Table of Contents

MIB Compilers and Loading MIBs

Document ID: 26015

Introduction

Most network management systems (NMSs) provide a way for the user to load MIBs. Loading a MIB is a way that an NMS can learn about new MIB objects, such as their names, object identifiers (OIDs), and the kind of datatype (for example, `Counter`).

The MIB might be parsed when it is loaded or it might happen later, for instance, when an NMS application runs. The software that performs the parsing is a MIB compiler.

Any syntactically correct MIB should be successfully parsed by any vendor s MIB compiler. Unfortunately, different MIB compilers may exhibit different quirks.

Cisco makes continuous efforts to ensure that the MIBs published to customers are syntactically correct. Cisco also avoids MIB constructs that have proven to be problematic in popular NMS products. Despite these efforts, it is not possible to satisfy the quirks in all of the MIB compilers in the field.

This document addresses some of the common problems and suggests workarounds. If you encounter any of these problems with your vendor s MIB compiler (with the exception of the RFC 14xx versus RFC 19xx issue), it is due to a deficiency in that MIB compiler. You may wish to urge your vendor or vendors to fix their compilers.

Prerequisites

Requirements

Readers of this document should be familiar with MIBs.

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

Common MIB Loading Problems

Load Ordering

The loading order is the most important and common problem when you are loading MIBs. Many MIBs use definitions that are defined in other MIBs. These definitions are listed in the `IMPORTS` clause near the top of the MIB.

For example, if MIB **mumble** imports a definition from MIB **bumble**, some MIB compilers require you to load MIB **bumble** prior to loading MIB **mumble**. If you get the load order wrong, the compiler will claim that the imported MIBs are undefined.

This is a list of MIBs from which many other MIBs import, and the order in which you should load them. This probably takes care of 95 percent of load order issues (most of the other MIBs can be loaded in any order):

- SNMPv2−SMI.my
- SNMPv2−TC.my
- SNMPv2−MIB.my
- RFC1213−MIB.my
- IF−MIB.my
- CISCO−SMI.my
- CISCO−PRODUCTS−MIB.my
- CISCO−TC.my

Note: If you are loading the v1 versions of these MIBs, the MIB filename will actually look like IF−MIB**−V1SMI**.my (−V1SMI is added to the name of MIBs that have been converted from v2 to v1). The exception to this is the RFC1213−MIB.my MIB, which only exists as a v1 version (that is, there is no RFC1213−MIB−V1SMI.my).

If you attempt to load another MIB, and if the compiler complains about undefined items, then identify from which MIBs this MIB is importing and verify that you have loaded all the other MIBs first.

Note: For each MIB, you can see the exact list of the MIBs that need to be loaded prior to it with the exact compilation order in SNMP Object Navigator > View & Download MIBs; select **View MIB dependencies and download MIB**.

Mismatches on Datatype Definitions

Although Cisco MIB datatype definitions will not be mismatched, you may find that to be the case for some standard RFC MIBs. For example:

- MIB **mumble** defines: `SomeDatatype ::= INTEGER(0..100)`
- MIB **bumble** defines: `SomeDatatype ::= INTEGER(1..50)`

This example is considered to be a trivial error and the MIB loads successfully with a warning message.

The next example is considered to be a non−trivial error (even though the two definitions are essentially equivalent), and the MIB is not successfully parsed.

- MIB **mumble** defines: `SomeDatatype ::= DisplayString`
- MIB **bumble** defines: `SomeDatatype ::= OCTET STRING (SIZE(0..255))`

If your MIB compiler treats these as errors, or if you wish to get rid of the warning messages, then edit one of the MIBs that define this same datatype so that the definitions match.

Object Identifier Redefinitions

You may encounter OID redefinitions if you load these MIBs (although there may be other instances where this error occurs):

- OLD−CISCO−CPU−MIB.my
- OLD−CISCO−ENV−MIB.my
- OLD−CISCO−MEMORY−MIB.my
- OLD−CISCO−SYSTEM−MIB.my

For example:

- OLD−CISCO−CPU−MIB.my defines: `lcpu OBJECT IDENTIFIER ::= { local 1 }`
- OLD−CISCO−ENV−MIB.my defines: `lenv OBJECT IDENTIFIER ::= { local 1 }`

When you load these two MIBs, the MIB compiler may complain about the `lcpu OBJECT IDENTIFIER` being redefined with a new name `lenv`. The OLD−CISCO−MEMORY−MIB.my and OLD−CISCO−SYSTEM−MIB.my similarly give new names to `{ local 1 }`.

This is treated as a trivial error and the MIB successfully loads with a warning message.

If the MIB does not load successfully, or if you wish to get rid of the warning message, then edit one of the MIBs so that all of the MIBs use the same name.

