APPENDIX A

MIB Files and Objects

This appendix contains information on how to modify existing Management Information Base (MIB) files and establish new MIB files, as well as a brief example of MIB objects and their use with CiscoWorks polling capabilities.

For more detailed information on MIB objects, refer to the *Cisco MIB User Quick Reference* publication. For a list of MIB reference sources relating to building MIB object structure, refer to the appendix "References and Recommended Reading."

For consistency, this guide uses the term *object* to represent such terms as MIB objects, MIB object instances, and so on. Other publications might use different terms, but they can be used interchangeably.

MIB Objects Overview

There are many MIB objects that can help you to manage your network. All objects are documented in either the RFC standard or a vendor-specific MIB (for example, the Cisco MIB). Here are a few MIB objects to consider using:

- Bandwidth—To obtain information about Internet Control Message Protocol (ICMP) echo packets (ping) input and output from the device, use *icmpInEchos*, *icmpInEchoReps*, *icmpOutEchos*, and *icmpOutEchoReps* objects. This information is useful to measure the bandwidth being consumed by simply answering ping packets.
- Configuration Information—To verify configuration information, use the MIB objects *sysDescr*, *sysName*, *sysUpTime*, *sysContact*, *sysLocation*, *ifNumber*, *romId*, and *whyReload*.
- Interface Error Rates—For fault management, it is often useful to monitor interface error rates. For an Ethernet interface, use *locIfInCRC*, *locIfCollisions*, *ifInErrors*, *ifOutErrors*, *locIfInRunts*, or *locIfInGiants*. For a serial interface, use *locIfInFrame*, *locIfInAbort*, *locIfInIgnored*, *locIfResets*, *locIfRestarts*, *ifOutErrors*, and *ifInErrors*.
- SNMP Packet Data—To determine how much time the router is spending answering or sending SNMP packets, use *snmpInGetRequests*, *snmpInGetNexts*, *snmpInGetResponses*, *snmpOutGetRequests*, *snmpOutGetNexts*, *snmpOutGetResponses*, *snmpInTraps*, and *snmpOutTraps*.
- Traffic Flow—To measure traffic flow for performance management, use the Cisco MIB objects *locIfInBitsSec*, *locIfOutBitsSec*, *locIfInPktsSec*, and *locIfOutPktsSec* to determine basic traffic flow on an interface in bits per second (bps) and packets per second (pps). Use *lifTable* in the Cisco MIB to monitor on a specific port for a specific protocol.
- Unreachable Address—To determine how often a router is being asked to send information to an unreachable address, use the *icmpOutDestUnreachs* object.

MIB Source Files

The textual MIB files must be written in the ASN.1 subset format described in RFC 1212. This guide does not describe all aspects of the ASN.1 language.

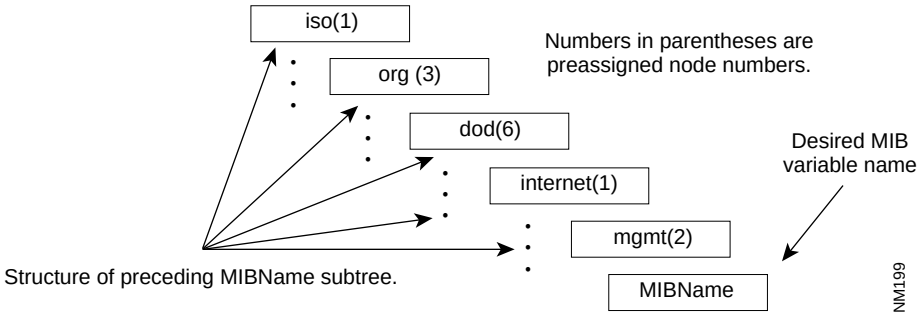
A MIB is organized into a tree structure consisting of labeled nodes. A complete object identifier is built by traversing the tree from the root node, collecting labels as each node is passed. The root of the MIB tree is *iso(1)*. (See Figure A-1.)

Except for the MIB tree root *iso(1)*, all nodes in the tree must have the parent node defined before a child node can be defined. Upper level nodes (those above *mib(1)*) can be most conveniently defined with the following statement:

```
MIBName { iso org(3) dod(6) internet(1) mgmt(2) 1 }
```

MibName is an arbitrary name. The numbers in parentheses are preassigned node numbers. The previous statement is interpreted as follows: **mgmt** is the beginning of a subtree (called *MIBName*) under **internet**, which is under **dod**, which is under **org**, which is under **iso**.

