



[<< Previous Section](#) [Next Section >>](#)


1. [Table of Contents](#)
2. [Release Notes](#)

Supported Platforms

3. [Linux Installation Instructions](#)
4. [Windows Installation Instructions](#)

Getting Started

5. [MIMIC Installation Instructions](#)
6. [Quick Start Guide](#)

This button  is used throughout the documentation as a shortcut into the Quick Start Guide.

Core

7. [Simulator Guide](#)
8. [Recorder Guide](#)
9. [Compiler Guide](#)

Extras

10. [Wizards Guide](#)
11. [WEBUI Guide](#)
12. [Virtual Lab Guide](#)
13. [Utilities Guide](#)

APIs

14. [Tcl API Guide](#)
15. [Java API Guide](#)
16. [Perl API Guide](#)
17. [C++ API Guide](#)
18. [Python API Guide](#)
19. [PHP API Guide](#)
20. [Javascript API Guide](#)
21. [Go API Guide](#)
22. [OpenAPI Guide](#)

Protocols

23. [Cable Modem Simulator Guide](#)
24. [DHCP Protocol Module Guide](#)
25. [TFTP Protocol Module Guide](#)
26. [Time-Of-Day Protocol Module Guide](#)
27. [IOS Simulator Guide](#)
28. [Telnet Protocol Module Guide](#)
29. [SSH Protocol Module Guide](#)
30. [SYSLOG Protocol Module Guide](#)
31. [IPMI Protocol Module Guide](#)
32. [Server Proxy Protocol Module Guide](#)
33. [NETFLOW Protocol Module Guide](#)
34. [SFLOW Protocol Module Guide](#)
35. [WEB Services Module Guide](#)
36. [MQTT Protocol Module Guide](#)
37. [CoAP Protocol Module Guide](#)

Appendices

38. [Appendix A: Directory Structure](#)

- 39. [Appendix B: Pre-compiled MIBs](#)
- 40. [Appendix C: Common Error Messages](#)
- 41. [Appendix D: Frequently Asked Questions](#)

This documentation is best viewed with a commercial web browser such as Opera, Arena, Mozilla or Internet Explorer.

If you have one of these, you can configure it in the MIMICView [Configuration Wizard](#). Otherwise, MIMIC includes a simple, built-in HTML documentation viewer.

[<< Previous Section](#) [Next Section >>](#)

Copyright © 1996-2019 Gambit Communications, Inc. All Rights Reserved.

MIMIC® is a registered trademark of Gambit Communications, Inc.

All other product and brand names are trademarks or registered trademarks of their respective holders.



[<< Previous Section](#) [Next Section >>](#)

Table of Contents

[Table of Contents](#)

[MIMIC Release Notes](#)

- [New Functionality in 19.20](#)
 - [Simulator](#)
 - [Miscellaneous](#)
- [New Functionality in Previous Releases](#)
- [Known Problems](#)

[Linux Installation Instructions](#)

- [Overview](#)
- [Swap Space](#)
- [1000/2000/5000/10000 agent support](#)
- [10000/20000/50000/100000 agent support](#)
- [Yellow Pages with large configurations](#)
- [Performance impact of certain daemons](#)
- [IPv6 support](#)
- [Core Dumps](#)
- [Filesystem](#)
- [NFS](#)
- [Shared libraries](#)
- [64-bit support](#)
- [Bonded Interfaces](#)
- [ARP Cache](#)
- [Known Problems](#)

[Windows Installation Instructions](#)

- [Overview](#)
- [Account Privileges](#)
- [Firewalls](#)
- [Windows 95 Environment Space](#)
- [Disk Space](#)
- [Assigning IP Addresses](#)
- [Windows NT 4.0 Service Pack 3](#)
- [5000 / 10000 agent support](#)
- [20000 agent support](#)
- [500/1000/2000 Agent Support on Windows NT](#)
- [Virtual Memory](#)
- [Duplicate IP Address](#)
- [Media Sense on Windows 2000 and newer](#)
- [IPv6 Support](#)
- [Performance impact of Windows services](#)
- [64-bit support](#)
- [Windows Vista and newer](#)
- [Crashes](#)
- [Known Problems](#)

[MIMIC Installation Instructions](#)

- [Overview](#)
- [Step 1: Download the MIMIC distribution](#)
- [Step 2: Uncompress and extract the distribution files](#)
- [Step 3: Read the README file for further instructions](#)

[Quick Start Guide](#)

- [Chapter 1: Overview](#)
 - [About MIMIC](#)
 - [MIMIC Tool Suite](#)
- [Chapter 2: Using MIMICView](#)
 - [Starting MIMIC](#)
 - [Introducing Agent Properties](#)
 - [Using the GUI](#)
 - [Understanding Agent Properties](#)
 - [Running Agent Simulations](#)
 - [The MIMIC Value Space](#)
 - [Generating Traps](#)
 - [Recording a Device](#)
- [Chapter 3: Troubleshooting](#)
 - [Online Help](#)
 - [Inspect the Log](#)
 - [Common Errors](#)

- Common Questions
- Diagnostic Wizard
- Crashes
- Chapter 4: Process
 - Important Concepts
 - MIMIC Process Overview: When to Use the Tools
 - References for Further Reading

1. MIMIC SNMP Agent Simulator Guide

- Overview
- MIMICView Reference
 - Startup
 - Title Bar
 - Canvas
 - Status Bar
 - Log Window
 - Menu Bar
 - File Menu
 - File->Open...
 - File->New...
 - File->Save
 - File->Save As...
 - File->Log
 - File->Exit
 - File->Terminate
 - Edit Menu
 - Edit->Add->Agent...
 - Device Selection
 - Device Configuration
 - Edit->Add->Map...
 - Edit->Configure...
 - Edit->Delete
 - Edit->Cut
 - Edit->Copy
 - Edit->Paste
 - Edit->Find
 - Edit->Select
 - Edit->Invert Selection
 - View Menu
 - View->Type
 - View->Up
 - View->Home
 - View->Previous
 - View->Next
 - View->Refresh
 - Agent Menu
 - Agent->Start
 - Agent->Stop
 - Saving Changed Simulations
 - Agent->Halt
 - Agent->Pause...
 - Agent->Resume
 - Agent->Value Space...
 - Agent->Evaluate Value...
 - Agent->Actions...
 - Simulator Actions
 - Agent->Script...
 - Agent->Timer Scripts...
 - Agent->Trap Destinations...
 - Agent->Generate Traps...
 - Agent->Manage Traps...
 - Agent->Trap Series...
 - Agent->Save...
 - Agent->Statistics...
 - Run Menu
 - Run->Timer Scripts
 - Run->Script Generator
 - Simulation Menu
 - MIB Menu
 - MIB Directory
 - Wizard Menu
 - Access Menu
 - Access->Load...
 - Access->Edit...
 - Access->Save As...
 - Speed Bar Buttons
- MIMICShell Reference
 - MIMICShell Command Line Options
 - MIMIC Package
 - Global Commands
 - Session Commands
 - Agent Commands
 - Value Space Commands
 - Trap Configuration Commands
 - SNMPv3 Configuration Commands
 - Access Control Commands
 - Examples

- Access Control
 - Client Access Control
 - Agent Access Control
- Advanced Configuration
 - Configurable Number of Management Channels
- Updates
 - Update to 4.10
 - Update to 5.10
- Usage Profiling

h. MIMIC Recorder Guide

- Overview
- Reference
 - Live Mode
 - File Mode
 - Other Options
- Creating a Walkfile with mib2walk
- Walkfile Format

i. MIMIC Compiler Guide

- Overview
- Reference
 - Importing a MIB
 - Compiling a MIB
 - Editing a MIB
 - Creating a Custom Simulation
 - Compiling a Simulation
 - Editing a Simulation
 - Creating a Scenario
- MIMIC Simulation Language Reference
 - High-level Syntax
 - Types
 - Constants
 - Arithmetic Functions
 - Nested Expressions
 - System Functions
 - Conditional Operator
 - Value Space Lookup
 - Counter Statistics
 - MIB Tables
 - Relationship between Objects
 - Trap Generation

j. MIMIC Wizards Guide

- MIB Wizard Reference
 - Overview
 - Select MIBs To Import
 - Select Existing MIB Files To Compile
- Discovery Wizard Reference
 - Overview
 - Discovery Wizard Sessions
 - Discovery Method
 - IP Addresses
 - Filters to Limit Discovery
 - SNMP Parameters
 - Recorder Options
 - Discovered Devices
 - Recorded Targets
 - Create Lab Configuration
 - Interrupted Session
- Snapshot Wizard Reference
 - Overview
 - Snapshot Option
 - Simulation Option
 - Choose a Live Target
 - Recorder Options
 - Snapshot Constraints
 - Take Snapshots
 - Create Change Script
- Trap Wizard Reference
 - Overview
 - Series Options
 - Simulation Options
 - Choose a Live Target
 - Recorder Options
 - Trap Capture Options
 - Capture Trap Series
 - Edit Trap Series
- Simulation Wizard Reference
 - Overview
 - Select Walkfile To Load
 - Select MIBs
 - Specify Default Values
 - Edit Object Simulations

- [Select Walkfile To Save](#)
 - [Select Simulation To Create](#)
- [CLI Wizard Reference](#)
 - [Overview](#)
 - [Method Choice](#)
 - [Recording Parameters](#)
 - [Discover Commands](#)
 - [Issue Commands](#)
- [NetFlow Wizard Reference](#)
 - [Overview](#)
 - [Create or modify](#)
 - [Create Config file using PCAP file](#)
 - [Modify Config File](#)
- [sFlow Wizard Reference](#)
 - [Overview](#)
 - [Create or modify](#)
 - [Create Config file using PCAP file](#)
 - [Modify Config File](#)
- [IPMI Wizard Reference](#)
 - [Overview](#)
 - [Create or modify](#)
 - [Record Live Session](#)
 - [Modify Config File](#)
- [Web Wizard Reference](#)
 - [Overview](#)
 - [Create or modify](#)
 - [Create Config file using PCAP file](#)
 - [Modify Config File](#)
- [Tutorial Wizard Reference](#)
 - [Overview](#)
 - [MIBs](#)
 - [Create Simulations](#)
 - [Run Simulations](#)
 - [Customize Simulations](#)
- [Configuration Wizard Reference](#)
 - [Overview](#)
 - [MimicView Configuration](#)
 - [MIMIC Log Configuration](#)
 - [Java Configuration](#)
 - [E-mail Configuration](#)
 - [Profiling Configuration](#)
- [Update Wizard Reference](#)
 - [Introduction](#)
 - [Select Update](#)
 - [Download Images](#)
 - [Prepare For Installation](#)
 - [Extract Files](#)
 - [Move Files](#)
- [Diagnostic Wizard Reference](#)
 - [Introduction](#)
 - [Select Configuration Files](#)
 - [Select MIB Files](#)
 - [Select Simulation Files](#)
 - [Select Walkfiles](#)
 - [Select Snapshot Files](#)
 - [Select Discovery Files](#)
 - [Select Trap Series Files](#)
 - [Select Log Files](#)
 - [Review Selected Files](#)
 - [Send Files](#)
- [Sanity Wizard Reference](#)
 - [Introduction](#)
 - [Select Items](#)
 - [Detection Results](#)
- [Performance Wizard Reference](#)
 - [Introduction](#)
 - [System Information](#)
 - [Setup Test](#)
 - [Performance Results](#)
 - [Send Files](#)
- [Protocol Wizard Reference](#)
 - [Introduction](#)
 - [Select Protocols](#)
 - [Supply License Keys](#)
 - [Results Installing/Uninstalling Protocols](#)

[MIMIC WEBUI Guide](#)

- [Requirements](#)
- [Installation](#)
- [WEBUI Server](#)
- [User Guide](#)

[MIMIC Utilities Guide](#)

- [grapher](#)
- [oidinfo](#)

- [walkfile converters](#)

4. [MIMIC Tcl API Guide](#)

5. [MIMIC Java API Guide](#)

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)

6. [MIMIC Perl API Guide](#)

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)

7. [MIMIC C++ API Guide](#)

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)
- [Actions](#)

8. [MIMIC Python API Guide](#)

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)
- [Jython](#)

9. [MIMIC PHP API Guide](#)

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)

10. [MIMIC Javascript API Guide](#)

- [Requirements](#)
- [Class Reference](#)
- [Javascript API Server](#)
- [Examples](#)

11. [MIMIC Go API Guide](#)

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)

12. [MIMIC OpenAPI Guide](#)

- [Requirements](#)
- [API Reference](#)
- [Examples](#)

13. [MIMIC Cable Modem Simulator Guide](#)

- [Overview](#)
- [Implementation](#)
- [Installation](#)
- [MIBs](#)
- [Action Scripts](#)

14. [MIMIC DHCP Protocol Module Guide](#)

- [Installation](#)
- [Using DHCP from MIMICView](#)
- [Using DHCP from MIMICShell](#)

15. [MIMIC TFTP Protocol Module Guide](#)

- [Installation](#)
- [Using TFTP from MIMICView](#)
- [Using TFTP from MIMICShell](#)
- [Compatibility](#)

16. [MIMIC Time-Of-Day Protocol Module Guide](#)

- [Installation](#)
- [Using TOD from MIMICView](#)
- [Using TOD from MIMICShell](#)

32. MIMIC IOS Simulator Guide

- Overview
- Implementation
- Installation
- IOS Explorer
- IOS Recorder

33. MIMIC Telnet Protocol Module Guide

- Installation
- Using Telnet from MIMICView
- Using Telnet from MIMICShell

34. MIMIC SSH Protocol Module Guide

- Installation
- Using SSH from MIMICView
- Using SSH from MIMICShell

35. MIMIC SYSLOG Protocol Module Guide

- Installation
- Using SYSLOG from MIMICView
- Using SYSLOG from MIMICShell
- Compatibility

36. MIMIC IPMI Protocol Module Guide

- Installation
- Using IPMI from MIMICView
- Using IPMI from MIMICShell
- Message Dictionary
- IPMI Proxy
- IPMI Recorder
- Interoperability
- Limitations

37. MIMIC Server Proxy Protocol Module Guide

- Installation
- Using PROXY from MIMICView
- Using PROXY from MIMICShell
- Compatibility

38. MIMIC NETFLOW Protocol Module Guide

- Installation
- Using NETFLOW from MIMICView
- Using NETFLOW from MIMICShell
- Recording Netflow
- Simulation Configuration
- Compatibility

39. MIMIC SFLOW Protocol Module Guide

- Installation
- Using SFLOW from MIMICView
- Using SFLOW from MIMICShell
- Simulation Configuration
- Compatibility

40. MIMIC WEB Services Module Guide

- Installation
- Using WEB from MIMICView
- Using WEB from MIMICShell
- Recording WEB Sessions
- Simulation Configuration
- Compatibility

41. MIMIC MQTT Protocol Module Guide

- Installation
- Using MQTT from MIMICView
- Using MQTT from MIMICShell
- Recording MQTT Sessions
- Simulation Configuration
- Compatibility

42. MIMIC CoAP Protocol Module Guide

- Installation

- [Using CoAP from MIMICView](#)
- [Using CoAP from MIMICShell](#)
- [Recording CoAP Sessions](#)
- [Simulation Configuration](#)
- [Compatibility](#)

[37. Appendix A: Directory Structure](#)

[38. Appendix B: Pre-compiled MIBs](#)

[39. Appendix C: Common Error Messages](#)

- [Agent Simulator](#)
- [Recorder](#)
- [Compiler](#)

[40. Appendix D: Frequently Asked Questions](#)

[<< Previous Section](#) [Next Section >>](#)



[<< Previous Section](#) [Next Section >>](#)

MIMIC Release Notes

New functionality in 20.00

Simulator

Agent host address can now be IPv6 address
NETCONF simulation

Miscellaneous

New functionality in previous releases

New functionality in 19.20

Simulator

[File versioning](#) of lab configuration files
[Go API](#)
[MIMIC OpenAPI REST API](#)

Miscellaneous

Major CLI Wizard improvements for default rule generation

New functionality in 19.10

Simulator

OpenSSL 1.1.1b for TLS1.3

Miscellaneous

Support OpenAPI in WEB module

New functionality in 19.00

Simulator

New [WEBUI](#) web-based user interface
New [MQTT 5.0](#) support, in addition to existing MQTT 3.1.1

Miscellaneous

Match tool in [speed bar](#) and dialogs to quickly filter from large numbers of items.
Newest tool in [File Browser dialog](#) to sort by modification.

New functionality in 18.10

Simulator

New [Javascript API](#)

Miscellaneous

Control MQTT publish simulation at runtime with [mimic agent protocol msg MQTT client message](#) command.
Latest [IPFIX](#) fields from [IANA](#)
[Agent -> Statistics](#) graphing to Graphite / Grafana

New functionality in 18.00

Simulator

New [CoAP Simulator](#)
SIMULATE language extensions for [Variable Store access](#) for parametrized simulations

Miscellaneous

New functionality in 17.10

Simulator

Remove support for Solaris platform.

Miscellaneous

MIMICview now can control a subset of all agents, allowing multiple parallel GUI instances
Tcl/Tk 8.6.6 for MIMICview, MIMICshell

New functionality in 17.00

Simulator

SIMULATE language extensions for [Value Space access](#) for faster complex built-in simulations
[NetFlow](#) module now exports over IPv6 in addition to IPv4
[MQTT](#) client can now retrieve authentication parameters, eg. from [OAuth 2.0](#), for CONNECT

Miscellaneous

Parallel management channel threads for multi-threaded MIMIC clients
MIMICview Explorer View: more than 10000 agents in root map causes them to automatically be moved to submaps; configurable in [Maps](#) tab of [Configuration Wizard](#)

New functionality in 16.20

Simulator

SNMPv3 SHA-2 authentication as specified in [RFC 7860](#)

Miscellaneous

[MQTT Recorder](#) to create simulations from real-world sensors

New functionality in 16.10

Simulator

[NetFlow module](#) now supports Cisco PFR Reporting fields; 8-byte fields; dynamic flow generation changes at runtime
New [PHP API](#)

Miscellaneous

[Enhanced Topology Wizard](#) with topology visualization

New functionality in 16.00

Simulator

New [MQTT Simulator](#)
Sped up loading large lab configuration files.
Major performance enhancements on 8+-CPU hosts for all protocols

Miscellaneous

[Python API](#) now supports Python 3.x in addition to 2.x
Self-documenting lab configuration files with [user editable comments](#)

New functionality in 15.20

Simulator

Better IPv6 support in Windows 7 and later: 10 times faster agent start
Content type negotiation in [WEB services module](#)
[SSH](#) can now be the transport for applications other than CLI / IOS / JUNOS (TELNET module).

Miscellaneous

Favorites in [Value Space Browser](#)

New GUI [Variable Store dialog](#) for [variable store commands](#)
[Web Wizard](#) changes WEB module configurable values in real-time

i. New functionality in 15.10

Simulator

[WEB module](#) supports SPNEGO-based Kerberos and NTLM HTTP Authentication on Windows
WEB module can now define multiple services per agent

Miscellaneous

Optional 64-bit [Recorder](#) to handle large walkfiles (greater than 4GB)

i. New functionality in 15.00

Simulator

MIMIC now also runs on [Windows 10](#)
[30,000 agents](#) on Windows 8, Windows 10

Miscellaneous

Tcl/Tk 8.6.3 for MIMICview, MIMICshell
[10000-foot view](#) for large-scale simulations
better support for [saving](#) and [merging](#) subsets of lab configurations / maps

i. New functionality in 14.20

Simulator

Simulate [DHCP](#) relay

Miscellaneous

i. New functionality in 14.11

Simulator

Major improvements in performance on Windows.

Miscellaneous

MIMICView [File -> Connect](#) dialog to connect to other instances of MIMIC
[IPMI Wizard](#) can now change IPMI simulations at run-time
Improved [SFLOW Wizard](#)

i. New functionality in 14.00

Simulator

Major performance enhancements targetted to multi-CPU systems
100,000 agents on [64-bit Linux](#)
[sFlow](#) module now generates up to 60000 exports per minute

Miscellaneous

New [Web Wizard](#) to create Web Services simulations
[Configurable NetFlow information elements](#)
major improvements to [NetFlow Wizard](#)
[Discovery Wizard](#) now also imports previously recorded walkfiles to generate a network simulation
[Discovery Wizard](#) also discovers LLDP Remote data and OSPF Neighbors
[Diagnostic Wizard](#) now dumps MIMIC thread information
New [IPMI Wizard](#) to create IPMI simulations

i. New functionality in 13.20

Simulator

Major performance enhancements targetted to multi-CPU systems

Miscellaneous

New [SOAP Recorder](#) to create WEB services simulations
[sFlow Recorder](#) now also records expanded flow samples

[Discovery Wizard](#) now also uses Cisco's CDP cache table

1. New functionality in 13.10

Simulator

Multiple flow sources in [NetFlow Simulator](#)
New [sFlow Recorder](#)
New WSDL compiler for the [WEB Services](#) module
Tcl 8.5.14

Miscellaneous

1. New functionality in 13.00

Simulator

New [WEB Services Simulator](#)

Miscellaneous

[MIMIC Recorder](#) now uses version 2c GETBULK walk by default
[Script Generator](#) now generates Python code in addition to Tcl, Java, Perl and C++
[Agent -> Script...](#) now runs Perl and Python scripts
[Snapshot Wizard](#) can now schedule snapshots

1. New functionality in 12.20

Simulator

[SFLOW module](#) supports nested structures, discriminant unions
Tcl 8.5.12

Miscellaneous

New [SFLOW Wizard](#)
Improved [NetFlow Wizard](#)
[File -> Open](#): more control to add lab configurations into running configuration
[CLI Wizard](#) now supports discovering Force10 CLI

1. New functionality in 12.10

Simulator

SFLOW module supports XDR fields

Miscellaneous

MIMIC now runs on Windows 8, Windows Server 2012, Fedora 17.

1. New functionality in 12.00

Simulator

New [sFlow Simulator](#)
Load lab configurations incrementally with new [mimic load](#) `starting-agent` optional argument

Miscellaneous

[MIMICView](#): [Agent Change Simulation dialog](#) to easily customize simulation of MIB objects
[MIMICView](#): re-design menu items to make features more accessible
[QuickStart](#): 1-minute videos on YouTube
MIMIC now runs on Fedora 16 and Ubuntu 11.10.

1. New functionality in 11.40

Simulator

Improvements in [Netflow protocol module](#):

- optionally generate template and data flowsets in same packet
- variable-length records in IPFIX
- private enterprise numbers in IPFIX

Miscellaneous

h. New functionality in 11.30

Simulator

Minor improvements in [Netflow protocol module](#):

- enumerated addresses in addition to ranges

Miscellaneous

Renamed IOS Wizard to [CLI Wizard](#)
CLI Wizard can now create rules interactively or from a transcript

h. New functionality in 11.20

Simulator

Support for IPFIX (NetFlow version 10) in [Netflow protocol module](#).
Licensing for running on virtual machines (eg. VMWare)
Tcl 8.5.9

Miscellaneous

trapper can now import from PCAP files containing traps

h. New functionality in 11.10

Simulator

Major improvements in [Netflow protocol module](#):

- multiple flow definitions per flowset
- bi-directional flows
- 64-bit values
- sequential values/addresses for predictable flows
- dynamic reconfiguration while running
- 1000 flowsets / second / agent
- IPv6 addresses

Miscellaneous

NetFlow Wizard

h. New functionality in 11.00

Simulator

50,000 agents on [64-bit Linux](#)
[Netflow protocol module](#)
UDP support in [PROXY server](#)
MIMIC now runs on Fedora 13 and 14.

Miscellaneous

--engineid parameter to MIMIC Recorder to specify Engine ID rather than discovering it

h. New functionality in 10.30

Simulator

Deterministic server switching in PROXY server
Dynamic content manipulation in PROXY server

Miscellaneous

h. New functionality in 10.20

Simulator

MIMIC now runs on Fedora 11 and 12.
IPv6 support for PROXY server; also proxies IPv4 to IPv6 and viceversa

Miscellaneous

i. New functionality in 10.10

Simulator

Source address indexing for TCP.

Miscellaneous

SNMPv3 support in Discovery Wizard.

. New functionality in 10.00

Simulator

[Server Proxy Protocol Module](#)
Tcl 8.5.7
[IPMI](#) multi-session support
[Telnet](#) NAWs negotiation

Miscellaneous

[Protocol Wizard](#) to ease installation of optional protocol modules
[Java API](#) on Unix uses Java Unix Domain Sockets for local management channel yielding another 20% performance improvement
Hyperlinks to detailed description in [Log Viewer](#)

i. New functionality in 9.40

Simulator

Complete support for Windows 7.

Miscellaneous

[C++ API performance improvements](#)

i. New functionality in 9.30

Simulator

SNMPv3 context-engine-id indexing in addition to engine-id indexing.
Tcl 8.5.6

Miscellaneous

[Performance Wizard](#)

i. New functionality in 9.20

Simulator

Support Fedora 10.
Support AES-192 and AES-256 privacy.
Perl interface is now 30% faster on Windows due to local channel.
Support [Solaris Zones](#).

Miscellaneous

i. New functionality in 9.10

Simulator

Support OpenSolaris.
Perl interface is now 50-100% faster.

Miscellaneous

Improved Topology Editor.

. New functionality in 9.00

Simulator

Support Windows Server 2008.
Enable full tracing to better diagnose encrypted PDUs.
Tcl 8.5.2

Miscellaneous

Create fast simulations with --fast option to Recorder

New functionality in 8.40

Simulator

Can now send multiple varbinds with same object in TRAPs.

Miscellaneous

Major performance improvements in SNMP PDU handling.

New functionality in 8.31

Simulator

Complete support for Windows Vista.
64-bit support for Solaris x86.

Miscellaneous

Track available virtual memory.
Tcl/Tk 8.4.14
Tix 8.4.2
Tcllib 1.9
Major performance enhancement starting agents with SNMPv3.

New functionality in 8.20

Simulator

MIMIC Universal: 20,000 agent support on Windows, Solaris and Linux.

Miscellaneous

Major performance enhancements in MIMICView.
[Organize your devices in categories.](#)
[MIB directory](#) organized by manufacturer.

New functionality in 8.11

Simulator

[IPMI 2.0 support](#) in addition to IPMI 1.5
64-bit executable support on [Windows](#), [Solaris/Sparc](#) and [Linux](#), allowing for large simulations going beyond 4GB of virtual memory.
New platform support for [Windows Vista](#)

Miscellaneous

[Tutorial Wizard](#)
Configure [startup action scripts](#) for agents.
More than 1800 pre-compiled MIBs.

New functionality in 8.00

Simulator

[IPMI support](#)
IPv6 support for Windows XP, 2003.
Complete control over OID traversal through [mimic action jump](#) in GETNEXT action script.
Latest Tcl/Tk 8.4.13

New functionality in 7.30

Simulator

SNMPv3 USM module supports the CFB128-AES-128 privacy protocol, as specified in RFC 3826.
SSHv2 support in addition to SSHv1.5.
Latest Tcl/Tk 8.4.11

New functionality in 7.20

Simulator

IPv6 support for Solaris 10.
SNMPv3 user management support.

Miscellaneous

Disconnect links in Virtual Lab.

New functionality in 7.11

Simulator

New platform support for Solaris 10.
5,000 agents on Windows 2000 and XP. Up from 2,000 agents.
SNMPv3 engine-id indexing allows multiple agents with same IP address and port but different engine-id.
Latest Tcl/Tk 8.4.8

Miscellaneous

Platform-native look-and-feel for MIMICView.
Major performance improvements in MIMICView.

New functionality in 7.00

Simulator

Full support for IPv6 on [Linux](#), [Solaris](#).
[Secure shell \(ssh\)](#)
[SYSLOG](#)

Miscellaneous

More IOS commands.

New functionality in 6.30

Simulator

Miscellaneous

Telnet server can now operate on [multiple IP aliases](#) for an agent.
More IOS commands.

New functionality in 6.20

Simulator

Load-time transform of data for wholesale changes to data (eg. network address change in VLAB).

Miscellaneous

Virtual Lab on Unix platforms.
[Improved exercises for Virtual Lab](#).

New functionality in 6.10

Simulator

New platform support for [Windows Server 2003](#) for upto 10,000 agents
[SNMP over TCP](#)

New Functionality in 6.00

Simulator

[Source address indexing](#) allows multiple simultaneous virtual networks with overlapping addresses to be accessible from different management stations.
[TL/1 Support in Telnet module](#)
New platform support for RedHat 9.

Miscellaneous

[Virtual Lab GUI](#).
[IPv6 support in Recorder](#).

l. New Functionality in 5.40

Simulator

[Enable/disable MIB subtrees](#).
New platform support for Solaris 9, RedHat 8.

Miscellaneous

[C++ API](#).
C++ Actions.
Integrated Development Environment (IDE).
[Script Generator](#).
Link into MIB source from Simulation Wizard.

l. New Functionality in 5.30

Simulator

New platform support for [Windows XP](#).
[Variable store](#).
[Multiple agent start/stop action scripts](#).
Option negotiation for Telnet.

Miscellaneous

[IOS Explorer](#).
[Trap Wizard](#).
[64-bit arithmetic in MIMICShell scripts](#).
[--static option in MIMIC Recorder](#).
[Sanity Wizard](#).
[Usage Profiling](#).

l. New Functionality in 5.20

Simulator

[Cisco IOS Simulator](#).
[MGCP Protocol module](#).
Persistent, per-user lab configuration.
Agent access control.
[Configurable number of management channels](#).

Miscellaneous

Recorder can now create simulations of [unknown MIB objects](#).
[IOS Recorder](#)
[Snapshot Wizard](#).

l. New Functionality in 5.10

Simulator

[Telnet Protocol module](#).
Per agent timer scripts.
[Action scripts on INFORM response and timeout](#).
Major performance improvements.
New platform support for [Windows Me](#).

Miscellaneous

[Perl API](#)

l. New Functionality in 5.00

Simulator

[Cable Modem Simulator for DOCSIS-compliant cable modems](#).
[TFTP Protocol module](#).
[Time-of-day Protocol module](#).
Generate INFORM PDU in addition to TRAP and NOTIFICATION.
[SIMULATE {random \(low, high\)}](#) returns random value for static objects.
Per-agent trap destinations in addition to global trap destinations.
[Remote host-based access control](#).

Recorder

SNMPv2c, SNMPv3 support, in addition to SNMPv1

Miscellaneous

Trap management in MIMICView.

Trap Recorder records INFORM PDU in addition to TRAP and NOTIFICATION.
Java-based Topology Editor with major enhancements.
Tcl/TK 8.3.

i. New Functionality in 4.40

Simulator

Scalable DHCP.
AUGMENTed tables.
Protocol-dependent statistics.

Miscellaneous

Graphing capability.
Java API for MIMIC clients.

j. New Functionality in 4.30

Simulator

DHCP client simulation.
MIMIC Global: 10,000 agent support on Solaris and Linux
New platform support for Solaris 8 (both Sparc and Intel).
Better simulation visualization/control through Value Space Browser.
Major performance enhancements.

Miscellaneous

New Topology Editor to display/modify device interconnection.

k. New Functionality in 4.20

Simulator

New platform support for Windows 2,000 and Solaris on Intel.
MIMIC Industrial: 5,000 agent support on Solaris and Linux
Ability to simultaneously run multiple versions of a MIB.
More control from action scripts: drop PDUs, variable binding processing,...

Miscellaneous

l. New Functionality in 4.10

Simulator

MIMIC Millennia: 2,000 agent support on Solaris and Linux
SNMPv3: USM with SHA1; tested with SNMP Research BRASS
Update Wizard: update to new versions of MIMIC
Configuration Wizard: configure MIMIC
Actions on traps.

Miscellaneous

m. New Functionality in 4.00

Simulator

MIMIC Millennia: 2,000 agent support on Windows NT
SNMPv3: USM with MD5, full VACM
GUI support for large-scale simulations: hierarchical maps, Explorer-like mode
MIB Wizard: compile MIBs, handle dependencies among MIBs
Discovery Wizard: record large networks
Actions on agent start/stop, timer-based.

Miscellaneous

More MIBs: Accelerated Networks, APC, AT&T, Compaq, Intel, Lucent, Nortel, Sonoma Systems, latest RFCs
Now more than 1,000 precompiled MIBs.

n. New Functionality in 3.30

Simulator

Support multiple users simultaneously.
Simulation Wizard: easy-to-use front-end to create simulations.

Miscellaneous

Redback, IBM MIBs
Now more than 800 precompiled MIBs.

New Functionality in 3.20

Simulator

Configurable transit delay per agent to simulate geographically dispersed WAN scenarios.
MIMICShell action scripts can now be invoked upon receipt of PDUs to accomplish desired side effects.
Community string indexing allows multiple agents with same IP address and port but different community string.
More scalability choices: MIMIC Lab product with 25 agents.

Miscellaneous

Xylan MIBs
Now more than 650 precompiled MIBs.

New Functionality in 3.10

Simulator

SNMPv3 with noAuth, noPriv USM.
Efficient loading of large tables (1,000+ entries).
Windows 98 support.

Miscellaneous

Trap recorder.

New Functionality in 3.00

Simulator

Full support for SNMPv2c and SNMPv2, including GETBULK, authentication, parties, contexts, views, v2 TRAP.
Multi-lingual agents (SNMPv1, SNMPv2c, SNMPv2).
1,000 agent support on Solaris.
SET PDU syntax checking.

Compiler

Full SMIv2 support.

Miscellaneous

Trap generation dialog.

New Functionality in 2.40

Simulator

Higher scalability: MIMIC Enterprise edition can now simulate up to 1,000 agents on one Windows NT PC. The MIMIC Campus edition simulates up to 500 agents.
Multi-NIC support: each agent instance can be configured independently to use a specific network interface card.

Recorder

Improved robustness: recover from error responses.

Miscellaneous

Automated creation of simulation from set of MIBs.
More than 600 precompiled MIBs are supplied.

New Functionality in 2.30

Recorder

Options to handle large MIBs (maximum table size, exclude object trees).
Option to slow down retrieval (in order not to overwhelm sensitive device with management requests).

Recorder now automatically recovers from agent's inability to handle multi-variable GETNEXT PDUs.

Simulator

Multiple community strings per agent.
Value space lookup failure now returns DEFVAL clause of MIB object. This simulates correct dynamic row creation semantics.
Solaris native threads support takes advantage of multi-processor systems.
More efficient/accurate trap generation.

Miscellaneous

TCL package with MIMIC extensions loadable into any TCL-based shell.
More robust scripting with TCL-style error handling (catch).
Better diagnostics for all tools.
More than 550 precompiled MIBs are supplied.

New Functionality in 2.23

Windows Support

Windows NT 4.0 is fully supported. Windows 95 is supported with minor limitations.

Computer Associates Unicenter(R) TNG Support

Support for Unicenter TNG agents.

Year 2000

Minor Year 2000 fixes.

Miscellaneous

Agents can be instructed to drop PDUs at a configurable rate.
Recorder can record subsets of large tables.
More than 500 precompiled MIBs are supplied.

New Functionality in 2.20

RedHat Linux 5.x Support

RedHat Linux 5.x is fully supported.

Configurable Network Mask

You can now set any subnet mask for each agent instance.

Miscellaneous

Scripting language has been improved: you can now create/delete agents, pause at any time, etc.
Performance improvements.
MIMICView is now based on the latest Tcl/Tk 8.0.

New Functionality in 2.10

SMIv2 BITS

The SMIv2 BITS construct is now supported in MIBs such as RMON-2, HCRMON, etc.

Maximum PDU size

The maximum PDU size is configurable per agent instance. MIMIC responds with the TOO_BIG error status for PDU overflow.

PDU Tracing

SNMP PDUs can be traced per agent instance. The PDU type and each variable are logged.

New Functionality in 2.00

SNMP SET

This release adds support for SNMP SETs. A management application can perform an SNMP SET on all objects that have read-write access in the view with the agent's write-community name.
MIMIC performs syntactical checking on the values, to make sure that they conform in type and that the value is legal given SIZE and range restrictions.
New entries are added to tables by performing a SET on any column in the new row. New objects assume their default value as specified in the DEFVAL clause.

Traps

This release provides for TRAP PDU generation. The "trap_periodic" simulation generates periodic traps with the specified frequency. The following aspects of trap generation are configurable at run-time for each trap for each agent instance:

- frequency
- cutoff time
- variable values

For example, you can generate the linkUp trap every 15 seconds for the next 300 seconds, and send a different ifIndex for each.

250 Agents

MIMIC Professional edition scales up to 250 agent simulations on all supported platforms.

RMON2 support

MIMIC fully supports MIB objects that are 64-bit counters, such as those in the IF-MIB (RFC 1573).

Persistence

MIMIC will now give you the option of saving changed simulations when stopping a simulation. Any simulation that is changed via user-interface Set Value or SNMP SET requests will ask you to save it when you use Agent-->Stop. The changed values are accessible in subsequent simulation sessions.

Scripting

In addition to the interactive graphical user interface in MIMICView, simulations are now accessible in batch-mode scripting via the new [MIMICShell](#) tool `mimicsh`. `mimicsh` provides a TCL-based scripting environment with access to MIMIC. All operations that can be done with MIMICView in interactive mode can be done in scripts in `mimicsh`.

Miscellaneous

At least 100 more MIBs have been added bringing the total above 400.

Several GUI improvements have been implemented:

- Agent icons are now displayed with configurable bitmaps. This gives at-a-glance indication of the simulation. The bitmaps are determined by the device name, and the regular expressions of bitmaps in the `bitmap/` subdirectory.
- Right-clicking on an agent icon brings up the Agent menu.
- Per-agent statistics are now available with [Agent-->Statistics](#).

Known problems

1. [Linux limitations](#)
2. [Solaris limitations](#)
3. [Windows limitations](#)

Please consult the [Frequently Asked Questions](#) section for resolved problems.

[<< Previous Section](#) [Next Section >>](#)



Linux Installation Instructions

Table of Contents

- [Overview](#)
- [Graphical User Interface](#)
- [Swap Space](#)
- [1000/2000/5000/10000 agent support](#)
- [20000/50000/100000 agent support](#)
- [Yellow Pages with large configurations](#)
- [Performance impact of certain daemons](#)
- [IPv6 support](#)
- [Core Dumps](#)
- [Filesystem](#)
- [NFS](#)
- [Shared libraries](#)
- [64-bit support](#)
- [Bonded Interfaces](#)
- [ARP Cache](#)
- [Known Problems](#)

1. Overview

MIMIC runs on any version of Linux with a kernel version greater than 2.0, but preferably [Red Hat Linux](#) Fedora Core 19 or later (kernel version 3.14 or later) or Red Hat Enterprise Linux 7. MIMIC runs on newer versions of Linux (Fedora 19 or later) in 64-bit mode.

Although we have reports of MIMIC running on a variety of distributions (such as Debian, SuSE, Caldera, etc), the latest release has been tested at Gambit to run out-of-the-box with

- Fedora 19 through 30
- Enterprise Linux 7, 8
- CENTOS 7, 8

1. Graphical User Interface

Even though MIMIC can be run entirely without a graphical user interface (GUI), using a GUI initially will speed your tasks significantly. The native [MIMICview](#) GUI is a X Windows client and requires an [X server](#) either on the local desktop or remotely.

To display on Microsoft Windows we have had success with the [Cygwin/X](#) or [MobaXterm](#) X servers, but other implementations like Xming do not work as well. On headless machines you can run a [RDP](#) implementation such as [Xrdp](#) to display on Windows desktops.

Additionally, any window manager running on the desktop will impact the behavior of the MIMICview GUI. For example, keyboard focus is entirely dependent on the window manager. If you use keyboard control of MIMICview, the window manager focus policy impacts the behavior after dismissing dialogs.

1. Swap Space

MIMIC is a specially memory-intensive application, and it needs plenty of RAM and swap space for the more complex device simulations. We recommend an absolute minimum 64MB of RAM and 128MB of swap space to start. Verify that you have enough swap space with the following command from any shell:

```
cat /proc/meminfo
```

It should display something like

```
% cat /proc/meminfo
MemTotal:      1000448 kB
MemFree:       37100 kB
...
SwapTotal:    1000448 kB
SwapFree:     1000448 kB
...
```

This system has 1 GB of RAM (MemTotal), and 1 GB of swap space (SwapTotal).

You should as a rule of thumb have twice the swap space as your physical memory.

If you need more swap space, you can easily create it with the `mkswap(8)` and `swapon(8)` commands. A 2 GB swap file on the local filesystem can be created with the following commands:

```
# dd if=/dev/zero of=SWAP bs=1M count=2048
2048+0 records in
2048+0 records out
2147483648 bytes (2.1 GB) copied, 63.9475 s, 33.6 MB/s

# mkswap SWAP
Setting up swapspace version 1, size = 2147483648 bytes
# swapon [ABSOLUTE-PATH-TO]/SWAP
```

Different versions of Linux on Intel systems impose different limits on the amount of virtual memory accessible for the MIMIC Simulator. If you run [64-bit Linux](#) the VM limits are of no concern, but on 32-bit systems the absolute limit is below 4 GB due to 32-bit addressing limitations. We have run MIMIC on common Red Hat distributions with upto approximately 3 GB of virtual memory.

1. 1000/2000/5000/10000 agent support

You need to perform an extra kernel configuration step on Linux prior to running MIMIC with a large numbers of agents, to increase the global file descriptor table. Eg. to enable 5000 agents, we recommend:

```
# cat /proc/sys/fs/file-max
4096
# echo "8192" > /proc/sys/fs/file-max
# cat /proc/sys/fs/file-max
8192
```

To apply this parameter permanently, you need to add this command to the boot startup files (eg. in `/etc/rc*`). Contact your MIS department on this issue.

1. 20000/50000/100000 agent support

You can run 20000 and more agents only on [64-bit Linux](#), and need to apply the appropriate configuration step detailed in the [previous section](#).

1. Yellow Pages with large configurations

If you are running MIMIC on a system that is using Yellow Pages for name resolution, then you may encounter the following problem: the Yellow Pages client may lock up if you are running a large-scale configuration. The symptoms include delayed command response, error messages in the system log of the form `ypbind: YPBINDPROC_DOMAIN: Domain not bound`.

This is likely due to ypbind using broadcast to find the Yellow Pages server, as configured in the file `/etc/yp.conf` with this entry

```
domain SOMETHING broadcast
```

To solve the problem, comment out the broadcast line and specify an explicit Yellow Pages server, with an entry such as

```
ypserver 192.9.200.1
```

1. Performance impact of certain daemons

Linux is a general-purpose operating system, and as such runs a variety of daemons to provide services. Certain daemons interfere with the operation of MIMIC. This is a partial list, if you encounter any others, please let us know:

- `avahi-daemon`
The `avahi-daemon` hogs the CPU for large configurations, because it tries to configure DNS entries for the configured IP aliases.
- `NetworkManager`
The `NetworkManager` hogs the CPU trying to maintain all those IP aliases that get created and removed when starting and stopping agents.
- `dnsmasq`
You can stop this since you don't need to run a DNS server.
- `ibus-daemon`
We have seen GUI actions slow down considerably with this running. You can stop this if you don't need multilingual input.

To diagnose the problem, look at CPU consumption of running processes. To solve the problem, stop the daemon.

1. IPv6 Support

IPv6 is supported on all Linux platforms that MIMIC runs on.

Prior to running agents with IPv6 addresses, the loadable IPv6 protocol kernel module needs to be installed. This is a boot-time configurable, eg.

```
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:A0:24:07:C7:8A
          inet addr:192.9.200.23  Bcast:192.9.200.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```



```

RX packets:2740928883 errors:80 dropped:0 overruns:125 frame:80
TX packets:2196208374 errors:0 dropped:0 overruns:0 carrier:96
collisions:51506073 txqueuelen:100
RX bytes:286144626 (272.8 Mb) TX bytes:1540102588 (1468.7 Mb)
Interrupt:10 Base address:0x220

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:1836705765 errors:0 dropped:0 overruns:0 frame:0
      TX packets:1836705765 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:3869230693 (3689.9 Mb) TX bytes:3869230693 (3689.9 Mb)

# modprobe ipv6
# ifconfig -a
eth0  Link encap:Ethernet HWaddr 00:A0:24:07:C7:8A
      inet addr:192.9.200.23 Bcast:192.9.200.255 Mask:255.255.255.0
      inet6 addr: fe80::2a0:24ff:fe07:c78a/10 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:2740929031 errors:80 dropped:0 overruns:125 frame:80
      TX packets:2196208477 errors:0 dropped:0 overruns:0 carrier:96
      collisions:51506073 txqueuelen:100
      RX bytes:286160140 (272.9 Mb) TX bytes:1540116436 (1468.7 Mb)
      Interrupt:10 Base address:0x220

lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:1836705775 errors:0 dropped:0 overruns:0 frame:0
      TX packets:1836705775 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:3869232221 (3689.9 Mb) TX bytes:3869232221 (3689.9 Mb)

sit0  Link encap:IPv6-in-IPv4
      NOARP MTU:1480 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)

```

When using ping6 to check connectivity to an IPv6 target, for link-local IPv6 destinations you must supply the -I option, eg.

```

% ping6 -I eth0 fe80::a00:20ff:feb0:2780
PING fe80::a00:20ff:feb0:2780(fe80::a00:20ff:feb0:2780) from fe80::d0:b760:a3f7:eth0: 56 data bytes
Warning: time of day goes back, taking countermeasures.
64 bytes from fe80::a00:20ff:feb0:2780: icmp_seq=0 hops=255 time=4.618 msec
64 bytes from fe80::a00:20ff:feb0:2780: icmp_seq=1 hops=255 time=244 usec
64 bytes from fe80::a00:20ff:feb0:2780: icmp_seq=2 hops=255 time=235 usec

```

```

--- fe80::a00:20ff:feb0:2780 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.235/1.699/4.618/2.064 ms

```

When using netstat or route to check or configure routing tables, you need to use the -A inet6 command line option, eg.

```

# netstat -A inet6 -r -n
Kernel IPv6 routing table
Destination          Next Hop             Flags Metric Ref    Use Iface
::1/128              ::                  U      0     404    0 lo
3ffe::1/128          3ffe::1             UC     0     1      1 eth0
3ffe:bc0:c56:1::1/128  ::                  U      0     0      0 lo
3ffe:bc0:c56:1::2/128  ::                  U      0     0      0 lo
fe80::/10            ::                  UA     256   0      0 eth0
ff00::/8             ::                  UA     256   0      0 eth0
::/0                 ::                  UDA    256   0      0 eth0

# route add -A inet6 3ffe::/16 eth0

```

For more details see Peter Bieringer's Linux IPv6 HOWTO document, eg. [here](#).

Please check effects of duplicate address detection [below](#).

l. Core Dumps

To allow core dumps from MIMICD on new versions of Linux, you need to set the [ulimit for core dumps](#) to unlimited.

On Linux versions that change the [kernel core pattern](#) which may prevent core dumps, You can inspect the behavior with something like this:

```

$ sysctl kernel.core_pattern
kernel.core_pattern = |/usr/share/apport/apport %p %s %c %P

```

Now you have the choice of enabling the `apport` utility or restoring the core dump behavior:

```
$ sudo sysctl kernel.core_pattern=core
kernel.core_pattern = core
```

The rest of this section only applies to MIMIC before release 9.00.

The MIMIC Simulator `mimicd` runs as a `setuid-root` daemon on Linux. By default, core files are not produced if it terminates abnormally. If you see the `mimicd` crashing, you need to run it as root to enable core dumps to help us diagnose the problem:

```
su
export MIMIC_IGNORE_SIGSEGV=1
ulimit -c unlimited
./mimicd
```

Filesystem

If you are setting up a partition on your hard disk to contain MIMIC data, you will want to consider tuning the block and inode allocation of the filesystem. Since MIMIC data typically consists of many small files, on Unix filesystems it uses up a filesystem resource called "inodes". When these run out, you will get error messages such as "file system full", even though the output of `df (1)` shows plenty of space available.

To change the inode allocation from the default for the `mkfs (8)` utility, use the `-b` command line option to reduce the block-size to 1024 bytes (the smallest possible, since fragments are not supported), and the `-i` command line option to set the bytes-per-inode to 1024 (to allocate the maximum number of inodes).

For details see the `mkfs` or `mke2fs` man pages or consult your system administrator.

In modern Linux systems you can use a [loop device](#) to configure a file as a **virtual disk** to be formatted and mounted with the preferred parameters. This is the recommended procedure to setup a 10GB filesystem, eg. as root user in `/usr/local`:

```
# pwd
/usr/local

# dd if=/dev/zero of=mimic.img bs=1MB count=10000
10000+0 records in
10000+0 records out
10000000000 bytes (10 GB) copied, 119.391 s, 83.8 MB/s

# ls -la mimic.img
-rw-r--r-- 1 root root 10000000000 Mar  7 13:47 mimic.img

# mkfs -t ext3 -b 1024 -i 1024 mimic.img
mke2fs 1.42.7 (21-Jan-2013)
mimic.img is not a block special device.
Proceed anyway? (y,n) y
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
9762144 inodes, 9765624 blocks
488243 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=76995072
1194 block groups
8184 blocks per group, 8184 fragments per group
8176 inodes per group
Superblock backups stored on blocks:
    8185, 24553, 40921, 57289, 73657, 204601, 220969, 401017, 662905,
    1023001, 1988713, 2807113, 5115001, 5966137
```

```
Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
# mount -t auto -o loop /usr/local/mimic.img /usr/local/mimic
```

```
# df /usr/local/mimic
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/loop2      7289346  3983   6797120   1% /usr/local/mimic
```

By setting it up as root, the permissions prevent accidental corruption of the `mimic.img` file.

If you need to later **increase this virtual disk** you can use the `resize2fs` utility, eg. to add 10GB, first unmount the image, then resize and mount:

```
# umount /usr/local/mimic.img

# dd if=/dev/zero bs=1MB of=mimic.img conv=notrunc oflag=append count=10000
10000+0 records in
10000+0 records out
10000000000 bytes (10 GB) copied, 152.072 s, 65.8 MB/s

# e2fsck -f mimic.img
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
```

```
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
mimic.img: 518409/9762144 files (1.3% non-contiguous), 9276883/9765624 blocks
```

```
# resize2fs mimic.img
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on mimic.img to 19531248 (1k) blocks.
The filesystem on mimic.img is now 19531248 blocks long.

