Cisco Application Centric Infrastructure MIB Quick Reference

This document describes the private, or local, SNMP Management Information Base (MIB) files supported by the software.

MIBs and Network Management

The MIB list includes Cisco proprietary MIBs and many other Internet Engineering Task Force (IETF) standard MIBs. The IETF standard MIBs are defined in *Requests for Comments* (RFCs). To find specific MIB information, you must examine the Cisco proprietary MIB structure and related IETF-standard MIBs supported by the software.

Network management takes place between two major types of systems: those systems in control, called *managing systems*, and those systems that managing systems observe and control, called *managed systems.* The most common managing system is called a *network management system* (NMS). Managed systems can include hosts, servers, or network components such as switches and routers.

To promote interoperability, cooperating systems must adhere to a common framework and a common language, called a *protocol*. In the Internet-standard management framework, that protocol is the Simple Network Management Protocol (SNMP).

The exchange of information between managed network devices and a robust NMS is essential for reliable performance of a managed network. Because some devices have a limited ability to run management software, most of the computer processing burden is assumed by the NMS. The NMS runs the network management applications, such as Cisco Data Center Network Manager, that present management information to network managers and other users.

In a managed device, specialized low-impact software modules, called agents, access information about the device and make it available to the NMS. Managed devices maintain values for a number of variables and report those values, as required, to the NMS. For example, an agent might report such data as the number of

bytes and packets sent or received by the device or the number of broadcast messages sent and received. In SNMP, each of these variables is referred to as a *managed object*. A managed object is anything that can be managed or anything that an agent can access and report back to the NMS. All managed objects are contained in the MIB, which is a database of the managed objects.

An NMS can control a managed device by sending a request to an agent of that managed device, requiring the device to change the value of one or more of its variables. The managed devices can respond to requests such as set or get. The NMS uses the set request to control the device. The NMS uses the get requests to monitor the device. The set and get requests are synchronous events, which means that the NMS initiates the activity, and the SNMP agent responds.

The managed device can send asynchronous events, or SNMP notifications, to the NMS to inform the NMS of some recent event. SNMP notifications (traps or informs) which are included in many MIBs, and allow the NMS to less frequently send get requests to the managed devices.

SNMP Access Support in Cisco ACI

Note For the complete list of MIBs supported in Cisco Application Centric Infrastructure (ACI), see http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/mib/list/mib-support.html.

SNMP support in Cisco ACI is as follows:

- SNMP read queries (Get, Next, Bulk, Walk) are supported by leaf and spine switches and by the Cisco Application Policy Infrastructure Controller (APIC).
- SNMP write commands (Set) are not supported by leaf and spine switches or by the Cisco APIC.
- SNMP traps (v1, v2c, and v3) are supported by leaf and spine switches and by the Cisco APIC.

Note Cisco ACI supports a maximum of 10 trap receivers.

- SNMPv3 is supported by leaf and spine switches and by the Cisco APIC.
- SNMP using a Cisco APIC IPv6 address is not supported.

Table 1: SNMP Support Changes by Cisco APIC Release

Release	Description
1.2(2)	IPv6 support is added for SNMP trap destinations.
1.2(1)	SNMP support for the Cisco APIC controller is added. Previous releases support SNMP only for leaf and spine switches.

Accessing MIB Variables Through SNMP

The SNMP system consists of three parts: the SNMP manager, the SNMP agent, and the MIB. You can compile Cisco MIBs with your network management software. If SNMP is configured on a device, the SNMP agent responds to MIB-related queries sent by the NMS.

SNMPv1 was the initial version of the protocol. SNMPv2 added support for 64-bit counters, and SNMPv3 added increased security for access, authentication, and encryption of managed data.

This table describes SNMP operations.

Table 2: SNMP Operations

Operation	Description
get-request	Retrieves a value from a specific variable.
get-next-request	Retrieves the value following the named variable. Often used to retrieve variables from within a table.[1]
get-bulk[2]	Retrieves large blocks of data, such as multiple rows in a table, which would otherwise require the transmission of many small blocks of data.
trap	Sends an unsolicited message by an SNMP agent to an SNMP manager indicating that some event has occurred.

[1] With this operation, an SNMP manager does not need to know the exact variable name. A sequential search finds the next variable from within the MIB.
[2] The get-bulk command is not a part of SNMPv1

SNMP Traps

You can configure the software to send notifications as SNMP traps to SNMP managers when particular events occur.