Figure A-1 MIB Hierarchy



Defining MIB Files

Additional subtrees below *mgmt* can be defined using OBJECT IDENTIFIER statements as follows:

```
Child Node OBJECT IDENTIFIER
::= { ParentNode ChildNodeNumber }
```

For example, to define **mib** (an object type) as a child of **mgmt** and assign a node number of 1, enter the following:

```
mib OBJECT IDENTIFIER ::= { mgmt 1 }
```

Most object types in a MIB should be defined using the OBJECT-TYPE macro. Following are the basic structure elements and format:

```
ObjectDescriptor OBJECT-TYPE
SYNTAX ASN1_Type
ACCESS AccessType
STATUS StatusType
::= ObjectIdentifier
```

An example of defining an OBJECT-TYPE macro with the name *atNetAddress* follows:

```
atNetAddress OBJECT-TYPE
SYNTAX NetworkAddress
ACCESS read-write
```

```
STATUS mandatory
:= { atEntry 3 }
```

Note For a complete discussion of the structure of a MIB file, refer to RFC 1212 and RFC 1213. ISO document 8824 describes ASN.1 in detail.

MIB trees must define the parentage of every branch from terminal object IDs to the *iso(1)* tree root. All trees must have the same root (*iso(1)* is the recommended root) in order to determine how one MIB tree relates to another.

CiscoWorks MIB Files

The MIB object set read by CiscoWorks at the time it is started defines the objects that can be monitored and collected in the database for future analysis. At startup, CiscoWorks looks for the *mib.bin* and *mib.alias* files in the *$NMSROOT/etc/mibs* directory. These files define the MIB object set and the aliases used for polling and data collection, and in addition to *alias.gen*, *alias.master*, and the MIB source files (**.mib*) comprise the set of MIB-related files necessary for operating CiscoWorks.

The following MIB source files are shipped with CiscoWorks:

- A100-R1-MIB.mib— enterprise MIB for version 1.x of the A100 Hyperswitch
- BGP4-MIB.mib—Border Gateway Protocol Version 4 (BGP4) MIB from *draft-ietf-bgp-mibv4-05.txt*
- CISCO-CDP-MIB.mib—Cisco Discovery Protocol (CDP) MIB. Used for management of the CDP in Cisco devices.
- CISCO-CHANNEL-MIB.mib—Cisco Channel Interface Processor (CIP) MIB, which allows management of the CIP.
- CISCO-DSPU-MIB.mib—MIB definitions for Cisco downstream physical unit (DSPU)
- CISCO-IMAGE-MIB.mib—Cisco Image MIB, which allows CiscoWorks to query the capabilities of the image running on a Cisco box
- CISCO-PING-MIB.mib— Allows multiprotocol pings to be performed via Simple Network Management Protocol (SNMP). This MIB is a replacement for the ping MIB object ({ lsystem 7 }), which is now obsolete.
- CISCO-PRODUCTS-MIB.mib—Contains the sysObjectID assignments for Cisco hardware platforms.
- CISCO-SMI.mib— Cisco Structure of Management Information (SMI) MIB, which defines the top-level OBJECT IDENTIFIER assignments for the Cisco enterprise, as well as all Cisco sysObjectID values.
- CISCO-TC.mib—Cisco Textual Conventions, which defines some ASN.1 textual conventions that are used throughout Cisco MIBs.
- CISCO-TCP-MIB.mib—An extension to the “tcp” group defined in MIB-II (RFC 1213). This MIB is a replacement for the "ltcp" group, defined as { local 6 } in earlier MIBs,which is now obsolete.
- CISCO-VINES-MIB.mib—Objectsfrom the Cisco VINES command line interface, influenced by the Banyan VINES MIB. It now subsumes the obsolescent, temporary Cisco VINES MIB.

