

Table of Contents

<u>How To Use the CISCO-BULK-FILE-MIB</u>	1
<u>Document ID: 24304</u>	1
<u>Introduction</u>	1
<u>Before You Begin</u>	1
<u>Requirements</u>	1
<u>Components Used</u>	1
<u>Conventions</u>	2
<u>Background Information</u>	2
<u>Using the CISCO-BULK-FILE-MIB</u>	2
<u>Creating a BULK-FILE Operation</u>	3
<u>Step-by-Step Instructions</u>	3
<u>Transferring the File Using the CISCO-FTP-CLIENT-MIB</u>	4
<u>Step-by-Step Instructions</u>	4
<u>Verifying the Result</u>	5
<u>Troubleshooting the Result</u>	6
<u>Caveats</u>	6
<u>Related Information</u>	7

How To Use the CISCO-BULK-FILE-MIB

Document ID: 24304

Introduction

Before You Begin

- Requirements
- Components Used
- Conventions

Background Information

Using the CISCO-BULK-FILE-MIB

- Creating a BULK-FILE Operation
- Step-by-Step Instructions
- Transferring the File Using the CISCO-FTP-CLIENT-MIB
- Step-by-Step Instructions

Verifying the Result

Troubleshooting the Result

Caveats

Related Information

Introduction

This document explains how to use the CISCO-BULK-FILE-MIB and transfer files created by that Management Information Base (MIB) using the CISCO-FTP-CLIENT-MIB.

Starting from Cisco IOS® software release 12.0, Cisco has implemented a way to store an Simple Network Management Protocol (SNMP) object or table as a file on the device. This file can then be retrieved using the CISCO-FTP-CLIENT-MIB. This technology allows you to transfer large amounts of data using a reliable transport method.

Before You Begin

Requirements

Before attempting this configuration, ensure that you meet these requirements:

- You have a Cisco device running Cisco IOS® software release 12.0 or later. Check the MIB Locator Tool to make sure the CISCO-BULK-FILE-MIB is supported for your device. A link to the tool can be found on the Cisco IOS MIB Tools page.

Note: This MIB is not supported on Catalyst OS devices.

- SNMP must be configured on the device with both read-only and read-write community strings. This is not covered in this document. For information on configuring SNMP on IOS® devices, read How to Configure SNMP Community Strings on Routers, Cisco IOS Software-Based XL Switches, RSMs, MSFCs and Catalyst Switches.

Components Used

The information in this document is based on these software and hardware versions:

- The CISCO-BULK-FILE-MIB to store the ifTable from a 7507 router running 12.1(12) in a file, then use the CISCO-FTP-CLIENT-MIB to transfer that file from the router to a FTP server.
- The net-snmp SNMP command suite installed on UNIX or Windows.
- These MIBs are used:

- ◆ SNMPv2-TC
- ◆ SNMPv2-SMI
- ◆ SNMPv2-CONF
- ◆ SNMPv2-MIB
- ◆ IANAifType-MIB
- ◆ IF-MIB
- ◆ CISCO-SMI
- ◆ CISCO-TC
- ◆ CISCO-BULK-FILE-MIB
- ◆ CISCO-FTP-CLIENT-MIB

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 .

Background Information

Make sure you have the MIBs in this table loaded into your management platform. This allows you to use the object names and values listed above instead of the numeric Object Identifiers (OIDs). In general, this document refers to object names and not OIDs.

Version 1 SMI Format	Version 2 SMI Format
SNMPv2-SMI-V1SMI.my	SNMPv2-SMI.my
SNMPv2-TC-V1SMI.my	SNMPv2-TC.my
	SNMPv2-CONF.my
SNMPv2-MIB-V1SMI.my	SNMPv2-MIB.my
IANAifType-MIB-V1SMI.my	IANAifType-MIB.my
IF-MIB-V1SMI.my	IF-MIB.my
CISCO-SMI-V1SMI.my	CISCO-SMI.my
CISCO-TC-V1SMI.my	CISCO-TC.my
CISCO-BULK-FILE-MIB-V1SMI.my	CISCO-BULK-FILE-MIB.my
CISCO-FTP-CLIENT-MIB-V1SMI.my	CISCO-FTP-CLIENT-MIB.my

Using the CISCO-BULK-FILE-MIB

Creating a BULK-FILE Operation

In this example, we capture the `ifTable` from a router, and store it in a bulk file. However, you can use any MIB object or table.

Use the `net-snmp` version of `snmpset`. The IP address of the router is **14.32.8.2**. Its read-write community string is **private**. The read-only community string is **public**.

Each time you create a new bulk file operation, choose two random numbers for the row instance. They can be any number between 1 and 4294967295 inclusive. For the purposes of this example, use 333 and 444.