Notifications may contain a list of MIB variables, or *varbinds*, that clarify the status this is relayed by the notification. The list of varbinds associated with a notification is included in the notification definition in the MIB. In the case of standard MIBs, Cisco has enhanced some notifications with additional varbinds that further clarify the cause of the notification.

Use the SNMP-TARGET-MIB to obtain more information on trap destinations.

Note You must enable most notifications through the CLI.

Interpreting the MIB Structure

A MIB presents the managed data in a logical tree hierarchy, using an IETF standard syntax called the *Structure of Management Information* (SMI). Branches of this MIB tree are organized into individual tables, which contain the managed data as leaf objects.

Object Identifiers

The MIB structure is logically represented by a tree hierarchy. The root of the tree is unnamed and splits into three main branches: Consultative Committee for International Telegraph and Telephone (CCITT), International Organization for Standardization (ISO), and joint ISO/CCITT.

These branches and those branches that fall below each category have short text strings and integers to identify them. Text strings describe *object names*, while integers allow computer software to create compact, encoded representations of the names.

Each MIB variable is assigned with an object identifier. The *object identifier* is the sequence of numeric labels on the nodes along a path from the root to the object. For example, the MIB variable tftpHost is indicated by the number 1. The object identifier for tftpHost is iso.org.dod.internet.private.enterprise.cisco.workgroup products.stack group.tftp group.tftpHost or .1.3.6.1.4.1.9.5.1.5.1. The last value is the number of the MIB variable tftpHost.

Tables

SNMP MIBs organize information into tables. When network management protocols use names of MIB objects in messages, each name has an appended suffix. This suffix is called an *instance identifier*. It identifies one occurrence of the associated MIB object. For simple scalar objects, the instance identifier 0 refers to the instance of the object with that name (for example, sysUpTime.0).

A MIB can also contain tables of related objects. For example, ifOperStatus is a MIB object inside the ifTable from the IF-MIB. It reports the operational state for an interface on a device. Because devices may have more than one interface, it is necessary to have more than one instance of ifOperStatus. This instance value is added to the end of the MIB object as the instance identifier (for example, ifOperStatus.2 reports the operational state for interface number 2).

Each object in a table is constructed with a set of clauses defined by the SMI. These clauses include the SYNTAX clause, MAX-ACCESS clause, STATUS clause, and DESCRIPTION clause.

An excerpt of the information in the VSAN table (known as vsanTable) from CISCO-VSAN-MIB follows:

```
vsanTable OBJECT-TYPE
          SYNTAX     SEQUENCE OF VsanEntry
          MAX-ACCESS not-accessible
          STATUS     current
          DESCRIPTION
          "A table of VSANs configured on this device."
  ::= { vsanConfiguration 3 }
  vsanEntry OBJECT-TYPE
          SYNTAX     VsanEntry
          MAX-ACCESS not-accessible
          STATUS     current
          DESCRIPTION
                 "An entry (conceptual row) in the vsanTable."
          INDEX { vsanIndex }
          ::= { vsanTable 1 }
  VsanEntry ::= SEQUENCE {
          vsanIndex                VsanIndex,
          vsanName                 SnmpAdminString,
  }
```

In the example, vsanTable contains two variables: vsanIndex and vsanName. (There are more values in the actual vsanTable.) The index for this table is the ID of the VSAN, or vsanIndex. With n number of VSANs

configured, n rows are present in the table. If you want to retrieve the vsanName that matches VSAN ID 3 (vsanIndex is 3), then you would issue an SNMP get for vsanName.3.

SYNTAX Clause

The SYNTAX clause describes the format of the information, or value, that is returned when you monitor or set information in a MIB.

The Cisco MIBs are defined with the SNMPv2 Structure of Management Information version 2 (SNMPv2-SMI) defined in RFC 1902. Some examples of SNMPv2-SMI syntax are as follows:

Counter32

A nonnegative integer that increases until it reaches some maximum value. After reaching the maximum value, it rolls over to zero. For example, the variable ifInOctets, with a Counter32 syntax, counts the number of input octets on an interface.

Gauge32

A nonnegative integer that increases until it reaches some maximum value. After reaching the maximum value, it stays fixed (no rollover).

Counter64

A nonnegative 64-bit integer that increases until it reaches some maximum value. After reaching the maximum value, it rolls back to zero. Counter64 is used for MIB objects that can reach high values in a short period of time (for example, a packet counter for a Gigabit Ethernet port).

Integer32

An integer from -232 to 232-1.

IPAddress

An octet string that represents an IP address. For example, the variable hostConfigAddr indicates the IP address of the host that provided the host configuration file for a device.