- FDDI-SMT73-MIB.mib—RFC 1512 Fiber Distributed Data Interface (FDDI) MIB for Station Management (SMT) 7.3.
- LS100-R2-MIB.mib—Enterprise MIB for Version 2.x of the LS100 Hyperswitch (formerly known as the A100).
- NOVELL-IPX-MIB.mib—Virtually identical to the IPX MIB distributed as a part of the Novell NetWare Link Services Protocol (NLSP) Specification 1.0, Novell Part Number 100-001708-002, 2nd Edition, February 1994.
- NOVELL-RIPSAP-MIB.mib—Virtually identical to the RIPSAP MIB distributed as a part of the Novell NetWare Link Services Protocol (NLSP) Specification 1.0, Novell Part Number 100-001708-002, 2nd Edition, February 1994.
- OLD-CISCO-APPLETALK-MIB.mib— Cisco AppleTalk MIB, extracted from the Cisco IOS 10.0 monolithic MIB
- OLD-CISCO-CHASSIS-MIB.mib—Cisco Chassis MIB, extracted from the Cisco IOS 10.0 monolithic MIB
- OLD-CISCO-CPU-MIB.mib—Cisco CPU MIB. This MIB represents a portion of the *lsystem* group that was present in the Cisco IOS 10.0 monolithic MIB. These are objects used to monitor the agent CPU usage.
- OLD-CISCO-DECNET-MIB.mib—Cisco DECnet MIB, extracted from the Cisco IOS 10.0 monolithic MIB.
- OLD-CISCO-ENVMON-MIB.mib—Cisco Environmental Monitor MIB. This MIB represents a portion of the *lsystem* group that was present in the Cisco IOS 10.0 monolithic MIB. These are objects used to monitor the Environmental Monitor, when present.
- OLD-CISCO-FLASH-MIB.mib—Cisco Flash Memory MIB, extracted from the Cisco IOS 10.0 monolithic MIB.
- OLD-CISCO-INTERFACES-MIB.mib—Cisco Interfaces MIB, extracted from the Cisco IOS 10.0 monolithic MIB.
- OLD-CISCO-IP-MIB.mib—Cisco IP MIB, extracted from the IOS 10.0 monolithic MIB
- OLD-CISCO-MEMORY-MIB.mib—Cisco Memory MIB. This MIB represents a portion of the *lsystem* group that was present in the IOS 10.0 monolithic MIB. These are objects used to monitor the Dynamic Memory Manager.
- OLD-CISCO-NOVELL-MIB.mib—Cisco Novell/IPX MIB, extracted from the Cisco IOS 10.0 monolithic MIB.
- OLD-CISCO-SYSTEM-MIB.mib—Cisco System MIB. This MIB represents a portion of the *lsystem* group that was present in the Cisco IOS 10.0 monolithic MIB. These are all general-purpose MIB objects.
- OLD-CISCO-TCP-MIB.mib—Cisco TCP MIB, extracted from the Cisco IOS 10.0 monolithic MIB. This MIB is deprecated, having been replaced with *CISCO-TCP-MIB.my*, and is obsolete.
- OLD-CISCO-TS-MIB.mib—Cisco Terminal Service MIB, extracted from the Cisco IOS 10.0 monolithic MIB.
- OLD-CISCO-VINES-MIB.mib—Cisco Banyan Vines MIB, extracted from the Cisco IOS 10.0 monolithic MIB.
- OLD-CISCO-XNS-MIB.mib—Cisco Xeros Network Systems (XNS) MIB, extracted from the Cisco IOS 10.0 monolithic MIB.
- RFC1213-MIB.mib—MIB-II from RFC 1213.

- RS-232-MIB.mib—RS-232 MIB from RFC 1659.
- SNA-SDLC-MIB.mib—Synchronous Data Link Control (SDLC) MIB. This MIB is almost identical to RFC 1747, but was implemented prior to the RFC being finalized. In the future it will be updated to implement the finalized RFC.
- SNMP-REPEATER-MIB.mib—Hub MIB from RFC 1516.
- SOURCE-ROUTING-MIB.mib—Source Route Bridging MIB from RFC 1525 and RFC 1573
- cisco-adapter-mib—ASN.1 description of Cisco-specific MIB variables for Cisco network adapter cards.
- cisco-stack.mib—ASN.1 description of Cisco-specific MIB variables for Cisco's Catalyst concentrators and switches.
- ciscoworks.mib

Individual objects in the MIB files appear on a lines with the OBJECT-TYPE keyword.

The *alias.master* file contains the list of aliases upon which CiscoWorks depends. It is located in *$NMSROOT/etc/mibs* directory.

Cisco Private MIB Objects

The Cisco private MIB object file, *cisco.mib*, is located in the CiscoWorks directory *$NMSROOT/etc/mibs.* This section lists the Cisco private MIB objects that have been introduced starting with Software Release 8.1.

Note As shipped, CiscoWorks 3.0.3 does not support objects newly introduced *after* Cisco IOS Release 10.2. With patch software, however, CiscoWorks will support network devices running future releases of the Cisco IOS.

Software Release 8.1

Following are the objects introduced with Software Release 8.1:

actAge
ipCkAccountingTable
ipckactSrc
ipckactDst
ipkactPkts
ipckactByts
ckactAge

Software Release 8.2

Following are the objects introduced with Software Release 8.2:

writeMem
writeNet
busyPer
avgBusy1
avgBusy5
idleCount
idleWired
locIfCarTrans
locIfReliab
locIfDelay
locIfLoad
locIfCollisions
tsLineNoise
dnAreaTable
dnACost
dnAHop
dnAIfIndex
dnANextHop
dnAAge
dnAPrio
vinesInput
vinesOutput
vinesLocaldest
vinesForwarded
vinesBcastin
vinesBcastout
vinesBcastfwd
vinesNotlan
vinesNotgt4800
vinesNocharges
vinesFormaterror
vinesCksumerr
vinesHopcout
vinesNoroute
vinesEncapsfailed
vinesUnkown
vinesIcpIn
vinesIcpOut
vinesMetricOut
vinesMacEchoIn
vinesMacEchoOut
vinesEchoIn
vinesEchoOut

Software Release 8.3

Following are the objects introduced with Software Release 8.3:

bufferHgsize
bufferHgTotal
bufferHgFree
bufferHgMax
bufferHgHit
bufferHgMiss
bufferHgTrim
bufferHgCreate
locIfInputQueueDrops
locIfOutputQueueDrops
ipNoaccess
actCheckPoint
tsMsgTtyLine
tsMsgIntervaltim
tsMsgDuration
tsMsgTest
tsMsgTmpBanner
tsMsgSend
dnIfTable
dnIfCost