Step-by-Step Instructions

To create a BULK-FILE operation, complete these steps:

1. Set up the file to be created.

```
$ snmpset -c private 14.32.8.2 cbfDefineFileEntryStatus.333 i 5
$ snmpset -c private 14.32.8.2 cbfDefineFileName.333 s ifTable.txt
$ snmpset -c private 14.32.8.2 cbfDefineFileFormat.333 i bulkASCII
```

2. Specify the MIB object to capture.

This object requires two indices for correct operation. The 333 is the 333 from the file creation table above. The 444 is a new random number used for the primary index in the `cbfDefineObjectTable`.

This example demonstrates using an object name for `cbfDefineObjectID` (`ifTable`). You could use a fully-qualified OID here as well.

```
$ snmpset -c private 14.32.8.2 cbfDefineObjectID.333.444 o ifTable
```

3. Activate the newly created rows.

You must have both indices for your `cbfDefineObjectTable` row.

```
$ snmpset -c private 14.32.8.2 cbfDefineObjectEntryStatus.333.444 i 1
$ snmpset -c private 14.32.8.2 cbfDefineFileEntryStatus.333 i 1
```

4. Create the file.

```
$ snmpset -c private 14.32.8.2 cbfDefineFileNow.333 i 3
```

The bulk file is created.

5. Verify the file was created successfully by using `snmpget` on the `cbfStatusFileState` object.

This object requires two indices. The first index is the random number chosen for the File table (333 in this example). The second index depends on how many files you created in your router. Since this is your first file, the index is 1. Therefore, use the command:

```
$ snmpget -c public 14.32.8.2 cbfStatusFileState.333.1
```

A value of `running(1)` means that the file is in the process of being created. A value of `ready(2)` means that the file was created successfully, and is waiting to be read.

This file is not directly accessible from the router, however. Use the `CISCO-FTP-CLIENT-MIB` to read this

file.

Transferring the File Using the CISCO-FTP-CLIENT-MIB

For each FTP Client operation, you must select a random number for the row instance. You can use one of the same random numbers you used above. This example uses 555.

Step-by-Step Instructions

To transfer the file using a CISCO-FTP-CLIENT-MIB, complete these steps:

1. Create a row instance of the FTP Client.

```
$ snmpset -c private 14.32.8.2 cfcRequestEntryStatus.555 i 5
```

2. Fill in the required parameters. The LocalFile **must** be the same name as the file you created above! Use **putASCII** to transfer bulkASCII files.

If you set the cbfDefineFileFormat to bulkBinary above, you must set cfcRequestOperation to putBinary.

```
$ snmpset -c private 14.32.8.2 cfcRequestOperation.555 i putASCII
$ snmpset -c private 14.32.8.2 cfcRequestLocalFile.555 s ifTable.txt
$ snmpset -c private 14.32.8.2 cfcRequestRemoteFile.555 s /home/Marcus/ifTable.txt
$ snmpset -c private 14.32.8.2 cfcRequestServer.555 s 172.18.123.33
$ snmpset -c private 14.32.8.2 cfcRequestUser.555 s Marcus
$ snmpset -c private 14.32.8.2 cfcRequestPassword.555 s marcus123
```

3. Begin the transfer by setting the row to active.

```
$ snmpset -c private 14.32.8.2 cfcRequestEntryStatus.555 i 1
```

The FTP transfer begins. When complete, the file is saved to **/home/Marcus/ifTable.txt**.

4. To get the status of the FTP transfer, use **snmpget** again on the cfcRequestResult object.

This object uses the same index that you used with the other FTP objects.

```
$ snmpget -c public 14.32.8.2 cfcRequestResult.555
```

A value of `pending(1)` means the file is still transferring. A value of `success(2)` means the file transferred successfully. Any other value is an error.

5. When the file is done transferring, try the **snmpget** of the cbfStatusFileState object again. It now has a different value.

```
$ snmpget -c public 14.32.8.2 cbfStatusFileState.333.1
enterprises.cisco.ciscoMgmt.ciscoBulkFileMIB.ciscoBulkFileMIBObjects.cbfStatus.
cbfStatusFileTable.cbfStatusFileEntry.cbfStatusFileState.333.1 = emptied(3)
```

The value of `emptied(3)` means that the file has been successfully read. The file cannot be transferred again.

6. It is now safe to delete this file by destroying the file status row. This object takes the same indices as the cbfStatusFileState above.

```
$ snmpset -c private 14.32.8.2 cbfStatusFileEntryStatus.333.1 i 6
```

7. Once the file is deleted, delete the corresponding Object and File rows.

```
$ snmpset -c private 14.32.8.2 cbfDefineObjectEntryStatus.333.444 i 6
```

```
$ snmpset -c private 14.32.8.2 cbfDefineFileEntryStatus.333 i 6
```

In this manner, you can use the CISCO-FTP-CLIENT-MIB to transfer any file off of the router using FTP.

Verifying the Result

This section guides you through reading some of the syntax for this file.

1. The first line is the prefix line. For our ifTable example, it is:

```
prefix 1.3.6.1.2.1.2.2.1
```

This corresponds to the OID for the ifEntry object. The ifTable is composed of one or more ifEntries.