Timeticks

A nonnegative integer that counts the hundredths of a second that have elapsed since an event. For example, the variable loctcpConnElapsed provides the length of time that a TCP connection has been established.

MAX-ACCESS Clause

The MAX-ACCESS clause identifies the maximum access level for the associated MIB object. This clause can represent one of the following five states: read-create, read-write, read-only, accessible-for-notify, and not-accessible.

read-create

You can read, modify, or create objects as rows in a table.

read-write

You can read or modify this object

read-only

You can only read this object.

accessible-for-notify

You cannot read or write to this object. SNMP notifications can send this object as part of their event information.

not-accessible

You cannot read or write to this object. Table indices are typically objects that are not accessible.

About Cisco MIB Files

MIB II is documented in RFC 1213, *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*. Portions of MIB-II have been updated since RFC 1213. See the IETF website http://www.ietf.org for the latest updates to this MIB.

If your NMS cannot get requested information from a managed device, such as a Cisco switch, the MIB that allows that specific data collection might be missing. Typically, if an NMS cannot retrieve a particular MIB variable, either the NMS does not recognize the MIB variable, or the agent does not support the MIB variable. If the NMS does not recognize a specified MIB variable, you might need to load the MIB into the NMS, usually with a MIB compiler. For example, you might need to load the Cisco private MIB or the supported RFC MIB into the NMS to execute a specified data collection. If the agent does not support a specified MIB variable, you must find out what version of system software that you are running. Different software releases support different MIBs.

Note Cisco and IETF MIBs are updated frequently. You should download the latest MIBs from Cisco.com whenever you upgrade your software.

MIB Loading Order

Many MIBs use definitions that are defined in other MIBs. These definitions are listed in the IMPORTS section near the top of the MIB.

For example, if MIB B imports a definition from MIB A, some MIB compilers require you to load MIB A prior to loading MIB B. If you get the MIB loading order wrong, you might get an error message about what was imported, claiming it is undefined or not listed in IMPORTS. If you receive an error, look at the loading order of the MIB definitions from the IMPORTS of the MIB. Make sure that you have loaded all the preceding MIBs first.

Here is a list of MIBs from which many other MIBs import definitions. The MIBs are listed in the order in which you should load them:

- SNMPv2-SMI.my
- SNMPv2-TC.my
- SNMPv2-MIB.my
- RFC1213-MIB.my
- IF-MIB.my
- CISCO-SMI.my
- CISCO-TC.my
- CISCO-ST-TC.my

- ENTITY-MIB.my

If you load the MIBs in this order, you can eliminate most of your load-order definition problems. You can load most other MIBs (those not listed here) in any order.

Accessing and Downloading Cisco MIB Files

You can use your browser to access and download the Cisco MIB files.

- You can access and download Cisco MIB files at the following GitHub site:

  https://github.com/cisco/cisco-mibs/tree/main/v2

- You can access and download Cisco MIB files using the SNMP Object Navigator tool located at the following site:

  https://snmp.cloudapps.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en

  You can use this tool to translate SNMP object identifiers (OIDs) into object names, search object names and descriptions, browse OID trees, and download MIB files. Cisco also includes supported IETF-standard MIBs at this site.

Understanding the ENTITY-MIB and Extensions

The ENTITY-MIB provides basic management and identification of physical and logical entities within a network device. Cisco NX-OS support for the ENTITY-MIB focuses on the physical entities within a device. This MIB provides details on each module, power supply, and fan tray within a switch chassis. It gives enough information to correctly map the containment of these entities within the switch, building up a chassis view.

Cisco has developed a number of private extensions to the ENTITY-MIB to provide more details for these physical entities. Each MIB extensions shares the common index value, entPhysicalIndex, which allows the management application developer to link information across multiple MIBs.

This table lists the Cisco MIB extensions that are linked to the ENTITY-MIB by entPhysical Index.

Table 3: ENTITY-MIB Extensions

MIB	Description
CISCO-ENTITY-EXT-MIB	Extends the entityPhysicalTable for modules with processors. For each of these modules, this MIB provides memory statistics and LED information.
CISCO-ENTITY-FRU-CONTROL-MIB	Manages field-replaceable units, such as power supplies, fans, and modules.
CISCO-ENTITY-SENSOR-MIB	Provides sensor data for environmental monitors such as temperature gauges.

Extending the IF-MIB

The IF-MIB provides basic management status and control of interfaces and sublayers within a network device. Multiple standard and Cisco-specific MIBs use ifIndex from the IF-MIB to extend management for specific interface types.