Software Release 9.0

Following are the objects introduced with Software Release 9.0:

netConfigProto
hostConfigProto
sysConfigAddr
sysConfigName
sysConfigProto
sysClearARP
sysClearInt
envPresent
envTestPt1Descr
envTestPt1Measure
envTestPt2Descr
envTestPt2Measure
envTestPt3Descr
envTestPt3Measure
envTestPt4Descr
envTestPt4Measure
envTestPt5Descr
envTestPt5Measure
envTestPt6Descr
envTestPt6Measure
locIfDescr
locIfPakmon

Software Release 9.1

Following are the objects introduced with Software Release 9.1:

envTestPt4MarginPercent
envTestPt5MarginPercent
envTestPt6MarginPercent
envTestPt1last
envTestPt2last
envTestPt3last
envTestPt4last
envTestPt5last
envTestPt6last
envTestPt1MarginVal
envTestPt2MarginVal
envTestPt3MarginVal
envTestPt4MarginVal
envTestPt5MarginVal
envTestPt6MarginVal
envTestPt1warn
envTestPt2warn
envTestPt3warn
envTestPt4warn
envTestPt5warn
envTestPt6warn
envFirmVersion
envTechnicianID
envType
envBurnDate
envSerialNumber
locIfSlowInPkts
locIfSlowOutPkts
locIfSlowInOctets
locIfSlowOutOctets
locIfFastInPkts
locIfFastOutPkts
locIfFastInOctets
locIfFastOutOctets
locIfotherInPkts
locIfotherOutPkts
locIfotherInOctets
locIfotherOutOctets
locIfipInPkts
locIfipOutPkts
locIfipInOctets
locIfipOutOctets
locIfdecnetInPkts
locIfdecnetOutPkts
locIfdecnetInOctets
locIfdecnetOutOctets
locIfxnsInPkts
locIfxnsOutPkts
locIfxnsInOctets
locIfxnsOutOctets
locIfclnsInPkts
locIfclnsOutPkts
locIfclnsInOctets
locIfclnsOutOctets
locIfappletalkInPkts
locIfappletalkOutPkts
locIfappletalkInOctets
locIfappletalkOutOctets
locIfnovellInPkts
locIfnovellOutPkts
locIfnovellInOctets

locIfnovellOutOctets
locIfapolloInPkts
locIfapolloOutPkts
locIfapolloInOctets
locIfapolloOutOctets
locIfvinesInPkts
locIfvinesOutPkts
locIfvinesInOctets
locIfvinesOutOctets
locIfbridgedInPkts
locIfbridgedOutPkts
locIfbridgedInOctets
locIfbridgedOutOctets
locIfsrbInPkts
locIfsrbOutPkts
locIfsrbInOctets
locIfsrbOutOctets
locIfchaosInPkts
locIfchaosOutPkts
locIfchaosInOctets
locIfchaosOutOctets
locIfpupInPkts
locIfpupOutPkts
locIfpupInOctets
locIfpupOutOctets
locIfmopInPkts
locIfmopOutPkts
locIfmopInOctets
locIfmopOutOctets
locIflanmanInPkts
locIflanmanOutPkts
locIflanmanInOctets
locIflanmanOutOctets
locIfstunInPkts
locIfstunOutPkts
locIfstunInOctets
locIfstunOutOctets
locIfspanInPkts
locIfspanOutPkts
locIfspanInOctets
locIfspanOutOctets
locIfarpInPkts
locIfarpOutPkts
locIfarpInOctets
locIfarpOutOctets
locIfprobeInPkts
locIfprobeOutPkts
locIfprobeInOctets
locIfprobeOutOctets
flashSize
flashFree
flashcontoller
flashcard
flashVPP
flashErase
flashEraseTime
flashEraseStatus
flashToNet
flashToNetTime
flashToNetStatus
netToFlash
netToFlashTime
netToFlashStatus
flashStatus
flashEntries

flashDirName
flashDirSize
flashDirStatus

Software Release 9.21

Following are the objects introduced with Software Release 9.21:

locIfDribbleInputs
vinesProxy
vinesProxyReply
vinesNet
vinesSubNet
vinesClient
vinesIfMetric
vinesIfEnctype
vinesIfAccesslist
vinesIfPropagate
vinesIfArpEnabled
vinesIfServerless
vinesIfServerlessBcast
vinesIfRedirectInterval
vinesIfSplitDisabled
vinesIfLineup
vinesIfFastokay
vinesIfRouteCache
vinesIfIn
vinesIfOut
vinesIfInBytes
vinesIfOutBytes
vinesIfRxNotEnabled
vinesIfRxFormatError
vinesIfRxLocalDest
vinesIfRxBcastin
vinesIfRxForwarded
vinesIfRxNoRoute
vinesIfRxZeroHopCount
vinesIfRxChecksumError
vinesIfRxArp0
vinesIfRxArp1
vinesIfRxArp2
vinesIfRxArp3
vinesIfRxArpIllegal
vinesIfRxIcpError
vinesIfRxIcpMetric
vinesIfRxIcpIllegal
vinesIfRxIpc
vinesIfRxRtp0
vinesIfRxRtp1
vinesIfRxRtp2
vinesIfRxRtp3
vinesIfRxRtp4
vinesIfRxRtp5
vinesIfRxRtp6
vinesIfRxRtpIllegal
vinesIfRxSpp
vinesIfRxUnknown
vinesIfRxBcastHelpered
vinesIfRxBcastForwarded
vinesIfRxBcastDuplicate
vinesIfRxEcho
vinesIfRxMacEcho
vinesIfRxProxyReply
vinesIfTxUnicast

vinesIfTxBcast
vinesIfTxForwarded
vinesIfTxFailedEncaps
vinesIfTxFailedAccess
vinesIfTxFailedDown
vinesIfTxNotBcastToSource
vinesIfTxNotBcastNotlan
vinesIfTxNotBcastNotgt4800
vinesIfTxNotBcastPpcharge
vinesIfTxBcastForwarded
vinesIfTxBcastHelpered
vinesIfTxArp0
vinesIfTxArp1
vinesIfTxArp2
vinesIfTxArp3
vinesIfTxIcpError
vinesIfTxIcpMetric
vinesIfTxIpc
vinesIfTxRtp0
vinesIfTxRtp1
vinesIfTxRtp2
vinesIfTxRtp3
vinesIfTxRtp4
vinesIfTxRtp5
vinesIfTxRtp6
vinesIfTxSpp
vinesIfTxEcho
vinesIfTxMacEcho
vinesIfTxProxy
chassisType
chassisVersion
chassisId
romVersion
romSysVersion
processorRam
nvRAMSize
nvRAMUsed
configRegister
configRegNext
cardTable
cardTableEntry
cardIndex
cardType
cardDescr
cardSerial
cardHwVersion
cardSwVersion
cardSlotNumber
chassisSlots

Cisco Internetwork Operating System (Cisco IOS) Release 10.0

Following are the objects introduced with Software Release 10.0:

ipxThresh
ipxactLostPkts
ipxactLostByts
ipxactSrc
ipxactDst
ipxactPkts
ipxactByts
ipxactAge
ipxckactSrc
ipxckactDst
ipxckactPkts
ipckactByts
ipxckactAge
ipxactCheckPoint
vinesIfInputNetworkFilter
vinesIfInputRouterFilter
vinesIfOutputNetworkFilter
cardIfIndex
cardIfSlotNumber
cardIfPortNumber

Cisco Internetwork Operating System (Cisco IOS) Release 10.2

Following are the objects introduced with Software Release 10.2:

cipCardClawEntry
cipCardClawIndex
cipCardClawConnected
cipCardClawConfigTable
cipCardClawConfigEntry
cipCardClawConfigPath
cipCardClawConfigDevice
cipCardClawConfigIpAddr
cipCardClawConfigHostName
cipCardClawConfigRouterName
cipCardClawConfigHostAppl
cipCardClawConfigRouterAppl
cipCardClawDataXferStatsTable
cipCardClawDataXferStatsEntry
cipCardClawDataXferStatsBlocksRead
cipCardClawDataXferStatsBlocksWritten
cipCardClawDataXferStatsBytesRead
cipCardClawDataXferStatsBytesWritten
cipCardClawDataXferStatsHCBytesRead
cipCardClawDataXferStatsHCBytesWritten
cipCardClawDataXferStatsReadBlocksDropped
cipCardClawDataXferStatsWriteBlocksDropped
cipCardClawDataXferStatsBufferGetRetryCount
cipCardDtrBrdIndex
cipCardDtrBrdType
cipCardDtrBrdStatus
cipCardDtrBrdSignal
cipCardDtrBrdOnline
implicitIncidents

Modifying the MIB Files

Many networks are built with equipment from various vendors, and there are custom objects that you as a network manager may want to monitor. You can add these non-Cisco custom MIBs to the standard object set using the *makemib* script.

CiscoWorks uses two files to define how it accesses MIB object information from devices. The first, *mib.bin*, is a database of all the objects currently defined in the MIB source files. The second file, *mib.alias*, allows Cisco to define a protocol-independent application environment.

The *mib.alias* file also allows you to assign custom names to existing MIB objects. For instance, you could assign the name *UpTimeInterval* to the MIB object *sysUptime*. You also could assign several names to the same MIB object. The *mib.alias* file provides the translation matrix for the object information you want to collect.

Note If you have Cisco devices only on your network, or if you want to collect standard MIB object information only from non-Cisco devices, you do not have to modify the *mib.bin* or *mib.alias* files.