2. The next line lists the number of objects in the table. The line consists of the keyword table followed by the number of objects in the table, followed by the index of each object.

For example:

```
table 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

This line states that the table contains 22 objects, and each object has an incrementing index. These objects are from the ifTable example:

```
ifIndex  
ifDescr  
ifType  
ifSpeed  
...
```

3. After this line, there are multiple row entries. In the ifTable example, each row corresponds to an interface. The rows begin with the keyword row, followed by their index identifier, and followed by the objects enumerated by the previous table entry.

For example:

```
row 1 1 546F6B656E52696E67302F30 9 4464 16000000 0008B0851800 2 2 6551 0 0 0 0 0 0 0 0
```

4. The fourth entry is the ifDescr for interface 1. However, this is the ifDescr in hexadecimal encoded ASCII.

To translate this line into a more readable format, use this Perl command:

```
$ perl -e 'print pack("H*", "546F6B656E52696E67302F30")'  
TokenRing0/0
```

This entry corresponds to interface TokenRing0/0. All objects that are normally strings are displayed as hexadecimal encoded ASCII in the bulk files. You can use this Perl command to translate any hexadecimal ASCII string into readable text. If you do not have Perl, use this ASCII character table to translate the string.

5. Some entries show ~ characters for values. This means that the value for that object is NULL. That is, the object is not instantiated on the device.

For example:

```
row 9 9 41544D312F302F302D61746D206C61796572 37 ~ 0 1 1 5971 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
```

This corresponds to the ATM1/0/0-atm layer interface. Notice that `ifMtu` is NULL for this interface. Since this is a virtual interface, it makes sense that it does not have an MTU. If you prefer, you can replace these NULLs with 0 by adding this command to the device's configuration:

```
Router(config)#no snmp-server sparse-table
```

Troubleshooting the Result

When polling the `cbfStatusFileState` object, if you receive a value other than `running(1)`, `ready(2)`, or `emptied(3)`, your operation has encountered an error. These are causes for the errors:

<code>noSpace</code>	no data due to insufficient file space
<code>badName</code>	no data due to a name or path problem
<code>writeErr</code>	no data due to fatal file write error
<code>noMem</code>	no data due to insufficient dynamic memory
<code>buffErr</code>	implementation buffer too small
<code>aborted</code>	short terminated by operator command

If the number of objects in the file is less than you expect, `cbfDefineMaxObjects` from the `CISCO-BULK-FILE-MIB` may be set too low. To determine the object's current value, use `snmpget`.

```
$ snmpget -c public 14.32.8.2 cbfDefineMaxObjects.0
```

A value of 0 means that no limit is configured. The value can be set to any integer between 0 and 4294967295, inclusive. To set the maximum objects per file to 10, use the `snmpset` command. The index for this object is always 0.

```
$ snmpset -c private 14.32.8.2 cbfDefineMaxObjects.0 u 10
```

This object may not be configurable on all platforms. If `snmpset` fails with this error, the object is not configurable on your platform:

```
Error in packet.  
Reason: (noSuchName) There is no such variable name in this MIB.  
Failed object: enterprises.cisco.ciscoMgmt.ciscoBulkFileMIB.ciscoBulkFileMIBObjects.cbfDef
```

When polling the `cfcRequestResult` object, if you receive a value other than `pending(1)` or `success(2)`, the FTP operation encountered an error. These are causes for the errors:

<code>aborted</code>	user aborted the transfer
<code>fileOpenFailLocal</code>	local bulk file was not found
<code>fileOpenFailRemote</code>	remote file could not be opened for writing
<code>badDomainName</code>	FTP server's hostname could not be resolved
<code>unreachableIpAddress</code>	route to the FTP server could not be found
<code>linkFailed</code>	connection could not be made to the remote server
<code>fileReadFailed</code>	local file could not be read
<code>fileWriteFailed</code>	remote file could not be written

Caveats

- Currently there is no supported way to access the bulk files directly. You must go through the `CISCO-FTP-CLIENT-MIB` to read the files.
- The `cbfDefineFileStorage` object defines three types: `ephemeral`, `volatile`, and `permanent`. Currently, the only type supported in IOS is `ephemeral`. Ephemeral files exist in small amounts until read.

- Once the files are read, they cannot be reread. They must first be re-created.
 - The `cbfDefineFileFormat` object defines three types: `standardBER`, `bulkBinary`, and `bulkASCII`. The only supported formats are `bulkBinary` and `bulkASCII`. The default format is `bulkBinary`.
 - The Chameleon FTP server for Windows is known **not** to work with the `CISCO-FTP-CLIENT-MIB`, since it does not return correct result codes.
-

Related Information

- **How to Configure SNMP Community Strings on Routers, Cisco IOS Software-Based XL Switches, RSMs, MSFCs and Catalyst Switches**
 - **Technical Support – Cisco Systems**
-

All contents are Copyright © 1992–2005 Cisco Systems, Inc. All rights reserved. Important Notices and Privacy Statement.

Updated: Jun 22, 2005

Document ID: 24304
