the sync, it is possible to set alarms and push configuration templates for multiple wireless controllers at once.

On the other hand, Cisco WLC can also be integrated with multiple 3rd party monitoring solutions, as long as the OIDs are known. Programs like Grafana, PRTG Network monitor and SolarWinds server allow the MIBs or OIDs to be imported and values to be displayed in a user-friendly graph.

This integration might require some tweaking on monitoring server side. In the example below, the PRTG monitoring server is provided with the per core CPU utilization OID which returns the string “0%/1%, 1%/1%, 0%/1%, 0%/1%”. PRTG is expecting an integer value and raises an error.
Table of most commonly monitored OIDs

Considering that MIBs present the data in non-user-friendly syntax, the following table includes some of the most common object names and their OIDs Cisco customers are using.

<table>
<thead>
<tr>
<th>Description</th>
<th>Object name</th>
<th>OID</th>
<th>Expected response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall CPU usage in %</td>
<td>agentCurrentCPUUtilization</td>
<td>1.3.6.1.4.1.14179.1.1.5.1.0</td>
<td>INTEGER: 0</td>
</tr>
<tr>
<td>Per core CPU usage</td>
<td>clsAllCpuUsage</td>
<td>1.3.6.1.4.1.9.9.618.1.4.1.0</td>
<td>STRING: &quot;0%/1%, 0%/1%, 0%/1%, 0%/1%&quot;</td>
</tr>
<tr>
<td>RAM usage in %</td>
<td>clsSysCurrentMemoryUsage</td>
<td>1.3.6.1.4.1.9.9.618.1.8.6.0</td>
<td>Gauge32: 33</td>
</tr>
<tr>
<td>CPU temperature in °C</td>
<td>bsnSensorTemperature</td>
<td>1.3.6.1.4.1.14179.2.3.1.13.0</td>
<td>INTEGER: 76</td>
</tr>
<tr>
<td>Number of joined AP</td>
<td>clsSysApConnectCount</td>
<td>1.3.6.1.4.1.9.9.618.1.8.4.0</td>
<td>Gauge32: 2</td>
</tr>
<tr>
<td>Number of clients</td>
<td>clsMaxClientsCount</td>
<td>1.3.6.1.4.1.9.9.618.1.8.12.0</td>
<td>Gauge32: 0</td>
</tr>
<tr>
<td>Number of clients per WLAN</td>
<td>bsnDot11EssNumberOfMobileStations</td>
<td>1.3.6.1.4.1.14179.2.1.1.1.38.0</td>
<td>Counter32: 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Counter32: 2</td>
</tr>
</tbody>
</table>