Using the makemib Script

If you want to collect custom MIB information from non-Cisco SNMP devices in your network, use the *makemib* script to create new *mib.bin* and *mib.alias* files for your installation.

To use the *makemib* script, perform the following steps:

Step 1 Assemble all the additional MIB source files relevant to your application into the *$NMSROOT/etc/mibs* directory.

Step 2 Delete any of the source files supplied by Cisco that are not needed.

Step 3 Check that any MIB files you added to the *$NMSROOT/etc/mibs* directory include the *.mib* extension.

Step 4 Enter the following:

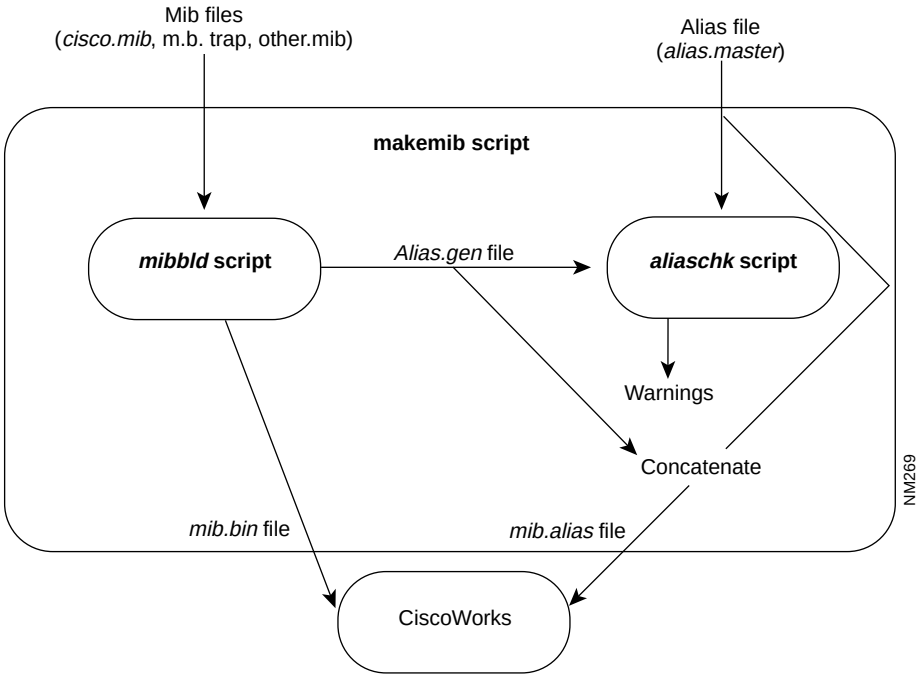
```
%makemib
```

Note *$NMSROOT/etc* must be in the *PATH* statement. If it is not, you must include the entire path in the command statement.

When the *makemib* process finishes, you are ready to start CiscoWorks.

The *makemib* script runs *mibbld* to generate the *mib.bin* file and the *alias.gen* file, and then runs *aliaschk* to check the *alias.gen* file. It then combines the *alias.gen* and *alias.master* files to form the *mib.alias* file. (See Figure A-2.)

Figure A-2 CiscoWorks MIB Build Process



The process of expanding the MIB object set starts with a series of files supplied by the vendors of the various network products. These files define the MIB objects and can be edited with any standard text editor, as long as you conform to the concise MIB format. You can store as many vendor files as you like. The file specification must include the *.mib* extension.

Note Building or modifying existing MIB tables requires a working knowledge of the ASN.1 language, the Concise MIB Notation, and the Structure of Management Information (SMI) specification.

Command Syntax for makemib

Use the following syntax to start the *makemib* script:

makemib -m *MasterAlias* **-a** *FinalAlias options*

MasterAlias is the master alias file (default: *alias.master*). *FinalAlias* is the final alias file (default: *mib.alias*). *Options* are described in Table A-1.

Table A-1 makemib Options

Option	Description
–f *MibFile*	Specifies the MIB file to be processed by filename *MibFile* (default **.mib*).
–o *OutMib*	Indicates the name of the output file for the MIB binary file (default *mib.bin*).
–p *PathToMibs*	Processes all MIB files found in the directory specified by the path name *PathToMibs* (default *$NMSROOT/$MIBDIR*).
–a *AliasFile*	Indicates the name of output file for the aliases (default *alias.gen*).

Option	Description
–w *WarnLevel*	Sets the warning message level at the level specified by *WarnLevel*. A level of 0 suppresses all warning messages (including "unknown variable type"). The default setting is 1. The *mibbld* script issues warning and error messages regarding syntax errors, bad keywords, and so on, so it is wise to run with the warning level set to 1 or greater.
–t	Displays MIB table after processing to standard output.
–n	Disables alias generation.
–v	Displays copyright and version information to standard output.
–h	Displays help information to standard error output (*stderr*).

Running *makemib* completes the creation of the *mib.alias* file in a single command step. You can, however, run the *mibbld* and *aliaschk* scripts individually instead of running the *makemib* script. These scripts are described later in this chapter under their individual headings.