# mount -t auto -o loop /usr/local/mimic.img /usr/local/mimic

# df /usr/local/mimic
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/loop2      14613461 6800269   6836706   50% /usr/local/mimic
```

i. NFS

We recommend not running MIMIC on NFS mounted file systems but on a local disk filesystem as detailed [above](#). When the simulator accesses persistent storage for its configuration or simulation data, NFS could slow things down considerably.

i. Shared libraries

If you are getting the error `error while loading shared libraries` while running MIMIC on a newer version of Linux, you need to install the compatibility libraries as detailed in this [Frequently Asked Question](#).

i. 64-bit support

By default, MIMIC runs in 64-bit mode (and requires 64-bit x86_64 Linux on 64-bit systems) on RHEL 7, Fedora 19 or later.

32-bit support has been discontinued.

i. Bonded Interfaces

MIMIC supports [NIC bonding](#) in Linux, that is we have tested that agents configured on bonded NICs perform correctly, although we have not seen much performance benefit.

i. ARP Cache

The ARP cache on a system resolves IP addresses to MAC addresses. In the case of a large number of MIMIC agents, the Linux ARP cache garbage collection mechanism on the system running the management application can interfere interoperability with MIMIC, limiting performance or even failing. This is the case if the management station and MIMIC system are on the same LAN (without routers in between). In the case of intermediate routers, the issue will apply to the router connected to the MIMIC system, but usually ARP tables are configured adequately on such systems.

The major clue will be kernel messages of the form

```
kernel: Neighbour table overflow.
```

In this case, you need to configure garbage collection on the other system to allow more ARP table entries, eg. as detailed at [this page](#), eg. to allow 100000 entries

```
echo "100000" > /proc/sys/net/ipv4/neigh/default/gc_thresh3
echo "100000" > /proc/sys/net/ipv4/neigh/default/gc_thresh2
```

i. Known Problems

We are constantly working to remove limitations, but currently we know of the following:

- Linux kernels upto 2.4.2 have problems dumping core of multi-threaded applications, including MIMIC. There is no such limitation on the 2.6.x kernels in Red Hat Linux 9 and later.
- Red Hat Linux 6.2 has problems with logging to automounted filesystems. In the default configuration, this will impact only the Trap Wizard if your private area is automounted. Red Hat Linux versions 7.0 and later have no such problems.
- If you use non-standard network masks with agents in the same network, stopping the agent that was started first will remove all the addresses in the same network. This is a limitation only on Linux kernel version 3.x, our tests show it seems fixed with kernel 4.x after Fedora 20.
- You can only start upto 200 agents with IPv6 addresses on Red Hat Linux 9.x. After that, the system hangs. There is no such limitation on Fedora Core 3 and later, or Red Hat Enterprise Linux 4 and later.

- As of kernel 2.6.18 there is a configurable limit of 4096 IPv6 route entries, which limits the number of IPv6 aliases to +/- 4096. This can only be overcome by increasing the number of route entries from a root shell:


```
# sysctl net.ipv6.route.max_size
net.ipv6.route.max_size = 4096
```

```
# sysctl -w net.ipv6.route.max_size=8192
net.ipv6.route.max_size = 8192
```

- Starting agents with IPv6 addresses on RedHat Linux 9 is slow (approx. 10 agents per second) due to default duplicate address detection. There is no such limitation on later versions of Red Hat Linux (Fedora or Enterprise). In order to disable duplicate address detection, you can change the `dad_transmits` kernel configurable as follows (assuming that `eth0` is your network interface):

```
# cat /proc/sys/net/ipv6/conf/eth0/dad_transmits
1
# echo "0" > /proc/sys/net/ipv6/conf/eth0/dad_transmits
# cat /proc/sys/net/ipv6/conf/eth0/dad_transmits
0
```

Still, while it takes a couple of seconds to start 1000 agents with only IPv4 addresses configured on a common PC, it takes on the order of 140 seconds with just one IPv6 alias per agent with `dad_transmits` set to 0, and on the order of 400 seconds with `dad_transmits` set to 1. Optimistic DAD settings don't seem to help performance.

On newer Linux versions (Fedora 25 and newer), duplicate address detection can interfere with the startup of agents with IPv6 addresses, leading to errors in the MIMIC log containing

```
cannot bind receive IP address IPV6-ADDRESS port 161
bind: Cannot assign requested address
```

for some IPv6 address IPV6-ADDRESS.

The workaround is to disable duplicate address detection with

```
sysctl -w net.ipv6.conf.INTERFACE.dad_transmits=0
sysctl -w net.ipv6.conf.INTERFACE.accept_dad=0
```

for your network interface `INTERFACE`. Eg. if your network is `eth0`, you would use

```
sysctl -w net.ipv6.conf.eth0.dad_transmits=0
sysctl -w net.ipv6.conf.eth0.accept_dad=0
```

On older Linux releases, although DAD does not make MIMIC fail, it does slows down starting many agents. We have sped up agent startup at least a factor of 10 by disabling DAD in our tests.

- If you want to run MIMIC inside virtual machine software such as VMWare or Xen, please contact Gambit Communications Technical Support (support@gambitcomm.com).
- Although MIMIC can run inside containers like Docker, there are limitations in the Docker network plugins that don't allow MIMIC to create new IP aliases. So, if all your simulated devices can run on the IP alias for the Docker instance (eg. certain MQTT Simulator scenarios), then you are ok running inside a container.
- On Linux releases older than RHEL 7 or Fedora 19, Linux on AMD64 systems needs 32-bit compatibility libraries installed for the 32-bit MIMIC executables to run. A tell-tale symptom of this problem would be:

```
% ./mimicd
-bash: ./mimicd: /lib/ld-linux.so.2: bad ELF interpreter: No such file or directory
```

```
% file mimicd
mimicd: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.6.9, dynamically linked (uses shared libs), stripped
```

```
% ldd mimicd
not a dynamic executable
```

On Ubuntu, to install these libraries, as root run

```
# apt-get install ia32-libs
```

On Fedora distributions, to install these libraries, as root run

```
# yum whatprovides /lib/ld-linux.so.2
Loaded plugins: refresh-packagekit
Importing additional filelist information
glibc-2.10.1-2.i686 : The GNU libc libraries
Repo                : fedora
Matched from:
Filename            : /lib/ld-linux.so.2
...
```

```
# yum install glibc-2.10.1-2.i686
...
```

- On Fedora Core 6, Java or Perl MIMIC clients will fail to connect to the local MIMIC daemon with an error such as

```
ERROR: socket - cannot connect. localhost, 9797
```

The root cause of this problem is that the `/etc/hosts` file by default defines `localhost` with an IPv6 address `::1`. Changing this back to an IPv4 address `127.0.0.1` as it has always been, or defining a new entry, will solve this problem. This has been fixed in Fedora 7.

- If you are running intensive TCP performance tests that open/close many connections in a short amount of time, then you will run into the well-known `TIME_WAIT` condition: after a while the agents will fail to respond to new connections even though the system seems idle. You can verify this condition with

```
% netstat -a -n | grep TIME_WAIT
```

and it would list many lines. For the details and solutions, search on the Internet for `Linux TIME_WAIT`, for example [this page](#).

- Firewalls interfere with the functioning of MIMIC. We have seen all kinds of limitations with different firewalls, among them:

- no connectivity to simulated devices due to restrictive filters
- throttling of rates via Linux Netfilter, resulting in spurious connection drops and rejections. This usually results in networking errors all over the system, like

```
do_yppcall: clnt_call: RPC: Unable to send; errno = Operation not permitted
or in /var/log/messages:
```

```
Mar 24 15:29:01 dmb kernel: [970079.411385] net_ratelimit: 50 callbacks suppressed
Mar 24 15:29:01 dmb kernel: [970079.411388] nf_conntrack: table full, dropping packet
If you want to diagnose connectivity problems, turn off your firewall first.
```

- Network interfaces with long labels interfere with MIMIC. If the active network interface to be used by MIMIC, eg. as shown with

```
% ip link show
1: lo: mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eno16780032: mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
   link/ether d4:be:d9:cb:58:9f brd ff:ff:ff:ff:ff:ff
```

```
% netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
lo         65536  1979263 0      0 0      1979263 0      0      0 LRU
eno16780032 1500  382210306 0      0 17720 0      542194465 0      0      0 BMRU
```

(in this case `eno16780032`) is more than 10 characters long, then it can overflow the 16-character limit for network interface names for the IP aliases that MIMIC creates, causing `ERROR` messages when agents start.

This is an OS limit, and the only workaround is to rename your network interface.

For any additional support, please contact Gambit Communications Technical Support (support@gambitcomm.com).

[<< Previous Section](#) [Next Section >>](#)



Windows Installation Instructions

Table of Contents

- [Overview](#)
- [Account Privileges](#)
- [Firewalls](#)
- [Assigning IP Addresses](#)
- [5000 / 10000 agent support](#)
- [20000 / 30000 agent support](#)
- [Virtual Memory](#)
- [Duplicate IP Address](#)
- [Media Sense on Windows Vista and newer](#)
- [IPv6 Support](#)
- [Performance impact of Windows services](#)
- [64-bit support](#)
- [Windows Vista and newer](#)
- [Crashes](#)
- [Known Problems](#)

1. Overview

MIMIC runs on

- Windows Vista Business Edition
- Windows Server 2008
- Windows 7
- Windows Server 2012
- Windows 8
- Windows 10
- Windows Server 2016 Standard

It is highly recommended to install the latest service pack for the respective OS. The following are some of the most common problems encountered on Windows, and their fixes:

1. Account Privileges

For Windows Vista, Server 2008 and Windows 7 and newer, please go to [the section below](#).

1. Firewalls

Due to pervasive security attacks against Windows systems connected to the Internet, it has become common to run a software firewall on recent versions of Windows.

MIMIC will coexist with a software firewall, provided that the firewall is configured to recognize MIMIC as a program allowed to access the network. MIMIC will, due to its very nature of simulating networked components, open network sockets and communicate with external applications (eg. network management applications, telnet clients, etc).

There are certain components of MIMIC that will access the Internet (eg. specific web sites to determine software updates, etc), but only if explicitly configured by the user.

1. Assigning IP Addresses

MIMIC requires at least one operational network interface card (NIC). On Windows Vista or later, as on the Unix platforms, MIMIC dynamically assigns IP addresses when starting each agent instance.

1. 5000 / 10000 agent support

Different versions of Windows have performance limitations while running large numbers of IP addresses. The following are the limits on the different versions of Windows:

- Windows Vista: 10000 agents;

Windows Server 2008: 10000 agents;

- Windows 7: 10000 agents;
- Windows Server 2012: 30000 agents;
- Windows 8: 30000 agents;
- Windows 10: 30000 agents;

In empirical tests, going beyond those limits introduced non-deterministic instability and performance problems.

l. 20000 / 30000 agent support

You can run 20000 agents only on Windows Server 2003, and 30000 agents only on Windows Server 2012, Windows 8 or Windows 10, and then only with [64-bit executables](#), and more on the Unix platforms.

l. Virtual Memory

MIMIC is a specially memory-intensive application, and it needs plenty of physical memory (RAM) and virtual memory (swap space) for the more complex device simulations. We recommend an absolute minimum 64MB of RAM and 128MB of swap space to start. Verify that you have enough swap space in the System Control Panel.

You should as a rule of thumb have twice the swap space as your physical memory.

Different versions of Windows on Intel systems impose different limits on the amount of virtual memory accessible for the MIMIC Simulator, but currently the absolute limit is below 4 GB due to 32-bit addressing limitations, which the [64-bit architecture](#) does not have.

On all 32-bit versions of Windows there is a 2 GB limit on per-process virtual memory.

On Windows Vista and later Microsoft has removed the boot.ini file and provided the BCDEdit command to enable 3 GB addressing as documented [in this document](#).

To enable this expanded capability on Windows Vista and later, as an Administrator in a Command Prompt window, type the command:

```
BCDEdit /set IncreaseUserVA 3072
```

And then restart your computer.

This will make "a full 3 GB of virtual address space available to applications and reduces the amount available to the system to 1 GB."

l. Duplicate IP Address

For Windows Vista, Server 2008 and Windows 7, please go to [the section below](#).

If Windows detects that an IP address on one of its Network Interface Cards (NICs) conflicts with another system (duplicate IP address), then it tries to resolve this problem by shutting down the NIC and displays a message such as:

```
The System has detected an IP address conflict with another system on the network. The local interface has been disabled. More details are available in the system event log. Consult your network administrator to resolve the conflict.
```

You must not have duplicate IP addresses on a connected network, neither with MIMIC or otherwise.

NOTE: you can disable duplicate IP address detection by following the instructions in the article in the Microsoft Knowledge Base titled [How to Disable the Gratuitous ARP Function](#). It is appended here:

How to Disable the Gratuitous ARP Function

View products that this article applies to.

```
Article ID      :      219374
Last Review    :      February 24, 2007
Revision       :      3.2
This article was previously published under Q219374
```

SYMPTOMS

When a Windows NT-based computer starts, a packet is broadcast on the network containing the computer's TCP/IP address to prevent the use of duplicate addresses on the same network. This is called a gratuitous Address Resolution Protocol (ARP) packet. Routers and other network hardware may cache routing information gained from multiple gratuitous ARP packets. For both performance and maintenance reasons, it is possible to disable this feature in Windows NT.

RESOLUTION

To resolve this problem, obtain the latest service pack for Windows NT 4.0. For additional information, click the following article number to view the article in the Microsoft Knowledge Base: 152734 (<http://support.microsoft.com/kb/152734/EN-US/>) How to Obtain the Latest Windows NT 4.0 Service Pack

STATUS

Microsoft has confirmed that this is a problem in the Microsoft products that are listed at the beginning of this article. This problem was first corrected in Windows NT 4.0 Service Pack 5.

MORE INFORMATION

To disable gratuitous ARPs after applying this hotfix:

1. Click Start, click Run, type regedt32, and then click OK.
2. On the Windows menu, click HKEY_LOCAL_MACHINE on Local Machine.
3. Click the \System\CurrentControlSet\Services\TcpIp\Parameters folder.
4. Double-click the ArpRetryCount value, type 0, (for Windows XP, type 1) and then click OK.
5. Quit Registry Editor, and then restart the computer.

). Media Sense on Windows Vista and newer

Newer versions of Windows (Windows Vista onwards) have a TCP/IP feature whereby it can sense if a NIC is actually connected to the network. By default, a NIC is disabled if it is not found to be on the network, which prevents agents from starting in MIMIC. There is a way to disable this behaviour so that you can work on standalone Windows machines. Attached is an excerpt of the [Microsoft KB](#) article on this topic... Please remember to make a copy of your registry before making any changes just to be on the safe side.

NOTE Windows Vista and newer have no way to disable this feature. You can only run MIMIC on Windows Vista and newer with the system connected to a network.

How to disable the Media Sensing feature for TCP/IP in Windows

SUMMARY

On a Windows-based computer that uses TCP/IP, you can use the Media Sensing feature to detect whether the network media are in a link state. Ethernet network adapters and hubs typically have a "link" light that indicates the connection status. This status is the same condition that Windows interprets as a link state. Whenever Windows detects a "down" state, it removes the bound protocols from that adapter until it is detected as "up" again. Sometimes, you may not want the network adapter to detect this state. You can set this configuration by modifying the registry.

Let me fix it myself

Important This section, method, or task contains steps that tell you how to modify the registry. However, serious problems might occur if you modify the registry incorrectly. Therefore, make sure that you follow these steps carefully. For added protection, back up the registry before you modify it. Then, you can restore the registry if a problem occurs. For more information about how to back up and restore the registry, click the following article number to view the article in the Microsoft Knowledge Base: 322756 How to back up and restore the registry in Windows

To prevent the network adapter from detecting a link state, follow these steps.

Note The NetBEUI protocol and the IPX protocol do not support Media Sensing.

Start Registry Editor.

Locate the following registry subkey:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters

Add the following registry entry to the

Parameters

subkey:

Name: DisableDHCPMediaSense

Data type: REG_DWORD (Boolean)

Value: 1

Note This entry controls the behavior of Media Sensing. By default, Media Sensing events trigger a DHCP client to take an action. For example, when a connect event occurs, the client tries to obtain a lease. When a disconnect event occurs, the client may invalidate the interface and routes. If you set this value data to 1, DHCP clients and non-DHCP clients ignore Media Sensing events.

Restart the computer.

. IPv6 Support

IPv6 is supported only on Windows Vista and newer. This is an OS platform limitation.

IPv6 is installed as detailed [here](#). IPv6 addresses use an additional Scope ID as detailed on the [Microsoft web-site](#).

If the IPv6 module gets corrupted, you can fix this with this command from the COMMAND prompt:

```
netsh interface ipv6 reset
```

NOTE: check the [Known Problems](#) section for limitations.

1. Performance impact of Windows services

Running unnecessary Windows services impacts performance adversely on a MIMIC system. The observed impact has been an approximate 300% slowdown in starting and 1000+% slowdown stopping agents, and a 10-20% slowdown in servicing requests. This is documented in a [Microsoft support page](#) titled "Memory usage by the Lsass.exe process on domain controllers that are running Windows Server 2003 or ...". The executables running the services end up hogging the CPU. This can easily be seen in a process listing in the Windows Task Manager.

Thus if you are running large lab configurations, we recommend to stop the following unnecessary services in the Services Control Panel:

Display Name	Service Name	Path to executable
IPSEC Services	PolicyAgent	WINDOWS\system32\lsass.exe
Protected Storage	ProtectedStorage	WINDOWS\system32\lsass.exe
Print Spooler	Spooler	WINDOWS\system32\spoolsv.exe
DHCP Client	Dhcp	WINDOWS\system32\svchost.exe -k NetworkService
DNS Client	Dnscache	WINDOWS\system32\svchost.exe -k NetworkService
TCP/IP NetBIOS Helper	LmHost	WINDOWS\system32\svchost.exe -k LocalService
Windows Time	W32Time	WINDOWS\system32\svchost.exe -k netsvcs

Additionally, on Windows Server 2003 we have seen the RPC service hog the CPU while MIMIC is busy servicing SNMP requests. On Windows XP SP2 we have seen the RPC service crash when stress testing MIMIC, thus rebooting the system by default. We recommend stopping all svchost.exe processes, and make sure that the services are configured not to restart or reboot the system.

See also [A description of Svchost.exe in Windows XP](#)

1. 64-bit support

The optional 64-bit MIMIC simulator and protocol module shared libraries can run on a x64 or AMD64 processor. To do so, you need to run a [64-bit version of Windows](#), and use [Update Wizard](#) to download the optional update package with the MIMIC 64-bit executables.

1. Windows Vista and newer

1. User Account Control

Windows Vista and later has the new User Account Control feature, which impacts the running of MIMIC. For details, consult this [Analysis of the Windows Vista Security Model](#) from Symantec. In order to enable to run MIMIC on Vista and later, you have 2 options:

- Disable User Account Control
This turns UAC off globally. NOTE: do this only if you are aware of the implications of this action.
 - On Windows 7:
Open User Accounts via Start->Control Panel->User Accounts->User Accounts->User Account Control settings.
Slide scrollbar to bottom (Never notify).
 - On Windows Vista and Server 2008:
Open User Accounts via Start->Control Panel->User Accounts->User Accounts.
Click on Turn User Account Control on or off
Clear the checkbox for Use User Account Control (UAC) to help protect your computer.
 - Click ok
 - A dialog will popup prompting you to Restart Now or Restart Later. Choose appropriately. User Account Control will be disabled once the system reboots.
 - On Windows Server 2012:
Follow directions at [this Microsoft article](#).
- Run MIMIC with User Account Control enabled
This involves changing the access control level of the MIMIC programs.
 - Change the privilege level of MimicView application in the Mimic Start Program group using the following steps:
 - Click Start->All Programs->Mimic Simulator x.xx
 - Move the cursor to the MimicView entry
 - Right click and select Properties.
 - In the Compatibility tab, check Run this program as an administrator
 - Once this is done, MimicView can be started as above or by running the MimicView.bat script in the bin folder of the MIMIC installation.
 - If MimicD.exe will be run directly, set the privilege level of it using the following steps:
 - In Windows Explorer, select it.

Right click and select Properties.

- In the Compatibility tab, check Run this program as an administrator

!. Duplicate Address Detection

On Windows Vista and later, the new TCP/IP stack tries to do "duplicate address detection" by default. This prevents MIMIC from starting agents, because IP aliasing is delayed, and even with a workaround in our software would unacceptably slow down the starting of agents. To correctly workaround the problem, you need to disable "duplicate address detection" for the network interface using the Windows netsh utility:

```
netsh interface ipv4 set interface "name or index" dadtransmits=0
```

The interface name and index info can be obtained by

```
netsh interface ipv4 show interfaces
```

For example:

```
H:\>netsh interface ipv4 show interfaces
```

Idx	Met	MTU	State	Name
1	50	4294967295	connected	Loopback Pseudo-Interface 1
7	20	1500	connected	Local Area Connection

```
H:\>netsh interface ipv4 set interface "7" dadtransmits=0
```

The same applies to IPv6.

```
netsh interface ipv6 set interface "name or index" dadtransmits=0
```

!. Services

Windows Vista and later has the problem of services interfering with MIMIC operation as documented [above](#).

Here is the list of already documented services and their paths:

Display Name	Service Name	Path to executable
IPsec Policy Agent	PolicyAgent	WINDOWS\system32\svchost.exe -k NetworkServiceNetworkRestricted
Protected Storage	ProtectedStorage	WINDOWS\system32\lsass.exe
Print Spooler	Spooler	WINDOWS\system32\spoolsv.exe
DHCP Client	Dhcp	WINDOWS\system32\svchost.exe -k NetworkServiceNetworkRestricted
DNS Client	DnsCache	WINDOWS\system32\svchost.exe -k NetworkService
TCP/IP NetBios Helper	lmhosts	WINDOWS\system32\svchost.exe -k NetworkServiceNetworkRestricted
Windows Time	W32Time	WINDOWS\system32\svchost.exe -k LocalService

Additionally, these new services in Windows Vista and later MUST be disabled for reasonable start/stop performance.

Display Name	Service Name	Path to executable
Function Discovery		
Resource Publication	FDRResPub	WINDOWS\system32\svchost.exe -k LocalService
IKE and AuthIP		
Keying Modules	IKEEXT	WINDOWS\system32\svchost.exe -k netsvcs
Network List Service	netprofm	WINDOWS\system32\svchost.exe -k LocalService
Network Location		
Awareness	NlaSvc	WINDOWS\system32\svchost.exe -k NetworkService
SSDP Discovery	SSDPDRV	WINDOWS\system32\svchost.exe -k LocalService
UPnP Device Host	upnphost	WINDOWS\system32\svchost.exe -k LocalService
Windows Remote Management (WS-Management)	WinRM	WINDOWS\system32\svchost.exe -k NetworkService

These are new services that may be disabled for better start/stop performance on Windows Vista and later.

Display Name	Service Name	Path to executable
Background Intelligent		
Transfer Service	BITS	WINDOWS\system32\svchost.exe -k netsvcs
IP Helper	iphlpvc	WINDOWS\system32\svchost.exe -k netsvcs

Service Details

- Function Discovery Resource Publication
Used for publishing network resources. Stopping will prevent other computers from discovering the network resources. Will impact applications that work with Network Connected Devices, Plug-n-Play Extensions and Web Services on Devices.
- IKE and AuthIP Keying Modules
Internet Key Exchange (IKE) and Authenticated Internet Protocol (AuthIP) keying modules are used for authentication and key exchange in IPSec. Stopping them will prevent IPSec from working.
- Network List Service
Used to determine the networks to which the system is connected, collect/store their properties and notify applications.

Stopping this will prevent the Network Status tray icon from working properly.

- Network Location Awareness
Used to collect/store network configuration information and notify applications.

Stopping this will prevent any applications (like Network List Service) that depend on this data from working.

- SSDP Discovery
Used to provide Simple Service Discovery Protocol based services. Stopping this will prevent UPnP from working.
- UPnP Device Host
Used to provide Universal Plug-n-Play services. Stopping this will prevent UPnP based devices from working.
- Windows Remote Management (WS-Management)
Used for Web Services for Management functionality. Stopping will disable Windows Remote Management and WS Management

i. Vista Power Management

The default power options will put the Windows Vista system to sleep after 1 hour of inactivity. To disable this, perform the following:

Open Power Options using Control Panel->System and Maintenance->Power Options.

Change Preferred Plan from Balanced to High Performance.

Verify by clicking on Change Plan Settings for High Performance. Ensure that Put the computer to sleep setting is Never.

i. Program Compatibility Assistant

After the install is completed or aborted, the Program Compatibility Assistant may prompt with the message

This program might not have installed correctly.
Please select This program installed correctly if the install completed. Else, select Cancel.

i. Crashes

Prior to Windows Vista, crashes can be analysed post-mortem using the [Dr. Watson Tool](#) crash dumps. This requires that Dr. Watson be enabled to handle any application exceptions on the system.

To install Dr. Watson as the default exception handler :

- Click Start->Run.
- Type drwtsn32 -i
- Click Ok.

A subsequent crash should popup the Dr. Watson dialog. Search for the following files in the Windows directory (this location can be changed using the Dr. Watson GUI) : drwtsn32.log and user.dmp . Send these to Gambit Technical Support (support@gambitcomm.com).

On Windows Vista and later, by default the Problem Reports and Solutions feature handles program crashes. Crash information, including minidumps when available, is automatically sent to Microsoft.

You can check if a MIMIC program crashed and if minidumps are available. Please use the following steps to extract any available MIMIC program crash data and forward it to Gambit Technical Support (support@gambitcomm.com).

- Go to Problem Reports and Solutions using Control Panel->System and Maintenance->Problem Reports and Solutions or using Control Panel->Problem Reports and Solutions
- Click on View problem history
- Find MIMIC programs in the list
- Double click on an entry to view the problem details
- If there is a Files that help describe the problem section, click on the View a temporary copy of these files link below that section
- The files will be extracted into a temporary directory and an Explorer window will be opened to view them
- Forward these to Gambit Technical Support

If the Problem Reports and Solutions settings are changed to check with the user before sending the crash information to Microsoft, Windows Vista will prompt the user when a program crash occurs. If you choose Close the program, no additional details are generated. If you choose Check online for a solution and close the program, crash data may be saved.

Microsoft's [Debug Diagnostic Tool](#) version 1.1 onwards may be used on Windows Vista and later to handle program crashes. If this is installed, please use the following steps to generate crash data.

- Open Debug Diagnostic Tool
- Click on the Rules tab
- Click on Add Rule button
- In the Select Rule Type dialog, choose Crash and click on Next
- In the Select Target Type dialog, choose A specific process and click on Next
- In the Select Target dialog, browse the process list, select Mimicd.exe and click on Next
- In the Advanced Configurations (Optional) dialog, change the number of userdumps as needed and click on Next
- In the Select Dump Location and Rule Name (Optional) dialog, change the path and name as needed and click on Next
- In the Rule Completed dialog, choose Activate the rule now and click on Finish

When a crash occurs, the Rules tab in Debug Diagnostic Tool will show the userdump count. Forward the available files from the configured dump location to Gambit Technical Support.

9. Known Problems

We are constantly working to remove limitations, but currently we know of the following:

- On Windows Vista and newer you cannot run MIMIC on a standalone PC (not connected to a network). This is due to the OS having removed the ability to [disable media sense](#). We have found no workaround for this limitation.
- On Windows NT, if the host is using DHCP to obtain its address, no agent instance can use that same address to export a MIB. The problem is that on stopping the agent this address is deleted which shuts off the TCP/IP services (ftp/telnet/internet). To restore working you either need to REBOOT or start the agent on the DHCP assigned address again (keep it running).
- On Windows Vista you can only configure upto 10000 agents and 15000 +/- IP addresses.
On Windows Server 2008 you can only configure upto 10000 agents and 12000 +/- IP addresses.
On Windows 7 you can only configure upto 10000 agents and 11000 +/- IP addresses.
Any subnets can be used for these addresses.
- On all versions of Windows you can only run upto 64 MIMIC clients over the local management channel [bug 2371].
The Unix versions of MIMIC have no such limitations on the number of clients.
- On Windows, certain network interface cards have limitations supporting multiple IP addresses. In particular, some adapters and/or drivers from 3com have been giving us trouble (eg. 3C905-TX or 3Com 3C90x Ethernet Adapter). The symptom is that a small number of agent instances can be started and polled correctly, but connectivity is lost to the box when starting more.
- On Windows, certain software is incompatible with MIMIC. In particular, if Novell Client is running, your machine may hang after starting and stopping a small number of agents (the System task will use 99+% of CPU). Just unchecking the box in Local Area Connection Properties is not sufficient - you have to uninstall it.
This problem is unrelated to MIMIC. Any test program (eg. `ifdiag` shipped with MIMIC) which uses the Windows API to register/unregister network addresses will reproduce the problem.
- If you want to run MIMIC inside virtual machine software such as VMWare, please contact Gambit Communications Technical Support (support@gambitcomm.com).
- `MAC Bridge Miniport` in Windows Vista and later is not supported. To remove the adapters from the network bridge, follow the instructions at [this Microsoft support page](#).
- IPv6 will only work on Windows Vista or later. It runs on all supported versions of Unix.
When adding or deleting IPv6 aliases to/from an agent it can take 2 to 3 seconds for the operation to complete per agent. This slowness is caused by the Domain Name System (DNS) performing a lookup to see if the address is actually a registered domain name. To resolve this problem you must follow both the steps listed below:
 - Do not use domain names and only use IPv4 or IPv6 addresses for all agents.
 - Do either of the following:
 - add the line "`hostname_resolve = 0`" to your `mimicd.cfg` file in your MIMIC config/ directory;
 - set the environment variable "`MIMIC_HOSTNAME_RESOLVE=0`" on your Windows system.

Even with this workaround, adding and deleting IPv6 aliases is approximately 5 to 10 times slower than on Linux or Solaris. If you are running thousands of agents, your performance will suffer. On Windows 7 or later the performance is comparable to Linux provided you have disabled [duplicate address detection](#).

Furthermore, our testing uncovered that the Microsoft `netsh` utility used to add aliases sometimes fails silently. For Windows 7 and later there is no such issue.

- On Windows you cannot add IPv6 addresses on disconnected NICs, contrary to IPv4, where there is no problem. This is an OS limitation that we cannot overcome. There is no such limitation on other OS platforms.
- On Windows 8.1 adding large numbers of IPv6 addresses crashes the Windows Explorer due to `STACK_OVERFLOW`. MIMIC continues to run correctly. Windows quickly recovers with no apparent harm, since Windows Explorer automatically restarts. This is reproducible with the Windows `netsh` tool, thus is a limitation outside of MIMIC.
- Performance on Windows Vista is much worse than Windows 7 and later, specially with large scenarios. For this reason we DO NOT recommend running MIMIC on Windows Vista.
- On all versions of Windows, there is a configurable limit on per-process "USER objects". The default limit of 10000 interferes with MIMICView for large-scale simulations. Eg. if you want to display a large number of device types (more than 1000) with `simulation->Devices`, the display may hang and leave droppings on the screen, in particular a line of text at the top of the display.
To work around this limitation, you have to increase the number of USER objects as described in this [Microsoft support page](#). The absolute limit is 18000, which you will NOT be able to overcome. With this limit, you'll be able to display about 2000 average-sized devices.

In addition, there is a limit on the "desktop heap", as detailed in this [Microsoft support page](#). Change this parameter in the registry to overcome heap limitations.

MIMIC on other platforms has no such limitation.

- On Windows, the `Nvidia ForceWare Network Access Manager` that is enabled by default for `NVIDIA nForce Networking Controller` cards causes problems when MIMIC is starting and stopping agents. The symptoms are that connectivity to the network is disabled. The only workaround we have found is to uninstall the program, since disabling it seems to have no effect.
- The native telnet client on Windows has problems as documented in detail in the [Virtual Lab Windows Installation Instructions](#).
- remote access via Remote Desktop gets disconnected when starting agent(s)

This applies to Windows 7 or later due to changes in Windows where Windows will automatically use the lowest IP assigned to an interface as the source IP address as explained in this [Microsoft Forum](#) and this [Microsoft article](#).

These are the possible workarounds for MIMIC to work in this scenario:

1. Use IP addresses for MIMIC agents that are higher than the system's IP address.
For example:

MIMIC system IP is 10.0.0.10 then use addresses like 10.0.0.11, 10.0.0.12 etc. for MIMIC agents.

2. Configure an extra interface on the computer for management/remote-access and having the agents bound to the other interface. Different interface can be assigned to MIMIC agents using Advanced tab of agent configuration dialog. Default interface to be used for MIMIC agents can also be configured by setting environment variable in the Control Panel-> System Properties as below:

Variable: MIMIC_DEFAULT_NETDEV
Value: Intel(R) 82566DM Gigabit Network

- MIMIC display tiny on high-resolution monitors
on newer high-resolution display monitors (eg. 3240x2160) you need to control the scaling of the MIMIC user interface. You can set the property on the wish86t.exe application as shown in this diagram.

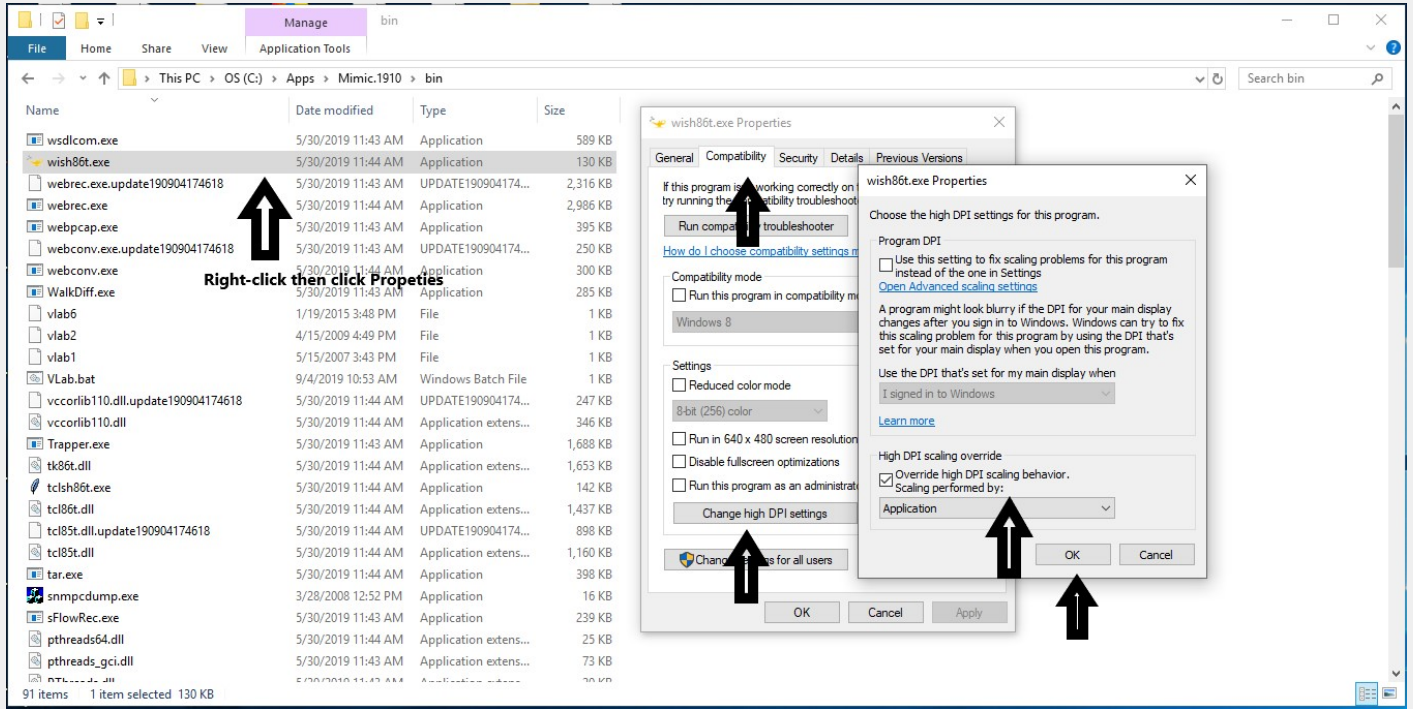


Figure 1: wish86t Properties

In Windows Explorer, right-click on the wish86t.exe file in the MIMIC install area bin folder, then select Properties.

The wish86t.exe Properties dialog pops up.

Click the Compatibility tab, then click Change high DPI settings.

In the dialog that pops up, select Override high DPI scaling behavior. Scaling performed by and select System Controlled (Enhanced).

Click ok to both dialogs, and when you invoked MIMICview, it should display correctly.

In case of difficulties, please contact Gambit Technical Support (support@gambitcomm.com).

Obviously, Windows is a trademark of Microsoft. All other product and brand names are trademarks or registered trademarks of their respective holders.



MIMIC Installation Instructions

Overview

MIMIC is distributed as a compressed file for each platform, either on CD-ROM or downloadable from the Web. This file needs to be uncompressed and its contents extracted on your system.

Follow the below 3 steps to install.

After installation, please review the [MIMIC Frequently Asked Questions page](#).

NOTE: you only need to download MIMIC when a new release is issued.

The evaluation and purchased versions are identical except for the license keys. If you have already downloaded MIMIC, determine the version you are running (eg. in MIMICview [Help->About Mimic...](#) or by looking at the contents of the `config/version` file), and download again only if the distribution listed below is newer.

Step 1: Get the MIMIC distribution

If you have a CD-ROM, copy the distribution file for your platform from the CD-ROM to a temporary directory. Go to Step 2.

To download from the Web from a URL that Gambit has provided, save the distribution files for your desired platform to a temporary directory (use right mouse-button click on the link for most web browsers).

NOTE: make sure the entire file was downloaded and verify size.

NOTE: there are scam versions of any software floating around the Internet with all sorts of malware. Download only authoritative versions of MIMIC from a URL that Gambit has given you directly.

Step 2: Uncompress and extract the distribution files

- For Windows:
Run the self-extracting `mimic-windows.exe` from an Administrator account.
- For Linux:

```
gunzip -c mimic-linux.tar.gz | tar xf -
```

The extracted files are:

README - further release-specific instructions
license.txt - the license agreement
install - the installation script
mimic.tar - the product

You are not done, please review the instructions in the next step.

Step 3: Further instructions

- If you are **updating from an older release**
 - make sure you **terminate MIMIC** (use `File->Terminate`) before running the install program.
 - **Do not uninstall the older release** (this way you can always fallback in case of problems).
 - Install this release in a **different directory from the older release**. MIMIC installs in these folders by default:
 - Windows: `C:\Apps\Mimic.VERSION-NUMBER` eg. `C:\Apps\Mimic.1800`
 - Linux: `/usr/local/mimic`
On Linux you can rename old install folder as:

```
mv /usr/local/mimic /usr/local/mimic.old
```

Then install in the default folder `/usr/local/mimic`.
 - You only need to **apply your license keys** and start the new MIMIC. New license keys are only necessary if the old ones don't work. All your MIMIC data files will be fetched out of your existing [private area](#).
 - When you start MIMICView from the new version, all your MIMIC data files will be fetched out of your existing [private area](#) which usually resides in a directory named

mimic under the user's HOME directory, and is displayed in the title bar of MIMICView. We recommend to backup this private data often (see this [FAQ](#)).

- Additionally, you may need to **upgrade any optional packages**, such as MIBs, Device Library and Network Library if you were using them in older release. They can be downloaded and installed using the [Wizard->Update](#) menu. The Update Wizard preselects packages from your older version to install them in your new version.
- If you are **installing for the first time**, the MIMIC installation program will display the HostID to be used in your evaluation license key request.
- If you are installing a Simulator other than SNMP, then you will have received **Get Started** instructions to download optional packages with sample simulations. Please refer to your [Get Started](#) instructions.

[<< Previous Section](#) [Next Section >>](#)



Quick Start Guide

Preface


This guide is an overview to using MIMIC Simulator. It describes the operations, concepts and processes of MIMIC. Its purpose is to introduce MIMIC's tool suite (see [About MIMIC](#)), which consists of the following components:

- **mimicd** -- MIMIC Simulator daemon
- **Graphical User Interfaces (GUI)**
 - **mimicview** -- MIMICView native GUI
 - **Wizards** -- Wizards
 - **WEBUI** -- WEBUI browser based UI
- **SNMP tools**
 - **mimicrec** -- MIMIC Recorder
 - **mimiccom** -- MIMIC Compiler
- **Protocol modules**
 - **Telnet** -- Telnet / command line interfaces
 - **SSH** -- SSH / NETCONF
 - **WEB** -- Web Services / XML / REST
 - **NETFLOW** -- NetFlow
 - **MQTT** -- Message Queue Telemetry Transfer
 - **CoAP** -- CoAP
- **Programming APIs**
 - **mimicsh** -- MIMICShell Tcl command line interface
 - **Python**
 - **Perl**
 - **Java**
 - **C++**
 - **PHP**
 - **Javascript**
 - **Go**
 - **OpenAPI**

In addition, it presents the general processes to:

- [Compile MIBs](#)
- [Create a Simulation](#)
- [Run a Simulation](#)
- [Customize a Simulation](#)

It assumes that you are familiar with networking and network management concepts, particularly Simple Network Management Protocol (SNMP) and Management Information Base (MIB).

If you prefer videos, you can consult our [video collection](#)  .

Organization

This guide is organized in the following major sections:

[Using MIMICView](#) gives an overview about the native MIMIC GUI and the overall functionality of the product.

[Troubleshooting](#) guides you through solving problems with MIMIC.

[Important Concepts](#) includes useful definitions.

[MIMIC Process Overview](#) presents flow charts to help you understand the process of using MIMIC.

Whenever you see the Youtube logo  you'll be able to view a video for that section.

Installing MIMIC

See the following online instructions for [installing MIMIC](#) on: [Linux](#) and [Windows](#).

Typography Conventions

Normal	Text
Typewriter	Computer output; names of functions and data types
Typewriter	Interface components; menus, buttons and entry fields
<i>Italics</i>	Values you can input; variable names, numbers, strings
Bold Normal	What you have to type correctly, for example, filenames, Unix commands, function names, command-line entries

1. Table of Contents

- Chapter 1: Overview
 - [About MIMIC](#)
 - [MIMIC Tool Suite](#)
- Chapter 2: Using MIMICView
 - [Starting MIMIC](#) 
 - [Using the GUI](#) 
 - [Controlling Agents](#) 
 - [Running Agent Simulations](#) 
 - [Changing a Simulation](#) 
 - [Generating Traps](#) 
 - [Recording a Device](#) 
- Chapter 3: Troubleshooting
 - [Online Help](#)
 - [Inspect the Log](#) 
 - [Common Errors](#)
 - [Common Questions](#)
 - [Diagnostic Wizard](#)
 - [Crashes](#)
- Chapter 4: Process
 - [Important Concepts](#)
 - [MIMIC Process Overview: When to Use the Tools](#)
 - [References for Further Reading](#)
- Chapter 5: Energy Savings

1. Chapter 1: Overview

Chapter Contents

- [About MIMIC](#)
- [MIMIC Tool Suite](#)

About MIMIC

MIMIC (Multiple Instance Management Information Concentrator) is a software simulation tool that helps to address the following issues:

- **Testing:** Crisis situations in your network
- **Evaluation:** Test before deployment and plan for future growth
- **Staff training:** Without impacting your live network
- **Demonstrating:** In an environment familiar to a customer
- **Development:** Parallel development of SNMP agent and management application

Network management systems (NMS) typically communicate with many manageable devices.

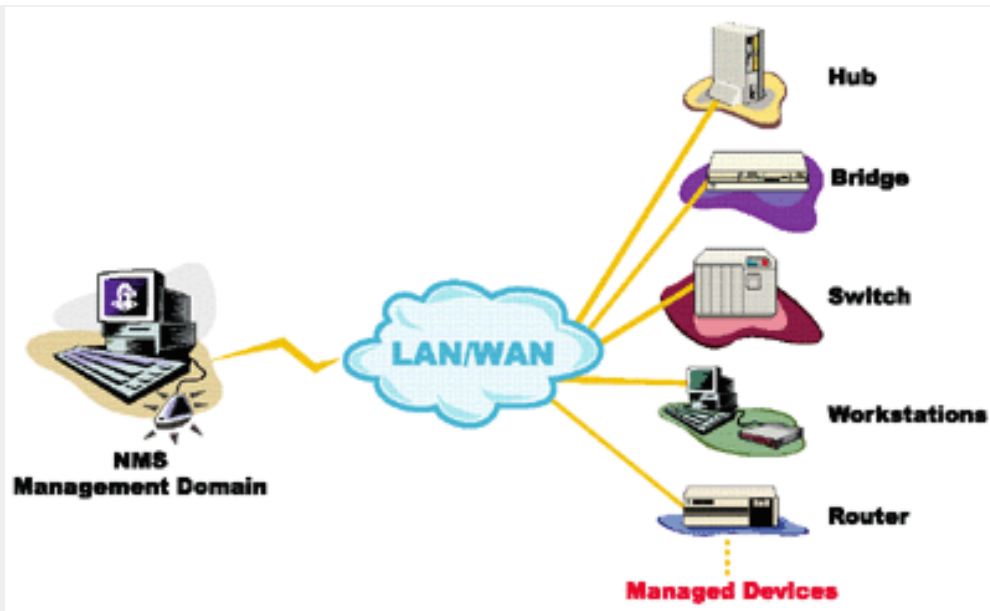


Figure - NMS with devices

To exercise (test, evaluate, train) the application, you would have to set up a lab with a facsimile of the expected environments. This entails such [problems](#) as buying all the devices relevant to your application; the interoperability of SNMP agents in various versions; and the nightmare of combinatorial scenarios of different devices and agents. MIMIC Simulator solves all these problems.

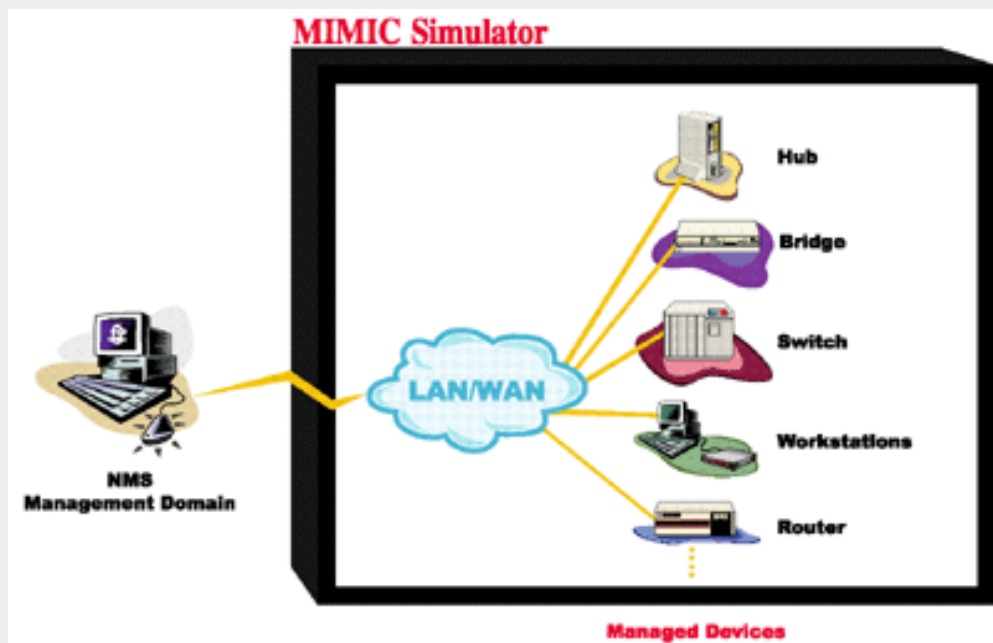


Figure - NMS with MIMIC

The [MIMIC SNMP Agent Simulator](#) lets you simulate up to 100,000 SNMP-manageable devices on one Intel-based PC. Your network management application can send SNMP (v1, v2, v2c, v3) requests to the simulated agent, which can return SNMP responses or traps. Any SNMP-based device is supported. You can run a variety of device configurations and customize them at runtime. Because MIMIC responds to SNMP queries on any of its configured IP addresses, it looks to the application as though it was communicating to actual devices.

The [MIMIC IOS Simulator](#) adds the capability to respond to Cisco IOS commands over [Telnet](#) or [SSH](#). It gives Network Engineers an ability to practice for certifications instead of just reading from the instructions. Included in this simulation are the [TFTP](#) and [SYSLOG](#) protocols.

The [MIMIC WEB Simulator](#) adds WEB Services simulation such as SOAP, XML, JSON and REST via HTTP/HTTPS.

The [MIMIC Cable Modem Simulator](#) extends the MIMIC SNMP Agent Simulator with the protocols necessary for simulating cable modems from an Operations Support System (OSS) perspective. The additional protocols are [DHCP](#), [TFTP](#) and [TOD](#).

The [MIMIC IPMI Simulator](#) simulates the Intelligent Platform Management Interface (IPMI) available on servers.

The [MIMIC Proxy Server](#) simulates any number of TCP- or UDP-based services by proxying existing servers.

The [MIMIC Netflow Simulator](#) allows simulating large flow-based monitoring environments

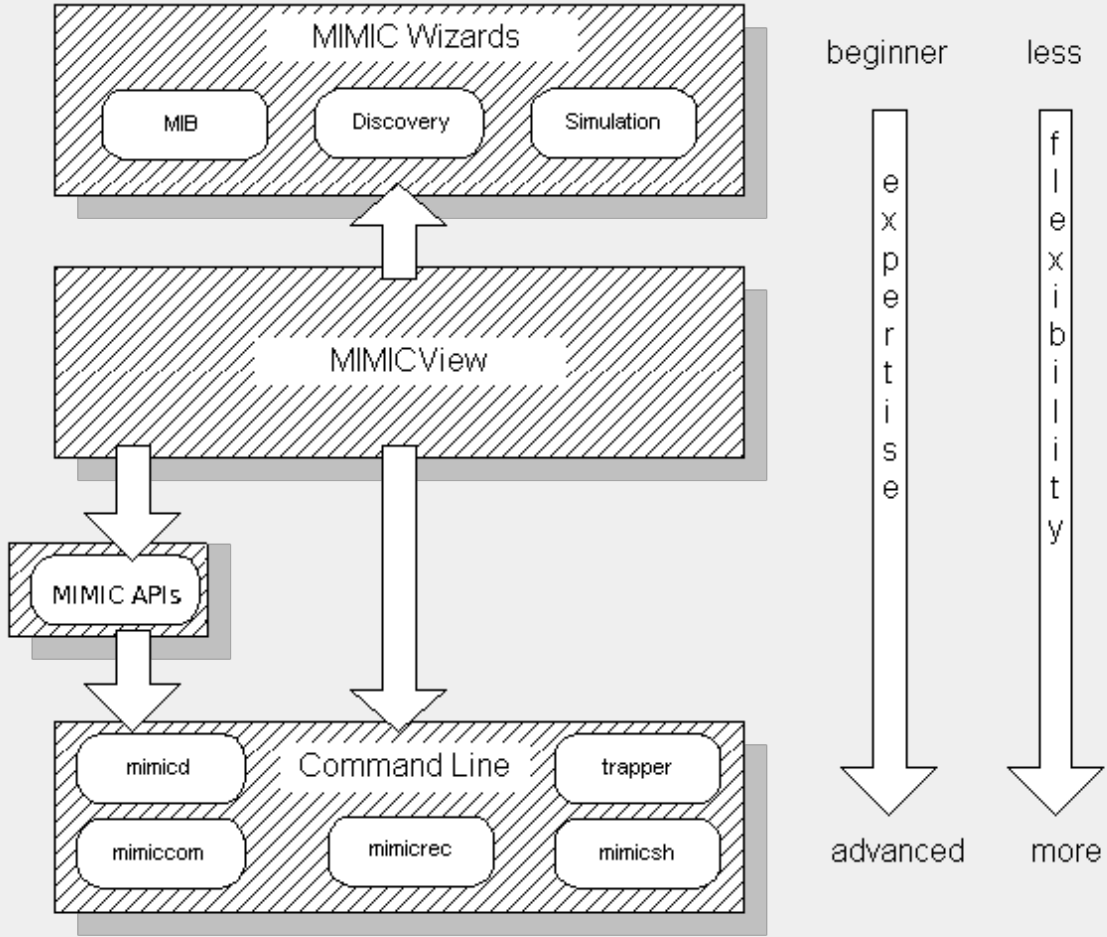
The [MIMIC sFlow Simulator](#) adds sFlow-based simulation.

The [MIMIC MQTT Simulator](#) simulates large MQTT-based sensor networks for the Internet of Things.

The [MIMIC CoAP Simulator](#) creates CoAP servers.

MIMIC Tool Suite

MIMIC provides a variety of user interfaces as shown:



The starting point for the interfaces is MIMICView, the GUI front-end to MIMIC. It provides a graphical interface to virtually all MIMIC functions.

MIMICView can launch the [MIMIC Wizards](#), which provide a powerful, yet user-friendly interface to the most common or most complicated tasks.

MIMIC contains the following main utilities:

MIMICView GUI	mimicview
Agent simulator daemon	mimicd
SNMP	mimicrec, mimiccom, trapper
WEB	webrec, webpcap, webconv
NetFlow	netflowrec
sFlow	sflowrec
IPMI	ipmiproxy, ipmirec

MQTT	mqttrec, mqttconv
APIs	mimicsh, Python, Perl, Java, C++ , PHP

For most purposes, you can launch all of these utilities, except MIMICShell, directly from MIMICView.

4. Chapter 2: Using MIMICView

Chapter Contents

- Starting MIMIC 
- Using the GUI 
- Controlling Agents 
- Running Agent Simulations
 - Running a Configured Simulation 
 - Verifying the Simulation 
 - Creating a Simulation 
 - Running a Different Simulation 
 - Shortcuts
 - Pausing an Agent
- Changing a Simulation 
- Changing a Value in the Value Space
- Generating Traps 
- Recording a Device 

Starting MIMIC

There are several options for controlling the MIMIC Simulator:

1. the MIMICView Graphical User Interface (GUI);
2. a command-line interface, such as the [Tcl-based MIMICShell](#), the [Perl command-line](#), or the [Python command-line](#).
3. a custom batch-script or application using the MIMIC API, such as [Tcl](#), [Perl](#), [Python](#), [Java](#), [C++](#) , or [PHP](#).

This guide will introduce the MIMICView GUI. For detailed information about MIMICView, there will be links to the [Simulator Guide](#), and for details about the scripting interfaces, such as the MIMICShell CLI, reference the [other sections](#) of the online help.

MIMICView is a user-friendly GUI for the simulator and most other MIMIC tools. It allows you to start/stop devices, create and modify simulations, view run-time activities, log and statistics, generate traps, compile MIBs, record real devices, control other protocols.

To start the GUI, invoke `mimicview` either from the MIMIC Program Group, or from the command line. The main front panel will be shown.

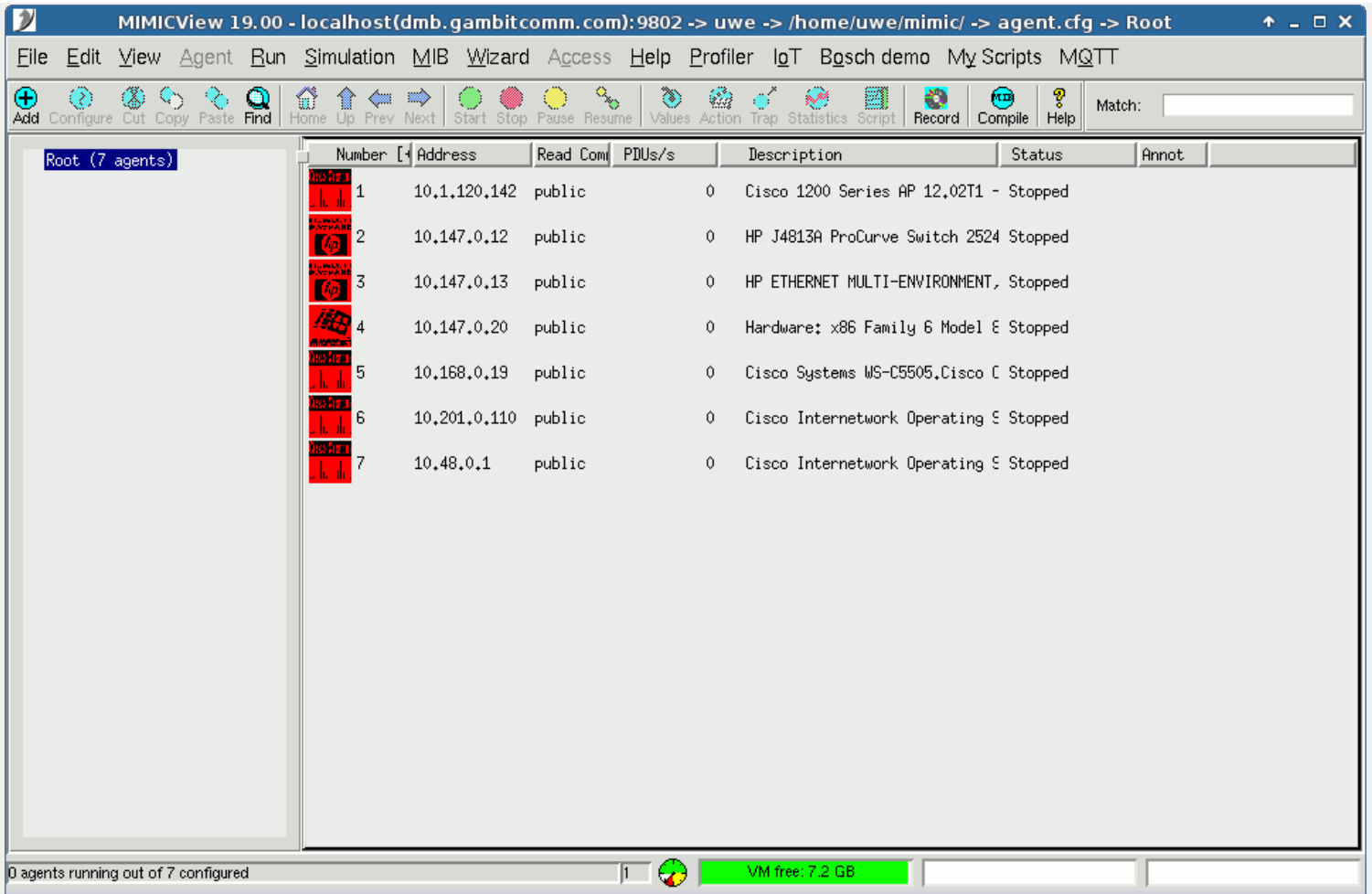


Figure - MIMICView Front Panel (default configuration)

The default Explorer view is fine for a small lab, but if you have a lot of agents you can switch to the more compact Classic view with the View -> Type submenu.

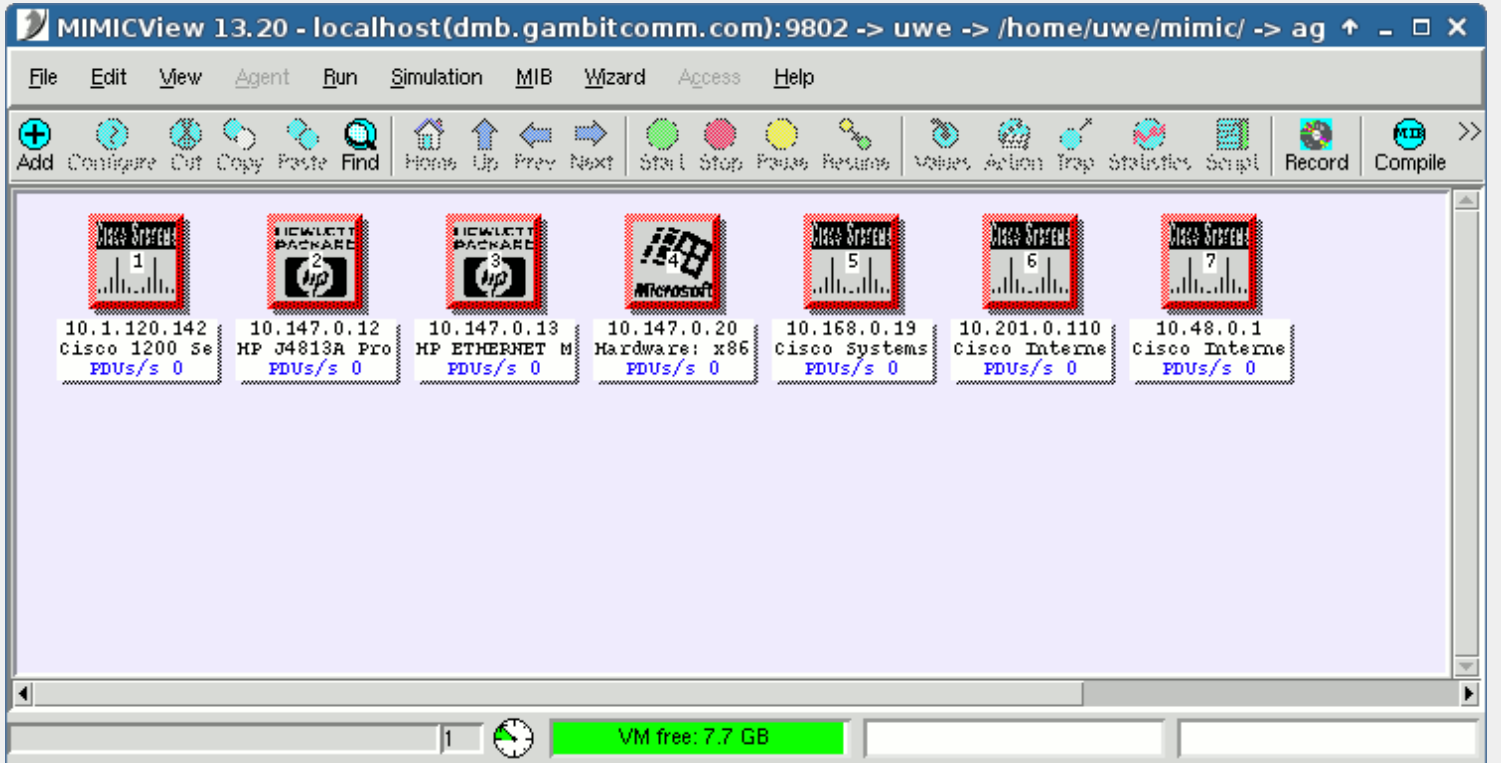


Figure - MIMICView Front Panel (Classic view)

When you first start mimicview, the MIMIC Simulator daemon will be invoked automatically if it is not already running. A log window will pop up with the output of the MIMIC Simulator daemon, mimicd. The log will start with lines similar to the following:

```
/usr/local/mimic/linux/mimicstart.sh
INFO 02/11.12:07:10 - MIMIC Simulator v14.00
INFO 02/11.12:07:10 - Copyright (c) 1997-2014 Gambit Communications, Inc.
INFO 02/11.12:07:10 - Registered individual license #2345
```

This running log is saved in the file shown in the title bar of the Log Window. It records important information and problems within the simulation.

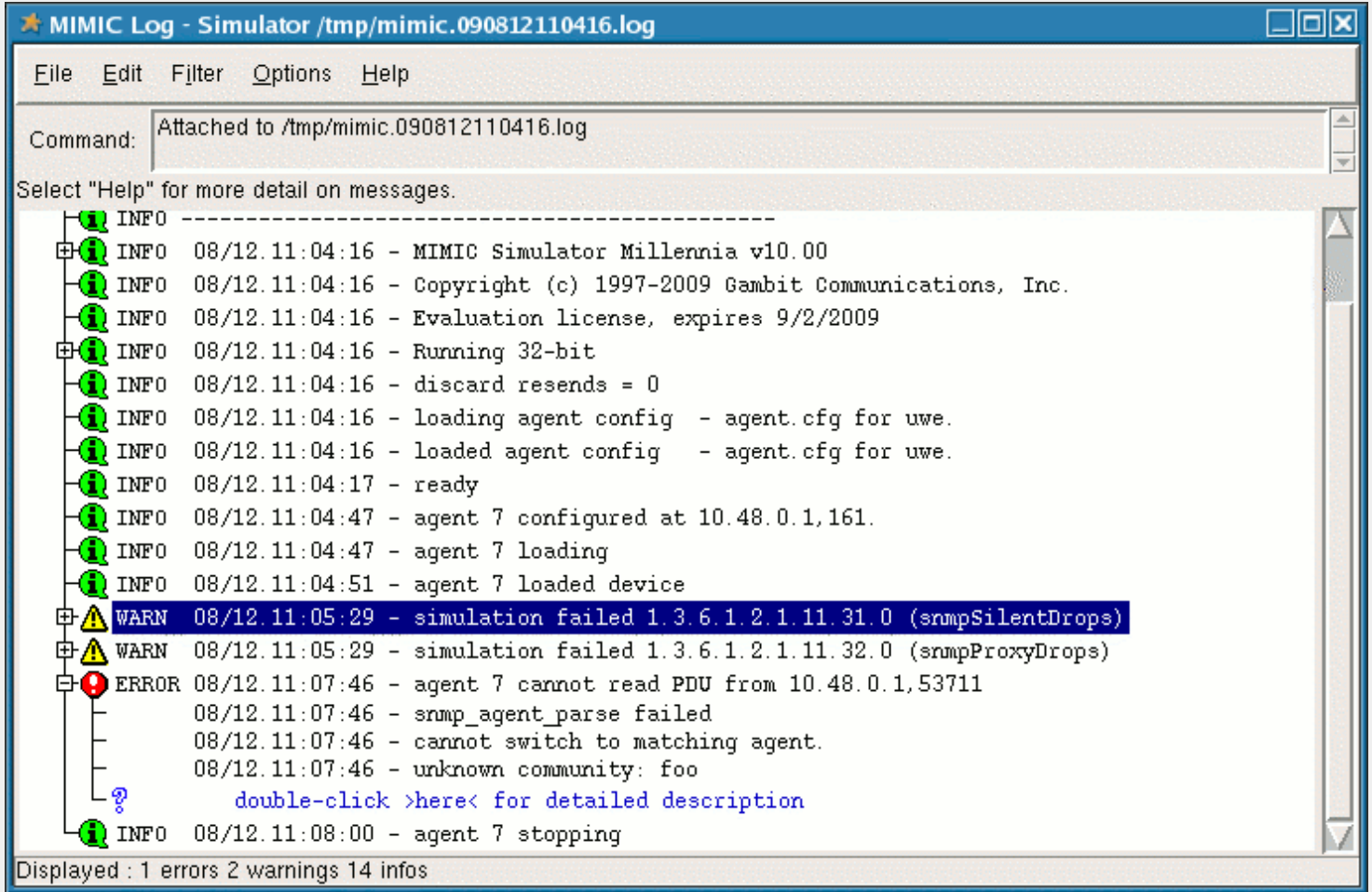


Figure - MIMICView Log Window

Using the GUI [YouTube](#)

The MIMICView GUI contains the following components:

1. the [main canvas](#);
2. the [title bar](#);
3. the [status bar](#);
4. the [menu bar](#); and
5. the [speed bar](#).

For details on their use see the [Simulator Reference Guide](#).

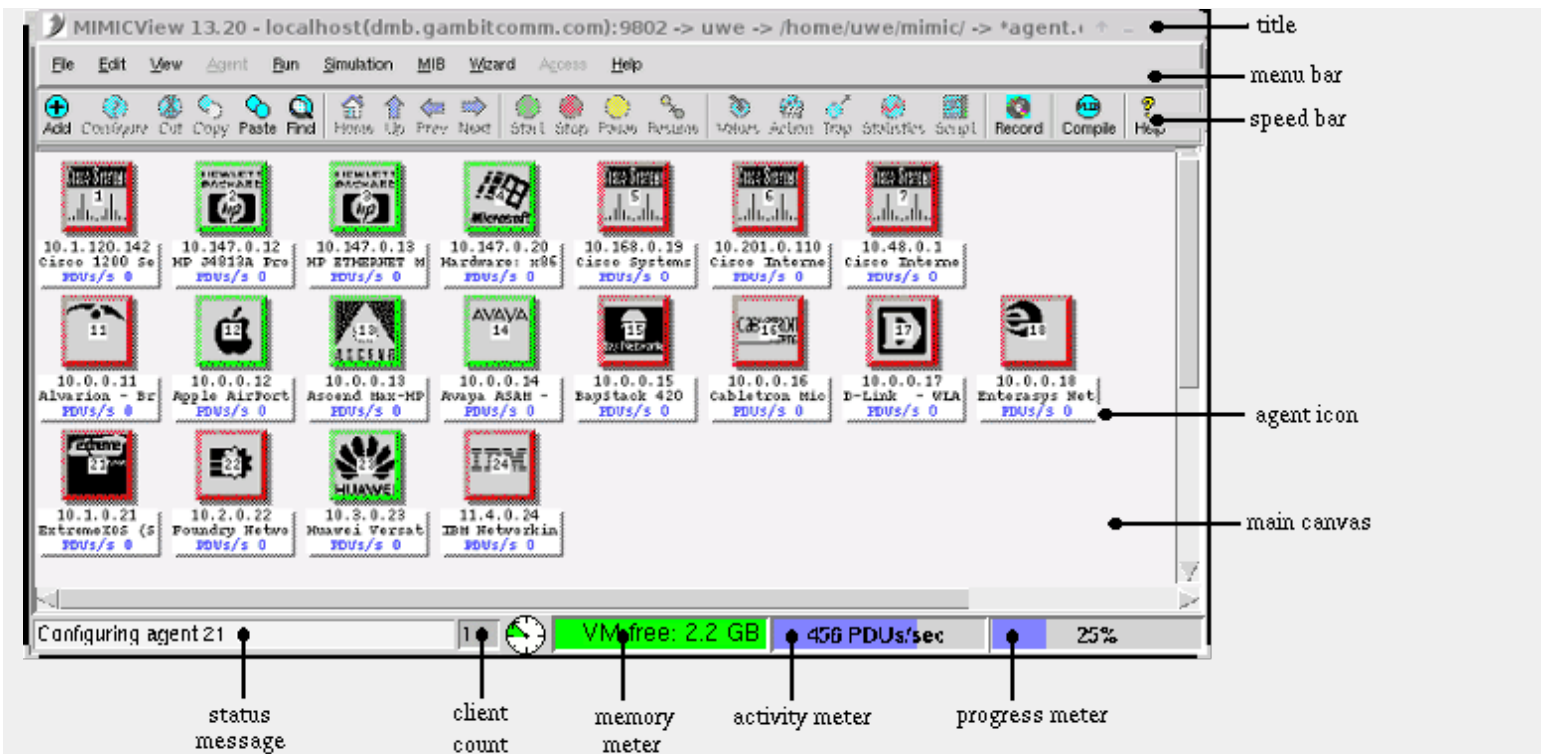


Figure - MIMICView Front Panel

The agents in the main panel are considered the **running configuration** that the simulator is running. In general, if you want to control the agents, you select their agent icons in the main canvas and perform actions with the **Agent menu items** menu items or **speed bar buttons**. The sections below introduce some of the tasks you can accomplish.

Controlling Agents

When you first start MIMICView, you will see several red icons in the main canvas. These are **agent instances** that have already been configured into the **running configuration**. Each icon shows the agent instance ID (a number from 1 to 100,000), its IP address, its **device type**, and its SNMP PDU statistics. The red color shows that the agents are stopped. We'll start them later. When you click on an agent icon, the blue border indicates that the agent is selected.



The menubar and speedbar are designed to give you a great deal of control over each of these agent instances, as described later.

For example, you can use **Edit->Configure** to display the general and advanced configuration information, as follows:

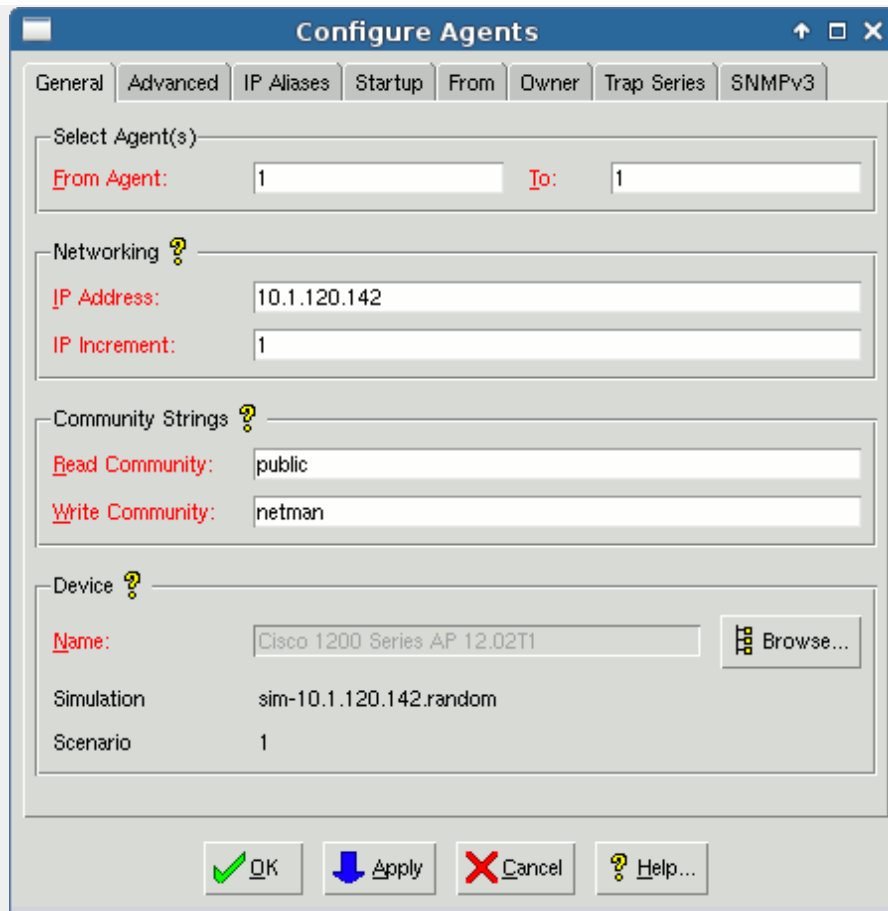


Figure - Dialogs for Configuring and Adding Agents

The selected agent instance number 1 is running at the specified IP Address, with the community strings and of device type shown.

Running Agent Simulations

Running a Configured Simulation [YouTube](#)

It is very easy to run a simulation that has already been configured. The icons in MIMICView's main canvas represent configured agents each running a simulation of a device.

To start them all, select **Run->Start**. All icons will turn green to show that the simulations on the instances are running.

To stop them all, use **Run->Stop**.

To start and stop subsets of the agent instances, select the subset and use **Agent->Start** and **Agent->Stop** as discussed later.

Verifying the Simulation [YouTube](#)

Once the agent is running, it becomes accessible to the network management application at the IP address shown below its icon in the main canvas (plus any configured [IP aliases](#)).

One of the easiest ways to verify the accessibility is by using the `ping` command. You can run this from the machine where the management application is running. If for some reason `ping` says it is unreachable, you need to use the `route` command to make the device visible, just like any real device. In case of problems, see the relevant [FAQ entry](#) for more details.

Now, you can try to access the device from the management application for example any MIB browser. You will be able to discover the device and perform management operations just as with the real device, like SNMP get/set, receive traps etc. You will see that you are using the simulator and your management application does not even realize it. In case of problems, see the relevant [FAQ entry](#) for more details.

With this, you have successfully run the application, started the simulations and connected your management application to query the MIB.

Creating a Simulation [YouTube](#)

A simulation of a device in MIMIC is more than just a set of MIBs. In database terms, the MIBs provide the schema of the information that the agent exports. The simulation needs more than the schema; it also needs the behavior, instances and values for each object in each MIB. For example, for the `ifType` MIB object in the `IF-MIB`, you have to know how many interfaces there are, and what type each is.

There are a number of ways of creating and customizing a new simulation of a device.

1. Use the ready simulated sample devices provided with the product.
2. Select from among many devices from the device and network libraries. They are included in the CD you have received or are downloadable with [Update Wizard](#) or from Gambit's website.
3. Record a real device using [MIMIC Recorder](#). The Recorder basically traverses the entire MIB of the real device, retrieves the value of each MIB object, and creates a basic simulation from that.
4. Modify the existing simulation to fit your requirements.
5. If you cannot record a device (for example, you don't have it, or it is under development), then you can create the simulation by hand, as detailed in the [Simulation Wizard Reference section](#).

Refer to the [Create Simulation](#) section below for details.

Running a Different Simulation

To run a new simulation, you need to configure a new [agent instance](#) in the [Simulator](#):

1. In MIMICView, use [Edit->Add->Agent...](#) to add an agent instance. A dialog with the title `Add Agents` pops up, where you can fill out configurable parameters for the new agent instance. The Agent ID is filled in with the first available agent instance number.
2. Fill in the desired IP address in the `IP Address` text field.
3. Pick a simulation by clicking the `Browse...` button in the `Device` section. A selection list of available device simulations pops up. There will be an entry for each successful recording you performed with the [MIMIC Recorder](#) plus the pre-defined simulations that are provided with MIMIC.

Click `OK` to accept the parameters and add the agent.
4. A new agent icon pops up in the main MIMICView canvas for the agent instance you just added. This icon is red, because the instance is not running yet.
5. To start the instance, select the icon with a left mouse click. (A blue border means that an agent instance is selected.)
6. Select [Agent->Start](#), and the agent is started. (Its icon turns green.)
7. When done, stop the instance with [Agent->Stop](#).

You can reconfigure a stopped agent instance with [Edit->Configure...](#), which brings up the `Configure Agents` dialog. The input fields are the same as in the `Add Agents` dialog.

Shortcuts

Besides the ALT+letter keyboard shortcuts for menu entries, MIMICView also accepts the Tab key as a shortcut to the most common actions, which are shown in the [speedbar](#) below the top menu bar.



In addition, you can right-click on an agent icon to select the agent, and pop up a copy of the `Agent` menu.

In this tutorial we will continue to use the menu entries for clarity. We suggest you use them until you get familiar, then start using the shortcuts.

Pausing an Agent

Once an agent instance is running, you may want to investigate certain behavior of a management application at a specific point in time, that is, as if time had stopped. MIMIC allows this by pausing the simulation running on an agent instance. This lets you compare values relative to each other or to values retrieved in prior sessions.

To pause an agent instance:

1. Select the desired agent icon(s).
2. Use [Agent->Pause...](#) to display a dialog with the title `Agent Pause`. Notice that the `Now` radio button is selected.
3. Click `Apply` to pause the simulation immediately.

Or, you can pause the simulation and set the time to a specific exact time:

1. Click the `Set Time` radio button.
2. Enter the time that you want the simulation to be set at. For example, to pause the agent as if it had been running for five days, enter 5 into the `Days` field.
3. Click `Apply`.
4. Click `Cancel` to exit the dialog.

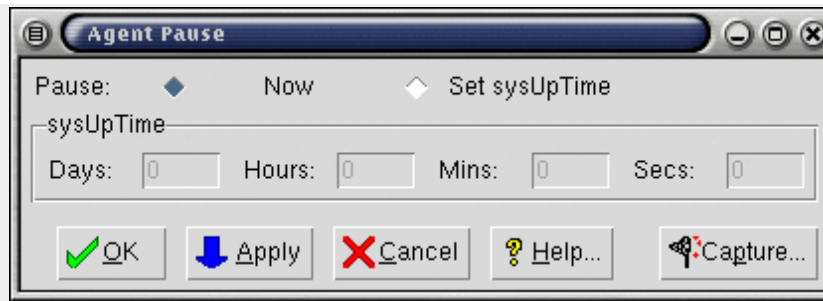


Figure - Dialog for Pausing Agents

The agent's icon turns yellow to show that its simulation is paused.

If you use any MIB browser, you will see that the `sysUpTime` MIB object is stuck permanently at the desired time. Other objects, especially counters, return the appropriate values for that time.

You can resume the simulation with `Agent -> Resume`. The simulation will resume at whatever time was set when the agent was paused.

Changing a Simulation [YouTube](#)

Changing a Value in the Value Space

MIMIC allows you to customize a running basic simulation in real-time by manipulating values returned by objects in the [Value Space](#).

You can inspect values by taking the following steps:

1. Select an agent icon that is green, i.e., running.
2. Use `Agent -> Value Space...` to pop up the `Value Browser` dialog.

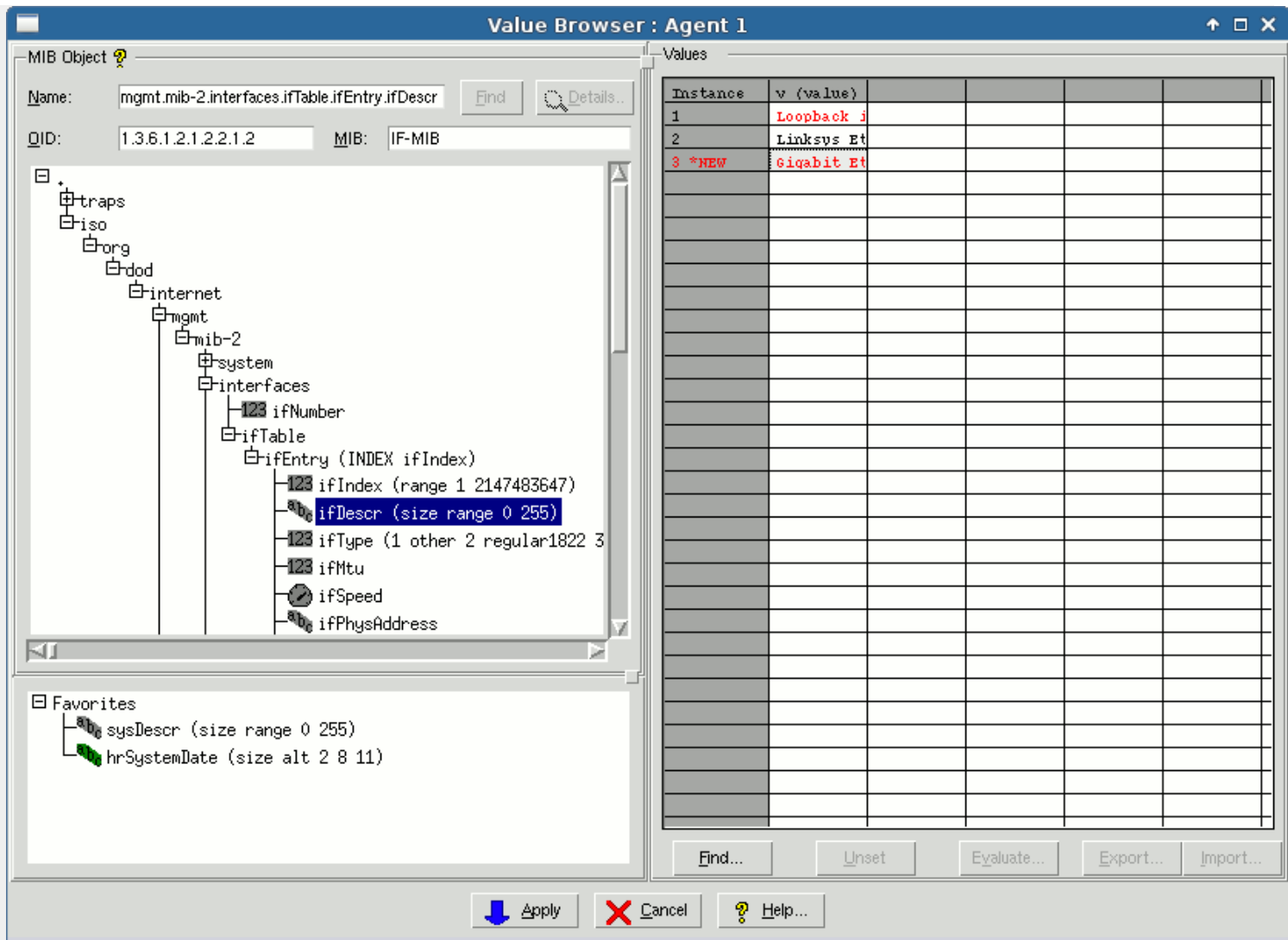


Figure - Value Browser Dialog

3. Enter or select the MIB object from the MIB tree; for example `sysDescr`. You can either type in the MIB object name in the Object field, then press `Find`, or browse through the MIB tree.
4. The right-hand matrix shows all instances of the object as rows. For scalar objects (such as `sysDescr`, the single 0 instance is shown).
5. The columns in the matrix show defined variables for this object. A non-counter object such as `sysDescr` should only have one variable, `v`. Since this MIB object is not a counter, this variable is used in its simulation.
6. The value of the variable is shown in the cell.

To change values in the Value Space:

1. Select the cell you want to change, by clicking on it once. The cell changes to edit mode.
2. Type in the desired value; for example `foo`.
3. Press `RETURN`. The cell returns from edit mode and shows the value in red to show its pending status.
4. Click `Apply` to commit the changes to the Value Space.

The value of the variable will be changed in the Value Space. Subsequent SNMP queries for this MIB object instance will return the new value.

This is uninteresting for the `sysDescr` MIB object, but useful to set the status of network interfaces via the `ifOperStatus` object, or to change rates of Counter objects in real time.

Generating Traps

Traps are generated with the `trap_periodic` simulation function, used by default for all traps in devices recorded with the MIMIC Recorder. The MIMICView Generate Traps dialog provides a simple interface to set the necessary parameters:

1. Use `Edit -> Trap Destinations...` to display the Global Trap Destinations dialog or `Agent -> Trap Destinations...` to display the Agent Trap Destinations dialog. With these, you can display, add and delete global or per-agent trap destinations.

2. In the `Destination IP,port` field type the address of your trap destination. This should be a host running a management application that is prepared to receive traps. The port number is optional, and should be used when your application is accepting traps at a non-standard port. Click `Add >>` to add the destination.
3. Click `OK` to accept it.
4. Select the agent icon that you want to start generating traps. To set the correct variables in the `Value Space` which control the traps you want to be sent, use `Agent ->Generate Traps...` `Agent ->Generate Traps...` to invoke the `Generate Traps` dialog.
5. Type the name, for example `linkDown`, of the trap into the `Object` text field, or click `Browse...` to browse for the trap name. SNMPv1 traps are listed in a separate hierarchy under the top node in the MIB OID tree. SNMPv2 traps are listed in their respective groups in the MIB OID tree. If there is no traps tree visible, double-click on the top node (the one marked `.`). For example, the `linkDown` trap is listed under `.traps_iso_org_dod_internet_mgmt_mib-2_snmp`. Notice that the path to the traps is analogous to the path to the enterprise OID under which they are defined. Keep double-clicking on the appropriate nodes until you reach `linkDown`. Double-click on it to select it.
6. Specify values to cause the traps. For the default trap simulation, the trap generation is controlled by three variables: the `Rate`, `Time Unit` and `Cutoff Time` parameters, which you can type in the text fields. The trap will be generated at the specified rate over time unit, for the time period specified in `Cutoff Time`.
7. To enable the `linkDown` trap at a frequency of 1 every 3 seconds for 5 minutes, set the `Rate` variable to 1, the `Time Unit` variable to 3, and the `Cutoff Time` variable to 300.

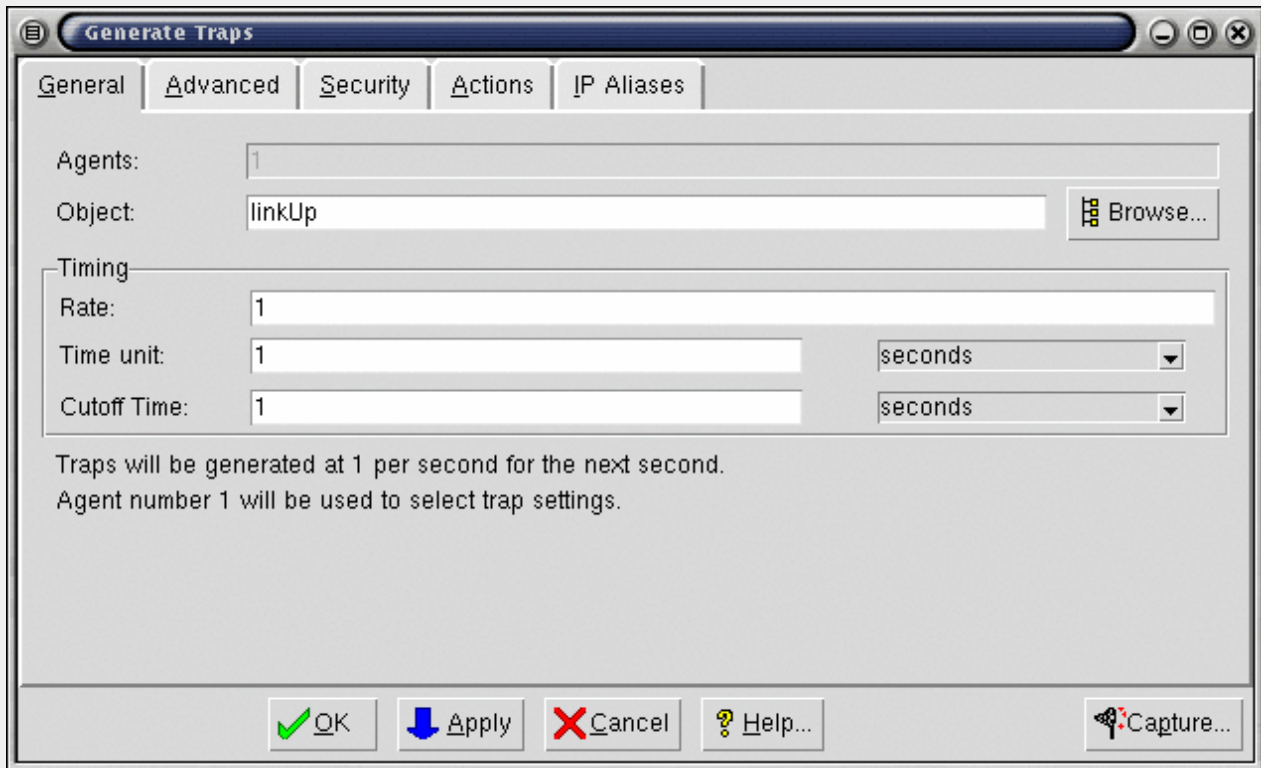


Figure - Dialog for Generating Traps

8. Click `Apply`. The specified traps will start to be generated for this agent instance.

To verify the trap generation, look at agent instance statistics `Agent ->Statistics` for this agent instance.

If you look at the traps generated with a protocol analyzer, or examine the log at the trap destination (if it has one), you will see that the `ifIndex` variable sent with the traps is assigned a `NULL` value. You can assign the value of any of the MIB variables to be sent with a trap by specifying them in the `Advanced` tab of this dialog.

To assign, for example, a value to the `ifIndex` variable with the `Generate Traps` dialog (all fields should be filled in by now), click the `Advanced` tab, and select `ifIndex` from the list. In the text field type the value to be set for this variable, for example 5. Then click `Modify`.

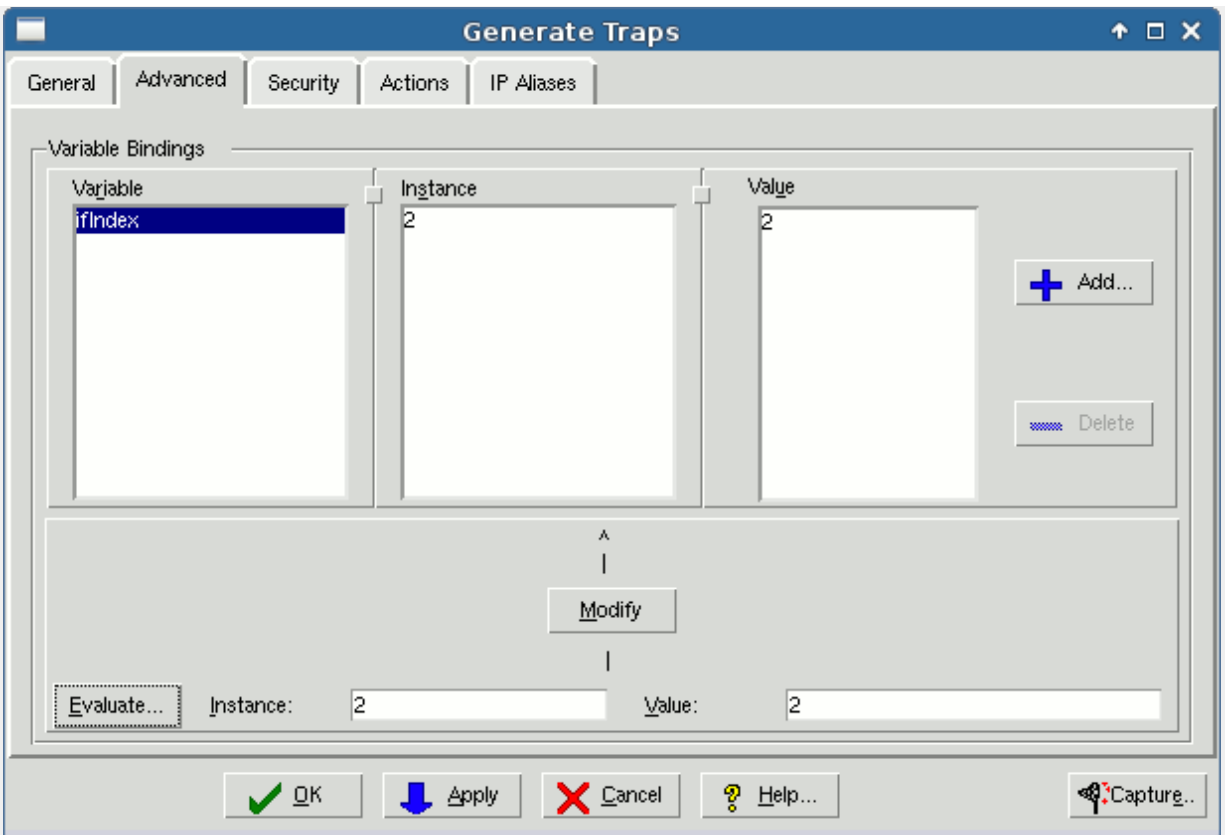


Figure - Dialog for Generating Traps

When you click **Apply**, the desired MIB variable value will be sent with the traps.

Recording a Device

The simplest way of creating a simulation is to record an actual real world device with the [MIMIC Recorder](#). There are other alternatives as described in the [Create Simulation](#) section below.

To record a device:

1. Use [Simulation -> Record Live...](#) in MIMICView.

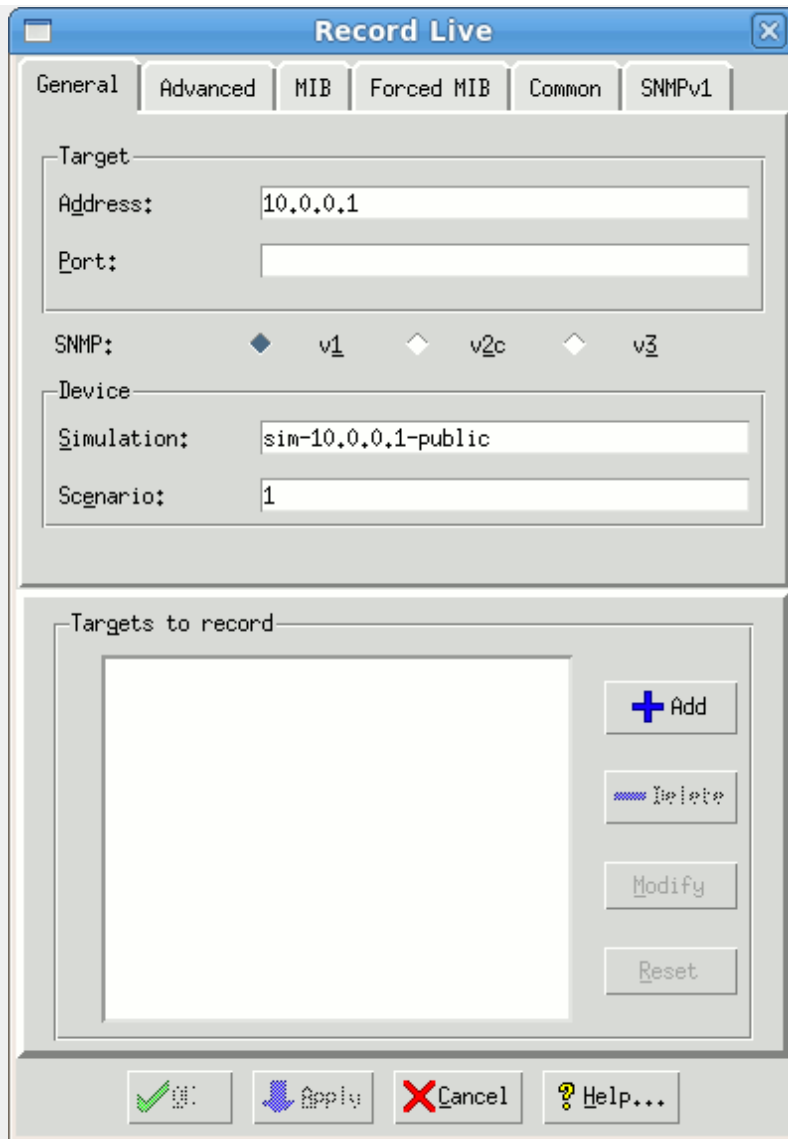


Figure - Dialog for recording a live device

2. In the Record Live dialog General tab, fill in the Target text field with the address or hostname of the target device to record. To use the default SNMP port (161), you do not need to fill in the Port field. Pick your SNMP version, the simplest being the SNMP v1 button.

3. The dialog suggests a default simulation name sim-address.community for your simulation. Alternatively, fill in any simulation name, such as test, or a short name for the device you are recording, or your first name. Use any Scenario name, such as a small number, for example, 1.

These names will be significant once you start doing advanced tasks. Pick easy names to remember.

4. Use the SNMPv1 tab to specify the Community String with which to access the device. The simplest read community string is public, but will depend on device configuration in your network.

5. Use the MIB tab to restrict the MIB objects to record, such as excluding certain trees, starting at a desired MIB object, etc. You rarely have to fill in this information.

6. Click Add to add the target to the list.

7. Click OK. The Recorder will record all targets in the list.

8. A recording session starts, and a log window of the ongoing recording is displayed. This log is also saved for future reference in a file whose name is shown in the title. The successful recording will end with lines such as:

```
INFO 06/17.17:58:29 - line 3660, 15 sec elapsed, 100 % done
INFO 06/17.17:58:29 - added device "Cisco Systems WS-C5000 - test, 1234"
INFO 06/17.17:58:29 - errors for 0 objects out of 3660
INFO 06/17.17:58:29 - simulation phase: done
```

Notice that the Recorder keeps track of the number of errors when attempting to simulate the device. The fewer errors, the better the simulation will be.


9. Choose **File->Close** to remove the log window.

10. Press **Cancel** to exit out of the **Record Live** dialog.

Once you have successfully recorded a device, it is added to the list of known devices with the name of its system description, as returned by the `sysDescr` MIB object. You can then assign it to an agent instance and run it as described in [Running a new Simulation](#).

i. Chapter 3: Troubleshooting

Chapter Contents

- [Online Help](#)
- [Known Problems](#)
- [Inspect the Log](#) 
- [Common Errors](#)
- [Common Questions](#)
- [Diagnostic Wizard](#)
- [Crashes](#)

This chapter lists the recommended troubleshooting procedures for quickest resolution of your problem.

Online Help

All MIMICView dialogs have a context-sensitive online help section, which you can invoke with the `Help` button.

Known Problems

Each of the supported platforms has known problems. Check there first to see if yours is one of them:

- [Linux](#)
- [Solaris](#)
- [Windows](#)

Inspect the Log

MIMIC logs all abnormal events in a [Log](#). In case anything goes wrong, inspect it first.

Common Errors

Common errors in the log are detailed in [Appendix C - Common Error Messages](#). Consult this section for details on your particular error.

Common Questions

Common questions and their answers are detailed in [Appendix D - Frequently Asked Questions](#).

Diagnostic Wizard

MIMIC, as any other complex software, sometimes will exhibit problems. If the other troubleshooting tips in this section don't help, then you are encouraged to submit a problem report to our Technical Support Department (support@gambitcomm.com). The [Diagnostic Wizard](#) makes it easy to submit a report for most problems.

Crashes

MIMIC, as any other complex software, occasionally terminates abnormally (crashes). In order to help us diagnose and fix the problem, we will request you to provide some additional information about the problem with the [Diagnostic Wizard](#), such as

- how did the crash occur?
- what simulation was running?
- how long had MIMIC been running?
- can you reproduce the crash?

In addition, we will request you to enable dumping of process memory on the crash. Details for Solaris are in the [Solaris Installation](#) and for Windows in the [Windows Installation](#) sections.

i. Chapter 4: Process

Chapter Contents

- [Important Concepts](#)
 - [What Is a Device Instance?](#)
 - [What Is a Simulation?](#)
 - [What Is an Agent Instance?](#)
 - [What Is a Lab Configuration?](#)
 - [What Is the MIMIC Value Space?](#)
 - [What Are Actions?](#)
 - [What Is the Private Data Area?](#)
 - [What are Maps?](#)

- [MIMIC Process Overview: When to Use the Tools](#)
 - [Overview](#)
 - [Compile MIBs](#)
 - [Create a Simulation](#)
 - [Run a Simulation](#)
 - [Customize a Simulation](#)
- [References for Further Reading](#)

Important Concepts

What Is a Device Instance?

In MIMIC terms, a **device to be simulated is a real-world entity on a network that supports a well-defined interface**, ie.

- managed primarily via the Simple Network Management Protocol ([SNMP](#)); or
- command-line interfaces (CLI) such as [Cisco IOS](#), JUNOS, TL/1; or
- [Web](#) interfaces such as SOAP, XML; or
- Intelligent Platform Management Interface ([IPMI](#)); or
- exporting [Netflow](#), IPFIX; or
- exporting [sFlow](#); or
- an Internet of Things (IoT) device publishing telemetry via [MQTT](#); or
- Constrained Application Protocol ([CoAP](#)).

or a combination of all the supported interfaces. Through these interfaces the **system under test** interacts with one or more devices. MIMIC simulates these devices for purposes of development, testing, technical support, training, pre-deployment sizing.

To be manageable via [SNMP](#), the device exports a Management Information Base (MIB) with embedded software called an SNMP agent. The MIB is usually composed of a collection of standard and enterprise-specific MIB fragments, for example, MIB-2, IF-MIB, or SNMP-REPEATER-MIB, which we just call MIBs. Each MIB is defined in a syntax called [Structure of Management Information \(SMI\)](#).

The system under test, ie. a SNMP-capable network management application, interacts with one or more SNMP agents by manipulating MIB objects.

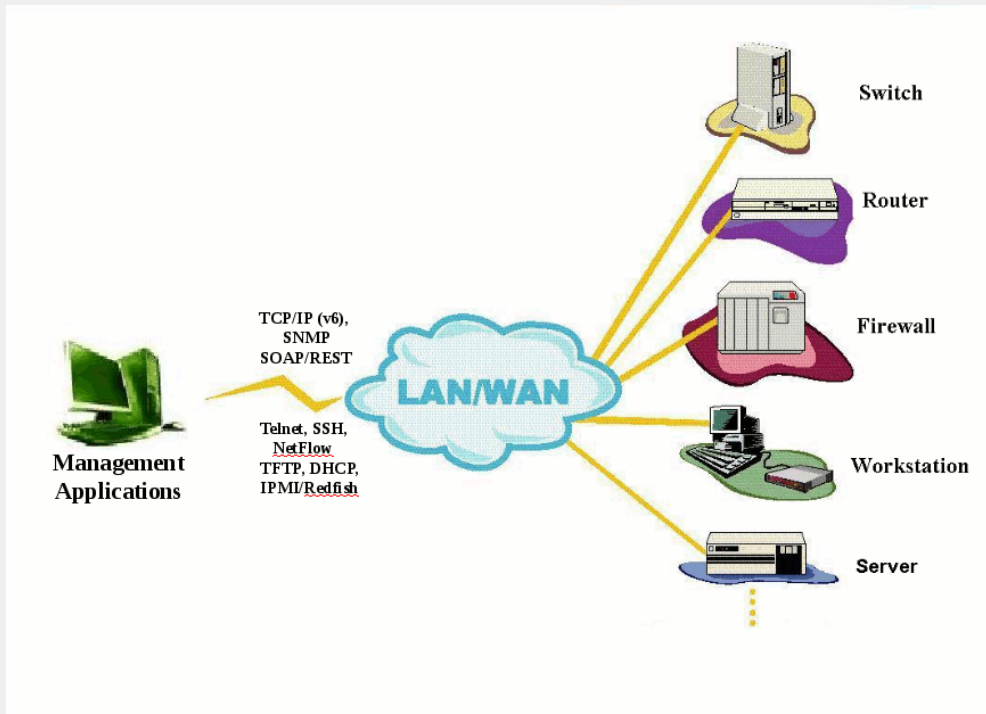


Figure - Network Management Topology

This can be generalized to all interfaces to say that **the system under test interacts with the simulated devices through the interface, exactly as it would to a real device**.

You use MIMIC to simulate one or more instances of a device from the interface perspective, eg. you simulate the SNMP agent, or command-line, or WEB server, etc. There are many different classes of devices, from data communications equipment to end systems, from tele-communications equipment to databases, to servers, to IoT sensors, gateways, etc.

A device has certain characteristics that distinguishes it from others:

- A **device type**, that determines which MIBs the device exports, or which command-line, or WEB interface. Examples include a Cisco Catalyst 5000 switch, a Cabletron MMAC hub, an HP Lanprobe, or a Microsoft Windows NT PC, etc. To be managed by SNMP network management applications, each of these device types exports, via SNMP, a different set of MIB objects. If accessed through a CLI, the set of commands is different for a router than for a switch or a firewall.
- A **version number**. A network typically has multiple versions of the same product coexisting at any point in time. Different versions of the same device type often have different MIBs, because the network management interface also evolves.

A **device instance (agent)** has additional characteristics particular to SNMP, among others:

- An IP address that identifies it on the network.
- A particular version of SNMP that it understands (v1, v2c, v2, v3).
- Access parameters (ie. read and write community strings for SNMP v1) that provide access to the MIB
- Specific instances and values of the MIB objects. For example, two devices of the same device type may have a different set of network interfaces.
- Uptime, that is the time that the SNMP agent on the device instance has been running. For example, an RMON probe will have different data after it has been running for a while, as compared to when it was first booted.

The device may use other secondary protocols such as [DHCP](#), [TFTP](#), or [TOD](#). to perform management functions, each with its own attributes.

What Is a Simulation?

A protocol simulation is the act of allowing protocol interface interaction with standard applications just as with a real-world device, but without the actual physical device.

For SNMP that means exporting MIB object instances and values, generating TRAPS.

For command-line interfaces (CLI) that means exporting a command set such as Cisco IOS via telnet or SSH.

For flow-based protocols the simulation exports via NetFlow or sFlow the template and data-flows.

For Web-based interfaces the protocol on top of HTTP or HTTP/S would be specific to the device type.

The network management applications interact with the simulations within MIMIC just as it would with real-world devices.

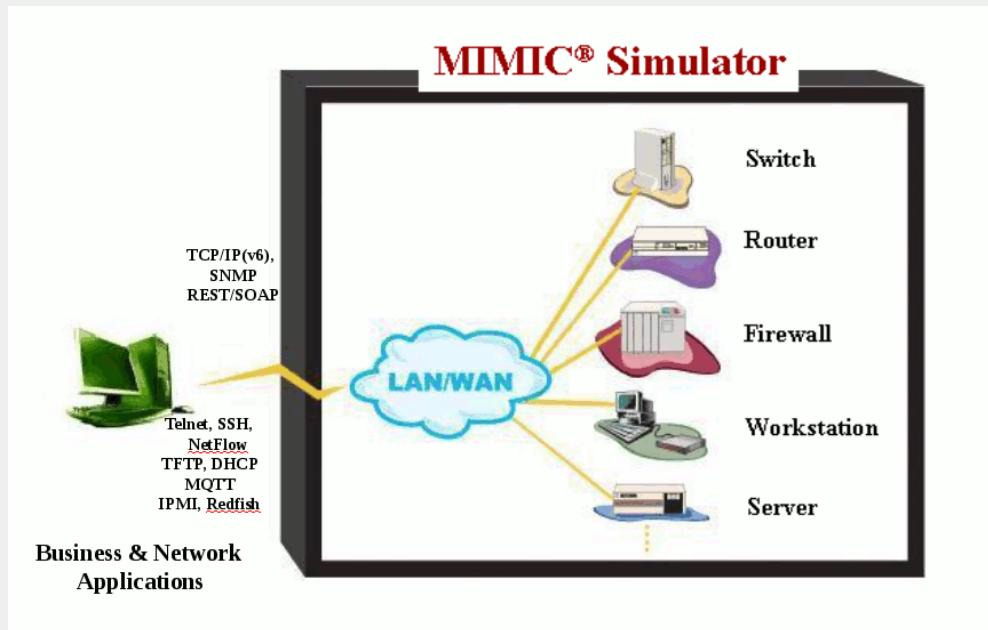


Figure - Simulations with MIMIC

The most common is a **"realistic"** simulation, i.e., it attempts to duplicate the behavior of a real-world device. Realistic simulations are good for demonstrating the capabilities of a network management application in a pre-sales situation, or as part of training exercises. MIMIC implements realistic SNMP simulations as follows:

- MIB objects that are Counters are simulated with an average rate and are perturbed with a random fudge factor;
- All other MIB objects assume the value that would be returned by the device.

For other interfaces, the values in responses are particular to the type and deployment of the device. For example, CLI commands or WEB requests yield responses with very specific values. You want the simulations of those interfaces to reflect the real responses.

For a particular device, you can run any number of simulations, just as in the real world there are a number of scenarios in which a device can be involved. For example, you can run a router simulation of a lightly loaded device or overburdened device, or any range of scenarios in between. Or, you can simulate an RMON probe on a healthy network segment, or a probe that is monitoring a segment with either a high traffic load or failing devices. From a network management perspective, the difference is seen merely in the instances and values of returned MIB objects.

Simulations with randomness have their limitations in the case of product development and testing, because the values of MIB objects are unpredictable. For this use, MIMIC allows to run "constant" simulations. This type of simulation makes Counter objects return a value with a constant rate.

What Is an Agent Instance?

An agent Instance is a simulation of a device instance within MIMIC. As such, the agent instance will have all the characteristics described in the previous section.

In addition, agent instances can be manipulated in ways that real-world devices cannot:

- Time can be stopped and restarted. This is useful in a technical support situation for investigating a problem at any particular point in time.
- Values of the simulation can be changed at will during the simulation. This allows "what-if" investigations and regression testing. For example, reaction time of an event correlation application to a certain set of events (as reflected in changes of MIB objects) can be accurately gauged.

What Is a Lab Configuration?

Since MIMIC can run multiple agent instances at once, the lab configuration describes what simulation each agent instance is running. For example, the first agent instance can be running a particular Cisco router simulation at a specific address, the next 50 agents could be running Windows end-node simulations. Or, you could have a different lab configuration with 10,000 sensors of different types publishing MQTT telemetry to an IoT application.

The lab configuration that is currently running on the Simulator is called the `running` configuration. MIMIC makes it very easy to use different lab configurations. You can load them, change them, and save them. This is the basis for the "virtual lab" concept: you can reconfigure your lab at any time with a different lab configuration.

What Is the MIMIC Value Space?

As described [above](#), a simulation can take on several flavors, just as a real-world device can be involved in a variety of situations. A MIMIC simulation can similarly run a variety of scenarios. For example, a realistic simulation of a device can exhibit high traffic rates in one scenario, and low rates in another, and high error rates in yet another. And in more complicated scenarios, values and rates can change dynamically during the simulation.

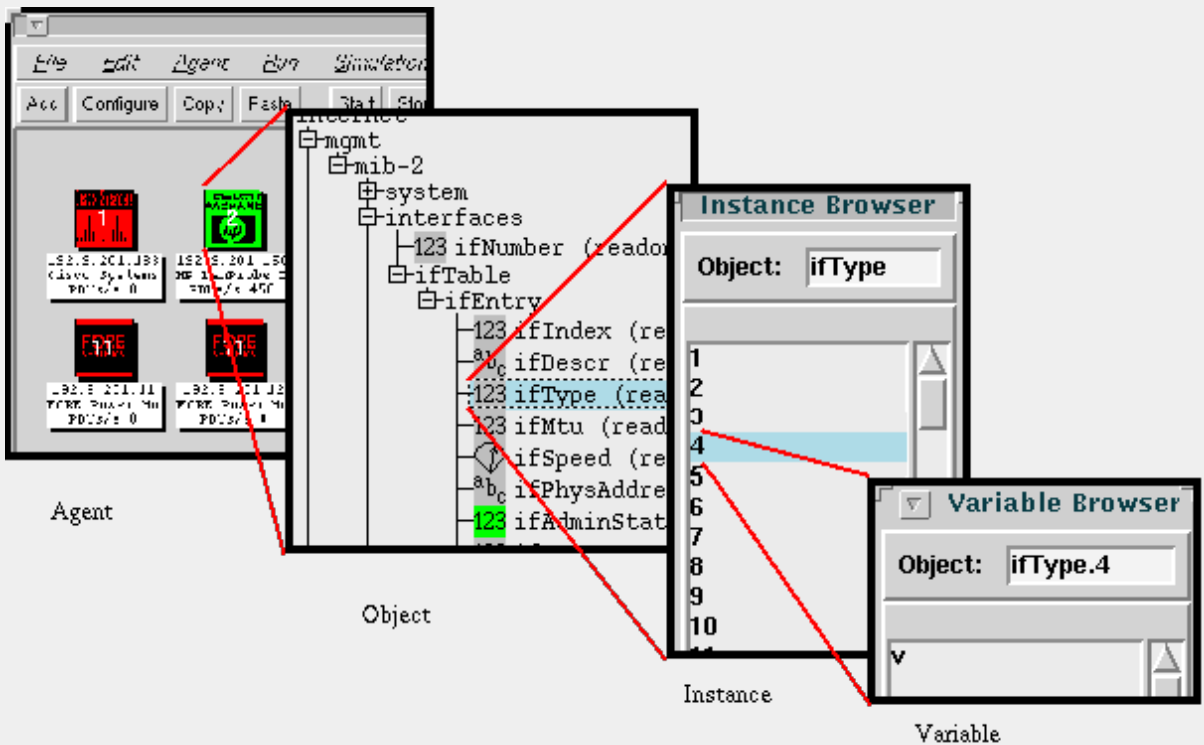


Figure - MIMIC Value Space

This is accomplished in MIMIC with the MIMIC Value Space. In simplest terms, the MIMIC Value Space contains the MIB object values for an agent.

More precisely, it is a 6-dimensional space which for every simulation scenario bound to an agent instance, stores for every MIB object instance, a set of variables each with its own value. For example, a different traffic rate for every interface of an agent can use a variable called "rate" in the MIMIC Value Space for the appropriate MIB objects; the simulation can then use this variable to simulate the different traffic rates.

MIMIC uses certain conventions for variable names in the Value Space (although any name can be used for any purpose). For all objects that are not Counters, MIMIC creates basic simulations that use the variable `v` (for value). For Counter objects, MIMIC uses the variables `r` (for rate) and `tu` (for timeunit). For Traps, MIMIC uses the variables `r` (for rate), `tu` (for timeunit) and `c` (for cutoff-time). (You will see how these are used later.)

The MIMIC Value Space allows you to manipulate variables at runtime; that is, change the way the simulation behaves in real time. For example, you can raise or lower the traffic rate seen on an interface by changing the values of the associated variables in the Value Space.

What Is the Basic Simulation?

By default, the MIMIC Recorder creates a **"basic simulation"** for a target device that is close to a realistic simulation described [above](#) . The basic simulation is created by default when recording a real device.

For SNMP here are the characteristics of this simulation:

- static objects

For static objects, the basic simulation will return the value that was recorded by the Recorder. For GET* SNMP requests, this is accomplished by retrieving from the [Value Space](#) a variable "v" with the value for the MIB object instance. For SET requests, the value is stored in the "v" variable.

- counter objects

The simulation for counter objects is rate-based. The variables in the Value Space that determine the value at any point in time are "r" (rate) and "tu" (time-unit in seconds). The value of the object is determined with the formula

$$\text{value} = \text{current-time} * r / \text{tu}$$

- traps

The simulation for trap objects is rate-based. The variables in the Value Space that determine the trap frequency at any point in time are "r" (rate), "tu" (time-unit in seconds) and "c" (cutoff-time in seconds). A trap is generated at a rate of r/tu for the specified period of time. There are some more variables that control advanced aspects of trap generation, as detailed in the [Agent ->Generate Traps](#) section.

For other protocol interfaces, the basic simulation is described in the specific protocol module guide.

What Are Actions?

As described in the previous section, by default the basic GET simulation of a MIB object looks up a variable in the [MIMIC Value Space](#) and returns its value. The basic SET simulation changes the appropriate variable in the MIMIC Value Space.

If you want to do more sophisticated things, like having side effects on a SET, or more complicated GET processing, you can accomplish this with **MIMIC actions**. These actions customize the behaviour of the Simulator for a variety of events. For example, if you wanted to change the values of other MIB objects as part of a SET or generate traps, you would use a SET action script for that object.

A MIMIC action script is a custom file that is executed by the simulator for the following pre-defined events.

1. SNMP Request action

The first type of action can be configured to be executed when a MIB object instance is accessed through SNMP requests (or Protocol Data Units (PDUs): GET, GETNEXT, SET, GETBULK). You can associate a request action script with the MIB object using the [Agent->Actions->On GET/SET...](#) dialog.

2. SNMP TRAP action

The second type of action can be associated with trap generation. Trap action scripts are run every time the associated trap is generated. These actions are executed before a trap is sent by an agent. They provide a good place to modify the contents of the trap as well as provide some side effects. E.g. trigger other traps, modify tables, modify varbinds etc. For details, see the [Agent->Generate Traps...](#) dialog.

3. Timer action

The third type of action can be scheduled to run periodically. These actions are executed repeatedly based on a timer specification. They can be executed either in a global context or an agent context. They provide an excellent way to implement constantly changing behaviour. For e.g. send traps or change table contents at varying interval. Timer action scripts can be scheduled with the [Run->Timer Scripts...](#) or [Agent->Timer Scripts...](#) menu item.

4. Agent startup/stop action

The fourth type of action script can be executed on starting or stopping an agent. If the simulation directory contains `start.mtc1` or `start+*.mtc1` scripts, then they are executed on agent start (in alphabetical order). Similarly, `stop.mtc1` or `stop+*.mtc1` scripts are executed on agent stop. These could be useful to issue certain traps on startup (eg. coldStart, warmStart) or shutdown.

5. Protocol action

The fifth type type of action script can be configured for other special events in the other protocol modules, such as on completion of downloading a file via TFTP, etc.

6. Simulator action

As an additional point of customization we support simulator- wide start and stop scripts. These actions are executed on startup or shutdown of the simulator. They provide a well defined point of customization to initialize and/or cleanup any simulator wide data. For e.g. a start action could be used to configure and start agents, while a stop action could be used to save any changes. All restrictions that apply to timer scripts or request action scripts apply to these too. These scripts will be located in `scripts` subdirectory in the searchpath.

Simulator start actions are named `mimicd+start.mtc1` or `mimicd+start+*.mtc1`. Stop actions similarly are named `mimicd+stop.mtc1` or `mimicd+stop+*.mtc1`. The actions are invoked in alphabetical order according to their names. The start actions are invoked after the steady state is reached. The stop actions are invoked as soon as the simulator is terminated.

Actions can be programmed using multiple languages. The following section details specifics for each of the supported languages :

- Tcl scripts

Tcl is a popular scripting language available as an open-source project. It is also extensible with a lot of packages available freely to automate a lot of common scripting activities. Tcl scripts are interpreted on the fly (not compiled) and hence one can use a trial-and-error approach to writing quick and efficient action scripts. The commands to control MIMIC are the same as in the [MIMICShell](#).

- C++ based DLLs

Although Tcl is flexible and easy-to-use, it has performance limitations because of being an interpreted language. The C++ API, exported from the 'Mimic' namespace, was designed to alleviate this problem. The C++ actions however need to be compiled into a DLL binary. It also needs to export a specific interface that can be invoked by the simulator. The details are in the [C++ API Guide](#)

What Is the MIMIC Private Data Area?

The data files used in MIMIC can live in either a shared or private data area. The **shared file space** is under the MIMIC directory that is created when you install MIMIC. After the initial install, it contains the precompiled MIBs, pre-recorded simulations, and other files that are supplied with MIMIC (see [Appendix A Directory Structure](#)). The shared area is accessible to all users using MIMIC.

The **private file space** is outside of the MIMIC installation directory, commonly in a directory called mimic/ under your home directory (although you can optionally have it stored anywhere). The private area contains any changes to the data after you install MIMIC. For example, when you import a MIB, it will reside in your private area. Or when you compile a MIB, the compiled MIB will be in your private area.

When MIMIC looks for data, it will look in your private area first, then in the shared area. For example, if you recompile a MIB with the changes you need, your changes will override the MIB in the shared area. The [File Browser](#) in MIMICView will show your private data in **red**, while the shared data is shown in **blue**.

The benefit of the private area is that multiple MIMIC users can each have their own private data without interfering with the other users. Also, when you upgrade from one version of MIMIC to the next, all your data is automatically accessible in the new version because it is in the private area.

What are Maps?

The MIMICView interface lets you organize your lab configuration hierarchically into maps, rather than all in a flat view. Maps are modelled after folders or directories in operating system environments, or maps in network management applications. They organize agent instances according to your choice either by subnets, manufacturer, device type, etc.

By default, all agent instances are placed into the topmost "root" map. You can add maps underneath the root. Maps are depicted with a blue network diagram icon. The Explorer view shows the map hierarchy in usual tree fashion.

Maps are self-contained, isolated workspaces. What you do in one map does not carry over to another. For example, if you use the selection menus to select agent instances, that selection is cleared when you change maps. Also, if your selection specifies agents outside the map, eg. by number, only agent instances in the map are selected. Currently, the MIMICview GUI has no provision to work on multiple maps at once.

MIMIC Process Overview: When to Use the Tools

Overview

This section describes a process view of the MIMIC Simulator. Each process consists of discrete tasks that are described in detail in the following sections.

1. The first [process](#) compiles all relevant MIB files for the device that you want to simulate. MIMIC includes a large number of precompiled MIB files, as well as the MIB files themselves in the mibs/ subdirectory. You can add any missing MIBs with this process.
2. The second [process](#) creates a [simulation](#), which can subsequently be run by the MIMIC Simulator. Each simulation represents a device, or part of a device (if you are only interested in a subset of the device).
3. The third [process](#) assigns simulations to [agent instances](#) in the simulator. This lets you configure any number of scenarios of device simulations. By separating the process of creating and running simulations, you are able to run one set of simulations while you are creating or customizing another.
4. The final [process](#) allows you to customize a simulation. Customizations can be transient or persistent, and they can be done while the agent instance is running or while it is stopped.

Although the order presented here is the natural flow, you can perform any of these processes simultaneously or in any order. MIMIC is also designed for feedback loops among the processes. For example, if you notice that you don't have the necessary MIBs while you are creating the simulation, you can easily go back to compiling new MIBs.

Follow the links in each of the sections to get more details.

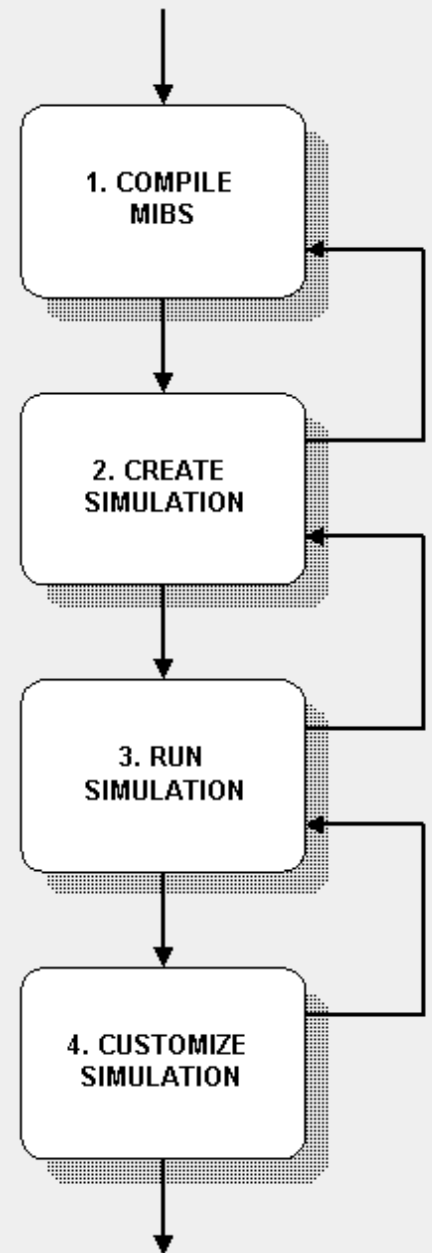


Figure - MIMIC Process Overview

Compile MIBs

MIMIC needs to know about the objects it is going to simulate, in order to return the correct instance, type and value, etc. This is specified in the MIB source file that defines each object. MIMIC includes a large number of [precompiled MIB files](#) in the `data/mibs/` subdirectory, as well as the MIB source files themselves in the `mibs/` subdirectory.

If there are many errors when you record a device, it may be exporting a MIB that MIMIC does not yet know about. The MIMIC Recorder will tell you about unknown MIB objects. If so, then you need to find out which MIBs the device exports, according to its manufacturer. You can then add any missing (enterprise-specific or standard) MIBs with this process. If you are not interested in certain MIBs, you can ignore them. If you don't have access to the MIBs, then as a last resort you can have the Recorder guess at the simulation of unknown objects with the `--unknown` option.

1. [import](#) the MIB source file into the MIMIC private data area. This is purely for organizational purposes, so that you can track down your MIB files. MIB source files are placed in the `mibs/` subdirectory.
2. [compile](#) the MIB source file. This translates the source file into internal data that can be efficiently used by MIMIC.

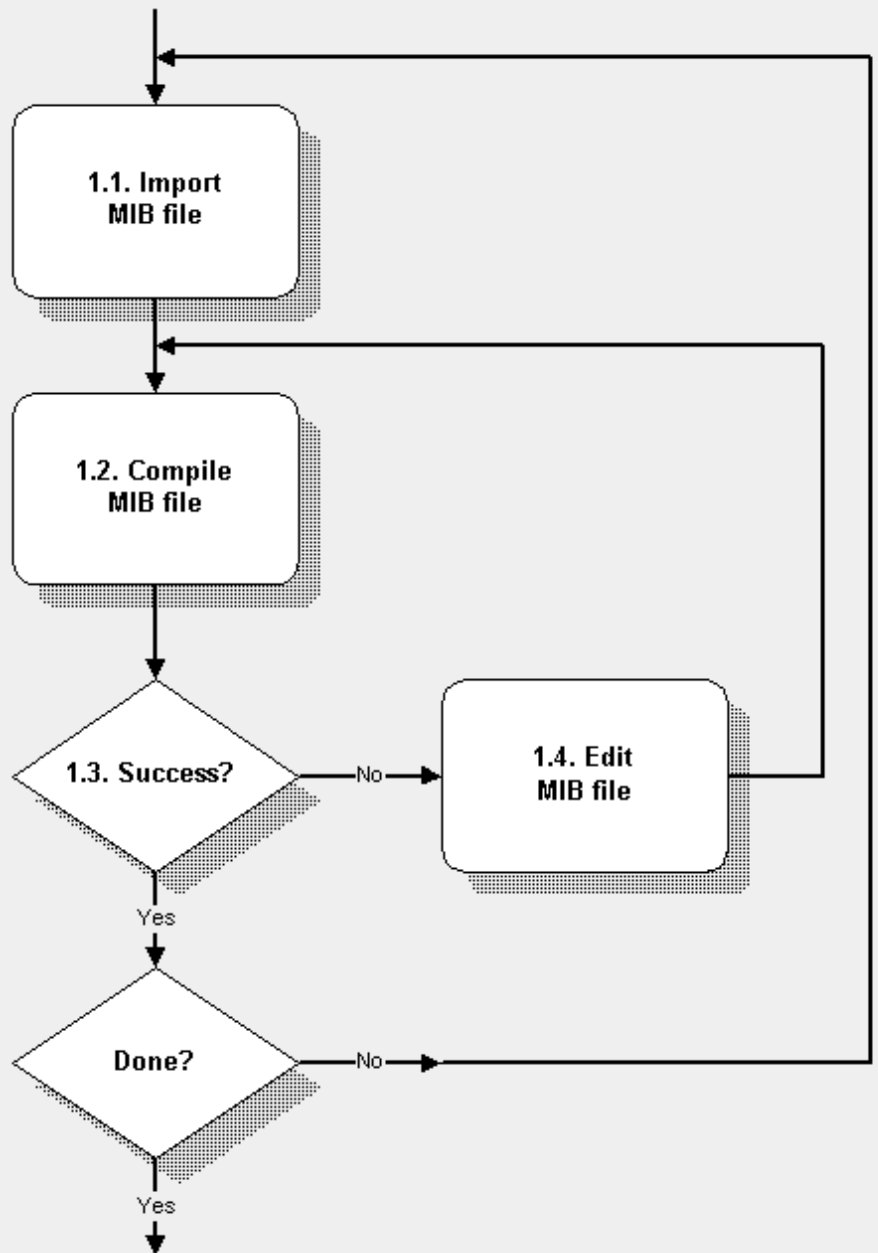


Figure - MIB Compilation Process

The internal MIB data are stored in the data/mibs/ subdirectory.

3. The compiler output will indicate whether the MIB was processed correctly. All MIMIC tools do extensive logging of diagnostic messages so that you can track exactly what they are doing.
4. **if there are errors, edit** the MIB source file to correct any them. Then recompile.

The [MIB Wizard](#) encapsulates these steps.

Create a Simulation

In this process you create a [simulation](#). In MIMIC, this means assigning simulation statements to each MIB object, and populating table entries and values for individual MIB object instances.

Before you create a simulation of a device, make sure that MIMIC knows about the set of [MIBs](#) that the device exports, as documented by the device manufacturer. If you are not interested in certain MIBs, you can ignore them.

1. The simplest way to create a simulation is to record a real device. The [MIMIC Recorder](#) will take care of creating simulation expressions for each discovered MIB object and populating tables and MIB values. The [Discovery Wizard](#) enables you to record an entire network of devices. The [Snapshot Wizard](#) allows you to take multiple snapshots of the same device, and automatically simulates the changes from one snapshot to the next. The [Trap Wizard](#) captures traps generated by your device and creates a trap sequence which you can include in any simulation.
2. The next simplest solution is to copy an existing simulation, either one that is supplied with MIMIC (in the

device and network libraries), or one that was previously recorded.

3. If you don't have the exact device available, or if the conditions are not exactly the way you want, you can still record the device and modify the simulation to suit your needs.
4. You can use the [Simulation Wizard](#) to populate a MIB with values. For example, you can automatically create default instances of variables having default values with the click of a few mouse buttons.
5. The last resort is to edit internal simulation files to [create](#) or change the simulation by hand. All internal MIMIC data resides in ASCII files, so that you can use your favorite editor or related text processing tools.

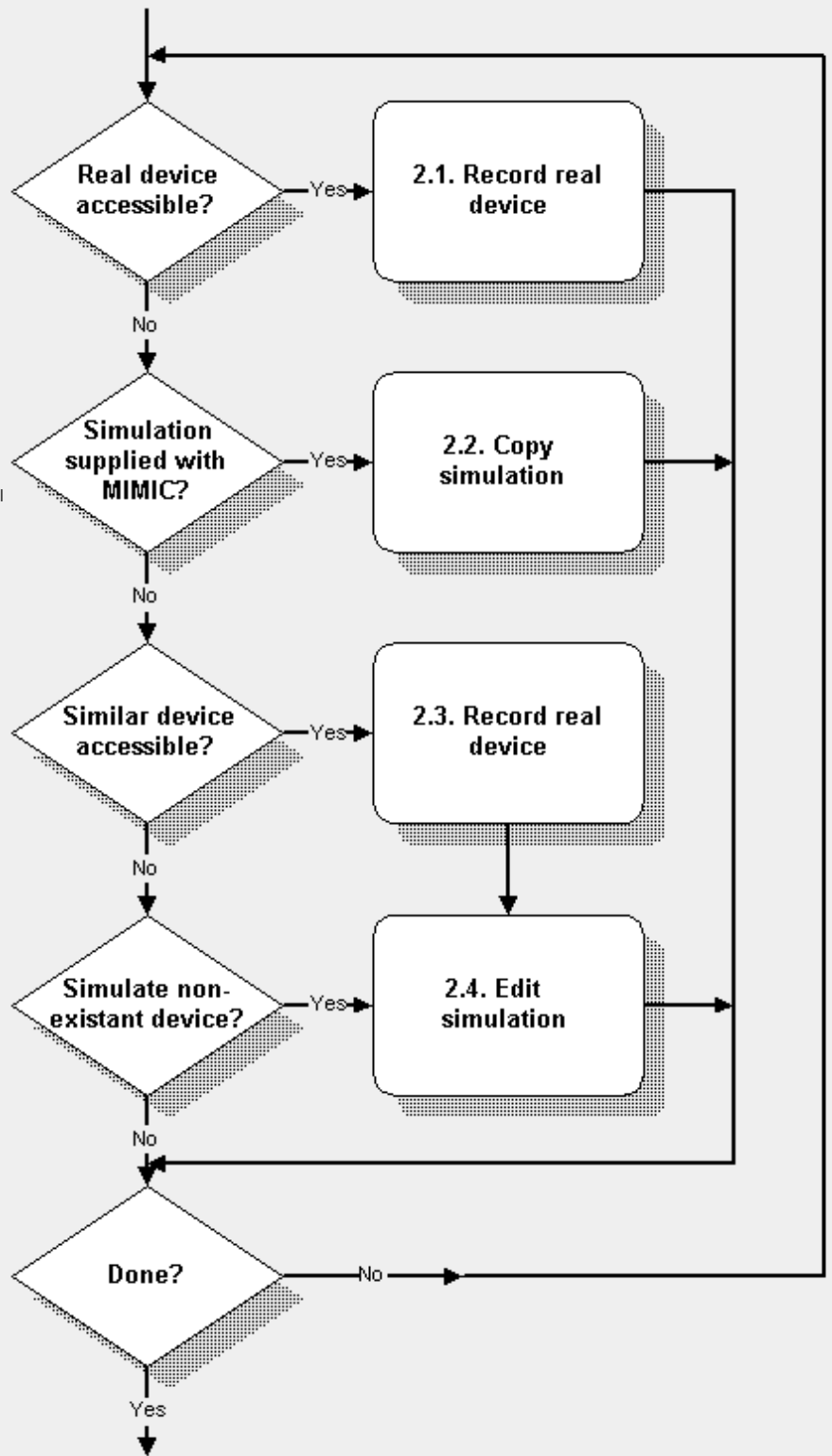


Figure - Creation Process

Run a Simulation

Once you have created the necessary simulations, you can run them in [agent instances](#). You can do any of these operations in any order on any number of agent instances, but for a specific instance the natural flow is:

1. First you need to [add](#) one or more agent instances with the desired device simulation.
2. Once an agent instance is configured, you need to [start](#) it. This is equivalent to booting your real device. Until it is running, an agent instance does not respond to queries. You can do many things with a running agent

instance, such as [pausing](#), [halting](#), and [changing the behavior](#) of the running simulation.

3. You can stop an agent instance at any point. This is equivalent to turning your real device off or shutting it down. Once an agent instance is stopped, you can [reconfigure](#) it or restart it.
4. Finally, you can [delete](#) your agent instance from the configuration.

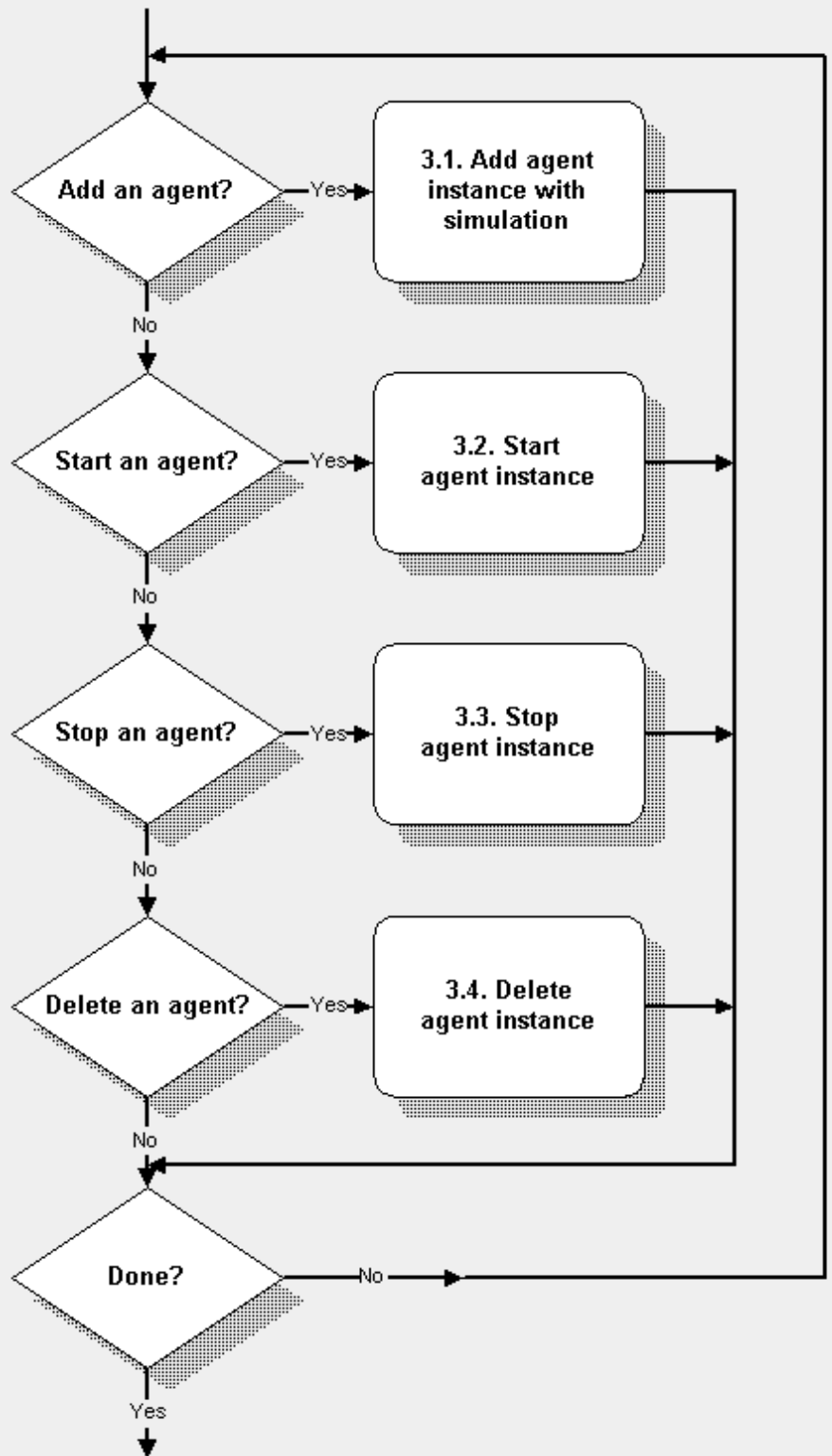


Figure - Simulation Process

Customize a Simulation

There are several ways to customize a simulation:

1. To change a simulation at runtime, you can use the graphical [MIMICView](#) or the scripting environments [Tcl-based MIMICShell](#), [Java API](#), or [Perl API](#), to change the [Value Space](#).
2. You can change the values and table entries for a scenario [manually](#). These values will take effect the next time you run the simulation.

3. Finally, you can [change](#) the simulation expression for a MIB object. This is only necessary for the most advanced uses. For example, you can create relationships between MIB objects, where the value of one MIB object affects another.

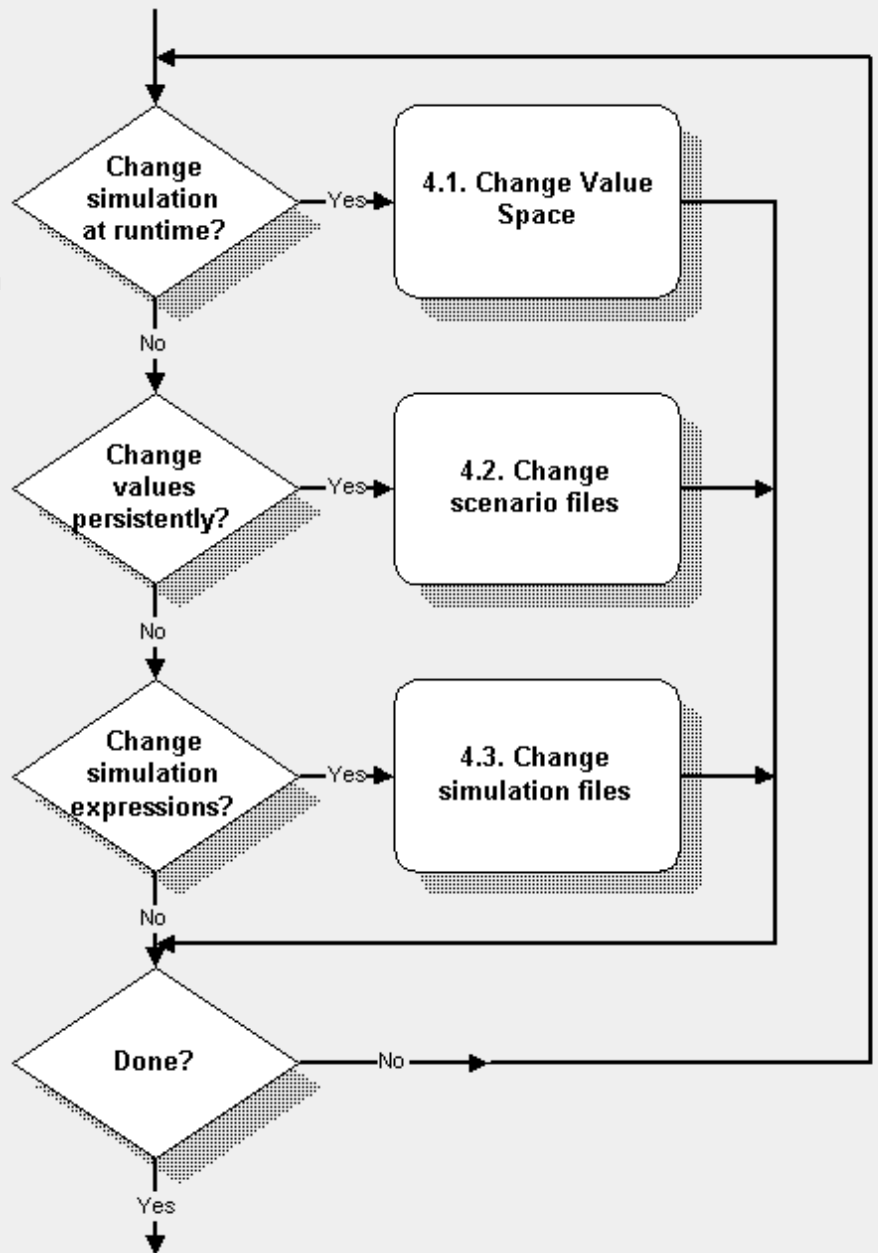


Figure - Customization Process

References for Further Reading

For more information on Network Management and SNMP, we recommend these books:

- Marshall Rose, *The Simple Book: An Introduction to Networking Management*, Prentice Hall, 1994
- David T. Perkins and Evan McGinnis, *Understanding SNMP MIBs*, Prentice Hall, 1996
- David T. Perkins, *RMON: Remote Monitoring of SNMP-Managed LANs*, Prentice Hall, 1999
- William Stallings, *SNMP, SNMPv2, and RMON: Practical Network Management*, Addison-Wesley, 1996
- William Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Addison-Wesley, 1999.

For more information on Tcl and Tk, which you will need to use to program simulation of device-specific behavior, we recommend these book titles

- John K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley, 1994
- Brent B. Welch, *Practical Programming in Tcl & Tk*, Prentice Hall, 1995

Chapter 5: Energy Savings

Chapter Contents

- [Overview](#)

Overview

MIMIC Simulator pays for itself very quickly even if one considers only energy savings.

One can calculate energy costs for a network, given a list of devices, the energy consumption for each, and energy prices. If you simulate this network, this translates into immediate energy savings. For a rough ballpark you can use [our energy savings calculator](#).

You can also download the optional [GREEN software update package](#) with the [MIMIC Update Wizard](#) to monitor energy savings while running MIMIC in real-time.

The energy savings matrix at [this page](#) shows savings given a sample network of 100 common devices.

[<< Previous Section](#) [Next Section >>](#)



MIMIC SNMP Agent Simulator Guide

Table of Contents

- Overview
- MIMICView Reference
 - Startup
 - Title Bar
 - Canvas
 - Maps
 - Status Bar
 - Log Window
 - Menu Bar
 - File Menu
 - Edit Menu
 - View Menu
 - Agent Menu
 - Run Menu
 - Simulation Menu
 - MIB Menu
 - Wizard Menu
 - Access Menu
 - Speed Bar
- MIMICShell Reference
 - MIMICShell Command Line Options
 - Interactive Command Line Editing
 - MIMIC Package
 - Global Commands
 - Session Commands
 - Agent Commands
 - Value Space Commands
 - Variable Store Commands
 - Timer Script Configuration Commands
 - Trap Configuration Commands
 - SNMPv3 Configuration Commands
 - Access Control Commands
 - Examples
- Access Control
 - Client Access Control
 - Agent Access Control
- Advanced Configuration
 - Configurable Number of Management Channels
- Updates
 - Update to 4.10
 - Update to 5.10
- Usage Profiling

Overview

The MIMIC SNMP Agent Simulator can simulate up to 100,000 SNMP-manageable devices simultaneously. Each device is simulated by an [agent instance](#).

Rules for the SNMP simulation are governed by RFCs [1157](#) , [1212](#) for SNMPv1, and RFCs [2578](#) , [2579](#) , [2580](#) , [3413](#) , [3415](#) , [3416](#) , [3417](#) , [3418](#) for SNMPv3.

Each agent instance is completely run-time customizable, both on an individual and collective basis. Since MIMIC responds to SNMP queries on any of its configured IP addresses, it looks to the network management application as if it is talking to actual devices.

Users can start the MIMIC SNMP Agent Simulator in two ways:

1. **Command line:**

In any command shell, start the MIMIC background daemon, e.g.,
`mimicd >& /tmp/mimicd.log &`

Simulations can be controlled in batch mode via the command-line Tcl-based [MIMICShell](#) or via programs written in [Java](#), [C++](#), [Perl](#), [Python](#), or [PHP](#).

2. **MIMICView Graphical User Interface (GUI) application:**

MIMICView is the graphical front-end to the main three MIMIC tools: Simulator, Recorder and Compiler. It also contains integration for all supported protocols. When you start MIMICView, the MIMIC Simulator will be invoked automatically if it is not already running. A [log window](#) will pop up with the output of the MIMIC Simulator daemon `mimicd`. The daemon will run in the background until explicitly stopped.

The MIMIC SNMP Agent Simulator functions are accessible via the **F**ile, **E**dit, **V**iew, **A**gent and **R**un menus in MIMICView. The most common tasks are accessible via the icons on the speedbar located below the menubar. Wizards attempt to simplify the most complex tasks.

MIMICView Reference

Startup **QuickStart**

Invoke MIMICView from a shell command prompt with
`mimicview`

or in Windows from the MIMIC Simulator program group in the taskbar, or by double-clicking on the MimicView.bat icon in Windows Explorer.

A splash screen will briefly display while MIMIC starts up. If the Simulator is not running, it is invoked and a [log window](#) for the Simulator will popup.

Title Bar

The MIMICView front panel will then appear. Its title indicates the host on which the Simulator is running, the user account name, [the private directory](#), the current [lab configuration](#) for the user and the current [agent map](#).

Canvas

The main canvas of MIMICView displays a representation of all agent instances currently running in MIMIC - the "running configuration". Each icon shows the type of agent (by image), the state of the agent (by color), its assigned IP address, an abbreviation of the device name it is simulating, and real-time statistics of SNMP PDUs handled per second.

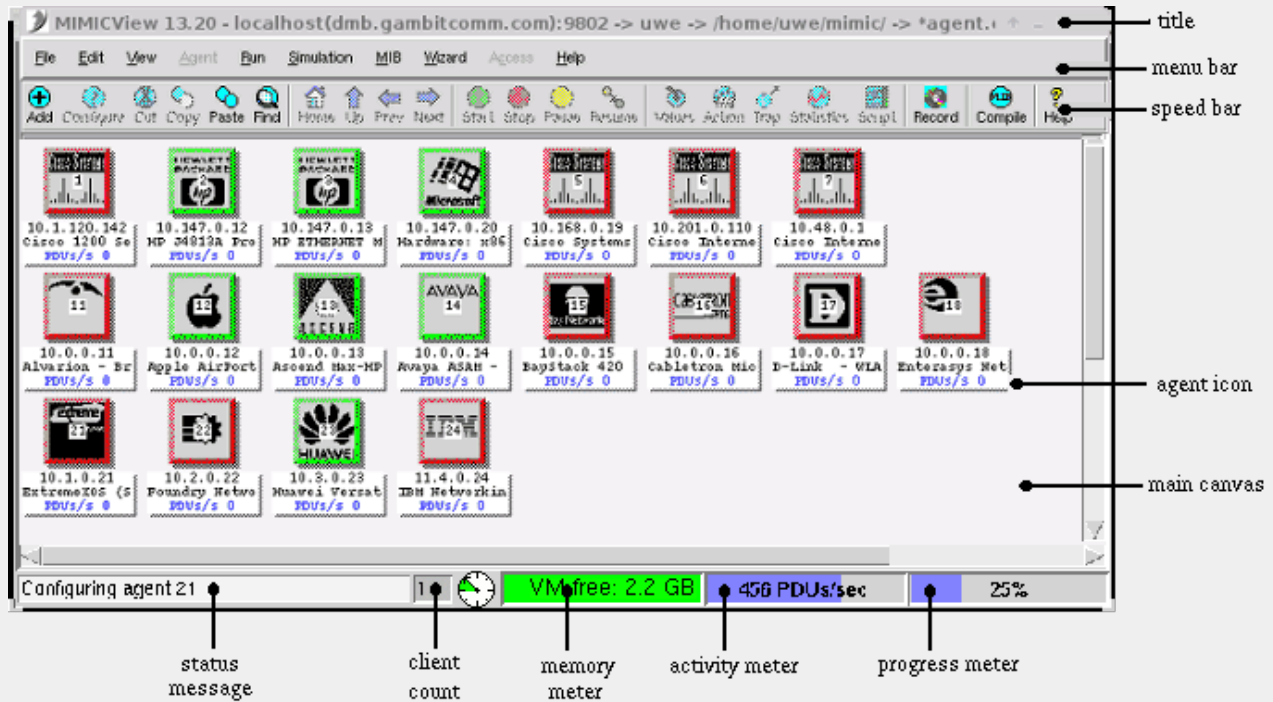


Figure 1: MIMICView Front Panel

In general, the GUI follows the "object/action" paradigm: to accomplish an action, you first select the object you want to act upon, then select the action.

For agent instances, this means you first select the group of agent instances in the main canvas, by either left-clicking on individual icons, or rubber-banding a group of contiguous icons. You can add to your selection of icons with **Shift** left-click. That way you can select non-contiguous agents as well as contiguous ranges of agents. Icons of selected agent instances are shown with heavy blue borders.

The actions are accomplished by the [menus](#) above the main canvas titled **File**, **Edit**, **View**, **Agent** and **Run**.

Operations on a single agent can be accomplished by right-clicking directly on the agent icon, which pops up the Agent menu.

Maps **QuickStart**

Agents shown in the canvas are contained in [maps](#). By default, agents are placed in the **Root** map, as shown below.

If you have only a few agents, that will probably suffice, but when you have to control tens of thousands or even 100,000 agents, you will probably want to organize them in maps.

Maps are like the hierarchical folder structure you are probably familiar with. You can add maps with the **Edit -> Add -> Map...** dialog, and move agents to those maps. The **View -> Up**, **Home**, **Next**, **Prev** menu items allow you to quickly traverse the history of maps you have viewed.

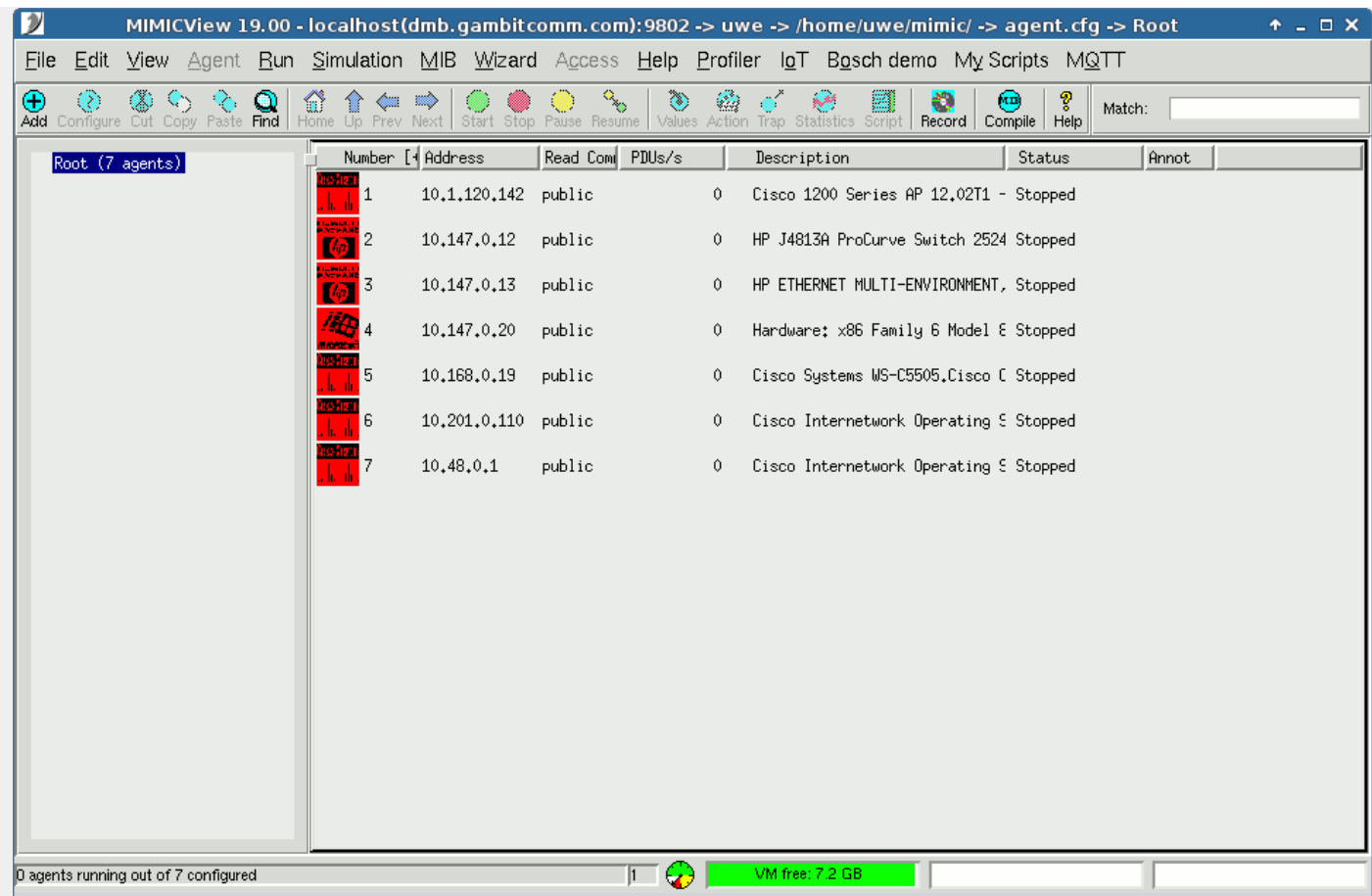


Figure 1b: MIMICView Root map

Status Bar

Underneath the main canvas is a dynamic status bar, with six components:

- A message area to the left, showing current status messages
- To its right, a text field showing the number of connected clients.
- To its right, a dial showing current activity between the MIMIC daemon and any connected clients. This reflects the amount of work the daemon is performing on behalf of clients.
- To its right, an meter tracking available virtual memory. It turns from green to yellow to red when virtual memory is exhausted. MIMIC will not run if there is not sufficient virtual memory. Consult the platform pages for setting up more swap space.
- To its right, an activity meter, showing current total SNMP activity (i.e., PDUs / second) at a glance in a logarithmic-scale meter from 0 to 10,000 PDUs per second
- A progress meter, showing current progress on tasks that take time to accomplish (such as adding lots of agents)

Log Window **QuickStart**

A log window pops up whenever you invoke one of the MIMIC tools from MIMICView. The log window captures the diagnostics output for that tool. All informational and error messages are displayed and output to a file given in the title of the window. The log window is displayed automatically when you start MIMICView with a fresh simulator daemon (see details), You can also display it anytime with **File->Log->View**. This log window or output file should be referred to whenever you have problems with MIMIC.

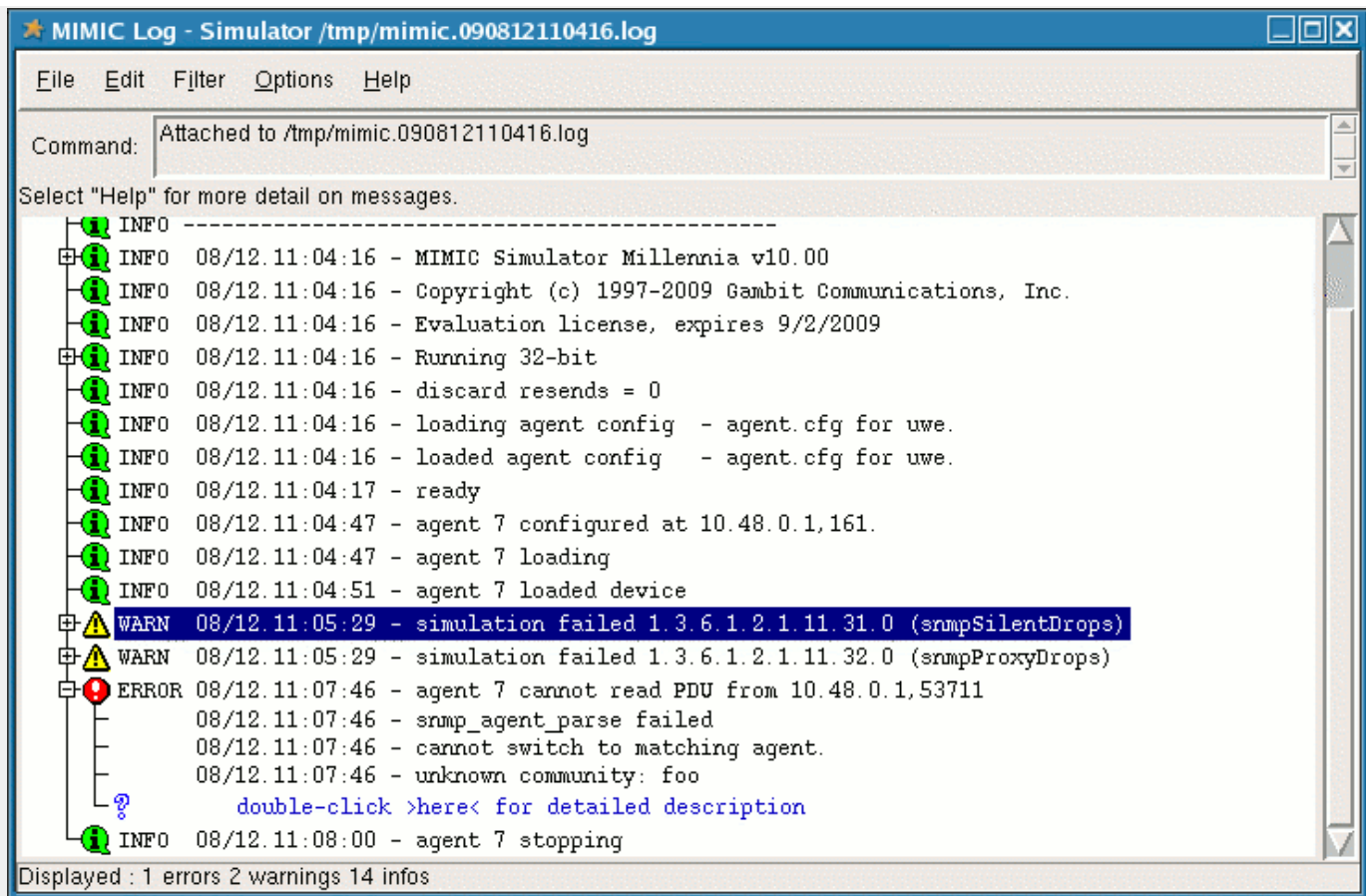





Figure 2: MIMICView Log Window

All log files are placed in /tmp/ and are ordered by date of invocation. The log file contains additional message details, which are filtered out in the log window for legibility.

In the log window, each diagnostic message is listed with the following severities:

1.  **INFO** - informational message (shows progress, no impact on correctness of results)
2.  **WARN** - warning message (some impact on correctness of results)
3.  **ERROR** - error message (more impact on correctness of results; possibly fatal)

NOTE: Severities are subjective under most circumstances. In general, you can ignore warnings, unless your management application relies on correctness of the impacted part of the simulation. Error messages merit closer inspection.

If the log windows resulted from invoking one of the MIMIC tools, you can use **File->Stop** to stop the invoked tool.

The log window will be displayed until you select **File->Close**.

A single selected message or a group of messages can be copied using **Edit->Copy** and then pasted into other applications. This makes it easy to e-mail related error message information to help Gambit support personnel diagnose your problem.

Edit->Find finds the first (or next) occurrence of the specified string.

You can filter out existing messages from being displayed with the **Filter** menu:

- **Message** filters out the individual selected message(s).
- **Type** filters out all messages of the same type as the selected message(s).
- **Severity** filters out all messages of the same severity as the selected message(s).
- **Off** turns off filtering for selected messages and (re)displays them.

This on-line help section is displayed with **Help->Contents...**

The **Options** menu lets you change the way the log window behaves. You can turn off the tracking of the end of the log (the "tail") by deselecting the **Tail** menu item.

Certain messages are multi-line because the underlying cause of the event is also logged. This leads to more accurate diagnosis of the problem. Multi-line messages are shown with a leading + sign. For legibility the log window shows only the first line of the diagnostic message. The causes of any error message can be expanded by clicking on the + or double-clicking on the message itself.

You can turn on the automatic expansion of nested multi-line messages with the **Expand** submenu:

- **All** - expands all messages
- **ERROR** - expands only ERROR messages (default)
- **None** - does not expand messages automatically, you can expand by clicking on the +

You can limit the number of lines displayed with the **Lines...** dialog. This is specially useful for large logs.

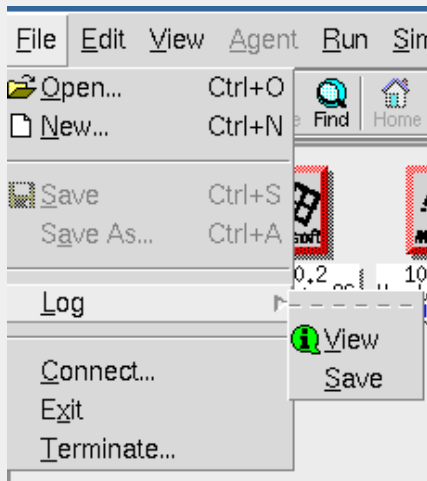
By default, the log viewer will raise itself to the top of the desktop if a  **WARN** or  **ERROR** message is logged. This behavior can be changed with the **Raise** submenu:

- **WARN** if unchecked will not raise the log viewer when a WARN message is logged
- **ERROR** if unchecked will not raise the log viewer when an ERROR message is logged
- **Deiconify** if unchecked will not deiconify the log viewer when a selected message severity is logged

The most common error messages are listed in [Appendix C](#). When you expand such messages in the Log Viewer, you'll be able to double-click on a hyperlink which will take you to the detailed description.

Menu Bar

File Menu



The File menu lets you load or save lab configurations and exit from MIMIC.

File->Open... **QuickStart**

MIMIC allows you to maintain multiple [lab configuration](#) files. All lab configuration files are in the `config/agents/` subdirectory. (The location `config/agents/` is historical.)

By default, on startup MIMIC loads the last loaded lab configuration for the user into the running configuration.

To load a different lab configuration, use **File->Open...** The **Load Configuration** dialog pops up, allowing you to specify the lab configuration file name. Alternatively, you can browse by clicking the **Browse...** button, opening the **File Browser** dialog.

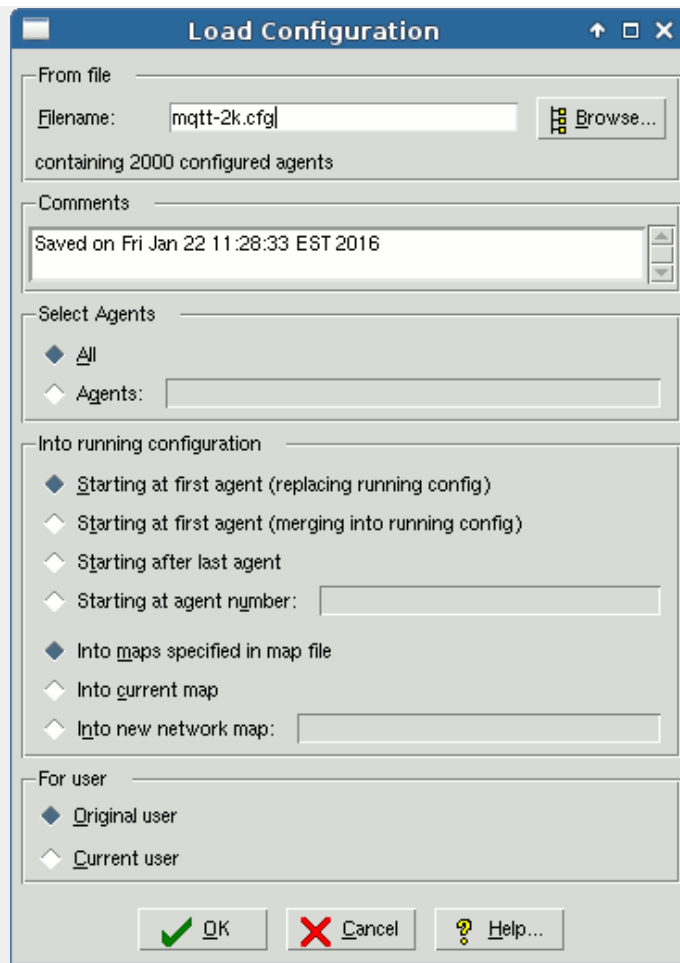


Figure 3a: Load Configuration dialog

The **File Browser** dialog lets you select a file by browsing through the **file space**. MIMIC supports two types of files: shared and private. Private files are any files you have changed, e.g., when you save a config file or when you import a MIB. Private files show up in **red** in the browser. Shared files are located in the installation area and show up in **blue**.

If you have many files to choose from, you can sort them by **Newest**: modification time instead of **Name** to show the newest files first.

In addition, you can filter the filename with the **match** field, which matches each filename against the regular expression you enter. For example, **^demo** matches all files that start with **demo**.

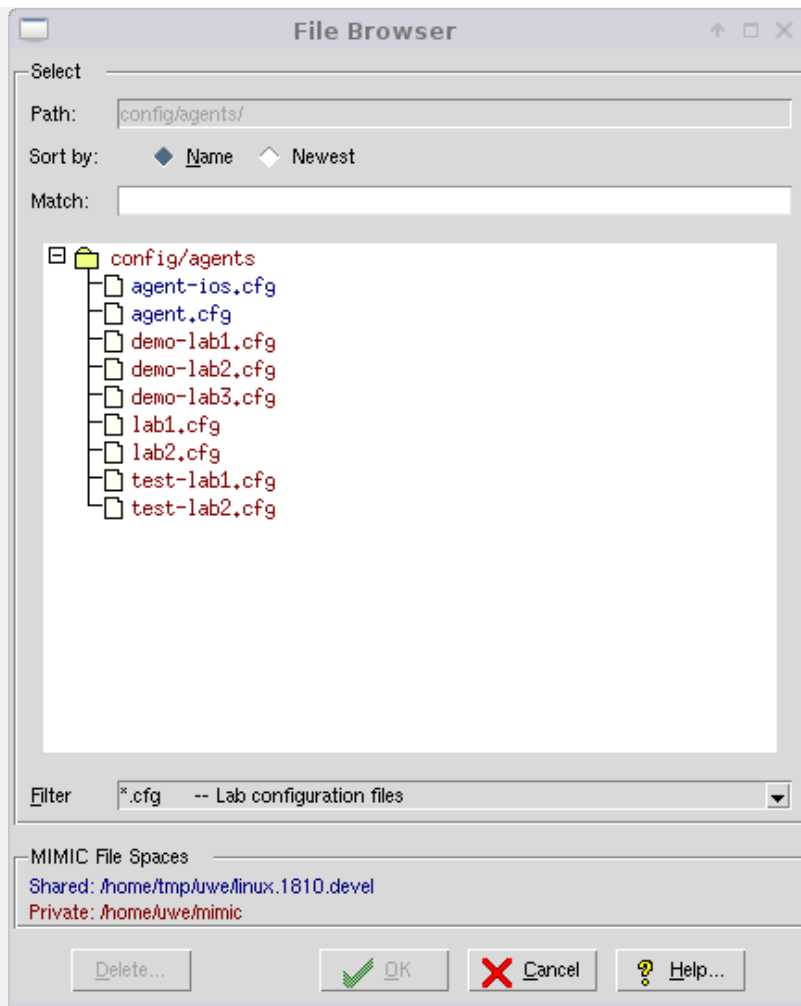


Figure 3b: File Browser

Once you have selected the `Filename` with the File Browser, the `Select Agents` field lets you load a specific set of agents from the file to load. By default, the `All` button is selected and it loads all the agents. Else, if you select the `Agents` button, the field accepts a comma-separated list of agent ranges, eg. to load agents 2 to 5 and 10 to 20 you would specify `2-5,10-20`.

The `Comments` field annotates the lab configuration, eg. tells you how and when it was created.

The `Into running configuration` fields specify different ways to add the agents in the configuration file into the running configuration. You can load agents at the beginning of the running configuration with `Starting at first agent` (replacing loaded configuration) by default. Thus, it will overwrite all agents that are configured with the agents in the file to be loaded.

The second option `Starting at first agent` (merging into loaded config) can be used to load agents into the current configuration, but merge in only the specified agents in the file. Empty agents in the file will not clear agents in the current running configuration. For example if you are running agents 1 through 10, and the file to be loaded specifies agents 11 through 20, then only those agents are merged in.

Or, you can add the new agents after the last configured agent with `Starting after last agent`. The `Starting at agent number` field allows you to load the new configuration at a specified agent number in the running configuration. To add at a certain spot in the running configuration, you would specify a different starting agent position. Agents are loaded starting at that position. For example, if you specify starting position 11, the first agent in the configuration file is loaded at position 11, then next agent at 12, etc.

By default, the `Into maps` specified in `map file` button is selected and the agents will be loaded and any map(s) previously saved with the lab configuration file will be loaded. If there is no map file, then all maps will be removed and the new agents will be loaded into the root map. You can load agents instead into the currently selected map with `Into current map`, or into a new submap in the current map with `Into new network map`.

The options under `For user control` who owns the agents that are loaded into the running configuration. The default, `Original user`, retains the ownership that was saved for the agent configuration. If you select `Current user`, then the agents that are loaded will be owned by you and their simulation data will be retrieved from your [private data area](#).

File->New...

This menu item creates a new lab configuration, closing the current running configuration. The `Agent` field lets you clear only a selected set of agents, eg. `2-5,10-20`.

Running agents will not be cleared. You will be warned in that case, and can stop them with [Agent->Stop](#) or [Run->Stop](#).

If you clear all agents, any maps defined in the running configuration are erased, too.

File->Save

Use this menu item to save the current [lab configuration](#).

The `save old file` option allows to do file versioning, saving the old version in a folder `OLD/timestamp` where `timestamp` is the current time. That way you can always go back to old lab configurations.

Old versioned lab configuration files can be loaded with `File->Open` the only limitation being that when you save it again, it will not overwrite the old file, instead being saved in the lab configuration folder `config/agents/` just as all other lab configuration files.

File->Save As...

This menu item allows you to save all or part of the current agent configuration as a different filename.

The `Save Configuration` dialog pops up, allowing you to specify the lab configuration file name. Alternatively, you can browse by clicking `Browse...`

Under `select Agents`, if you pick `All` (the default), then all the agents are saved. This will guarantee that you can load this exact configuration with `File->Open`.

Alternatively, the `Agent` field lets you save only a selected set of agents, eg. `2-5,10-20`.

All configured `maps` are saved in a map file (with the same root as the configuration file, but with `.map` suffix). For example, if the lab configuration file is `test.cfg`, then the map file is `test.map`.

If you select `Current map`, then only the agents in the current map are saved (as well as in the map file).

You can annotate the configuration in the `Comments` text input.

File->Log

The `File->Log->View` menu item lets you view the current `simulator log`.

In order to manage large log files, you can use `File->Log->Save` to switch log files. This is also done automatically around every midnight.

File->Connect...

By default, MIMICView connects to the MIMIC simulator running on the local system. With this menu item you can connect to and control MIMIC simulators running on other systems.

File->Exit

Exiting MIMICView does not stop MIMIC from running. The `mimicd` daemon will continue running until explicitly stopped, allowing you to continue running a simulation when no GUI interface is needed or desired.

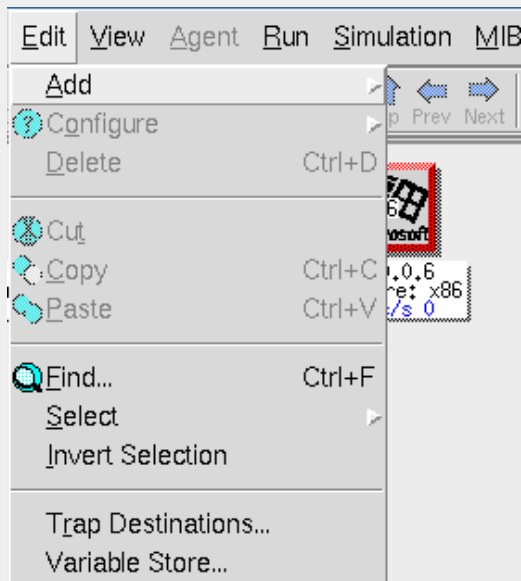
File->Terminate

This menu item terminates MIMIC. All running agent instances are stopped, and the `mimicd` daemon stops.

If there are any active agents, you should stop them before terminating in order to save any changes to their MIB data.

If there are any other clients connected to the simulator daemon, you should stop them before terminating in order to exit gracefully.

Edit Menu



The `Edit` menu lets you add, configure, delete and select agent instances and maps.

Edit->Add->Agent... **QuickStart**

To add new agent instances, use `Edit->Add->Agent...`

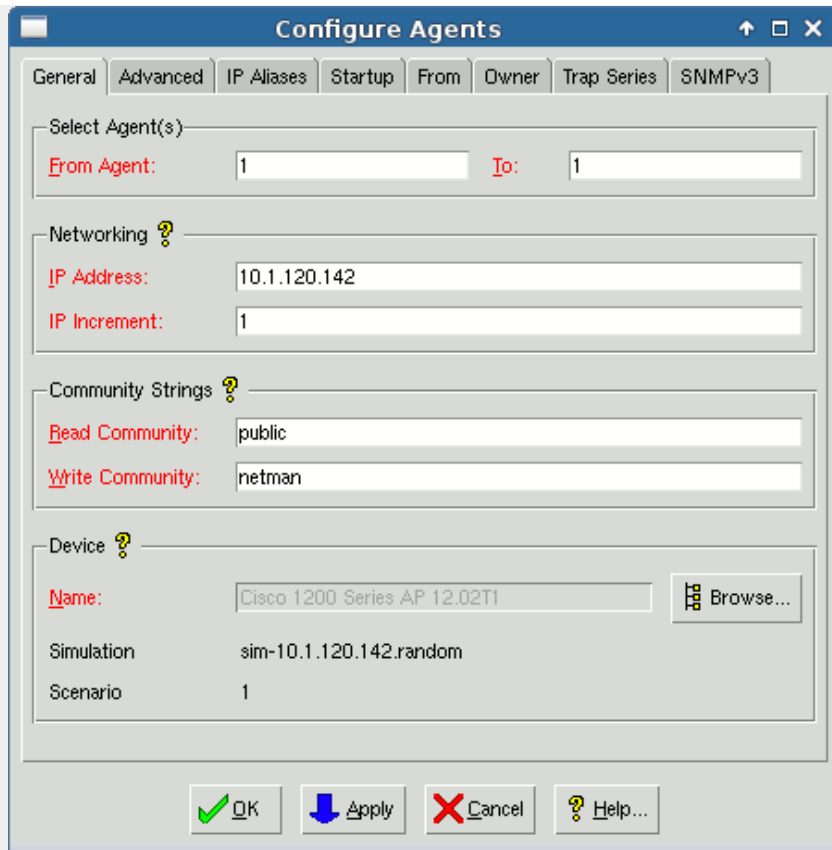


Figure 4: Dialogs for Configuring and Adding Agents

The Add Agents dialog pops up, with the following General configurable parameters (mandatory fields are in red):

- **From Agent** -- To -- The range of agent instances to add. By default it is set to the next available agent instance. If you select an incorrect ID, you will get an error dialog such as