Definitions of Built−in Datatypes

Many MIB compilers have built−in knowledge of some datatypes, such as DisplayString. Some of these compilers complain if they see a definition for these datatypes in a MIB. For example, DisplayString is defined in SNMPv2−TC.

The workaround is to remove or comment out the offending definition in the MIB file.

Alternate Sizes

This is a syntactically valid example, which indicates that a value of type `MyDatatype` will either be 0, 5, or 20 octets in length:

```
MyDatatype ::= OCTET STRING (SIZE(0 | 5 | 20))
```

Some MIB compilers do not accept this syntax. Usually, a sufficient workaround is to pick one of the sizes and remove the others. You should keep the largest size. For example, the previous example would be changed to this:

```
MyDatatype ::= OCTET STRING (SIZE(20))
```

Odd Object Identifiers

Some OIDs are considered odd because they do not refer to a node in the SMI (like most `OBJECT IDENTIFIER`s). However, they are syntactically valid. A common example is the null object identifier, for example, `{ 0 0 }`. Some MIB compilers do not care for `OBJECT IDENTIFIER`s that do not correspond to a node in the SMI. These are examples of MIB syntax that could cause problems for these compilers:

```
zeroDotZero OBJECT IDENTIFIER ::= { 0 0 }
myMIBObject OBJECT-TYPE
DEFVAL { {0 0} }
```

The workaround is to remove or comment out those types of references in the MIB file.

Trap Definitions

In SNMPv1 MIBs, traps are defined with the `TRAP-TYPE` macro. In SNMPv2 MIBs, traps are defined using the `NOTIFICATION-TYPE` macro.

Some MIB compilers do not like these definitions in the MIB files that they are parsing (they do not support these macros).

If this is the case, you can remove the trap definitions or comment out the definitions (for instance, put the MIB comment delimiter `--` at the beginning of the lines).

RFC 14xx Based Compilers versus RFC 19xx Based Compilers

RFCs 1442 through 1452 define the party-based SNMPv2. These RFCs are obsoleted by the newer Draft Standard RFCs 1902 through 1908.

With regards to MIB syntax, there are very few differences between these two versions of SNMPv2; there are some, however. Cisco MIBs are currently based on RFC 19xx rules.

Note: A few years ago, when Cisco MIBs were RFC 14xx-based, some RFC 19xx-based compilers would complain about the `Unsigned32 ::= TEXTUAL-CONVENTION` line in CISCO-TC.my and PNNI-MIB.my MIBs. This is because Unsigned32 is a predefined datatype in RFC 19xx. For this reason, Cisco used to have alternative versions of these MIBs (CISCO-TC-NO-U32.my and PNNI-MIB-NO-U32.my) with no definition for Unsigned32, to load in the compilers that already know about this data type. This is no longer applicable.

Loading and Compiling MIBS into Third−Party NMS

The best and most efficient way to load Cisco MIBs, traps, and icons into third−party NMS is to use CiscoWorks Integration Utility (Integration Utility), which is available as part of CiscoWorks Common Services (or standalone from http://www.cisco.com/cgi−bin/tablebuild.pl/cw2000−utility), with the corresponding Integration Utility Adapter from http://www.cisco.com/tacpage/sw−center/cw2000/cmc3rd.shtml and the latest Network Management Integration Data Bundle (NMIDB). Check Integration Utility documentation for more details.

Alternatively, you can consult the documentation of third−party NMS on MIB loading and compilation. This document includes instructions for HP OpenView and IBM NetView; but you should still consult HP or IBM documentation, as the products may change.

From the GUI of HP OpenView or IBM NetView

Follow these steps to load the Cisco MIBs that you want:

1. Copy the files into the directory /usr/OV/snmp_mibs of the network management station.

   This is the default directory where HP OpenView and IBM NetView look for MIB documents. If you place them elsewhere, specify the explicit path names in the loadmib graphical interface.
2. Set the permissions so that you have read access to the MIBs.
3. From the GUI menu, choose **Options > Load/Unload MIBs**.
4. Follow the instructions in the platform documentation, to compile or load the Cisco MIBs.

From the Command Line Interface of HP OpenView or IBM NetView

Issue the **/opt/OV/bin/xnmloadmib −load *filename*** command, to load the MIB file.

NetPro Discussion Forums − Featured Conversations

Networking Professionals Connection is a forum for networking professionals to share questions, suggestions, and information about networking solutions, products, and technologies. The featured links are some of the most recent conversations available in this technology.

NetPro Discussion Forums − Featured Conversations for Network Management
Network Infrastructure: Network Management
Virtual Private Networks: Network and Policy Management

Related Information

- **SNMP Tech Notes**
- **IP Services Technology Support**
- **Technical Support & Documentation − Cisco Systems**

Updated: Jun 29, 2005 Document ID: 26015