We recommend that you use the *makemib* script.

Using MIB Aliases

MIB aliases are used by CiscoWorks to create protocol-independent MIB objects. The *mib.alias* file defines a name in terms of a MIB object name and a protocol. Aliases also allow one or more protocol-specific object identifiers to be associated with a single MIB object. Currently, only SNMP is supported.

The *mib.alias* file is an ASCII file. Each alias entry takes the following general format:

```
Alias { Protocol::ObjectID [ Protocol::ObjectID [ ...]] }
```

The *ObjectID* can be either numeric or textual, but must be fully qualified. MIB trees must define the parentage of every branch from terminal object IDs to the *iso(1)* tree root. All trees must have the same root. The recommended root is *iso(1)*. As a side effect of ASN.1 object identifier encoding, the first two subidentifiers are compressed into a single numeric value. Thus, *iso(l) – org(3)* becomes 43. rather than 1.3. You follow this convention when specifying a numeric object identifier within an alias entry.

For uniformity, *iso(l)* through *org(3)* also are compressed to *isoorg* in a textual object identifier. Note that in actual application, each textual subidentifier must be introduced with the underbar delimiter (_), even the first subidentifiers (such as *_isoorg*).

An example of a portion of a *mib.alias* file follows:

```
UpTime { SNMP::43.6.1.2.1.1.3 }
ItsTime { SNMP::_isoorg_dod_internet_mgmt_mib_system_sysUpTime }
EgpNeighbor { SNMP::43.6.1.2.1.8.5.1.1, SNMP::43.6.1.2.1.8.5.1.2 }
RouteMetric { SNMP::43.6.1.2.1.4.21.1.3, SNMP::43.6.1.2.1.4.21.1.4,
SNMP::43.6.1.2.1.4.21.1.5, SNMP::43.6.1.2.1.4.21.1.6 }
```

If you list more than one object identifier for a given alias, CiscoWorks searches for the ID in the order listed.

To include comments in the MIB alias file, place two dashes (--) in front of the comment text, as follows:

```
-- This is a comment.
```

Modifying the mib.alias File

The *makemib* script builds the alias entries in *alias.gen* and then concatenates *alias.gen* and *alias.master* into *mib.alias*. It generates a unique name for each object by prefixing the MIB file identifier (defined within the *mib* file) to the object name with a hyphen (-). If the MIB filename is *new.mib*, its file name identifier is *NEW-MIB*, and the object name is *newMibVar,* the following alias name is generated:

```
NEW-MIB-newMibVar
```

The protocol and object identifier is defined in the file as follows:

```
NEW-MIB-newMibVar {SNMP::43.6.1.2.4.33.2.5}
```

However, if the filename identifier is already part of the object name, it is not prefixed to the object name. For example, if the MIB filename is *new.mib*, its filename identifier is *new* and the object name is *newMibVar,* the following alias name is generated:

```
newMibVar
```

One-to-One Name Definition

You can also modify the *mib.alias file* with any text editor. This is the most convenient way to add a name definition to an already existing MIB object. For instance, if you want to define the name *UpTimeInterval* as the standard MIB object *sysUptime,* enter the following in the *mib.alias* file:

```
UpTimeInterval{SNMP::_isoorg_dod_internet_mgmt_mib_system_sysUpTime}
```

Many-to-One Name Definition

At times, you may want to have several names for the same MIB object. For instance, if you want to define the names *Myhost*, *Ciscohost,* and *Routername* as the Cisco private MIB object *CISCO-MIB-hostName*, enter the following in the *mib.alias* file:

```
Myhost{SNMP::_isoorg_dod_internet_private_enterprises_cisco_local_lsystem_hostName}
CiscoHost{SNMP::_isoorg_dod_internet_private_enterprises_cisco_local_lsystem_hostName}
Routername{SNMP::_isoorg_dod_internet_private_enterprises_cisco_local_lsystem_hostName}
```

Timesaver The easiest way to construct these many-to-one name entries is to find the appropriate MIB object definition in the *mib.alias* file, copy it, and modify it.

One-to-Many Name Definition

At times, you may want to have a single name for several MIB objects. For instance, if you want to define the name *Routemetric* to include four objects—*SNMP::43.6.1.2.1.4.21.1.3*, *SNMP::43.6.1.2.1.4.21.1.4*, *SNMP::43.6.1.2.1.4.21.1.5*, and *SNMP::43.6.1.2.1.4.21.1.6*—enter the following in the *mib.alias* file:

```
Routemetric {
SNMP::43.6.1.2.1.4.21.1.3
SNMP::43.6.1.2.1.4.21.1.4
SNMP::43.6.1.2.1.4.21.1.5
SNMP::43.6.1.2.1.4.21.1.6
}
```

When you start CiscoWorks, your changes to the *mib.alias* file are used when you ask CiscoWorks to poll for that object.