```
Bad value for agent ID: 2 (expected: 1 <= ID <= 1)
```

This usually means your license does not allow adding more agents. You will have to re-configure an existing agent.

- **IP Address** -- This specifies the agent IPv4 address in "dot-value" notation (e.g., 192.9.200.1). Both IPv4 and IPv6 addresses are allowed as the main address of the agent, as well as [IP aliases](#) for the agent.
- **IP Increment** -- If configuring more than one agent, subsequent agent instances will be assigned an IP address with an offset specified by this number. For example, if the starting address is 192.9.200.1, and the increment is 2, the second instance will be assigned address 192.9.200.3, the third will be 191.9.200.5, and so on. When reaching the end for a IP address field, the least-significant field wraps, and the next higher field is incremented. For example, if the starting address is 192.9.200.254, and the increment is 1, then the next address is 192.9.201.1 (the broadcast address is not assigned).

NOTE: It may be dangerous to assign addresses that are already used by existing devices in your network. This can potentially cause network outage. Make doubly sure that the IP addresses you give to MIMIC agent instances are unique within the network on which MIMIC is running. If you are unsure, please talk to your network administrator first.

NOTE: resolving increments with clashing network and broadcast addresses is non-obvious. The rule is that only addresses at the specified increment are considered, if they are not a network or broadcast address for the given `Network Mask` parameter. For example, if the starting address is 192.9.200.1, the increment 255, and 3 agents are configured, then only addresses 192.9.200.1, 192.9.202.254 and 192.9.203.253 are used, since addresses 192.9.201.0 and 192.9.201.255 are not valid host addresses.

- **Read Community** -- This is the SNMP v1/2c access control parameter for read access to the MIB on the agent. SNMP PDUs of type GET and GETNEXT will only be accepted with this community string. A group of acceptable community strings is separated by "," (comma). Eg. if the agent is configured with `public1,public2`, it will accept requests with either `public1` or `public2`. A special case is the community string `*` (i.e., containing a single star character) which will accept any request. By default, agents use the string `public`.
- **Write Community** -- This is the SNMP v1/2c access control parameter for write access to the MIB on the agent. SNMP PDUs of type SET will only be accepted with this community string. By default, agents use the string `netman`.
- **Device Browse...** -- This button pops up a dialog to select the [Device Type](#) of the agent instance, i.e., the MIBs it exports. This dialog is explained in the following [section](#). The `Name`, `Simulation` and `Scenario` are filled in from the device definition you select.

The Advanced page lets you select the advanced parameters of the agent instance:

- **Interface** -- On machines with multiple network interfaces, you can select the network interface adapter for the agent instance. An empty field will match the first active NIC on your system (default). The interface name depends on the OS platform:
 - On Solaris, it is the device name as displayed with the `ifconfig` command. For example, `hme0` and `hme1` can be used on the machine below:

```
# ifconfig -a
lo0: flags=849 mtu 8232
```

```

    inet 127.0.0.1 netmask ff000000
hme0: flags=863 mtu 1500
    inet 192.9.200.35 netmask ffffffff broadcast 192.9.200.255
hme1: flags=863 mtu 1500
    inet 192.9.201.35 netmask ffffffff broadcast 192.9.201.255
ipdptp0: flags=28d1 mtu 8232
    inet 127.0.0.1->127.0.0.1 netmask ffffffff

```

- On Windows NT, the interfaces can be listed with the `ifdiag` utility supplied with MIMIC. For example, the adapters **RTL80291** and **RTL80292** can be used on the machine below:

```

C:\MIMIC\BIN> ifdiag
$print

```

```

System has 2 adapters
-----

```

```

Adapter [RTL80291] details

```

```

Addresses (1001) :

```

```

[0] - 192.9.200.50,255.255.255.0
Rest (1000) Mimic Addresses.

```

```

Gateways (1) :

```

```

- 192.9.200.11

```

```

Adapter [RTL80292] details

```

```

Addresses (261) :

```

```

[0] - 192.9.201.50,255.255.255.0
Rest (260) Mimic Addresses.

```

```

Gateways (1) :

```

```

- 192.9.200.11

```

```

$quit

```

```

Goodbye!

```

NOTE: on Windows, MIMIC will allow a partial, case-insensitive match. Eg. if your NIC has the name **NDIS 4.0 driver**, then you can use **ndis** in the `Interface` field to match it.

- **Port Number** -- This field specifies the port at which the agent instance listens for SNMP requests.
- **Network Mask** -- The network mask allows you to do subnet masking.
- **Max. PDU Size** -- This field specifies the maximum PDU size accepted and sent. PDUs which exceed this size are discarded. The limit for this configurable is 65536, the default is 4500.
- **Protocol** -- You can enable a set of protocols for an agent instance. For multi-lingual agents, check off multiple buttons. The currently supported SNMP protocols are:
 - SNMPv1
 - SNMPv2c
 - SNMPv2
 - SNMPv3

The default is `v1` and `v2c`.

When you select a protocol, a new tab may be added to the configuration dialog which lets you configure additional protocol-specific parameters. The [SNMPv2 tab](#) and [SNMPv3 tab](#) are detailed below.

- **Delay** -- This parameter specifies the one-way transit delay in msec, to simulate distant nodes. The round-trip delay is twice this value. The minimum granularity is 10 msec.
- **Drop Rate** -- You can configure an artificial drop rate (every N-th PDU) to simulate faulty network paths to nodes. The default is no drops.
- **Relative Start Time** -- You can have agents starting up as if they had been running for a while. This parameter specifies the time that the agent has been running at startup.

The **IP Aliases** page lets you add additional IP addresses for the agent instance. The agent will respond to requests on the IP aliases. Each IP alias is an IPv4 or IPv6 address, optionally followed by a port number (default 161), optionally followed by a network mask, optionally followed by the network interface. Each of these parameters is described above, and is separated by a comma. Eg. 192.9.201.6,161

The **Startup** page lets you customize the agent instance at startup. This is useful if multiple agent instances are running the same simulation. Rather than returning the same values, the startup customization changes each agent to return unique information.

You can select any of the pre-configured startup scripts in the list. Select an entry if you want to run the script at agent startup, deselect if not. If you double-click on any entry, an editor window will open with the source code of the script, allowing you to further customize the startup script. If you do this, be aware that other agent instances may share this startup script. If you require further customization then the scripts provided here, it is advised to use [Agent->Action->Start...](#) instead.

Some scripts require one or more optional protocol modules for the agent. If the agent is not running the protocol, then the entry will be disabled and shown in grey.

The **From** page configures source-address-indexing for an agent instance. Source-address-indexing in effect creates disjoint virtual networks in the simulation from the perspective of different management stations.

It lets you add one or more `from` entries for the agent. These entries identify a source-address from which the agent will accept requests for processing. Each entry will be composed of an IP address/hostname and/or port number of the source entity (eg. a management application). The IP address can be empty or '0.0.0.0' both of which implies a wildcard ('any') address. Similarly an empty port or '0' both imply a wildcard ('any') port. Both address and port cannot be wildcards. If there are 'from' entries on two agents with identical IP address, port and community/context-engine-id, then the agents will be deemed unique as long as there is at least one 'from' entry on each of these agents and none of the 'from' entries are the same. See also this [Frequently Asked Question](#).

The **Owner** page lets you review the ownership parameters of the agent instance:

- **Owner** -- Displays the user who configured the agent instance. When multiple users are using the same MIMIC daemon, their agents will have different ownership information.

- `Private Directory` -- Shows the path to the [private area](#) directory for the data of the agent instance.

The `Trap Series` page configures the trap series configured with the [Trap Wizard](#). You can have the traps generated automatically when the agent starts, or control the trap generation with the [Agent -> Trap Series...](#) dialog.

The `SNMPv2` page configures SNMPv2-specific parameters for the agent instance. This tab is only visible if you have enabled SNMPv2 for the agent in the [Advanced](#) tab.

- `Party Database` -- specifies a file containing the party definitions, as detailed in RFC [1445](#) and [1446](#).
- `Context Database` -- specifies a file containing the context definitions, as detailed in RFC [1445](#) and [1446](#).
- `Access Control Database` -- specifies a file containing the access control definitions, as detailed in RFC [1445](#) and [1446](#).
- `View Database` -- specifies a file containing the view definitions, as detailed in RFC [1445](#) and [1446](#).

These files are self-documented and editable.

The `SNMPv3` page configures SNMPv3-specific parameters for the agent instance. This tab is only visible if you have enabled SNMPv3 for the agent in the [Advanced](#) tab.

- `Engine ID` -- specifies the `snmpEngineID` for the agent. It is in ASCII format by default, or hexadecimal format if it starts with `\x`. If left empty, it defaults to a proprietary engine ID which by default is the same for all agents.
- `Context Engine ID` -- specifies the `contextEngineID` for the agent. It is in ASCII format by default, or hexadecimal format if it starts with `\x`. If left empty, it defaults to the same value as the Engine ID.
- `USM Database` -- specifies a file containing the User-Based Security Model definitions, as detailed in RFC [2574](#), then [3414](#), then [5590](#).
- `VACM Control Database` -- specifies a file containing the View-based Access Control Model definitions, as detailed in RFC [2575](#), then [3415](#).

These files are self-documented and editable.

Device Selection

The `Devices` dialog lets you assign a device simulation to an agent instance. The dialog lists all the devices that your copy of MIMIC knows about. These include the devices that were preconfigured with the distribution of MIMIC you received, and any devices you have recorded with the [MIMIC Recorder](#) or created manually.

Device Catalog

The list of known device simulations is configurable with [Simulation->Devices...](#). In the resulting `Edit Devices` dialog, you can

- categorize your devices into any of the categories in the left-hand list;
- add/delete categories and subcategories;
- assign labels to device simulations;
- control the list of simulations, MIBs and scenarios that get loaded for a specific device simulation; and
- add descriptive notes to the simulation.

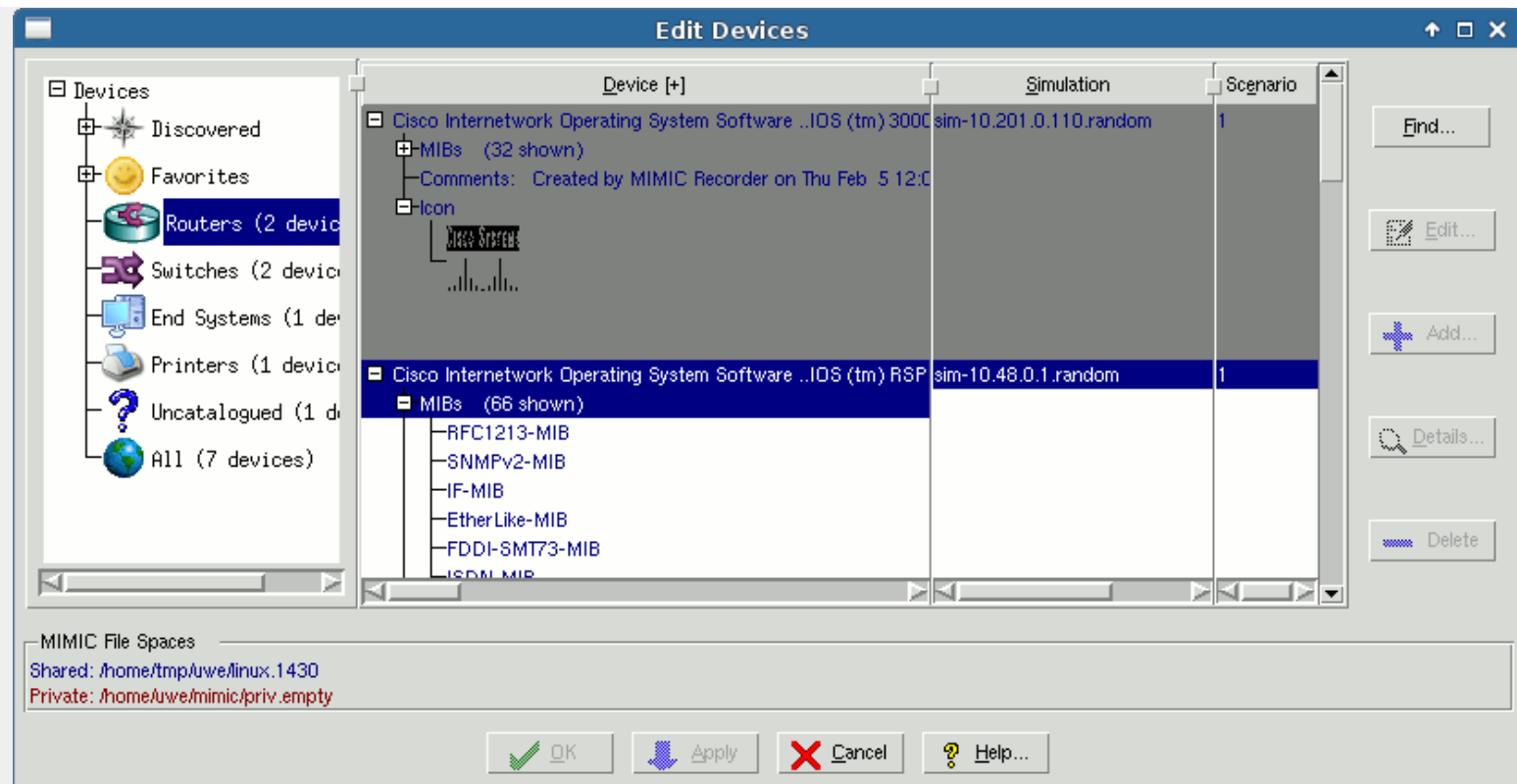


Figure 4-2: Manage Devices

The hierarchical list in the left-hand pane shows the catalog of device categories. If you have many devices, this allows a way of organizing them, so that you can easily find them. We have categorized the sample devices that ship with MIMIC by device type, ie. *Routers*, *Switches*, etc. But, you can create any classification you want. For example, you could add a *Favorites* category, and classify in there the devices you work with most. Or, if you want to classify devices by manufacturer, you can create a *By Vendor* category, and under it create a category for each of the manufacturers. Or, you could organize by project that you are working on, and create a *By Project* tree.

All unclassified devices are listed in a category called *Uncatalogued*. Initially, your newly created devices will be in here, until you classify them.

The *All* category contains a list of all the devices, whether they are classified or not. This allows you to find a device in the global list, if you don't remember where you put it.

The devices that ship with MIMIC are shown in **blue**, while the devices that you create are shown in **red**. This is due to the difference between the **shared and private data areas**. You can only modify a device in your private data area, ie, if it is shown in **red**.

To add/remove categories, use the right mouse-button click in the categories pane, and use the **Add...** or **Delete** menu items of the popup menu. This allows you to customize the categories to your taste. To move a device to a category, select it and right-click **Cut** it. Then right-click **Paste** it in the desired category.

The list of device simulations is shown on the right with these fields:

- The **Device** column has first the name that determines the user-friendly label that is shown in the MIMICView main canvas. This is only a convenience for the user, and is not used internally by the simulator. Underneath is a **MIBs** node that can be opened to display the MIBs simulated in this device as detailed below.
- The **Simulation** and **Scenario** fields are used by the Simulator to find the simulation data.
- The **MIBs** selection list determines the set of MIBs in the simulation. You can remove unwanted MIBs from the simulation with the **Delete** button. If you add a MIB with the **Add...** button, you need to make sure to provide simulation data for this MIB.
- The **Comments** field lets you add comments to the simulation. The MIMIC Recorder places a default comment.

To change a simulation, select it from the selection list. Click the **Edit...** button to modify the simulation attributes for the selected device. Click **Delete** to delete the simulation. All changes are applied when you click the **OK** or **Apply** button.

Edit->Add->Map...

To add new submaps in the current map, use **Edit->Add->Map...**

The **Identifier** field displays the next available map number.

You can select a name for the map in the **Description** field.

Edit->Configure

This submenu allows to configure entities (agents and maps).

Edit->Configure->All...

To re-configure existing agent instances or maps, use **Edit->Configure->All...** on the selected agent instance or map icons.

This displays essentially the same dialog as **Edit->Add->Agent...** or **Edit->Add->Map...**

NOTE: Currently only the first agent will be modified. There are too many non-obvious side-effects of allowing this dialog on multiple agents.

Edit->Configure->General

Edit->Configure->Advanced

Edit->Configure->Startup...

Edit->Configure->PROTOCOL

The dialogs in these menus configure just one of the configurables for agents from [Edit->Configure->All...](#)

The improvement of these dialogs over [Edit->Configure->All...](#) is that they can be invoked on multiple agents.

NOTE: These dialogs can be invoked for any group of selected agents, including non-contiguous agent ranges.

The `Agents` text entry field by default contains the selected agent(s), but can be changed to any agent range(s). This is specially useful for large groups of agents, eg. 1-100000.

Edit->Delete

This menu item lets you delete the selected agent instances or maps.

NOTE: Only stopped instances will be deleted.

Edit->Cut

You can easily move an existing agent instance or map with this menu item. Use [Edit->Paste](#) to paste the cut item into a different map.

Edit->Copy

You can easily duplicate an existing agent instance with this menu item. Use [Edit->Paste](#) to create the duplicate. Notice that only the persistent (on disk) simulation data is duplicated. In MIMIC, any changes to a running agent are individual for that agent.

Edit->Paste

Pastes an agent instance or map based on a previous [Edit->Cut](#) or [Edit->Copy](#).

If the paste is from an [Edit->Cut](#), then the items are pasted immediately.

If the paste is from an [Edit->Copy](#), then all attributes except IP address are copied. A `Paste Agent` dialog appears, which works like the [Add Agent](#) dialog.

Edit->Find...

This menu item finds agent instances by searching for the specified expression in their configured data.

Only agents in the current map are searched, starting with the first agent in the map. Successively pressing `Next` or `Prev` adds the found agent to the selection, so that afterwards something can be done with all the selected agents. When no more agents are found, the `Next` or `Prev` button is disabled.

If `In Variable Store` is selected, then it searches the variables for each agent and their value, too.

Edit->Select

This submenu lets you select different subsets of the configured agent instances in the current map, eg. by range, running status. This is useful if the subset is large, preventing [interactive selection on the main canvas](#). Notice, that the subsets (except for `None`) are additive, ie. the subset will be added to the already selected agents. If you want to replace the current selection, you have to first clear it with `None`, then select your desired subset(s). To make this clear, the dialogs will prompt whether you want to clear the selection first.

Edit->Invert Selection

This menu item inverts the selection of agent instances. This easily lets you manipulate all the agents that you have not yet manipulated.

Edit->Trap Destinations...

The `Edit->Trap Destinations...` dialog lets you configure the global trap destinations. Any traps generated by an agent instance, which does not have its own per-agent trap destination, will be sent to all the global trap destinations.

Each entry is a comma-separated IP address and port number. The default trap port number is 162. A trap destination should specify the host running a management application prepared to receive traps.

Edit->Variable Store...

The MIMIC `Variable Store` dialog allows you to edit variables in the [Variable Store](#).

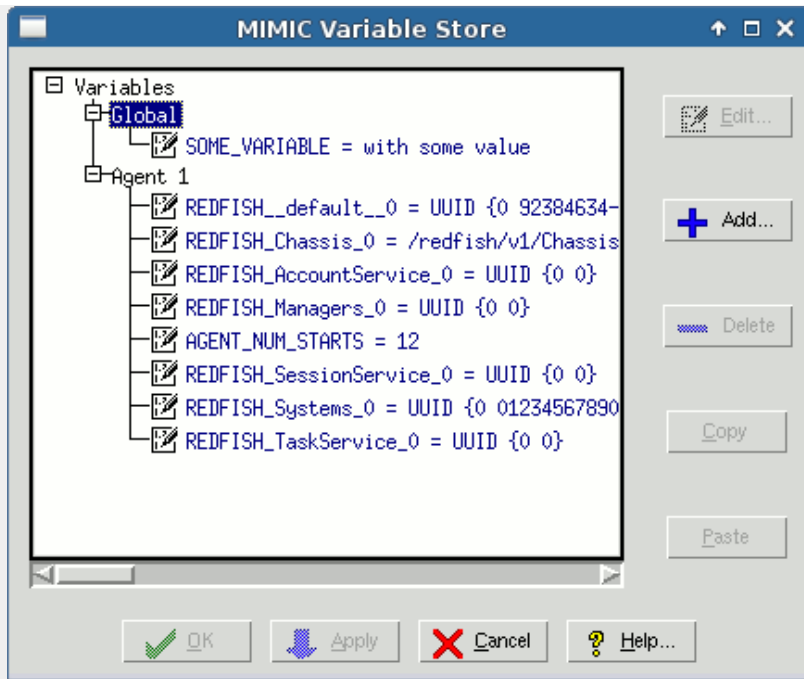
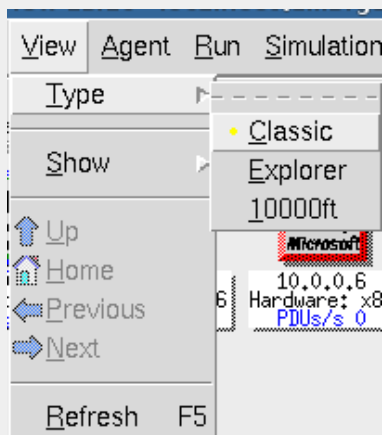


Figure 4-3: Variable Store

View Menu



The view menu lets you change views into the simulation.

View->Type

To change the view mode use View->Type. You can change between the Classic, Explorer and 10000-foot view types. The Classic view (Figure 1) presents an icon per agent, just like many large-scale network management applications. Alternatively there is an Explorer view (Figure 5), which presents the information in a format similar to the Windows Explorer. By default, MIMICView will use the Explorer view.

NOTE: For large configurations, eg. between 100 and 1000 agents and more, the Classic view performs substantially better than the Explorer view.

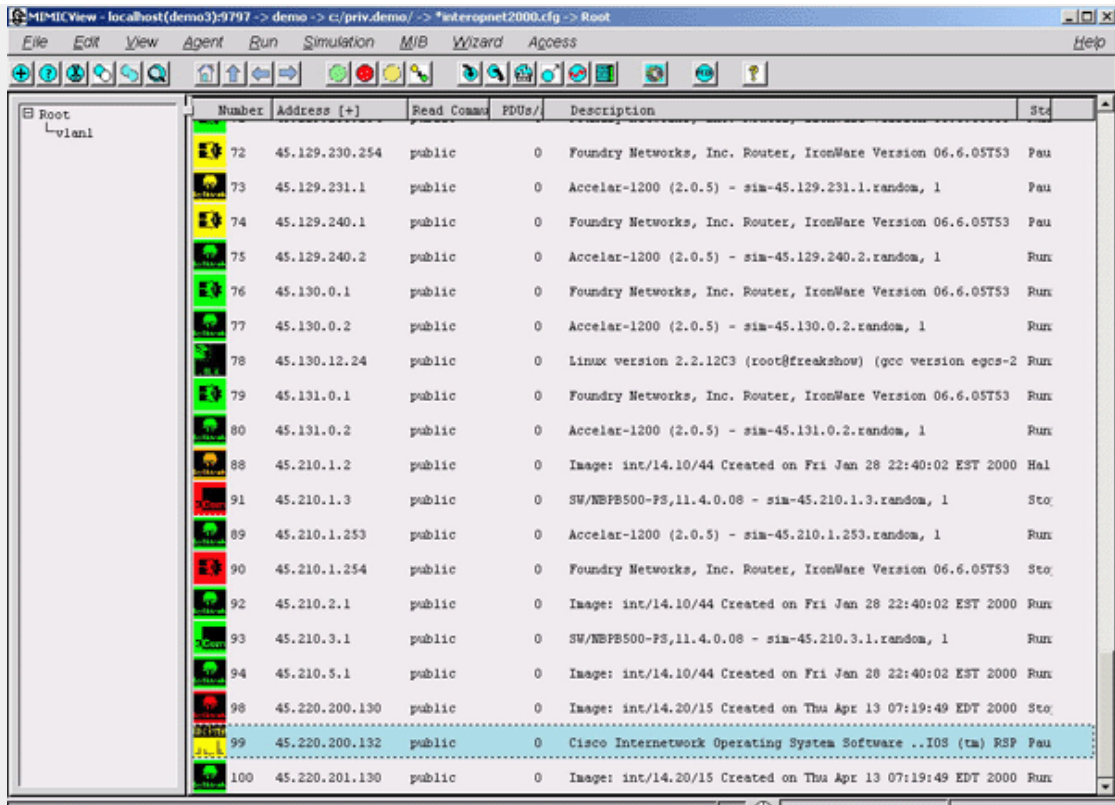


Figure 5: MIMICView Front Panel (Explorer view)

While these 2 views are adequate for simulations up to 1000 agents, a new experimental "10000-foot" view is designed for large-scale simulations, up to 10000 agents visible on a common display. This view will visualize activity patterns on your agents, allowing you to focus your attention on the agents that are being accessed by your management app.

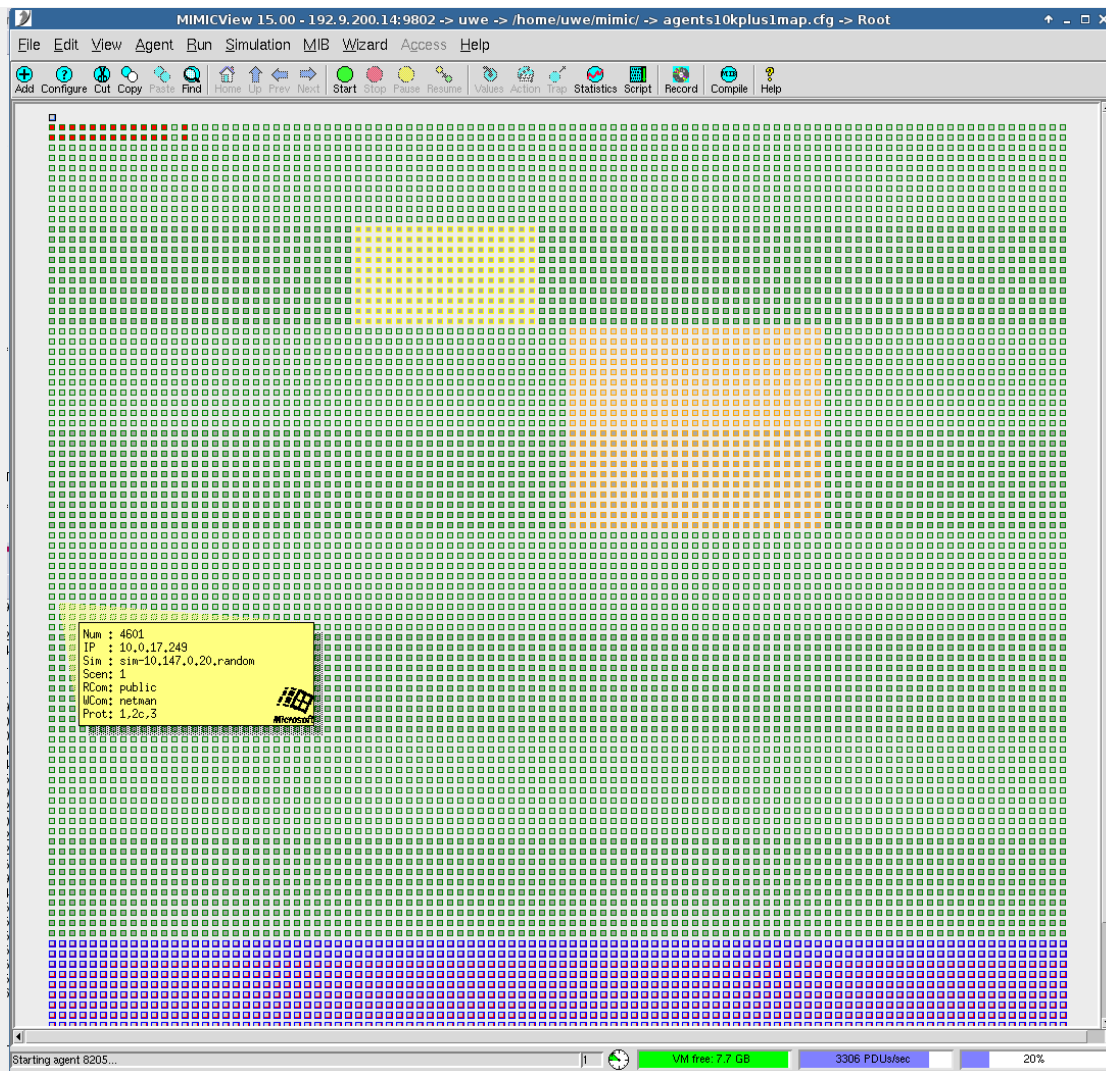


Figure 5-2: MIMICView Front Panel (10000 ft view)

View->Up

MIMIC maintains hierarchical **maps** of agent instances, which you can traverse by double-clicking on the map icon. To go back to the parent map, use **View->Up**.

View->Home

The **View->Home** menu item brings you to the top-most Root map.

View->Previous

MIMICView maintains a history of the traversed maps, which you can navigate with **View->Previous** and **View->Next**

View->Next

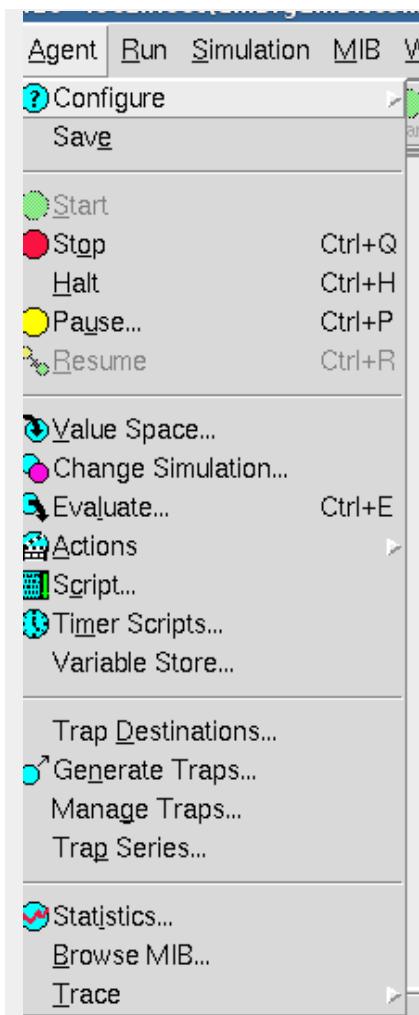
MIMICView maintains a history of the traversed maps, which you can navigate with **View->Previous** and **View->Next**

View->Refresh

The **View->Refresh** menu item refreshes the view. Agent icons are drawn in numeric order in the Classic view, and in sort order in the Explorer view.

Agent Menu

Individual or groups of agent instances can be manipulated with the menu items in the **Agent** menu. This menu also appears when you right-click on one or more selected agent instances.



Agent->Configure

This is the same as [Edit -> Configure](#).

Agent->Save

This menu item saves the simulation values for the selected agent(s). Any MIB object variables are saved for the simulations running on the selected agent(s). This is different from the [File->Save](#) menu item, which saves the current [lab configuration](#).

Agent->Start, Agent->Stop **QuickStart**

To start or stop agent instances, first select the desired instances in the main canvas. Then select the `Start` or `Stop` menu items. The icons for the agent instances change color as follows:

- **Purple** for instances in the process of starting or stopping
- **Green** if started successfully
- **Red** if stopped

Agent instances that are not selected are unaffected.

You can do certain things, for example, configuration, only on stopped agent instances.

When an agent is started, the simulator runs its configured [agent start action scripts](#) in the `STARTED` state, after it has done all its configuration (IP addresses, etc), but just before accepting messages to the agent. These scripts are named `start.mtc1` and `start+*.mtc1` and reside in the agent's simulation directory. They are executed in alphabetical order (`start.mtc1` first).

When an agent is stopped, the simulator runs its configured [agent stop action scripts](#) in the `STARTED` state, just before unconfiguring the agent. These scripts are named `stop.mtc1` and `stop+*.mtc1` and reside in the agent's simulation directory. They are executed in alphabetical order (`stop.mtc1` first). Since the stop action script runs in a critical region, one limitation is that the stop action script must not wait on other agents to stop, because deadlock will occur.

Agent->Halt

Use `Agent->Halt` to temporarily halt the simulation of the selected agent instances -- they will not respond to any requests, and the simulation will be stopped while the agent is halted. The icon in the main canvas turns **orange**, to show that the instance is halted.

This is useful if you are investigating a situation with specific agent instances and want to disable others so that they don't interfere. For example, if you are simulating subagents on a device, you can halt some while still running others.

This can also simulate failure of PDU processing of agents, eg. if the agent software is down.

Note that TRAP generation suspends for halted agents.

Halted agents can be restarted with `Agent->Resume`. The simulation will resume at the point in time at which the agent was halted.

Agent->Pause... **QuickStart**

Once an agent instance is running, you may want to investigate certain behavior of a management application at a specific point in time, that is, as if time had stopped. MIMIC allows this by pausing the simulation running on an agent instance. This lets you compare values relative to each other, or even to values retrieved in prior sessions.

To pause agent instance(s), use `Agent->Pause...` for the selected icon. Pausing an agent freezes time, i.e., `sysUpTime` does not advance. The `Agent Pause` dialog pops up, with the following configurable parameters:

- `Now` -- Selecting this radio button immediately pauses the agent when you press the `OK` or `Apply` button.
- `Set sysUpTime` -- Selecting this radio button allows you to specify the effective time to be set for the paused agent, i.e., `sysUpTime` is set to the specified time. This parameter lets you manipulate time, either into the future or into the past.
- `Days Hours Mins Secs` -- This input fields lets you set the effective time for the paused agent. For example, if you set `Days` to 1, and the rest of the fields to 0, the agent instance will be paused as if it had been running exactly one day.

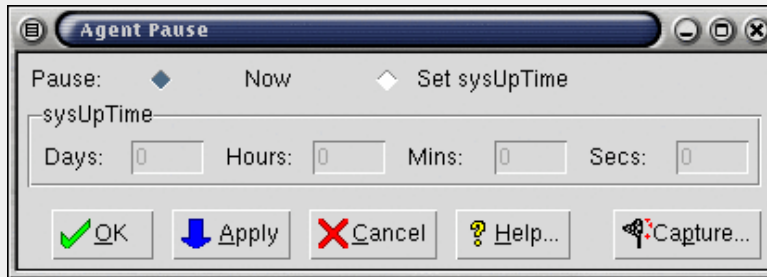


Figure 6: Agent Pause dialog

Click `OK` or `Apply` to pause the selected agent instance(s). The icon of each paused agent turns `yellow` to show that it is paused. Paused agents can be resumed with `Agent->Resume`. The simulation will resume at the point in time at which the agent was paused.

Agent->Value Space... **QuickStart**

MIMIC allows you to customize a running `basic simulation` by manipulating values returned by objects in the `Value Space`.

You can inspect and change values in the Value Space via `Agent->Value Space...` for the selected agent icon. A dialog with the title `Value Browser` pops up with 3 main components detailed below:

- `MIB Browser` on left
- `value matrix` on right
- `Favorites` on bottom

It shows a `MIB Browser` in the left `MIB Object` pane, which displays a tree diagram of the MIB object hierarchy. Each node in the tree is either a subtree or a leaf MIB object. The `Find` button lets you find the object name you typed in the `Name` field. The `Details...` button will be enabled if the MIB source file for the currently selected object is available, and lets you inspect the MIB definition for this object.

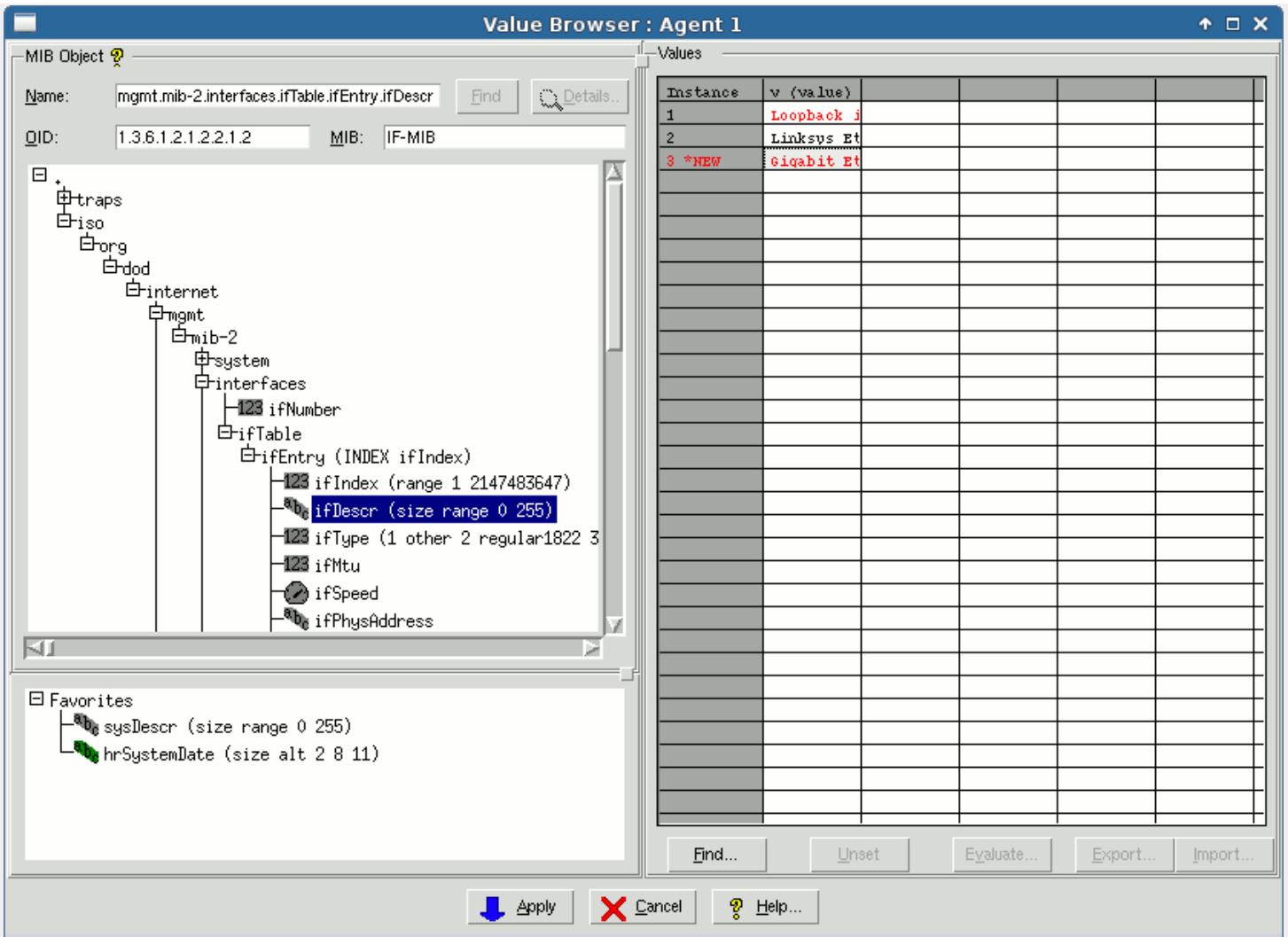


Figure 7: Value Browser

You can open subtrees in the hierarchy by double-clicking on branch nodes that are preceded by a plus (+) box, or by single-clicking on the plus box itself. You can close subtrees by single-clicking on the minus (-) box before a branch node.

MIB object leaf nodes contain information for the object. A symbol denotes the type of the MIB object. These are the currently displayed object types:

- 123 - Integer
- abc - OctetString
- [Gauge icon] - Counter, Counter64
- [Gauge icon] - Gauge
- [IP icon] - IpAddress
- [Triangle icon] - OBJECT IDENTIFIER
- [Clock icon] - TimeTicks
- [Trap icon] - SNMPv1 Trap
- [Trap icon] - SNMPv2 Trap
- [D icon] - Address
- [0110 icon] - BITSTRING
- [Net icon] - NetAddress
- [H icon] - Opaque

In addition, the color indicates the access to the object:

- gray - read-only, not-accessible, accessible-for-notify
- green - read-write, write-only
- yellow - read-create

You select a MIB object by clicking on the leaf node in the tree.

You can type a object name in the Object field, and click the Find button to directly select it.

For example, for the outgoing octets counter of a network interface you would use ifOutOctets.

The right side displays a value matrix with all instances of an object as rows. The columns list the variables in the Value Space with their value. You can expand and contract columns with Shift left-button-click on the top row. The simulation expression for an object can use any number of variable names, but the basic MIMIC simulation expressions generated by the [MIMIC Recorder](#) use some simple conventions:

- for all MIB objects which are not of type Counter the only variable used in the expression is v (for value).
- For all objects of type Counter, the variables used in the expression are r (for rate) and tu (for timeunit).
- For TRAP objects, the variables used in the expression are r (for rate), tu (for timeunit), c (for cutoff-time) and the names of any MIB objects to be sent in the trap.

To change a variable, just select (click) its cell and type the desired value. Use CTL-C to copy the selected text, and CTL-V to paste it. Press RETURN or click another cell to change the value. The value will become red to show its pending status.

As soon as you click Apply the value of the variable will be committed to the Value Space. Subsequent SNMP queries for this MIB object instance will return simulations using the new value.

This is useful to set the status of network interfaces via the ifStatus object, or to change rates of Counter objects in real time.

You can also add and remove MIB table entries for the selected agent. In the Instance column you can specify a MIB object instance index to be added to the table, by typing a value in the row after the last displayed row. Its value will show * NEW and will become red to show its pending status. To delete a row, clear its index number in the Instance column. Its value will show * DEL and will become red to show its pending status.

As soon as you click Apply the changes to the rows will be committed to the Value Space. Subsequent SNMP queries for this MIB table will return simulations using the new instances.

The Unset button unsets the previously set value of the currently selected cell.

The Find... button lets you find instances / values in the grid. Type the search string in the Instance/value field, then press Next or Previous to find the next or previous cell.

For counter objects, the Evaluate... button is a convenience to display the Agent->Evaluate... dialog to show the current simulation values that will be retrieved from the agent.

The Export... button lets you export a table (instances vs. variables) for an object as a comma-separated-value (CSV) file, which can be loaded into any popular spreadsheet programs. This allows you to do more advanced editing, specially for large tables. The Import... button imports the CSV file back into this table.

The Favorites pane underneath the MIB tree allows you to add shortcuts to your favorite MIB objects. To add a favorite, right-click on the MIB object and select Add to Favorites. A left-click on a favorite item will show the values in the value matrix for that object, without moving the position in the MIB tree. A double-click on the favorite item will position the MIB tree at the selected object. Thus you can quickly access your favorite MIB objects.

Agent->Change Simulation...

This menu item allows you to change the running simulation from basic to a pre-configured group of advanced simulations. The advanced simulations are implemented in plug-in scripts that automate the steps you used to do manually to accomplish the same effects.

You can change the simulation of the MIB Objects with Agent->Change Simulation... for the selected agent icon. A dialog with the title Change Simulation pops up. It shows the same MIB Browser on the left as the Value Space dialog above. The MIB Browser allows you to select the MIB object for which to change the simulation.

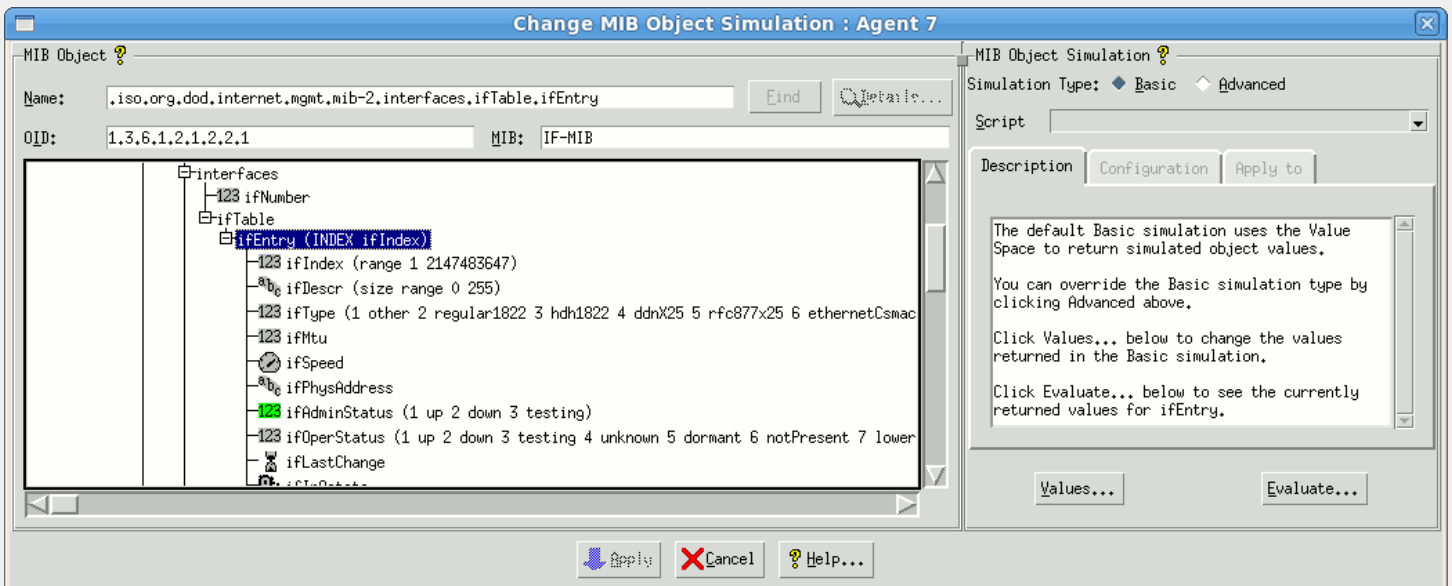


Figure 8: Change Simulation

Once a MIB object is selected, the MIB Object Simulation pane on the right shows the currently configured simulation. By default, this is the basic simulation which returns values directly based on the MIMIC Value Space.

You can change this to an Advanced simulation, which lets you pick from a list of preconfigured advanced simulations for this object from the Script selection list.

The Description tab will describe the currently selected choice, and will guide you through the steps. If the simulation is configurable, the Configuration tab will let you pick

the configurable values. The **Apply To** tab will allow you to deploy this simulation to either

- the selected object
- all the objects of the same type in this object group
- or all objects of the same type in the MIB

The **Source** tab lets you inspect the source code of the advanced simulation, either to make changes you need or to create new plug-ins.

Agent->Evaluate...

This dialog evaluates the simulations for MIB Object instances for an agent just as if they were returned via SNMP.

The MIB **Browser** behaves just as for [Agent->Value Space...](#)

The right side contains the familiar matrix, which displays all columns in a MIB table, or scalar objects underneath a branch. Shift left-button-click on the top row expands or contracts the width of a row between 3 possible states: wide enough to display all values in the column, 10 characters wide, contracted. Clicking on a value has no effect - the edit mode is allowed only to inspect the entire value.

To redisplay the value (eg. to monitor increasing Counter objects), just click on the object and the values will be updated.

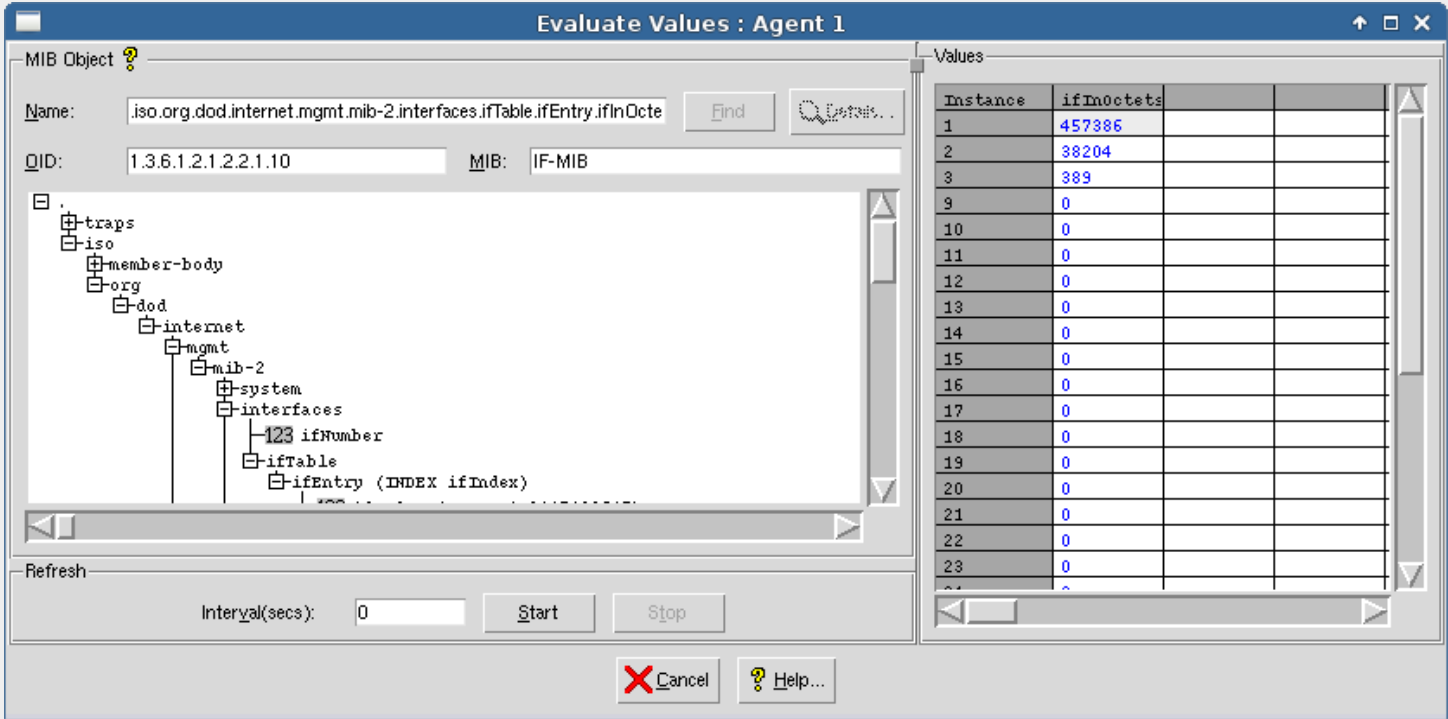


Figure 8-2: Evaluate Values

Agent->Actions **QuickStart**

The **Agent->Actions** menu allows you to setup various action scripts for an agent instance.

Agent->Actions->On GET/SET...

The **Agent->Actions->On GET/SET...** dialog lets you configure SNMP request action scripts for a MIB object on an agent instance.

An action script is a **MIMICShell** script file or **C++** dynamically loaded library that is executed in the Simulator on certain events. This menu item lets you configure actions to be executed when a MIB object instance is accessed through SNMP requests. A **GET** action will be executed on **GET**, **GETNEXT** or **GETBULK** requests; a **SET** action will be executed on **SET** requests. The exact rules for actions are defined [at the end of this section](#).

- **Agent ID** -- Displays the ID of the selected agent.
- **Object** -- This field specifies the MIB object to which you want to assign an action. You can either type the MIB object name in the Object field, or browse with the **MIB Browser** dialog by clicking on **Browse...**

For example, to assign a **SET** action to the **RMON2 Higher-Level Matrix Control Table RowStatus** object, you would use `hlMatrixControlStatus`.
- **Get script** -- Specifies the filename of the **GET** action script. If you click **Browse...**, you can browse for the file. The **File Browser** dialog also lets you create a new script and edit it. To clear the action, clear the text entry field.
- **Set script** -- Specifies the filename of the **SET** action script. If you click **Browse...**, you can browse for the file. The **File Browser** dialog also lets you create a new script and edit it. To clear the action, clear the text entry field.

When you click **Apply** or **OK**, the action script is associated with this object.

NOTE: The **MIMICShell** script is cached by the **MIMIC** daemon. In order to force a reload of the script, you must click **Apply** or **OK** after changing it in the editor.

SNMP Request Actions

The action script is executed when an SNMP PDU is processed.

- For GET* requests, the action script is run after looking up the Value Space, from which the current value is passed into the script. The script has the option of modifying the returned value in the response.
- For SET requests, the action script is run, then the returned value is stored in the Value Space.
- For both types of scripts, an error return from the script execution will result in an error response PDU.

Data is exchanged by the MIMIC daemon with the action scripts using a predefined set of global variables, detailed below :

- **gCurrentAgent** (IN): This variable is used to pass the number of the agent for which the script is being executed. The first agent is 1.
- **gCurrentRequest** (IN): This variable is used to indicate what kind of SNMP request triggered this script. Possible values are SET, GET, GETNEXT, GETBULK and UNKNOWN.
- **gCurrentObject** (IN): This variable indicates the MIB object for which the script was invoked. This is a user-friendly name like sysDescr, ifIndex etc.
- **gCurrentInstance** (IN): This variable gives the instance for which the script was invoked. This will be 0 for scalar variables and the index value for table variables.
- **gCurrentValue** (IN/OUT): This variable is used to pass the current valuespace content for a GET operation and the value to be set for a SET operation. The user can override the value of this variable. For the GET operation, the new user-specified value is sent back, while in case of the SET operation, the new user-specified value is the one that is set.
- **gSharedDir** (IN): This variable specifies the shared directory.
- **gPrivateDir** (IN): This variable specifies the private directory for the current agent.
- **gOwner** (IN): This variable specifies the owner for the current agent.
- **gPduInfo** (IN): This variable is used to pass additional PDU information for this action. Currently, this is only a single boolean (0 or 1) which indicates whether this variable is the last in the PDU. This can be used to do special PDU processing for the last variable in a PDU (ie. to approximate multi-stage SET processing).
- **gError** (IN/OUT): This variable is primarily used to indicate any SNMP error from the script. When passed in it is always 0 which means no-error. If modified to any other number, this translates to an error condition. The following table lists all error values that the script can return:

```
SNMP v1/v2/v3
-----
noError          0
tooBig           1
noSuchName       2
badValue         3
readOnly         4
genError         5

SNMP v2/v3
-----
noAccess         6
wrongType        7
wrongLength      8
wrongEncoding    9
wrongValue       10
noCreation       11
inconsistentValue 12
resourceAvailable 13
commitFailed     14
undoFailed       15
authorizationFailed 16
notWritable      17
inconsistentName 18

MIMIC specific
-----
drop PDU         -1
jump to different OID -2
set varbind value to
  noSuchObject   -3
  noSuchInstance -4
```

All Tcl commands (but no other Tcl extensions) are supported inside a MIMICShell action script. To exit the script, use the `return` statement instead of `exit`. The `exit` statement has no effects for embedded interpreters. In addition to this, a subset of the [MIMICShell commands](#) is also available to the user. The following is the list of the commands that can be used from an action script:

```
mimic get
mimic agent assigned
mimic agent assign
mimic agent start
mimic agent stop
```

NOTE: the `mimic agent stop` command will not work in a start action script since the agent is not running yet when the action script executes.

```
mimic agent pause
mimic agent halt
mimic agent resume
mimic agent get
mimic agent mget
mimic agent set
```



```

mimic agent mset
mimic agent save
mimic agent ipalias list
mimic agent ipalias add
mimic agent ipalias delete
mimic agent ipalias start
mimic agent ipalias stop
mimic agent store
mimic store
mimic value get
mimic value set
mimic value mget
mimic value mset
mimic value unset
mimic value munset
mimic value eval
mimic value meval
mimic value add
mimic value remove
mimic value pos
mimic value oid
mimic value name
mimic value mib
mimic value info
mimic value list
mimic value instances
mimic value variables
mimic value split
mimic timer script list
mimic timer script add
mimic timer script delete
mimic trap config list
mimic trap config add
mimic trap config delete
gci expr

```

In addition, the following MIMICShell commands can ONLY be used in an action script:

- **mimic action version**
Returns the SNMP version number. One of 1, 2c, 2 or 3.
- **mimic action auth**
Returns the authentication information for the request, for SNMPv1 and SNMPv2c that is the community string, for SNMPv3 that is the user name.
- **mimic action varbinds**
List the variable bindings in this PDU. This is provided to be able to process other variable bindings in the same PDU, eg. for semantic checking of other table columns.
- **mimic action reqid**
Retrieve the request ID in this PDU.
- **mimic action from**
Retrieve the source address in this PDU.
- **mimic action to**
Retrieve the destination address in this PDU. This is useful if an agent has IP aliases, and you want to behave differently for each.
- **mimic action jump OID**
Jump to a different OID while doing GETNEXT processing. The action script must return -2 in the `gError` variable. This is a way to violate lexicographic ordering of MIB objects. For example, this script will cause the OID processing to jump to `sysName`:

```

mimic action jump 1.3.6.1.2.1.1.4
set gError -2
return

```

For example, the following script can be associated with a `rowStatus` variable of a table. When a manager creates a row it initially sets the `rowStatus` to 2 (`createRequest`), upon which the agent creates the row and sets it to 1 (`valid`). Also when the manager sets the `rowStatus` for a row to 4 (`invalid`), the agent deletes that row. This can be simulated using the following script

```

# This action implements generic SNMPv2 RowStatus semantics
# minus the error checking. Our assumption is that the
# management application works correctly.