MIB Directory Pathing

There are two forms of directory pathing:

- Default MIB alias file, where *$MIBDIR* has the following relative path name: `$NMSROOT/$MIBDIR/mib.alias`
- Default MIB alias file if *$MIBDIR* is not defined: `$NMSROOT/etc/mibs/mib.alias`

CiscoWorks requires the *mib.alias* file to appear in the same directory as the *mib.bin* file. CiscoWorks software looks for the MIB alias file in the directory *$NMSROOT/etc/mibs* or *$NMSROOT/$MIBDIR*. *NMSROOT* defines the root directory for all Network Management System (NMS) software. *MIBDIR* is a relative path name that defines the directory below the NMS root directory, which contains one or more MlB files and the *mib.alias* file.

If *MIBDIR* is not defined, the MIB directory defaults to *$NMSROOT/etc/mibs.* If a full path to the MIB directory or a MIB filename is specified using the **–p** or **–f** options, *NMSROOT* and *MIBDIR* are ignored.

The base name of the output MIB file defaults to *mib.bin* unless a specific binary output file is defined. The directory in which the output file appears is determined as follows:

- If the output MIB file includes a path, the file appears in the indicated directory.
- If the output MIB file did not include a path (or no output MIB file was specified), the following rules apply:
  - If an input MIB file is specified and it includes a path, the output MIB file appears in that directory.
  - If an input MIB file is specified and it does not include a path, the output MIB file appears in the current directory.
  - If an input MIB directory is specified and it does include a path, the output MIB file appears in that directory.
  - If neither an input MIB file nor path is specified, the output MIB file appears in the default directory, *$NMSROOT/$MIBDIR*.

Using mibbld

The script used to build the binary MIB file is *mibbld*. The script checks the contents of all the files with the *.mib* extension in the specified directory for syntax errors and creates binary format *mib.bin* and *alias.gen* files usable by CiscoWorks.

The format of the statement used to invoke *mibbld* follows:

mibbld [**-f** *MibFile*] [**-o** *OutMib*] [**-P** *PathToMibs*] [**-a** *AliasFile*] [**-w** *WarnLevel*] [**-t**] [**-n**] [**-v**] [**-h**]

The *mibbld* script options are described in Table A-2.

Table A-2 mibbld Options

Option	Description
-f *MibFile*	Specifies the MIB file to be processed by filename *MibFile* (default **.mib*).
-o *OutMib*	Provides the name of the output file for the MIB binary file (default *mib.bin*).

Option	Description
-p *PathToMibs*	Processes all MIB files found in the directory specified by the path name *PathToMibs* (default *$NMSROOT/$MIBDIR*). The environment objects *NMSROOT* and *MIBDIR* are ignored.
-a *AliasFile*	Provides the name of the output file for the aliases (default *alias.gen).*
-w *WarnLevel*	Sets the warning message level at the level specified by *WarnLevel*. A level of 0 suppresses all warning messages (including "unknown variable type"). The default setting is 1. The *mibbld* script issues warning and error messages regarding syntax errors, bad keywords, and so on, so it is wise to run with the warning level set to 1 or greater.
-t	Displays MIB table after processing to standard output.
-n	Disables alias generation.
-v	Displays copyright and version information to standard output.
-h	Displays help information to standard error.

If you run *mibbld* without specifying full path names, *mibbld* looks for the environment objects *NMSROOT* and *MIBDIR*.

Using showmib

showmib displays the contents of the binary MIB file in a tree structure. The command also checks the contents of the MIB alias file against the MIB tree. The form of **showmib** follows:

showmib [**–a** *AliasFile*] [**–f** *MibFile* | **–p** *PathToMibs*] [**–s m** | **a**] [**–h**]

showmib processes the contents of a binary format MIB file and displays the contents in a hierarchical tree structure starting from the tree's root. The MIB alias file is then processed. Aliases are displayed with corresponding fully qualified object IDs.

showmib options are described in Table A-3.

Table A-3 showmib Command Options

Option	Description
-a *AliasFile*	Provides name of MIB alias file.
-f *MibFile*	Processes the single MIB file specified by filename (path name) *MibFile.*
-p *PathToMibs*	Processes all MIB files found in the directory specified by path name *PathToMibs*. The environment objects *NMSROOT* and *MIBDIR* are ignored.
-s *SuppressO*	Suppresses output. If *SuppressO* is set to **m**, the output of the MIB tree is suppressed. If *SuppressO* is set to **a**, the output of aliases is suppressed. If set to **-h**, displays the **showmib** options list.
-h	Displays help information to standard error.

If you run **showmib** with no arguments, it looks for the environment objects *NMSROOT* and *MIBDIR*.

Using aliaschk

The *aliaschk* script ensures that alias names defined in *alias.gen* do not conflict with alias names in *alias.master.*

To complete the last step in the *makemib* process, concatenate the *alias.master* and *alias.gen* files together as *mib.alias* by entering the following:

```
% cat alias.master alias.gen > mib.alias
```