# value checking
if { $gCurrentValue < 1 || $gCurrentValue > 6 } {
    # bad value
    set gError 3
    return
}

# createAndGo
if { $gCurrentValue == 4 } {

    # set rowStatus to "active"
    set gCurrentValue 1

```

```

    return
}

# destroy
if { $gCurrentValue == 6 } {

    # position at the xxxTable point in the table
    mimic value pos $gCurrentObject
    mimic value pos ..
    mimic value pos ..
    set entry_name [mimic value list]

    # delete the row
    mimic value remove $entry_name $gCurrentInstance

    return
}

return

```

Agent->Actions->Start...

The Agent->Actions->Start... dialog lets you configure startup action scripts for an agent instance. The script will be run at agent startup.

The In Simulation field lets you browse the existing scripts in the agent simulation. The From Template field lets you copy one of the templates from the `scripts/` folder.

Agent->Actions->Stop...

The Agent->Actions->Stop... dialog lets you configure stop action scripts for an agent instance. The script will be run at agent stop.

The In Simulation field lets you browse the existing scripts in the agent simulation. The From Template field lets you copy one of the templates from the `scripts/` folder.

Agent->Script...

You can run a batch-mode script on the selected agent instance with this menu item. The Agent Script dialog lets you select the script to run, either Tcl-based in a [MIMICShell](#), [Perl-based](#), or [Python-based](#). This script could have been previously generated with the [MIMICView script generator](#).

All Tcl scripts must reside in the `scripts/` directory. A [log window](#) will pop up with the output of the MIMICShell scripting tool, `mimicsh`, `mimicsh-perl` or `python`. The script will run in the background until it finishes or is explicitly stopped.

The Arguments field passes optional arguments to the script parser (via the command line option `--args`).

Agent->Timer Script...

The Agent Timer Scripts dialog schedules per-agent timer-based scripts. The script file has to exist in the simulation directory of the agent, the interval is in msec.

The In Simulation field lets you browse the existing scripts in the agent simulation. The From Template field lets you copy one of the templates from the `scripts/` folder.

For more details, see [the Timer Scripts section](#).

Agent->Variable Store... **QuickStart**

This is the same dialog as [Edit->Variable Store...](#) but it also lets you edit variables of the selected agent(s).

Agent->Trap Destinations... **QuickStart**

The Agent->Trap Destinations... dialog lets you configure the trap destinations for an agent instance. Any traps generated by the agent instance will be sent to all its configured trap destinations. This overrides any configured global trap destinations (see [Edit->Trap Destinations...](#)).

Each entry is a comma-separated IP address and port number. The default trap port number is 162. A trap destination should specify the host running a management application prepared to receive traps.

Agent->Generate Traps... **QuickStart**

The Agent->Generate Traps... dialog lets you generate traps for an agent instance. Any traps generated by the agent instance will be sent to all configured trap destinations in the same address domain (ie. only the subset of IPv4 trap destinations for an IPv4 source agent address, or only the subset of IPv6 trap destinations for an IPv6 source agent address).

- In the General tab you can specify the:
 - Agent ID for which to generate the traps.
 - Object -- This field specifies the trap object. You can either type the MIB object name in the object field, or browse with the [MIB Browser](#) dialog by clicking on `Browse...`
For example, for the trap generated when a network interface becomes active you would use `linkUp`.
 - Rate -- This parameters specifies the number of traps to generate per time unit. For example, if you want to generate 3 traps every 10 seconds, the Rate field should contain 3.
 - Time Unit -- This parameters specifies the frequency with which to generate traps at the specified rate. For example, if you want to generate 3 traps every 10 seconds, the Time Unit field should contain 10.
 - Cutoff Time --
This parameters specifies the time duration for the trap generation. For example, if you want to generate traps for the next 20 seconds, the Cutoff Time field

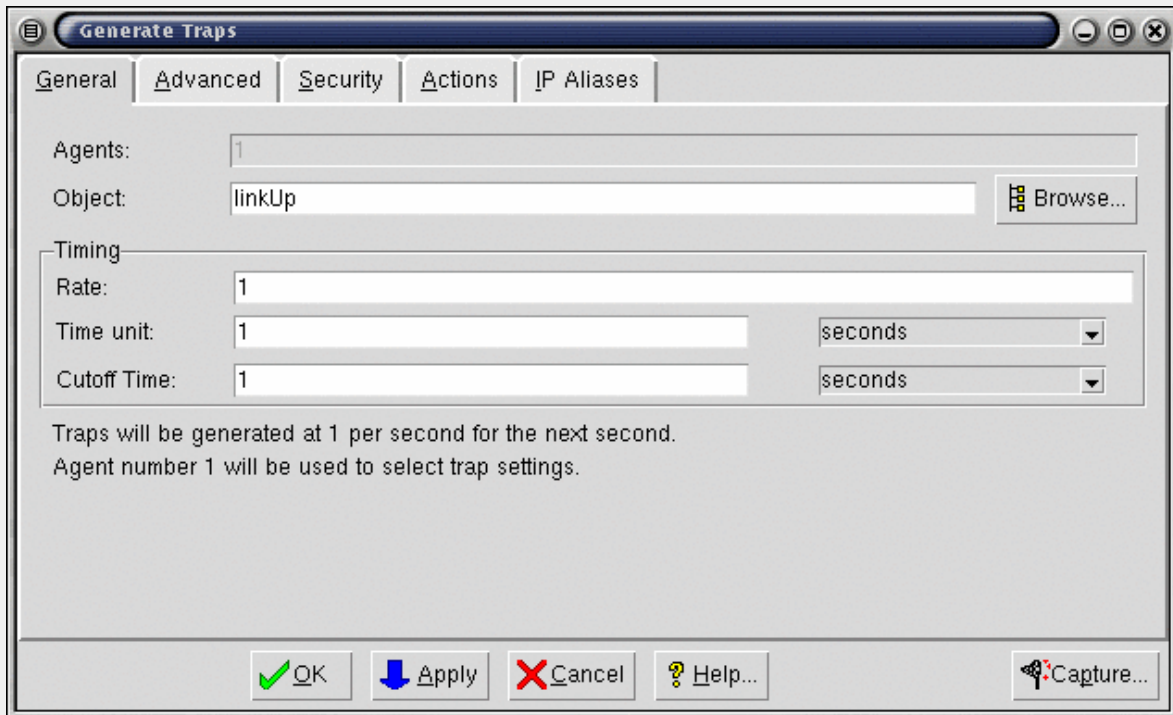


Figure 9: Generate Traps dialog

- In the **Advanced** tab you can specify the values of any variables to be sent with the trap.

Once you select a variable, you can enter an instance (if the variable binding is to be sent with an instance). Then set its value in the **Value** field, and click **Modify** to modify its value. Alternatively, if you click **Evaluate...**, it will invoke the **Agent->Evaluate...** dialog where you can pick the instance and value. When you dismiss that dialog, those are entered in this dialog.

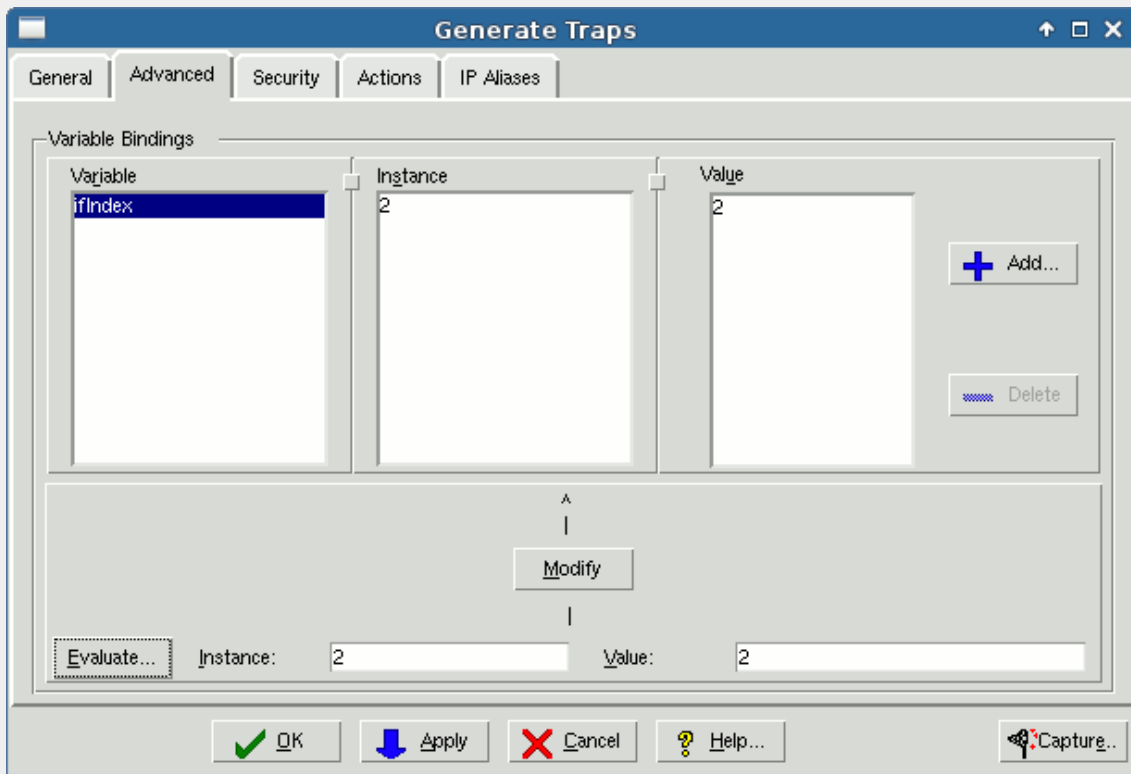


Figure 10: Generate Traps Advanced tab

- In the **Security** tab you can specify the SNMP version of the trap PDU and associated security information for the generated trap.

Choose the appropriate SNMP version using the radiobuttons. The security information for different SNMP versions is as follows:

- For **SNMPv1**, this is the community string (if specified empty, the read community of the agent), enterprise OID (if empty, the snmpOID or sysObjectID of the agent), and optionally the agent address.

- For SNMPv2c, this is the community string (by default, the read community string of the agent).
- For SNMPv2, this is the source and destination party, and the context (no default).
- For SNMPv3, this is the user name and context (no default).

You can choose to send this trap as a TRAP or INFORM.

- In the **Action** tab you can specify an action script to be invoked before packaging and sending each TRAP or INFORM request, as well as on response or timeout of an INFORM. This action script could increment a counter, set a variable to a dynamic value (eg. current time, or the value of a counter), etc.

In addition to the global variables

- gCurrentAgent
- gCurrentRequest
- gCurrentObject
- gSharedDir
- gOwner
- gPrivateDir

documented for [SNMP Request actions](#), trap action scripts receive these variables:

- **gStatus** (IN): This variable is used to pass the status of the TRAP or INFORM, ie. whether it is before sending the notification (value 0), or on INFORM response (1) or timeout (2).
- **gReqId** (IN): This variable is used to pass the request ID of the notification. This can be used to match up INFORM requests and responses.
- **gReportType** (IN): For a report, it indicates the type of report with these possible values:

```
1 == usmStatsUnsupportedSecLevels
2 == usmStatsNotInTimeWindows
3 == usmStatsUnknownUserNames
4 == usmStatsUnknownEngineIDs
5 == usmStatsWrongDigests
6 == usmStatsDecryptionErrors
```

- **gMsgAuthoritativeEngineID** (IN): engine id reported by the manager
- **gMsgAuthoritativeEngineTime** (IN): engine time reported by the manager
- **gMsgAuthoritativeEngineBoots** (IN): engine boots reported by the manager

For example, the `inform_action.mtcl` script in the `scripts/` directory shows an INFORM action script, or the following script increments a global counter, which is sent as `someVariable` with the `myTrap` TRAP:

```
global my_global
if { [info exists my_global] } {
    incr my_global
} else {
    set my_global 1
}
mimic value set myTrap 0 someVariable $my_global
```

- In the **IP Aliases** tab you can select which of the [IP Aliases](#) configured for the agent instance is to send the trap(s).

Click **OK** or **Apply** to start trap generation.

There is advanced functionality which is accessible only from the [MIMICShell](#).

Agent->Manage Traps...

The `Agent->Manage Traps...` dialog allows you to manage the currently active traps, ie. traps that are being generated. Inactive traps which were generated in the past, ie. the cutoff time has expired, will not show up in this dialog.

The list of active traps for the selected agent instance(s) is shown in a hierarchical tree. All relevant information is shown for each trap, such as it's current rate, duration and variables that are sent.

This dialog also lets you cancel any of the active traps.

Agent->Trap Series...

The `Agent->Trap Series...` dialog controls a trap series configured with the [Trap Wizard](#) for an agent.

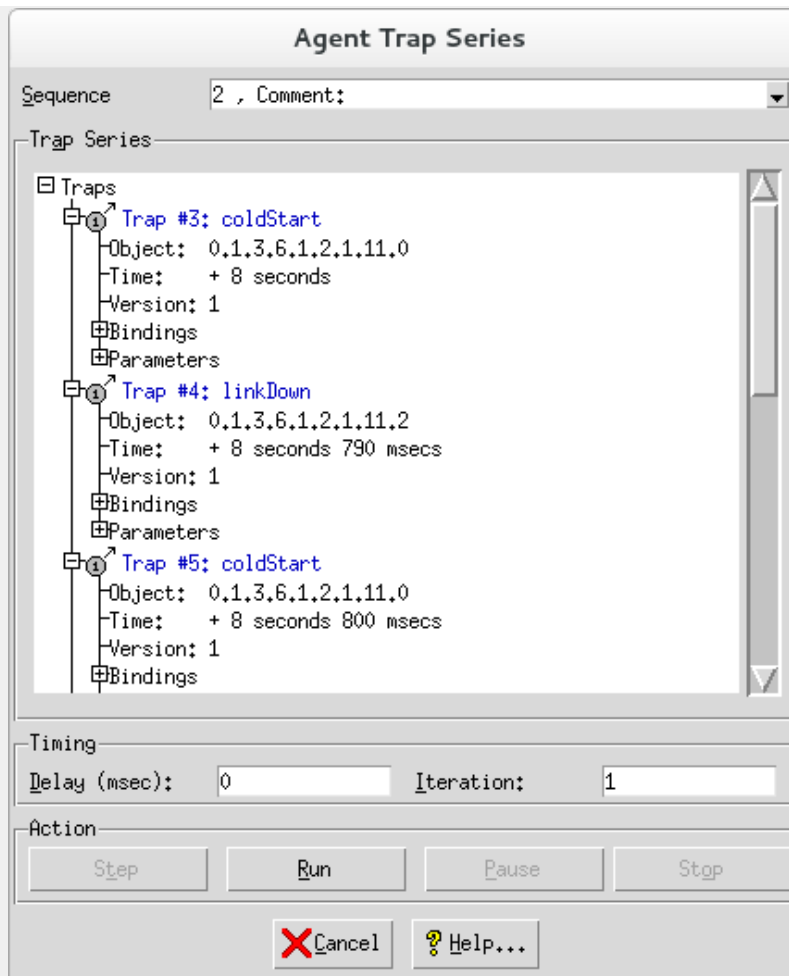


Figure 10-2: Agent Trap Series

You can click **Run** to run the sequence of traps at any time. Click **Step** to send a single selected trap. While a series is generating, click **Pause** to pause the sequence at the currently selected trap. Click **Stop** to stop the series.

Agent->Save

If an agent simulation changes during the session, either via SNMP SET requests or through the [Value Space](#) dialog, you will be able to save the changed simulation values when you stop the agent. The **Agent Stop** dialog pops up for every changed agent instance to be stopped, allowing you to save the changes. The **Yes to all** button lets you save all changed agents, the **No to all** button lets you ignore all changes. The **Cancel** button cancels the operation.

The **Agent->Save** menu item lets you save the changed agent value space data at any time.

Agent->Statistics...

This menu item displays detailed statistics for the currently selected agent instances. This allows quantitative diagnosis of a simulation running on an agent instance. The following statistics are shown:

- **Total PDUs** -- The total number of PDUs processed at the agent instance. This is a sum of the other statistic values.
- **Error PDUs** -- The number of PDUs returned with an error indication.
- **Discarded PDUs** -- The number of PDUs discarded due to error or retransmit.
- **GET PDUs** -- The number of GET PDUs processed.
- **GETNEXT PDUs** -- The number of GETNEXT PDUs processed.
- **SET PDUs** -- The number of SET PDUs processed.
- **GETBULK PDUs** -- The number of GETBULK PDUs processed.
- **TRAP PDUs** -- The number of TRAP PDUs generated.
- **GET variables received** -- The number of GET variables received.
- **GETNEXT variables received** -- The number of GETNEXT variables received.
- **SET variables sent** -- The number of SET variables sent.
- **GETBULK variables received** -- The number of GETBULK variables received.

- **INFORM PDUs sent** -- The number of INFORM PDUs sent.
- **INFORM PDUs resent** -- The number of INFORM PDUs that had to be resent.
- **INFORM PDUs timed out** -- The number of INFORM PDUs that timed out.
- **INFORM PDUs acked** -- The number of INFORM PDUs that were acknowledged.
- **INFORM Report PDUs received** -- The number of INFORM Report PDUs received.

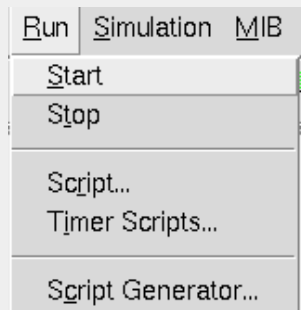
The statistics are refreshed every 5 seconds.

Besides built-in graphing of the selected statistics fields via the `Graph...` button in the `Plot` widget, you can graph to [Graphite](#) or [Grafana](#) by changing the `Graph` configurator in [Configuration Wizard](#).

Agent->Trace

The `Agent->Trace` menu lets you enable or disable tracing on an agent for the supported protocols (SNMP, DHCP, TFTP, TOD). The trace output will appear in the [Log Window](#).

Run Menu **QuickStart**



The `Run` menu allows starting and stopping of the SNMP Agent Simulator, and scheduling of global timer-based scripts.

- The `Start` menu item starts all agent instances.
- The `Stop` menu item stops all agent instances.
- The `Script...` menu item allows you to run a client script, either Tcl-based in a [MIMICShell](#), [Perl-based](#), or [Python-based](#).

All Tcl scripts must reside in the `scripts/` directory. A [log window](#) will pop up with the output of the MIMICShell scripting tool, `mimicsh`, `mimicsh-perl` or `python`. The script will run in the background until it finishes or is explicitly stopped.

The `Arguments` field passes optional arguments to the script parser (via the command line option `--args`).

- The `Timer Scripts...` menu item schedules global timer-based scripts. The script file has to exist in the `scripts/` directory, the interval is in msec.

Timer scripts provide an alternative to MIMICShell based poll scripts (which sit in a loop, sleeping and waking up periodically). Typically these are required when the user needs to repeat a task at various time intervals (eg. change various interface utilizations based on the time of the day). Another application of timer scripts is an action that is performed some fixed time after an event.

Since timer scripts run within the simulator, unlike MIMICShell scripts, they execute much faster. Also there is no limit on the number of timer scripts that can be configured at the same time (although a large number of timer scripts will degrade the performance of the simulations). For more details, see [the MIMICShell timer script command](#).

- The `Script Generator` menu item creates a batch script in the desired MIMIC API language, either a Tcl-based [MIMICShell script](#), [Perl script](#), [Python script](#), [Java program](#), or [C++ program](#). The script is generated from the dialogs you use in MIMICView. These scripts can be saved, further customized, and later be used in batch mode in the MIMICShell.

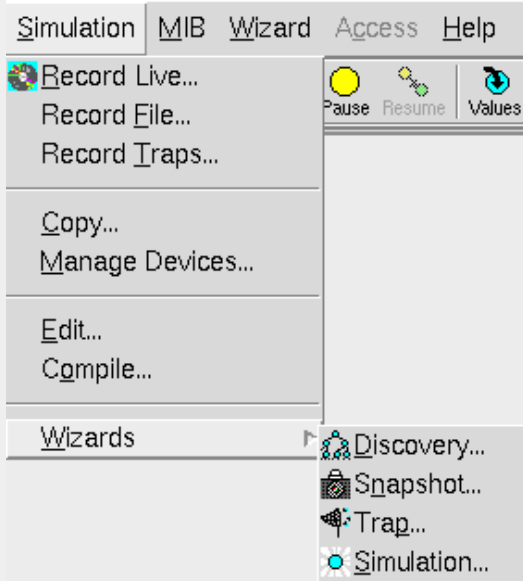
To start the scripting, press the `Capture` button. To stop it, press the `Stop` button. To clear the script, press `Reset` and to save it, press `Save...`

The following is a current list of dialogs/menu items that can generate scripts:

- `Run->Start`
- `Run->Stop`
- `Agent->Start`
- `Agent->Stop`
- `Agent->Pause`
- `Agent->Halt`
- `Agent->Resume`
- `Agent->Generate Traps`
- `Edit->Delete`

In general, a dialog that can be scripted will contain a `Capture...` button in the bottom button bar. This button is a shortcut to launch the `Script Generator`, if it is not already invoked.

Simulation Menu



The Simulation menu provides access to the simulations.

- The Record Live... menu item creates a simulation by recording a device with the [MIMIC Recorder](#).

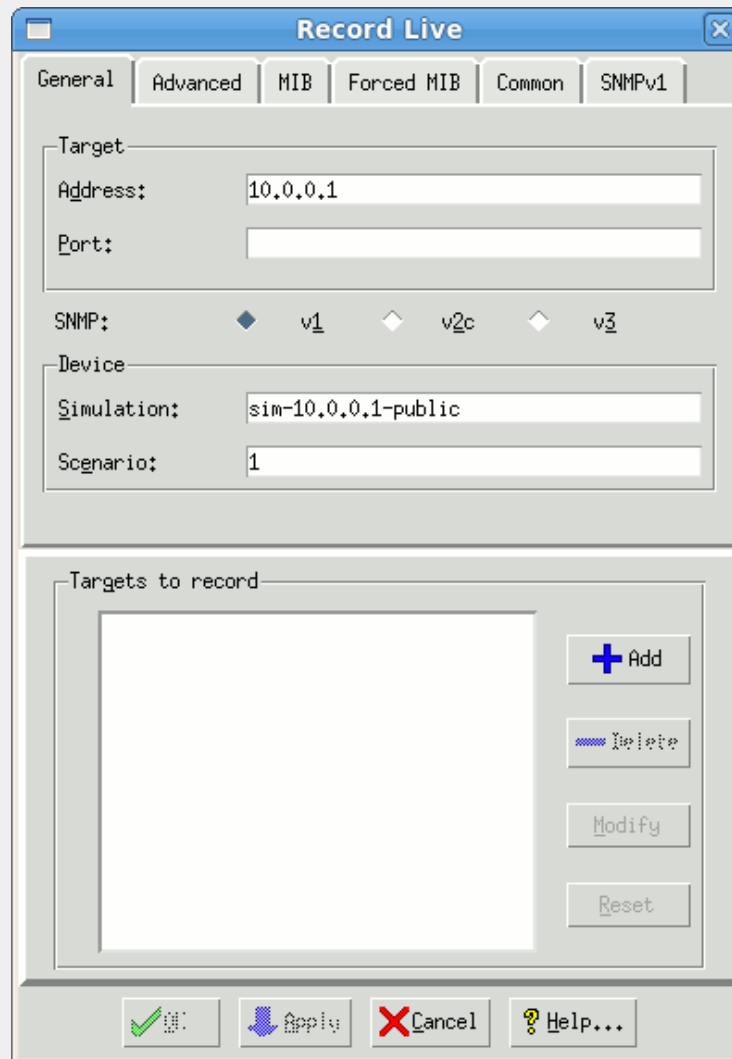


Figure 11: Record Live

- The Record File... menu item lets you create a simulation by recording from a file with the [MIMIC Recorder](#).
- The Record Traps... menu item invokes the Trap Recorder `trapper` to capture traps sent from a device. The output of the Trap Recorder will give you the traps in the order they were received, with timestamps (so that you can correctly space the trap generation), and the variables included in the trap PDU.

In the **General** tab specify the trap port, if different from the standard port 162.

In the **Filters** tab enter any IP addresses which you want to receive traps from or ignore.

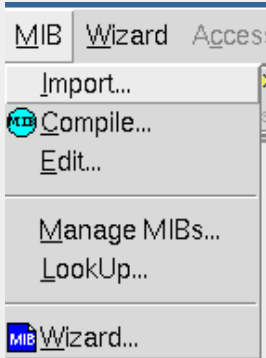
The **Trap Wizard** supports more advanced ways of recording traps and generating trap playback.

- The **Copy...** menu item allows you to copy a device simulation to a new simulation. This in effect duplicates the device. You can then experiment on the new device without affecting the original.

Type the existing simulation name in **From** or select with **Browse...**. Type the new simulation name in **To**, or you can create a new name within the file browser with **Browse...**, then **New...**

- The **Manage Devices...** menu item lets you review and modify the [Device Configuration](#).
- The **Edit** menu item lets you edit a simulation source file. For details see the section on [creating a simulation](#).
- The **Compile...** menu item compiles a simulation source file.
- The **Wizards** submenu is a shortcut into related wizards in the [Wizard](#) menu.

MIB Menu



The MIB menu provides access to loading, compiling and editing of MIB source files. For an overview, see [MIB Compilation Process](#).

- The **Import...** menu item imports a MIB source file as detailed in the [Compiler Guide](#).
- The **Compile...** menu item compiles a MIB source file as detailed in the [Compiler Guide](#).
- The **Edit...** menu item lets you edit a MIB source file as detailed in the [Compiler Guide](#).
- The **Manage MIBs...** menu item allows to edit the MIB Directory as detailed in the [next section](#).
- The **Lookup...** menu item lets you lookup MIB object information, as detailed in the section on [Oid Info](#). The **Details...** button will be enabled if the MIB source is available. The MIB source for all pre-compiled MIBs is downloadable via the [Update Wizard](#).
- The **Wizard...** menu item is a shortcut into the [Wizard->MIB...](#) menu item to invoke the MIB Wizard.

MIB Directory

The **Edit MIB Directory** dialog is an advanced feature which allows modification of the MIMIC MIB Directory. The MIB Directory is a database (stored in `data/mibs/mimic.dir`), which maps OIDs to MIBs. This is the database that lets MIMIC decide for example that the OID 1.3.6.1.2.1.1 is in RFC1213-MIB. The MIB Directory is built by compiling MIB source files using the [MIMIC Compiler](#). MIMIC does not know about a MIB until it is compiled with the MIMIC Compiler into the MIB Directory.

You would use this dialog to remove unwanted MIBs or change the order of compiled MIBs for object resolution. We recommend you do this after consultation with Gambit Technical Support.

On the left side you can manipulate the list of MIBs. The MIB directory lists the MIBs in order of compilation. The MIBs that come pre-compiled with MIMIC are shown in **blue**, while the MIBs that you compiled are shown in **red**. Your compiled MIBs in the private data area show first. This is due to the difference between the [shared and private data areas](#). You can only modify MIBs in your private data area, ie, if it is shown in **red**.

As a convenience, you can find MIBs by name with the **Find...** button. In the **Find MIBs by name** dialog enter parts of the MIB name, and press **Next** or **Previous** to search forwards or backwards. If the MIB is found, the found MIB is selected.

The **Up** or **Down** buttons in the left pane reorder the selected MIB in the list. The **Delete** button removes the selected MIB from the directory. The **Modify** button modifies the MIB with the changes in the right pane.

The right side of the dialog lists the details about the selected MIB. The interesting part is the list of "TOPOIDS", which are the "top" trees of the MIB. MIMIC can quickly map an OID to a MIB by checking which "top" trees they are under. The MIMIC Compiler stores these automatically when a MIB is compiled, but in rare circumstances you can add top OIDs, remove them or reorder them.

Wizard Menu



The Wizard menu provides access to the MIMIC Wizards detailed in the [MIMIC Wizards Guide](#).

- The MIB... menu item invokes the [MIB Wizard](#).
- The Discovery... menu item invokes the [Discovery Wizard](#).
- The Snapshot... menu item invokes the [Snapshot Wizard](#).
- The Trap... menu item invokes the [Trap Wizard](#).
- The Simulation... menu item invokes the [Simulation Wizard](#).
- The CLI... menu item invokes the [CLI Wizard](#).
- The NetFlow... menu item invokes the [NetFlow Wizard](#).
- The sFlow... menu item invokes the [sFlow Wizard](#).
- The IPMI... menu item invokes the [IPMI Wizard](#).
- The Web... menu item invokes the [Web Wizard](#).
- The Configuration... menu item invokes the [Configuration Wizard](#).
- The Update... menu item invokes the [Update Wizard](#).
- The Diagnostic... menu item invokes the [Diagnostic Wizard](#).
- The Sanity... menu item invokes the [Sanity Wizard](#).
- The Performance... menu item invokes the [Performance Wizard](#).
- The Protocol... menu item invokes the [Protocol Wizard](#).

Access Menu



The Access menu allows the MIMIC administrator to configure [agent access control](#). It is disabled for all other users.

- The Load... menu item allows to load a different access control configuration at run-time. This is useful, if your agent access control needs change over time. For example, the development team may have most of the agents in the morning, while the QA team has most of them in the afternoon.
- The Edit... menu item allows to edit the currently loaded access control parameters.
- The Save and Save As... menu items allow to save the currently loaded access control configuration.

Speed Bar



The most common actions have been included on the Speed Bar. The action to be performed by each button will be displayed as a hint in the status bar message area when the cursor is positioned over the button.

- Add Agent Instances...** Opens the Add Agents dialog box, which allows you to add and set the parameters of one or several new agent instances (shortcut to [Edit->Add->Agent...](#)).
- Configure Agent Instances...** Opens the Configure Agents dialog box, which allows you to change the parameters of the selected agent instance(s) (shortcut to [Edit->Configure...](#)).
- Cut Agent Instances...** Moves the selected agent instance or map (shortcut to [Edit->Cut](#)).
- Copy Agent Instances...** Duplicates the selected agent instance or map (shortcut to [Edit->Copy](#)).
- Paste Agent Instances...** Pastes an agent instance or map based on a previous Cut or Copy (shortcut to [Edit->Paste](#)).
- Find Agent Instances...** Finds an agent instance based on a specified expression in its configured data (shortcut to [Edit->Find](#)).
- Go Home...** Brings you to the top-most home map (shortcut to [View->Home](#)).
- Go to the Parent Map...** MIMIC maintains hierarchical maps of agent instances. This button will bring you to the parent map (shortcut to [View->Up](#)).
- Back in Map history...** MIMIC maintains a history of the traversed maps. This button will bring you to the previous map in the history (shortcut to [View->Previous](#)).
- Forward in Map history...** MIMIC maintains a history of the traversed maps. This button will bring you to the next map in the history (shortcut to [View->Next](#)).
- Start Agent Instances...** Starts the selected agent instance(s) (shortcut to [Agent->Start](#)).
- Stop Agent Instances...** Stops the selected agent instance(s) (shortcut to [Agent->Stop](#)).
- Pause Agent Instances...** Pauses the selected agent instance(s) (shortcut to [Agent->Pause...](#)).
- Resume Agent Instances...** Resumes the paused/halted agent instance(s) that you have selected (shortcut to [Agent->Resume...](#)).
- Browse Value Space...** Opens the Value Browser to change the Value Space of the selected agent instance(s) (shortcut to [Agent->Value Space...](#)).
- Evaluate Values...** Opens the Evaluate Values window, which evaluates the simulations for MIB Object instances for an agent just as if they were returned via SNMP (shortcut to [Agent->Evaluate...](#)).
- Actions...** Opens a dialog box, which lets you configure SNMP request action scripts for a MIB object on the selected agent instance(s) (shortcut to [Agent->Actions->On GET/SET...](#)).
- Generate Traps...** Opens a dialog box, which lets you generate traps for the selected agent instance(s). (shortcut to [Agent->Generate Traps...](#)).
- Agent Statistics...** Displays detailed statistics for the selected agent instances (shortcut to [Agent->Statistics...](#)).
- Agent Script...** Opens a dialog box, which allows you to select a batch-mode script to run on the selected agent instance (shortcut to [Agent->Script...](#)).
- Record Live Device...** Creates a simulation by recording a device with MIMIC Recorder (shortcut to [Simulation->Record Live...](#)).
- Compile MIB...** Compiles a MIB source file (shortcut to [MIB->Compile...](#)).
- Online Help...** Provides access to fully searchable and indexed online help.

The match: field lets you filter the agents in the current map by a regular expression. This is useful if you want to operate on a small subset of the agents. All future selection is done on the matched agents only.

4. MIMICShell Reference

The MIMIC TCL-based extensions provide an environment for programmatic and batch-oriented control over MIMIC simulations. There are 2 ways to use Tcl-based control over MIMIC:

- as a MIMIC client from the MIMICShell utility or from another Tcl-based interpreter and the MIMIC package.

The simplest form of programmatic control of MIMIC is through MIMICShell. MIMICShell is a ready-made TCL/Tk-based shell that is invoked from a shell command prompt with `mimicsh`

All commands in TCL, Tk, TclX, and Tix are supported, as well as the command set to control MIMIC. Just like any other Tcl-based shell, you can dynamically load additional packages (for example, the expect package).

The MIMIC package can be loaded into any TCL-based shell, such as `tclsh`, `wish`, `wishx`, `expect`, etc. It works with any other TCL extension package, such as `itcl`, `otcl`, etc. This way you have even more customization of your scripting environment. The MIMIC package encapsulates the MIMIC command set.

- as [MIMIC actions](#) which are run within the simulator based on a pre-defined set of events.

The MIMIC command set is invoked in either environment with the `mimic` command.

MIMICShell Command Line Options

- **--host hostname**

Open a session to the MIMIC daemon on the specified host. This is equivalent to the [mimic session open](#) command.

- **--agent agent-number**

Use specified agent-number as current agent. This agent-number can later be changed with the [mimic agent assign](#) command.

- **--script filename**

Run the script in the file specified by filename.

- **--args arguments**

Pass the specified arguments to the script. They will be available in the global variables `argc` and `argv`. Multiple arguments need to be passed as a list, ie. in curly braces. Eg. `--args {arg1 arg2 arg3}`. You may have to escape the curly braces in certain shells, such as with single quotes in `csh`.

The script needs to define `argc` and `argv` as globals, such as

```
global argc
global argv
```

- **--nosession**

Run without opening a session to the MIMIC Simulator daemon. This is useful if you want to open multiple sessions to multiple daemons.

- **--nogui**

On Unix, run without requiring an X display. This disables the graphical (Tk) functionality of the MIMICShell.

Interactive Command Line Editing

The MIMICShell on Windows already has command line editing built-in like the DOS command shell. On Unix, you can achieve the equivalent behavior with the `r1wrap` utility, which on Linux is installable as an extra.

MIMIC Package

In addition to the supplied MIMICShell utility, you can extend any TCL-based shell with the MIMIC extensions. To load the MIMIC package into a shell, do the following:

```
% set auto_path [linsert $auto_path 0 path-to-MIMIC-installation/lib]
% set auto_path [linsert $auto_path 0 path-to-MIMIC-installation/packages]
% package require Mimic
3.0
# from now on, the MIMIC command set is available, e.g.,
% mimic help
Usage: mimic
      where is one of
      help
      get
      value
      agent
      trap
      timer
      diag
      session
```

The following MIMIC command set is implemented:

Global Commands

- **mimic get info**

Get information about MIMIC, where `info` is one of the following:

- **max** -- The maximum number of agent instances. See [Examples](#).
- **last** -- The last configured agent instance. See [Examples](#).
- **version** -- The version of the MIMIC command interface.
- **clients** -- The number of clients currently connected to the daemon.
- **cfgfile** -- The currently loaded lab configuration file for the particular user. In the case of [multi-user access](#) this command returns a different configuration file loaded for each user.
- **cfgfile_changed** -- This predicate indicates if the currently loaded lab configuration file has changed since the last time this predicate was queried. This allows for a client to detect lab configuration changes and to synchronize those changes from the MIMIC daemon.
- **return** -- The return mode. MIMICShell can operate in two modes: `nocatch`, where error returns from MIMIC operations return error conditions as a return value; or `catch`, where the TCL `catch` semantics are used (these are similar to C++ exceptions). See [Examples](#).

- **configured_list** -- The list of {agentnum} that are currently configured. This list is guaranteed to be sorted into increasing order.
- **active_list** -- The list of {agentnum} that are currently active (running or paused). This list is guaranteed to be sorted into increasing order.
- **active_data_list** -- The list of {agentnum {statistics}} for agents that are currently active and whose statistics have changed since the last invocation of this command. This list is guaranteed to be sorted into increasing order.
- **changed_config_list** -- The list of {agentnum} for which a configurable parameter changed. This list contains at most 5000 agent(s), and is guaranteed to be sorted into increasing order.
- **changed_state_list** -- The list of {agentnum state} for which the state changed. This list contains at most 5000 agent(s), and is guaranteed to be sorted into increasing order.
- **log** -- The current log file for the Simulator.
- **protocols** -- The set of protocols supported by the Simulator.
- **interfaces** -- The set of network interfaces that can be used for simulations.
- **product** -- The product number that is licensed.
- **netaddr** -- The network address of the host where the MIMIC simulator is running.
- **netdev** -- The default network device to be used for agent addresses if the interface is not explicitly specified for an agent.
- **licensing** -- Get MIMIC licensing information.
- **threads** -- Get MIMIC threads information.

• **mimic mget info+**

Get multiple sets of information about MIMIC, where `info` is one of the parameters defined in the `mimic get` command. See [Examples](#).

• **mimic set info value**

Set global MIMIC information, where `info` is:

- **return mode** -- The return mode. `mode` is one of the following: `nocatch`, where error returns from MIMIC operations return error conditions as a return value; or `catch`, where the TCL `catch` semantics are used (these are similar to C++ exceptions). See [Examples](#).
- **log file** -- the log file. This changes the current log file for the Simulator. This allows to control the size of the log file by periodically changing it, so that older logs can be manipulated differently from newer logs. For example, a batch job can periodically backup and/or remove log files older than a certain age. By default, the MIMICView GUI changes the log file every day at midnight.
- **netdev device** -- The default network device to be used for agent addresses if the interface is not explicitly specified for an agent. This is persistent, ie. it will apply across restarts of the daemon.
- **persistent** -- This operation flushes all global objects which need to be made persistent to disk. The MIMIC daemon caches persistent objects and their changes, and writes them to disk at program termination. If it were to crash, these changes would be lost. This operation allows to checkpoint the cache, ie. write changes to persistent objects to disk. To save the lab configuration with per-agent persistent information the `mimic save` operation needs to be used.

• **mimic load file [agent-range] [starting-agent] [merge]**

Load the lab configuration file `file`. Same as [File->Open...](#)

The `agent-range` specifies the agents to load from the file. By default, or if "all" is specified, all agents are loaded.

The `starting-agent` specifies the relative agent position of the first agent to load in the file. Combined with the agent range, it lets you load any agent in the file at any position. Thus, to load the 6th through 9th agent in the file `agent.cfg` to the running configuration starting at agent number 100 would be

```
mimic load agent.cfg 6-9 100
```

By default, loading starts at agent 1 in the running config.

NOTE: loading a file is remembered persistently, but only if the file is loaded at starting agent position 1. Any other load operation at other relative positions is considered a non-persistent configuration change. Thus, if you want to preserve it you need to explicitly save the changed configuration after loading a config to a starting position other than 1.

The `merge` indicator specifies what to do with undefined agents in the file to be loaded. If 0 (the default), then agents are cleared in the running config for undefined agents in the file, else if 1 the agent in the running config is untouched.

• **mimic clear [agent-range]**

Clear the lab configuration. Same as [File->New...](#)

• **mimic saveas file [agent-range]**

Save the lab configuration in file `file`. Same as [File->Save As...](#)

- **mimic save**
Save the lab configuration. Same as [File->Save](#)

- **mimic start**
Start MIMIC. Same as [Run->Start](#)

- **mimic stop**
Stop MIMIC. Same as [Run->Stop](#)

- **mimic terminate**
Terminate the MIMIC daemon. Same as [File->Terminate](#)

Session Commands

- **mimic session open [server-host] [server-port]**
Open a session to a MIMIC daemon. All subsequent MIMIC commands control the simulation running on the specified target `server-host`. If `server-host` is not supplied, the local host is used. If `server-port` is not supplied, the default port is used.

- **mimic session close [session-number]**
Close a session to a MIMIC daemon. If no `session-number` is specified, the current session is closed.

- **mimic session assign session-number**
Select a session to a MIMIC daemon. The `session-number` has to be one of the currently open sessions. All subsequent MIMIC commands go to the selected session. This way, multiple sessions can be controlled simultaneously.

- **mimic session assigned**
Return the session-number of the currently assigned session.

- **mimic session list**
Return the session-numbers of the currently open sessions.

- **mimic session properties**
Returns the properties of the currently open sessions.

Agent Commands

- **mimic agent assign agent-number**
Set the current agent to the specified `agent-number`. See [Examples](#).

- **mimic agent assigned**
Return the number of the currently assigned agent. This is the same as the number passed to the `--agent` option to `mimicsh` or to the most recent [mimic agent assign](#) command. For example:

```
# do something with agent specified by "new"
proc do_something {new} {
    set old [mimic agent assigned]
    mimic agent assign $new

    # do something with the new agent
    #...

    # then restore the old agent
    mimic agent assign $old
}
```

• **mimic agent add address triplets**

Add an agent. Same as [Edit->Add->Agent...](#) See also [Examples](#).

• **mimic agent start**

Start the current agent. Same as [Agent->Start](#) For speed, this operation will complete asynchronously. A successful return from this command means the starting of the agent is in progress. If you need to rely on the agent to have completed startup, you should wait for it's state to become `RUNNING`. For example, this code to reliably start agents numbered between `START_I` and `END_I`

```
for {set agenti $START_I} {$agenti <= $END_I} {incr agenti} {
  mimic agent assign $agenti
  mimic agent start
}
for {set agenti $START_I} {$agenti <= $END_I} {incr agenti} {
  mimic agent assign $agenti
  while { [mimic agent get state] != $MimicData::AGENT_RUNNING } {
    sleep 1
  }
}
```

for a large number of agents is much faster than this loop

```
for {set agenti $START_I} {$agenti <= $END_I} {incr agenti} {
  mimic agent assign $agenti
  mimic agent start
  while { [mimic agent get state] != $MimicData::AGENT_RUNNING } {
    sleep 1
  }
}
```

because agent startup happens in parallel in the former case, whereas it is serialized in the latter.

• **mimic agent stop**

Stop the current agent. Same as [Agent->Stop](#) For speed, this operation will complete asynchronously. The same synchronization considerations apply as in [mimic agent start](#).

• **mimic agent remove**

Remove the current agent. Same as [Edit->Delete](#) For speed, this operation will complete asynchronously. The same synchronization considerations apply as in [mimic agent start](#).

• **mimic agent pause [when]**

Pause the current agent. Same as [Agent->Pause](#) If the `when` argument is supplied, the agent is paused at the specified time (in seconds). For example, if 300 is specified, the agent is paused at 5 minutes.

• **mimic agent halt**

Halt the current agent. Same as [Agent->Halt](#)

• **mimic agent reload**

Reload the current agent. This only works for halted agents. The net effect is the same as restarting an agent (ie. stop, start, halt), but without disconnecting the network (and thus existing connections).

• **mimic agent resume**

Resume the current agent. Same as [Agent->Resume](#)

• **mimic agent get info**

Get information about the current agent, where `info` is one of the following:

- **interface** -- network interface card for the agent. Same as [Edit->Configure](#).
- **host** -- host address of the agent. Currently, only IPv4 addresses are allowed as the main address of the agent, but both IPv4 and IPv6 addresses are allowed as [IP aliases](#) for the agent. See [Examples](#).
- **mask** -- subnet mask of the agent. Same as [Edit->Configure](#).
- **port** -- port number

- **protocol** -- protocols supported by agent as a comma-separated list
- **read** -- read community string
- **write** -- write community string
- **delay** -- one-way transit delay in msec. The minimum granularity is 10 msec.
- **start** -- relative start time
- **mibs** -- set of MIBs, simulations and scenarios
- **sim** -- first simulation name
- **scen** -- first scenario name
- **state** -- current running state of the agent:

0 Unknown
 1 Running
 2 Stopped
 3 Halted
 4 Paused
 5 Deleted
 6 Stopping

See [Examples](#).

- **statistics** -- current statistics of the agent instance

The statistics are returned as 64-bit decimal numbers for the following statistics:

- total PDUs
- discarded PDUs
- error PDUs
- GET PDUs
- GETNEXT PDUs
- SET PDUs
- GETBULK PDUs
- trap PDUs
- GET variables
- GETNEXT variables
- SET variables
- GETBULK variables
- INFORM PDUs sent
- INFORM PDUs re-sent
- INFORM PDUs timed out
- INFORM PDUs acked
- INFORM REPORT PDUs

The variable binding statistics are meaningful to keep track of the average number of variable bindings per PDU.

See [Examples](#).

- **changed** -- has the agent value space changed?
- **config_changed** -- has the lab configuration changed?
- **state_changed** -- has the agent state changed?
- **trace** -- SNMP PDU tracing
- **pdu_size** -- maximum PDU size. The limit for this configurable is 65536.
- **drops** -- drop rate (every N-th PDU). 0 means no drops.
- **owner** -- owner of the agent.
- **privdir** -- private directory of the agent.
- **oiddir** -- MIB directory of the agent.

- **validate** -- SNMP SET validation policy. Is a bitmask in which with the following bits (from LSB) check for
 - type
 - length
 - range
 - access

A default value of 65535 does all validation checking. For examples, see the [Frequently Asked Questions](#).

- **inform_timeout** -- timeout in seconds for retransmitting INFORM PDUs. The agent will retransmit INFORM PDUs at this interval until it has received a reply from the manager.
- **inform_retries** -- number of retries for retransmitting INFORM PDUs. The agent will retransmit at most this many INFORM PDUs or until it has received a reply from the manager.
- **num_starts** -- number of starts for the agent. This count is incremented each time an agent starts. It affects the SNMPv3 EngineBoots parameter.

5. **mimic agent mget info agent+**

Get information about the specified agents. `info` has the same choices as the `mimic agent get` command.

6. **mimic agent set info value**

Set information for the current agent. `info` has the same choices as the `mimic agent get` command.

7. **mimic agent mset info agent1 value1 ... agentN valueN**

Set information for the specified agents. `info` has the same choices as the `mimic agent get` command.

8. **mimic agent save**

Save agent MIB values.

9. **mimic agent ipalias list**

Lists all the additional ipaliases configured for the agent. The agent host address (set with `mimic agent set host`) is not in this list, since it is already accessible separately with `mimic agent get host`.

10. **mimic agent ipalias add ipaddress[,port[,mask[,interface]]]**

Adds a new ipalias for the agent. `port` defaults to 161 if not specified. `mask` defaults to the class-based network mask for the address. `interface` defaults to the default network interface.

If `port` is set to 0, the system will automatically select a port number. This is useful for client-mode protocols, such as [TFTP](#) or [TOD](#). Upon start of an IP alias with a 0 (auto-assigned) port number, its port will change to contain the value of the selected system port.

11. **mimic agent ipalias delete ipaddress[,port]**

Deletes an existing ipalias from the agent. `port` defaults to 161 if not specified.

12. **mimic agent ipalias start ipaddress[,port]**

Starts an existing ipalias for the agent. `port` defaults to 161 if not specified.

13. **mimic agent ipalias stop ipaddress[,port]**

Stops an existing ipalias for the agent. `port` defaults to 161 if not specified.

14. **mimic agent ipalias status ipaddress[,port]**

Returns the status (0=down, 1=up) of an existing ipalias for the agent. `port` defaults to 161 if not specified.

15. **mimic agent trap config list**

List the set of trap destinations for this agent instance.

Each trap destination is identified with an IP address and a port number. The default port number is the standard SNMP trap port 162. See [Agent->Trap Destinations....](#)

16. **mimic agent trap config add destination port**

Add a trap destination to the set of destinations.

17. **mimic agent trap config delete destination port**

Remove a trap destination from the set of destinations.

18. **mimic agent trap list**

List the outstanding asynchronous traps for this agent instance.

See [Agent->Manage Traps....](#)

19. **mimic agent from list**

List the source addresses that the agent will accept messages from. This in effect implements `source-address-indexing`, where 2 agents with the same address can be configured, each accepting messages from different management stations.

20. **mimic agent from add [ipaddress][,port]**
mimic agent from delete [ipaddress][,port]

Add/delete a source address that the agent will accept messages from. An empty ipaddress or 0.0.0.0 both imply any address. Similarly an empty port or 0 both imply any port. For agents with `source-address-indexing` enabled, messages which do not match any source address will be discarded with an ERROR message, similar to community string mismatches.

See [Edit->Add->Agent....](#)

Value Space Commands

• **mimic value pos object**

Enable MIB browsing of the MIB on the current agent. Set the current MIB position within the MIB hierarchy on the agent. The specified object can be any within the MIB being simulated on the agent instance. This command is similar to the `cd` or `chdir` operating system commands to traverse filesystem hierarchies. See [Examples](#).

• **mimic value list**

Enable MIB browsing of the MIB on the current agent. Display the MIB objects below the current position (set by the previous command). This command is similar to the `ls` or `dir` operating system commands to list filesystem directories. See [Examples](#).

• **mimic value oid object**

Return the numeric OID of the specified object. See [Examples](#).

• **mimic value name oid**

Return the symbolic name of the specified object identifier. See [Examples](#).

• **mimic value mib object**

Return the MIB that defines the specified object. This will only return a MIB name if the object is unmistakably defined in a MIB. See [Examples](#).

• **mimic value info object**

Return the syntactical information for the specified object, such as type, size, range, enumerations, and ACCESS. See [Examples](#).

• **mimic value minfo object+**

Return the syntactical information for the specified objects, such as type, size, range, enumerations, and ACCESS. This is a performance optimization over `mimic value info` when information about a number of objects is requested. See [Examples](#).

- **mimic value instances object**

Display the MIB object instances for the specified object. This enables MIB browsing of the MIB on the current agent. See [Examples](#).

- **mimic value eval object instance**

Evaluate the value of the specified instance `instance` for the specified MIB object `object` and return it as it would through SNMP requests. See [Examples](#).

- **mimic value meval [object instance]+**

Evaluate the values of the specified instance `instance` for each specified MIB object `object` and return it as it would through SNMP requests. See [Examples](#).

- **mimic value variables object instance**

Display the variables for the specified instance `instance` for the specified MIB object `object`. This enables variable browsing of the MIB on the current agent. See [Examples](#).

- **mimic value split object instance**

Split the numerical OID into the object OID and instance OID. This is useful if you have an OID which is a combination of `object` and `instance`. See [Examples](#).

- **mimic value get object instance variable**

Get a variable in the Value Space. Same as [Agent->Get Value....](#) See [Examples](#).

- **mimic value set object instance variable value**

Set a variable in the Value Space. Same as [Agent->Value Space....](#) See [Examples](#).

NOTE: to set a binary string value, specify a string starting with `\x` followed by pairs of hexadecimal digits, eg. "`\x 01 23 45`".

This command also assigns [SNMP PDU action scripts](#) for GET* and SET requests on a MIB object. The instance parameter must be 0. The following variables enable actions:

- **g**
The specified TCL script will be run on GET or GETNEXT requests. It has to exist under the simulation directory.
- **s**
The specified script will be run on SET requests. It has to exist under the simulation directory.

This command also controls advanced [trap generation](#) functionality. The following variables control trap generation:

- **r, tu, c**
These variables together represent the rate settings for the trap. `r` and `tu` is the actual per second rate and `c` represents the total duration in seconds for which the trap is sent. As soon as the `c` variable is set, the trap generation begins, for this reason it should be the **last** variable set for a particular trap.

e.g : The code below implies 3 traps per 10 seconds for 60 seconds.

```
set trap_oid "linkUp"
mimic value set $trap_oid 0 r 3
mimic value set $trap_oid 0 tu 10
mimic value set $trap_oid 0 c 60
```

The default generation of traps is "asynchronous": you schedule traps at a specific rate for a specific period, and they happen in the background, asynchronously. MIMIC also allows generation of "synchronous" traps, ie. individual traps to be sent immediately. This is accomplished by setting the `c` variable to contain the letter `s`. The other 2 variables are ignored in this case, and can be set to anything. The client can verify to make sure the trap got sent by polling the `c` variable for the value 0.

e.g : The code below sends a single synchronous linkUp trap and does not continue until it is actually sent:

```
set trap_oid "linkUp"
mimic value set $trap_oid 0 r 1
mimic value set $trap_oid 0 tu 1
mimic value set $trap_oid 0 c s
# wait until it is actually sent
while {[mimic value get $trap_oid 0 c] != "0"} {
    after 10
}
```

The following variables have to be set before setting the `c` variable to modify the behavior of the generated trap(s).

- **OBJECT**
An object name when used as a variable is looked up during the trap send and the value of that variable is included in the PDU.

e.g : The command below will associate '5' as value of IfIndex to be sent in the linkUp trap PDU generated.

```
mimic value set linkUp 0 ifIndex 5
```

- **OBJECT.i**

This type of variable will be used to assign an optional instance for the specified object in the traps varbind. The value of this variable identifies the index. e.g. The commands below will send ifIndex.2 with a value of 5 in the linkUp trap PDU.

```
mimic value set linkUp 0 ifIndex 5
mimic value set linkUp 0 ifIndex.i 2
```

- **i**

This variable is used to specify any extra version specific information to the trap generation code. Here is what it can be used to represent for various SNMP versions :

```
SNMPv1 : [community_string][,[enterprise][,agent_addr]]
SNMPv2c : community_string
SNMPv2 : source_party,destination_party,context
SNMPv3 : user_name,context
```

e.g. The command below will associate "mycommunity" as community and "1.3.6.1.4.1.1000.1" as the enterprise

```
mimic value set linkUp 0 i "mycommunity,1.3.6.1.4.1.1000.1"
```

- **v**

This variable lets the user override the version of the PDU being generated. The possible values are - "1", "2c", "2" and "3".

e.g. The command below forces the linkUp trap to be generated as a v2c.

```
mimic value set linkUp 0 v 2c
```

- **o**

This variable is used for traps that need extra variables to be added to the PDU along with the ones defined in the MIB as its variables. This lets the user force extra objects (along with instances if needed). All variables to be sent need to be assigned to the o variable.

E.g. the commands below force an ifDescr object with the linkUp trap being sent out with instance as 7 and value as "MyInterface".

```
mimic value set linkUp 0 o "ifDescr"
mimic value set linkUp 0 ifDescr "MyInterface"
mimic value set linkUp 0 ifDescr.i 7
```

The commands below force the ifDescr object as above and the ifType object with instance 7 and value "6".

```
mimic value set linkUp 0 o "ifDescr ifType"
mimic value set linkUp 0 ifDescr "MyInterface"
mimic value set linkUp 0 ifDescr.i 7
mimic value set linkUp 0 ifType "6"
mimic value set linkUp 0 ifType.i 7
```

NOTE: if multiple additional varbinds referring to the same object need to return different instances and values, they have to be specified in the form [INDEX]OBJECT, where INDEX is a small integer, and OBJECT is the name or OID of the object. Eg. the commands below send 3 additional varbinds for the ifAdminStatus object with the linkDown trap:

```
set linkDown 1.3.6.1.6.3.1.1.5.3 1.3.6.1.6.3.1.1.5.3
mimic value set $linkDown 0 o {[1]ifAdminStatus [2]ifAdminStatus [3]ifAdminStatus}
mimic value set $linkDown 0 {[1]ifAdminStatus.i} 11
mimic value set $linkDown 0 {[1]ifAdminStatus.i} 1
mimic value set $linkDown 0 {[2]ifAdminStatus.i} 12
mimic value set $linkDown 0 {[2]ifAdminStatus.i} 2
mimic value set $linkDown 0 {[3]ifAdminStatus.i} 13
mimic value set $linkDown 0 {[3]ifAdminStatus.i} 3
```

- **O**

To omit any variables which are defined in the MIB you can use the o (capital o) variable. This needs to be set to the list of OIDs of the variable bindings in the order defined in the MIB.

E.g. the commands below omit the ifIndex object from the linkUp trap.

```
mimic value set linkUp 0 O "1.3.6.1.2.1.2.2.1.1"
```

- **ip**

The variable ip is used for generating the trap from the N-th IP alias address.

E.g. the command below generates the linkUp trap from the 2-nd IP alias.

```
mimic value set linkUp 0 ip 2
```

- **a**

This variable associates an action script to the trap or INFORM request. The action script specified in the value of this variable has to exist in the simulation directory. It will be executed before each instance of the trap is sent out.

e.g. The command below causes the linkUp.mtcl action script to be executed for each linkUp trap being sent out.

```
mimic value set linkUp 0 a "linkUp.mtcl"
```

- **I**

This optional variable controls the generation of INFORM PDUs. An INFORM is sent only if the variable is non-zero, else a TRAP is generated.

e.g. The command below causes the linkUp even to be sent as an INFORM.

```
mimic value set linkUp 0 I 1
```

- **R, T, E**

This variable associates an action script to the INFORM request. The action script specified in the value of this variable has to exist in the simulation directory. The action script associated with the **R** variable will be executed on receiving a INFORM RESPONSE, the one associated with the **T** variable on a timeout (ie. no response),

the one associated with the **E** variable on a report PDU.

e.g. The command below causes the linkUp-response.mtcl action script to be executed for each linkUp INFORM RESPONSE received.

```
mimic value set linkUp 0 R "linkUp-response.mtcl"
```

- **eid.IP-ADDRESS.PORT**

control variable allows to configure message authoritative engine id for the destination specified by IP-ADDRESS and optionally by PORT.

e.g. to setup the engine ID for destination 10.10.1.1, you would use

```
mimic value set linkUp 0 eid.10.10.1.1 "\\x80 00 00 00 20 40 ab cd ef"
```

or if you had multiple trap receivers at that IP address, and you wanted to further discriminate by port

```
mimic value set linkUp 0 eid.10.10.1.1.9999 "\\x50 FF 20 40 ab cd ef"
```

- **eb.IP-ADDRESS.PORT**

control variable allows to configure message authoritative engine boots.

- **et.IP-ADDRESS.PORT**

control variable allows to configure message authoritative engine time.

- **mimic value mget [object instance variable]+**

Get multiple variables in the Value Space. This is a performance optimization of the `mimic value get` command, to be used when many variables are requested.

- **mimic value mset [object instance variable value]+**

Set multiple variables in the Value Space. This is a performance optimization of the `mimic value set` command, to be used when many variables are to be set.

- **mimic value unset object instance variable**

Unset a variable in the Value Space in order to free its memory. Only variables that have previously been set can be unset.

- **mimic value munset [object instance variable]+**

Unset multiple variables in the Value Space. This is a performance optimization of the `mimic value unset` command, to be used when many variables are to be unset.

- **mimic value add object instance+**

Add an entry to a table. The `object` needs to specify the MIB object with the INDEX clause, usually an object whose name ends with `Entry`. See [Examples](#).

- **mimic value remove object instance+**

Remove an entry from a table. The `object` needs to specify the MIB object with the INDEX clause, usually an object whose name ends with `Entry`. See [Examples](#).

- **mimic value state get object**
mimic value state set object 0 | 1

Get/set the state of a MIB object `object`. To disable traversal into a MIB object and any subtree underneath, set the state to 0, else set the state to 1. By default, traversal is enabled into all MIB objects. See [Examples](#).

Variable Store Commands

The Variable Store sits in the middle of the MIMIC storage hierarchy. Whereas the fast [Value Space](#) is directly related to simulations and MIB objects, and only accessible while agents are running; and whereas slower files can be used for larger storage, the Variable Store can efficiently store/retrieve variables, either global or per-agent. It can be used at any time that MIMIC is running. The variables in the Variable Store are analogous to "environment variables" in shells. Global variable store variables are accessible everywhere, but per-agent variables are only accessible for the currently assigned agent (see [mimic agent assign](#)).

Variables can be either persistent or non-persistent. Non-persistent variables are forgotten on simulator termination. Persistent variables are stored across invocations of the simulator, i.e. they are persistent even after termination of the simulator. Global persistent variables are automatically stored upon simulator termination, but the lab configuration has to be saved in order to store per-agent persistent variables.

- **mimic [agent] store set var value [persist]**
mimic [agent] store append var value [persist]

These commands allow the creation of a new variable, or changing an existing value. The `append` sub-command will append the value to an existing variable, or create a new one. The `set` sub-command will overwrite an existing variable, or create a new one. The optional `persist` flag can be used to indicate if the variable is to be persistent as described [above](#). By default a value of '0' will be implied for the `persist` flag. To avoid mistakes, for existing variables the `persist` flag can only be set. If you want to reset it, you first need to unset the variable. See [Examples](#).

- `mimic [agent] store lreplace var index value`
`mimic [agent] store mlreplace [var index value]+`

These commands treat the variable as a list, and allow to replace an entry in the list at the specified index with the specified value. The variable has to already exist. See [Examples](#).

- `mimic [agent] store unset var`

Deletes a variable which is currently defined. This will cleanup persistent variables if needed. See [Examples](#).

- `mimic [agent] store get var`

Fetches the value associated with a variable. The value will be returned as a string (like all Tcl values). See [Examples](#).

- `mimic [agent] store mget [var]+`

Fetches the value associated with a list of variables. The value will be returned as a list of strings (like all Tcl values). See [Examples](#).

- `mimic [agent] store exists var`

This command can be used as a predicate to ascertain the existence of a given variable. It returns "1" if the variable exists, else "0". See [Examples](#).

- `mimic [agent] store persists var`

This command can be used as a predicate to ascertain the persistence of a given variable. It returns "1" if the variable is persistent, else "0". See [Examples](#).

- `mimic [agent] store list`

This command will return the list of variables in the said scope. The list will be a Tcl format list with curly braces "{}" around each list element. These elements in turn are space separated. See [Examples](#).

- `mimic agent store copy other`

This command copies the variable store from the other agent to this agent.

Timer Script Configuration Commands

Timer scripts provide an alternative to MIMICShell based poll scripts (which sit in a loop, sleeping and waking up periodically). Typically these are required when the user needs to repeat a task at various time intervals (eg. change various interface utilizations based on the time of the day). Another application of timer scripts is an action that is performed some fixed time after an event.

Conceptually, rather than doing a busy-wait loop like

```
while { 1 } {
    # STEP 1: do some periodic task

    # STEP 2: optionally re-adjust the period, eg. $milliseconds

    # if done, break
    if { $milliseconds == -1 } {
        break
    }

    # delay for specific milliseconds, eg. $milliseconds
    after $milliseconds
}
```

you would schedule a timer script that does STEP 1 and STEP 2. Notice that the busy-wait loop performs the task immediately, then periodically thereafter, and timer scripts are designed to do exactly this.

Since timer scripts run within the simulator, unlike MIMICShell scripts, they execute much faster. Also there is no limit on the number of timer scripts that can be configured at the same time (although a large number of timer scripts will degrade the performance of the simulations).

A global timer script can be added using either the `mimic timer script add` command or by adding an entry to `timerscripts.cfg` in the config directory before starting the MIMIC daemon. Per- agent timer scripts can be manipulated with the equivalent `mimic agent timer script` commands.

When a new timerscript is added the user can associate a timer-interval with it (in msec). This is the default interval after which the script will be invoked. (NOTE: Only one timer script is executed at a given point of time so scripts that take a long time to complete will hinder the execution of other scripts).

All timer scripts are invoked immediately, and periodically thereafter. You can use state variables (a global variable, MIMIC store variable, etc) to determine first immediate invocation vs. successive invocations. The sample timer scripts in the `scripts/` folder illustrate this.

If the timer interval is -1 then the script gets invoked only once. This interval can be changed from within the script itself when it is being executed. The current value of interval is passed to the script as a global variable `gNextInterval`. The script can change this variable to some other interval. If it is set to -1 then the script is not executed again. The script can also be stopped by deleting it using the `mimic timer script delete` command.

In addition to the global variables

- gCurrentAgent
- gSharedDir
- gOwner
- gPrivateDir

documented for [SNMP Request actions](#), timer scripts receive these variables:

- **gNextInterval** (IN/OUT): This variable contains the time interval (in msec) that the script will be executed next. If the script sets it to -1, then this timer script will be deleted upon completion.
- **gArguments** (IN): any optional arguments to this timer script invocation.

NOTE: timer scripts have the following limitations:

- timer scripts run in a limited number of threads in the MIMICD daemon. Each thread handles expired timer scripts in round-robin fashion. Thus, if a timer script is slow, then all other scripts may be affected (slowed down). Thus, your timer script should be efficient. In particular, you should never `sleep` in your timer script. Also, polling loops are discouraged. User interaction is not possible.
- although you can `exec` other programs from timer scripts, the execution of external programs slows down the script. Find a way to use alternatives without the use of `exec`.

The timer script commands are

- **mimic [agent] timer script list**

List the timer scripts currently running along with the their intervals. The command `mimic timer script list` lists global timer scripts, the command `mimic agent timer script list` is the per-agent equivalent.

NOTE: Global timer scripts run globally but within them you can address individual agents using 'mimic agent assign'. To schedule timerscripts for an individual agent, use `mimic agent timer script`.

Each timer script is identified with the script name and an interval at which the timer script is to be invoked. The timer scripts are expected to be located in the scripts directory. See [Run->Timer Scripts...](#)

- **mimic [agent] timer script add script interval [arg]**

Add a new timer script to be executed at specified interval (in msec) with the specified argument.

- **mimic [agent] timer script delete script [interval] [arg]**

Remove a timer script from the execution list. The first scheduled script that matches the `script` name, and optionally the `interval` and `argument` will be deleted.

Trap Configuration Commands

- **mimic trap config list**

List the global set of trap destinations effective for any agent instances which do not have their own trap destinations.

Each trap destination is identified with an IP address and a port number. The default port number is the standard SNMP trap port 162. See [Edit->Trap Destinations...](#)

- **mimic trap config add destination port**

Add a trap destination to the global set of destinations.

- **mimic trap config delete destination port**

Remove a trap destination from the global set of destinations.

SNMPv3 Configuration Commands

The following SNMPv3 command set is available to the user allowing the manipulation of security settings and access to the engine configuration.

- **SNMPv3 Configuration**

- **mimic agent protocol msg snmpv3 get config**

Returns the SNMPv3 configuration. Eg.

```
mimicsh> mimic agent protocol msg snmpv3 get config
{engine_id=} {context_engine_id=} {usm_db=v3usm.conf} {vacm_db=v3vacm.conf}
```

- **mimic agent protocol msg snmpv3 set config [config]**

Changes the SNMPv3 configuration. Eg.

```
mimicsh> mimic agent protocol msg snmpv3 get config
{engine_id=} {context_engine_id=} {usm_db=v3usm.conf} {vacm_db=v3vacm.conf}
mimicsh> mimic agent protocol msg snmpv3 set config engine_id=12345678

mimicsh> mimic agent protocol msg snmpv3 get config
{engine_id=12345678} {context_engine_id=} {usm_db=v3usm.conf} {vacm_db=v3vacm.conf}
```

- Engine Parameters

`mimic agent protocol msg snmpv3 get engineid`

For started agents, retrieves the current engineID in use by the snmpv3 module. For stopped agents, this operation is meaningless. If not explicitly set by the user then the autogenerated engineID is returned. The format of the engineID is in the familiar hex format, eg. \x01 23 45 67 89...

`mimic agent protocol msg snmpv3 get engineboots`

Retrieves the number of times the agent has been restarted.

`mimic agent protocol msg snmpv3 get enginetime`

Retrieves the time in seconds for which the agent has been running.

`mimic agent protocol msg snmpv3 get context_engineid`

Retrieves the contextEngineID for the agent instance.

- User Configuration

`mimic agent protocol msg snmpv3 user list`

Returns the current user entries as a Tcl list.

`mimic agent protocol msg snmpv3 user add userName securityName authProtocol authKey privProtocol privKey`

Adds a new user entry with the specified parameters.

`mimic agent protocol msg snmpv3 user del userName`

Deletes the specified user entry.

`mimic agent protocol msg snmpv3 user clear`

Clears all user entries.

- Group Configuration

`mimic agent protocol msg snmpv3 group list`

Returns the current group entries as a Tcl list.

`mimic agent protocol msg snmpv3 group add groupName securityModel securityName`

Adds a new group entry with the specified parameters.

`mimic agent protocol msg snmpv3 group del groupName`

Deletes the specified group entry.

`mimic agent protocol msg snmpv3 group clear`

Clears all group entries.

- **SNMPv3 Access Configuration**

`mimic agent protocol msg snmpv3 access list`

Returns the current access entries as a Tcl list.

`mimic agent protocol msg snmpv3 access add groupName prefix securityModel securityLevel contextMatch readView writeView notifyView`

Adds a new access entry with the specified parameters.

`mimic agent protocol msg snmpv3 access del groupName prefix`

Deletes the specified access entry.

`mimic agent protocol msg snmpv3 access clear`

Clears all access entries.

- **View Configuration**

`mimic agent protocol msg snmpv3 view list`

Returns the current view entries as a Tcl list.

`mimic agent protocol msg snmpv3 view add viewName viewType subtree mask`

Adds a new view entry with the specified parameters.

`mimic agent protocol msg snmpv3 view del viewName`

Deletes the specified view entry.

`mimic agent protocol msg snmpv3 view clear`

Clears all view entries.

- **USM Configuration File**

`mimic agent protocol msg snmpv3 usm save`

Saves current user settings in the currently loaded USM config file.

`mimic agent protocol msg snmpv3 usm saveas filename`

Saves current user settings in the specified USM config file.

- **VACM Configuration File**

`mimic agent protocol msg snmpv3 vacm save`

Saves current group, access, view settings in the currently loaded VACM config file.

`mimic agent protocol msg snmpv3 vacm saveas filename`

Saves current group, access, view settings in the specified VACM config file.

- **Access Control Commands**

The following access control command set is available to the administrator to configure agent access control.

- `mimic access get adminuser`

Returns the current administrator. If nothing is specified in admin/settings.cfg then returns "".

- **mimic access get adminidir**

Returns the current admin directory. If nothing is specified in admin/settings.cfg then returns "". If no admin directory is specified then the shared area will be used where needed (e.g. for persistent info, access control data files etc.)

- **mimic access get acldb**

Returns the current access control database in use. If nothing is specified then this returns "".

- **mimic access get enabled**

Returns the state of access control checking. 0 indicates that it is disabled, 1 indicates it is enabled.

- **mimic access set acldb**

Allows setting the name of the current access control database. This will be used for subsequent load and save operations.

- **mimic access set enabled**

Allows the user to enable/disable the access control check. 0 indicates disabled, 1 indicates enabled.

- **mimic access load [file]**

Loads the specified file for access control data. If filename is not specified then the currently set 'acldb' parameter is used.

- **mimic access save [file]**

Saves current access control data in specified file. If filename is not specified then the currently set 'acldb' parameter is used.

- **mimic access list**

Returns a Tcl list of lists. Each entry consists of user, agents (in minimal range representation) and access mask (not used currently).

- **mimic access add user agents [mask]**

Adds/Overwrites the user entry in the access control database.

- **mimic access del user**

Clears a users entry from access control database. Using '*' for user clears all the users.

Arithmetic commands

This is an extension of the Tcl `expr` command to perform unsigned long and 64-bit arithmetic.

- **gci expr unsigned expression**

- **gci expr 64bit expression**

where `expression` is of the form

`operand1 operator operand2`

where `operand1` and `operand2` are numbers and `operator` is one of

- +
- -
- *
- /
- ==
- !=
- >
- >=
- <
- <=

Examples

This section lists a sample session. The comment lines begin with puts and can be omitted when you try the examples yourself.

```
% mimicsh
mimicsh> puts "----- COMMENT: global commands -----"
----- COMMENT:global commands-----
mimicsh> puts "COMMENT: we are running MIMIC Enterprise"
COMMENT: we are running MIMIC Enterprise
mimicsh> mimic get max
1000
mimicsh> set info [mimic mget max last version clients]
{1000} {841} {10.20} {3}
mimicsh> set maxagt [lindex $info 0]
1000
mimicsh> set lastagt [lindex $info 1]
841
mimicsh> set myversion [lindex $info 2]
10.20
mimicsh> set numclient [lindex $info 3]
3
mimicsh> puts "COMMENT: we are running $lastagt agents out of $maxagt on MIMIC $myversion and
$numclient clients are connected"
COMMENT: we are running 841 agents out of 1000 on MIMIC 10.20 and 3 clients are connected
mimicsh> puts "----- COMMENT: agent commands -----"
----- COMMENT:agent commands-----
mimicsh> set max_agents [mimic get max]
1000
mimicsh> puts $max_agents
1000
mimicsh> puts "COMMENT: but only 5 agents configured"
COMMENT: but only 5 agents configured
mimicsh> set last_agent [mimic get last]
5
mimicsh> for {set i 1} {$i <= $last_agent} {incr i} { mimic agent assign $i; puts "Agent $i:
[mimic agent get host]"}
Agent 1: 192.9.201.133
Agent 2: 192.9.201.150
Agent 3: 192.9.201.180
Agent 4: 192.9.201.181
Agent 5: 192.9.201.182
mimicsh> for {set i 1} {$i <= $last_agent} {incr i} { mimic agent assign $i; puts "Agent $i:
[mimic agent get state]"}
Agent 1: 2
Agent 2: 2
Agent 3: 2
Agent 4: 2
Agent 5: 2
mimicsh> mimic agent assign 1
mimicsh> mimic agent start
mimicsh> puts "COMMENT: it takes a while for the agent to start..."
COMMENT: it takes a while for the agent to start...
mimicsh> mimic agent get state
6
mimicsh> mimic agent get state
6
mimicsh> mimic agent get state
1
mimicsh> puts "COMMENT: duplicate agent 1"
COMMENT: duplicate agent 1
mimicsh> set mibs [mimic agent get mibs]
cisco-5k.random,RFC1213-MIB,1 cisco-5k.random,IF-MIB,1 cisco-5k.random,EtherLike-MIB,1
cisco-5k.random,FDDI-SMT73-MIB,1 cisco-5k.random,RMON-MIB,1 cisco-5k.random,BRIDGE-MIB,1
cisco-5k.random,cisco/CISCO-STACK-MIB,1 cisco-5k.random,cisco/CISCO-CDP-MIB,1
cisco-5k.random,cisco/CISCO-VTP-MIB,1
mimicsh> puts "COMMENT: concat removes extra white-space"
COMMENT: concat removes extra white-space
mimicsh> set mibs [concat [eval list $mibs]]
mimicsh> mimic agent assign [expr [mimic get last] + 1]
mimicsh> mimic agent get state
5
mimicsh> mimic agent add 192.9.201.6 $mibs
mimicsh> mimic agent get state
2
mimicsh> puts "COMMENT: since agent 1 has been running, check its stats"
COMMENT: since agent 1 has been running, check its stats
mimicsh> mimic agent assign 1
mimicsh> mimic agent get statistics
427 1 0 0 426 0 0 0 0 426 0 0 0 0 0 0
mimicsh> puts "----- COMMENT: value space commands -----"
----- COMMENT:value space commands-----
mimicsh> mimic value list
directory mgmt experimental private
mimicsh> mimic value pos mgmt
iso.org.dod.internet.mgmt.
mimicsh> mimic value list
mib-2
mimicsh> mimic value pos mib-2
iso.org.dod.internet.mgmt.mib-2.
mimicsh> mimic value list
system interfaces at ip icmp tcp udp egg transmission snmp ifMIB
mimicsh> mimic value pos ifEntry
iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.
mimicsh> mimic value list
ifIndex ifDescr ifType ifMtu ifSpeed ifPhysAddress ifAdminStatus ifOperStatus ifLastChange
ifInOctets ifInUcastPkts ifInNUcastPkts ifInDiscards ifInErrors ifInUnknownProtos ifOutOctets
ifOutUcastPkts ifOutNUcastPkts ifOutDiscards ifOutErrors ifOutQLen ifSpecific
```

```

mimicsh> mimic value oid ifType
1.3.6.1.2.1.2.2.1.3
mimicsh> mimic value name 1.3.6.1.2.1.2.2.1.3
ifType
mimicsh> mimic value mib 1.3.6.1.2.1.2.2.1.3
IF-MIB
mimicsh> mimic value mib 1.3.6.1.2.1
{Cannot find mib for object}
mimicsh> mimic value mib 1.3.6.1.2.1.2
IF-MIB
mimicsh> mimic value info ifAdminStatus
readwrite leaf integer enum 3 1 up 2 down 3 testing enabled
mimicsh> mimic value minfo ifAdminStatus ifDescr
{readwrite leaf integer enum 3 1 up 2 down 3 testing enabled} {readonly leaf octetstring
size range 0 255 enabled}
mimicsh> mimic value instances sysDescr
0
mimicsh> mimic value eval sysDescr 0
HP LanProbe III Ver. A.01.00, (Boot ROM A.00.01 4M)
mimicsh> mimic value meval sysDescr 0 ifSpeed 2
{HP LanProbe III Ver. A.01.00, (Boot ROM A.00.01 4M)} {1000000}
mimicsh> mimic value instances ifSpeed
1 2 3 4
mimicsh> mimic value variables ifSpeed 2
v
mimicsh> mimic value get ifSpeed 2 v
1000000
mimicsh> mimic value set ifSpeed 2 v 5000
0
mimicsh> mimic value get ifSpeed 2 v
5000
mimicsh> mimic value unset ifSpeed 2 v
0
mimicsh> mimic value get ifSpeed 2 v
1000000
mimicsh> mimic value add ifEntry 7
0
mimicsh> mimic value instances ifSpeed
1 2 3 4 7
mimicsh> mimic value remove ifEntry 2
0
mimicsh> mimic value instances ifSpeed
1 3 4 7
mimicsh> mimic value get ifType 7 v
6
mimicsh> set split [mimic value split 1.3.6.1.2.1.2.2.1.3.7]
1.3.6.1.2.1.2.2.1.3 7
mimicsh> mimic value get [lindex $split 0] [lindex $split 1] v
6
mimicsh> mimic value set ifType 7 v 39
mimicsh> mimic value get [lindex $split 0] [lindex $split 1] v
39
mimicsh> puts "---- enable/disable MIB tree --"
mimicsh> mimic agent assign 1

mimicsh> mimic value state get ifTable
enabled

mimicsh> # press CTL-Z (if your shell has job control, otherwise
mimicsh> # invoke in another terminal) to query the agent
Suspended
% snmpget 192.9.201.133 public .1.3.6.1.2.1.2.2.1.1.1
interfaces.ifTable.ifEntry.ifIndex.1 = 1

% snmpget 192.9.201.133 public .1.3.6.1.2.1.2.2.1.2.1
interfaces.ifTable.ifEntry.ifDescr.1 = sc0

% fg # to resume the MIMICShell
mimicsh> mimic value state set ifTable 0

mimicsh> mimic value state get ifTable
disabled

mimicsh> # press CTL-Z (if your shell has job control, otherwise
mimicsh> # invoke in another terminal) to query the agent
Suspended
% snmpget 192.9.201.133 public .1.3.6.1.2.1.2.2.1.1.1
Error in packet
Reason: (noSuchName) There is no such variable name in this MIB.
This name doesn't exist: interfaces.ifTable.ifEntry.ifIndex.1

% snmpget 192.9.201.133 public .1.3.6.1.2.1.2.2.1.2.1
Error in packet
Reason: (noSuchName) There is no such variable name in this MIB.
This name doesn't exist: interfaces.ifTable.ifEntry.ifDescr.1

% fg # to resume the MIMICShell
mimicsh>puts "----- variablestore examples"
----- variablestore examples
mimicsh> mimic store list

mimicsh> mimic store set last_agent [mimic get last]

mimicsh> mimic store list
{last_agent}

```

```

mimicsh> mimic store get last_agent
5
mimicsh> mimic store exists last_agent
1
mimicsh> mimic store unset last_agent

mimicsh> mimic store exists last_agent
0
mimicsh> mimic store list

mimicsh> mimic store set A 1

mimicsh> mimic store get A
1
mimicsh> mimic store mget A
{1}
mimicsh> mimic store lreplace A 3 4

mimicsh> mimic store get A
1 {} {} 4
mimicsh> mimic store lreplace A 2 3

mimicsh> mimic store get A
1 {} 3 4
mimicsh> mimic store lreplace A 2 5

mimicsh> mimic store get A
1 {} 5 4
mimicsh> mimic store set A "1"

mimicsh> mimic store set B "10"

mimicsh> mimic store mget A B
{1} {10}
mimicsh> mimic store mlreplace A 3 4 B 4 50

mimicsh> mimic store mget A B
{{1} {} {} {4} } {{10} {} {} {} {50} }
mimicsh> set A [lindex [mimic store mget A B] 0]
{1} {} {} {4}
mimicsh> llength $A
4
mimicsh> mimic store mlreplace A 2 "hello how are you" B 3 "I am fine"

mimicsh> mimic store mget A B
{{1} {} {hello how are you} {4} } {{10} {} {} {} {I am fine} {50} }
mimicsh> mimic agent assign 1

mimicsh> mimic agent store list

mimicsh> mimic agent store set ipaddr [mimic agent get host]

mimicsh> mimic agent store list
{ipaddr}
mimicsh> mimic agent store get ipaddr
192.9.201.133
mimicsh> mimic agent store exists ipaddr
1
mimicsh> mimic agent store unset ipaddr

mimicsh> mimic agent store exists ipaddr
0
mimicsh> mimic agent store list

mimicsh>puts "----- catchexamples"
----- catchexamples
mimicsh> mimic get return
nocatch
mimicsh> mimic agent assign 2
0
mimicsh> mimic agent start
0
mimicsh> mimic agent start
mimic agent: Agent 2 is already running
1
mimicsh> proc a {} {mimic agent start; puts hello}
mimicsh> a
mimic agent: Agent 2 is already running
hello
mimicsh> puts "NOTE: the hello statement is executed"
NOTE: the hello statement is executed
mimicsh> mimic set return catch
0
mimicsh> catch {mimic agent start} msg
1
mimicsh> puts $msg
mimic agent: Agent 2 is already running

mimicsh> a
mimic agent: Agent 2 is already running
mimicsh> puts "NOTE: the hello statement is NOT executed"
NOTE: the hello statement is NOT executed
mimicsh> exit

```

21. Access Control

MIMIC is a multi-user simulation environment, and as such provides access control mechanisms to enforce access to the shared resources it provides.

Client Access Control

MIMIC allows access control to limit clients on remote hosts that are allowed to control your simulation. By default, and for backward compatibility, access control is disabled, allowing clients on any remote host to control MIMIC. To enable access control, add any number of lines to the `config/mimicd.cfg` configuration file of the form

```
remote_host = HOST
where HOST is a host address specification (FQDN). To allow no remote host to control MIMIC (only clients on the local host), add a line

remote_host = NONE
```

Agent Access Control

Before MIMIC 5.20 the Simulator did not enforce any user-based access control over agent operations such as addition/deletion/configuration, allowing any user to change any agent instance at will.

This works fine in environments where complete cooperation among users exists. For environments where such cooperation is not possible, this allows for inadvertent (or deliberate) interference. In "production" simulation scenarios, lack of access control is undesirable.

At version 5.20, agent access control assigns each agent instance an owner (or group of owners) who has exclusive access rights to change it. This allows to partition the set of agent instances into mutually exclusive sets of resources.

A special user, called the "MIMIC administrator", can change access control at run-time with the [MIMICView Access Menu](#) or [MIMICShell Access Control commands](#). By default, access control is disabled and the user who installed MIMIC is the administrator.

To enable access control, specify the administrative settings at installation time. After installation, you can specify administrative settings by creating/editing the `config/admin/settings.cfg` file in your installation (shared) area. For example, assuming that MIMIC was installed in `/usr/local/mimic`, and you want to make the user `mimicadmin` the administrative user, it should contain

```
version = 5.20
admin = {
  user = mimicadmin
  dir = /usr/local/mimic.admin
}
```

After that, the administrative user `mimicadmin` can use the [Access Menu](#) in MIMICView to configure access control parameters.

22. Advanced Configuration

Normally you don't need to reconfigure the Simulator. But, sometimes users have special demands on MIMIC, that have to be optimized using extra configuration.

Configurable Number of Management Channels

MIMIC allows to configure the number of management channels for MIMIC clients from the default of 20. To change this limit, add a line to the `config/mimicd.cfg` configuration file of the form

```
max_mgmt_connections = N
where N the number of channels you want to allow. Each MIMIC client invocation (MIMICView GUI front-end, Tcl-based MIMICShell, Java, C++ or Perl client) will normally use up one or more of these channels. MIMICView uses one channel per invocation. The other programmable clients will use one channel per open session.
```

23. Updates

MIMIC is designed to be frequently updated with the minimal amount of pain. Normally, an update from an old version to a new version is transparent, not requiring any action from the user. This means that your data saved by the old version will be read by either old or new version of MIMIC, and your simulations will work identically in either version (barring fixed bugs).

Only rarely will there be an update procedure visible to the user. If so, the update procedure will be invoked on the first invocation of MIMICView in the new version, informing the user and prompting to complete the update. Below are recent update procedures.

Update to 4.10

The simulation data file hierarchy has changed in version 4.10 to handle special cases for enterprise-specific MIBs. The MIB directory name for enterprise-specific MIBs is now "enterprise"- "MIB". The update step which is invoked the first time you run MIMICView converts your simulation areas to this new naming scheme.

Update to 5.10

With MIMIC v5.10, lab configuration files live in `config/agents/`. This update moves existing lab configuration files from `config/` to `config/agents/`.

NOTE: after this update, older versions of MIMIC will not run with these lab configuration files. If you have to downgrade, you can move the files in `config/agents/` back to `config/`.

24. Usage Profiling

MIMIC is a complex product consisting of many subcomponents and internal subsystems. Usage profiling lets the Gambit support team empirically determine which subsystems are the most used, and thus the most important. This will let Gambit improve the product, by making better decisions as to which areas benefit most from improvement, or gauging as to how many customers are affected by proposed changes.

Profiling information is constantly collected by MIMIC. This collection is transparent to the end-user. The collected data is deposited in the `config/profile/` directory in the private data area. Nothing in the profiling data identifies a particular user or site. The only identifying mark is a timestamp in the ID file, so that multiple profiles from the same site can be aggregated correctly. The timestamp is generated the first time you run MIMIC, and is virtually unique among all users. All files are in ASCII format, ready to be inspected.

This profiling data can optionally be reported to Gambit every so often (we recommend an interval of 30 days). If your MIMIC host is not connected to the Internet, then you should [turn profile reporting off](#).

If you have enabled it (either when prompted or from the [Configuration Wizard](#), the reporting is done automatically via e-mail to a profile collection account at Gambit, or via HTTP to the profile collection page at the Gambit web site. For e-mail to work, your MIMIC host has to be connected to the Internet, and the mail gateway has to be configured with the [Configuration Wizard](#). The e-mail message inherently has some identifying information (such as the hostname where the e-mail originated), but is discarded by the profile collection software (you'll have to take our word for it). For HTTP transfer to work, your MIMIC host has to be connected to the Internet.

Once the profiling data is reported, it is deleted from your private data area, so that new data can be accumulated.

[<< Previous Section](#) [Next Section >>](#)



MIMIC Recorder Guide

Table of Contents

- [Overview](#)
- [Live Mode](#)
- [File Mode](#)
- [Other Options](#)
- [Creating a Walkfile with mib2walk](#)
- [Walk File Format](#)

Overview **QuickStart**

The MIMIC Recorder lets you easily create starting simulations of actual devices on your network. The starting simulation of a device has the following characteristics:

- MIB objects that are counters are simulated with the average rate detected during the recording. The Recorder has options to create constant or random value simulations.
- All other MIB objects assume the value detected during the recording.

This means that the simulated device will look like a copy of the real one: values returned by the MIMIC Simulator of a simulation created by the Recorder will return exactly the values of all MIB objects discovered, and for MIB objects of type Counter, return values that change at the average rate calculated by the recorder.

The MIMIC Recorder can run in either live or file mode. In live mode, the Recorder interacts with one or more running target devices on your network in real time. In the discovery phase, the Recorder walks the entire MIB (or, optionally, a subset of the MIB) of each target device, and stores the retrieved values. In the second simulation phase, the Recorder creates the simulation from the discovered values.

The file mode works by reading a walkfile created previously either with other commercial or public domain MIB browser utilities (SNMP Walk utilities) or from a previous run of the Recorder against a Live target device.

Once data is captured from a device, the MIMIC Agent Simulator can use the simulation generated by the Recorder to simulate the device realistically. Any variation on this basic simulation can later be done for use with the Simulator. This is accomplished by changing:

- Values for MIB objects, but not the simulation itself. This lets you create variations, such as high traffic rate; modify tables; etc.
- The simulation itself. This entails editing the simulation files and changing simulation expressions.

There are two ways of starting the MIMIC Recorder:

- The advanced way of invoking the Recorder is on the command line. In any command shell, start `mimicrec`, e.g.,
`mimicrec [OPTIONS]`
where [OPTIONS] are the Recorder command line options defined below.
- The simpler way is via the MIMICView GUI, the graphical front-end to all three MIMIC tools: Agent Simulator, Recorder and Compiler. The MIMIC Recorder functions are accessible via the `simulation` menu in MIMICView.

If you haven't done so yet, read the [Recording a Device](#) section in the

Reference

Live Mode **QuickStart**

The recorder can interact with any number of target devices simultaneously. The MIB walk of each target device in the discovery phase is interleaved for maximum performance. Each target is then simulated separately in the simulation phase (there is no performance benefit in handling them together).

The Record Live dialog allows you to start a recording for the specified list of targets.

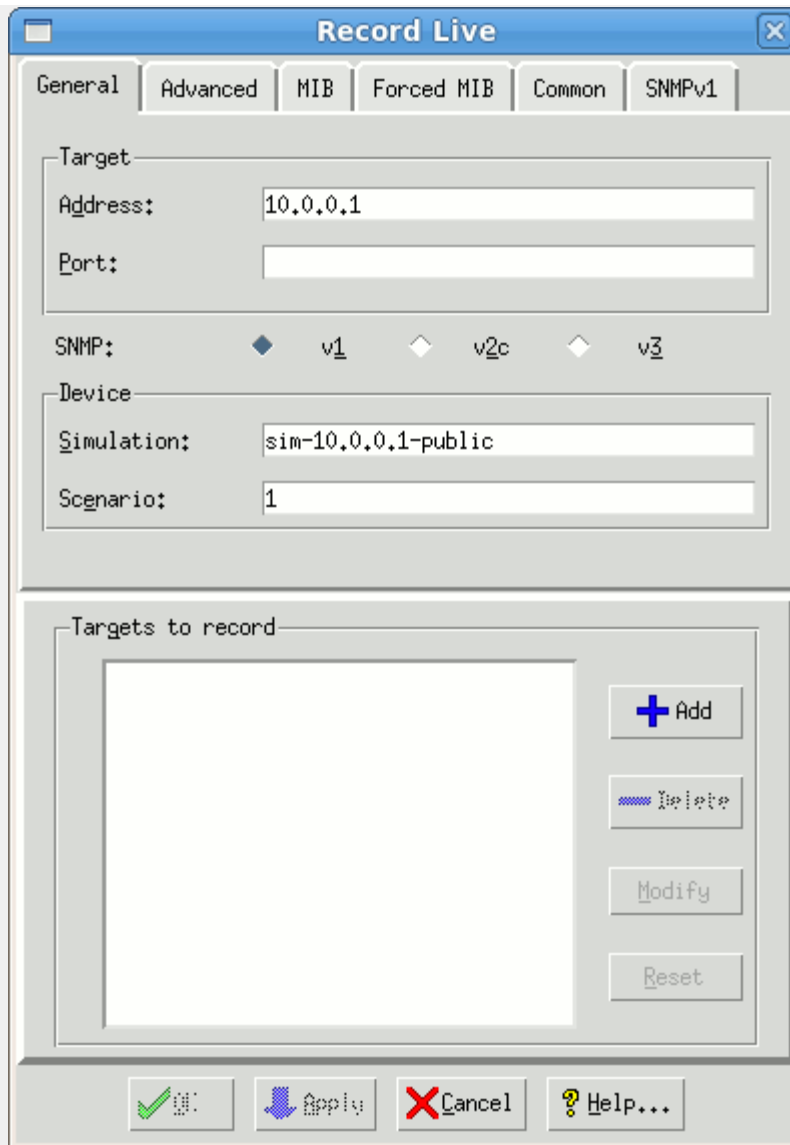


Figure - Dialog for recording a live device

- The **Add** >> button adds a target, with the options defined in the input fields, to the list of targets.
- The **Delete** << button deletes the selected target from the list.
- The **Modify** button modifies the selected target with the options defined in the input fields.
- The **Reset** button clears the input fields.

For each target the following basic options need to be supplied in the **General** tab:

• **Target Address (--target)**

Specifies the target IPv4 address, either as "dot-value" notation (e.g., 192.9.200.1), or as a hostname (e.g., gambit), or fully qualified domainname (e.g., gambit.gambitcomm.com) provided that they can be resolved to an address (via /etc/hosts, Yellow Pages or DNS).

For IPv6 addresses, the standard notation is used, eg. fe80::a00:20ff:fe15:57c. On Windows, Scope IDs need to be specified, as detailed in the [Windows installation section](#).

• **Port (--port)**

The Recorder talks to the standard SNMP port 161 on the target device. If the device is configured to use a non-standard port, you can specify it in this option.

• **Protocol (--version)**

This specifies the version of SNMP to use to access the target device. The possible options are "1", "2c" and "3". The default is SNMPv1.

- **Simulation (--simulation)**

This specifies the simulation name of the simulation to create. The dialog provides a default simulation name, which you can change. Using an already existing simulation causes the old simulation values to be overwritten with the new.

- **Scenario (--scenario)**

This specifies a scenario name. The dialog provides a default scenario name which you can change. Using an already existing scenario causes the old simulation values to be overwritten with the new.

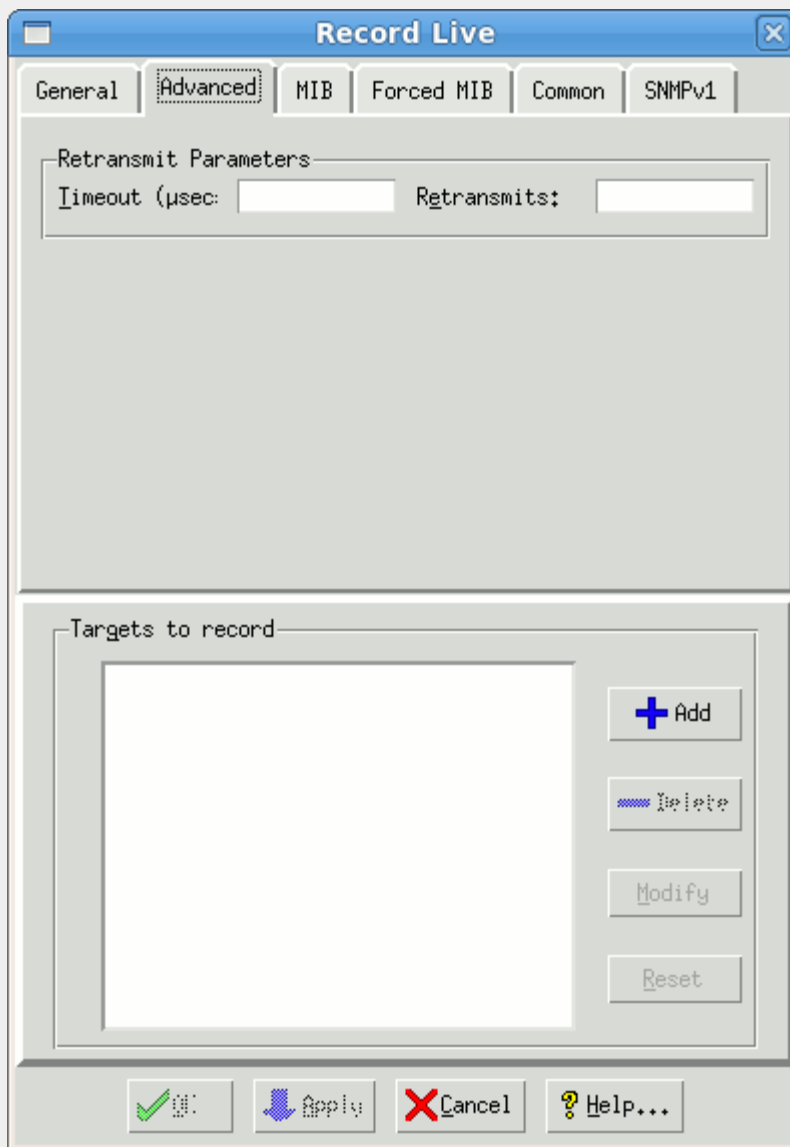


Figure - Advanced Tab

In addition, the following advanced options can be supplied for each selected target in the Advanced tab:

- **Timeout (--timeout)**

The minimum timeout to wait for a reply from the agent (in microseconds). The Recorder uses a binary exponential retransmission scheme to overcome unreliable communications. The default minimum retransmit timeout is 1/2 second (500000 microseconds). This retransmit timeout is doubled on every consecutive retransmit of a PDU.

- **Retransmits (--rextmits)**

By default, the Recorder tries at most four times for every SNMP PDU. If no response arrives after the number of retransmits, the PDU times out and the discovery phase is over for the specific target. Use this option for extra slow, distant or erratic devices.

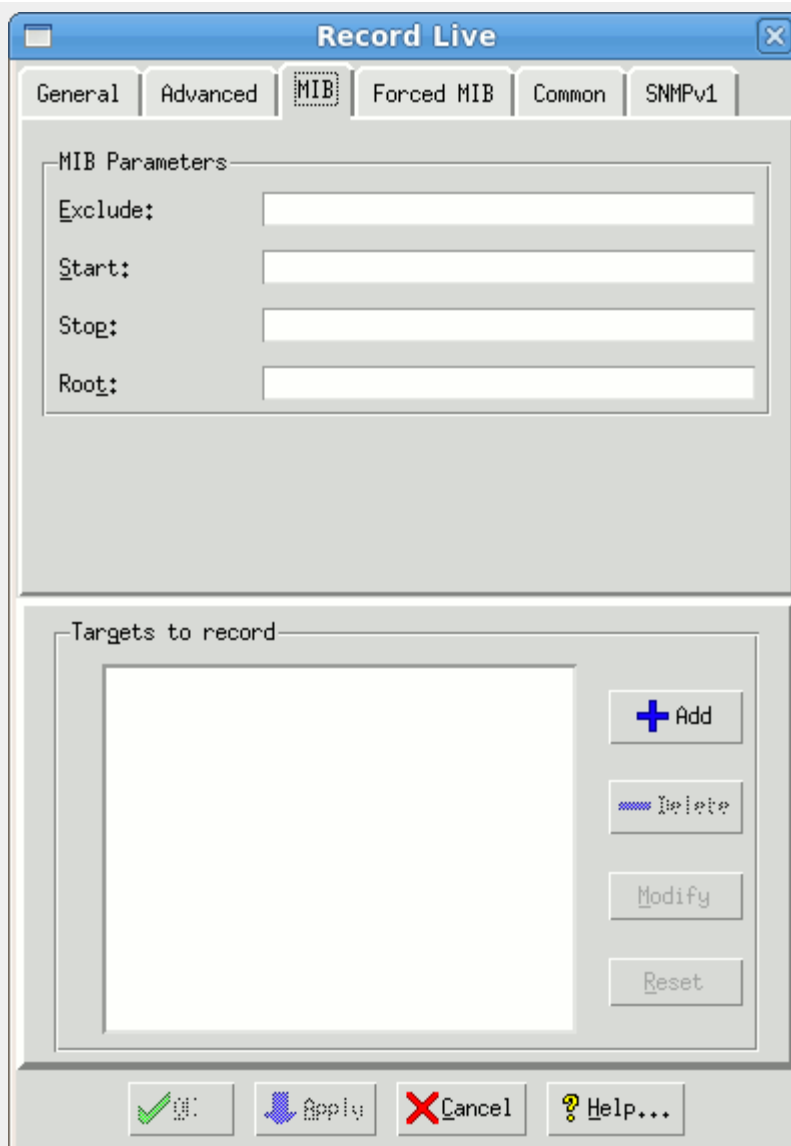


Figure - MIB Tab

The following advanced options control the recording of MIBs for each selected target through parameters in the MIB tab:

• **Exclude (--exclude)**

By default, the Recorder walks the entire MIB exported by the target device. This option is useful if you want to exclude parts of the MIB from being simulated. For example, some MIB tables (such as RMON history tables) can be extremely dynamic (and large). Other tables implement proprietary retrieval schemes (e.g., getTable). Some devices even crash when you access certain parts of the MIB. This option allows to skip these MIB subtrees.

• **Start (--start)**

By default, the Recorder walks the entire MIB exported by the target device. This options lets you specify the starting OID to start recording. This is useful if you want to restart a recording that previously failed for any reason.

• **Stop (--stop)**

By default, the Recorder walks the entire MIB exported by the target device. This options lets you specify the ending OID, ie. the object after which the Recorder stops recording. This option is useful if you are interested in only a small part of the MIB.

• **Root (--root)**

By default, the Recorder walks the entire MIB exported by the target device (starting at OID .1.3). This options lets you specify MIB subtrees to record. This option is useful if you are interested in only a small part of the MIB.



Figure - Forced MIB Tab

You can force the inclusion of certain MIBs for the selected target through the list in the `Forced MIB` tab.

• `Forced MIBs(--force)`

By default, the Recorder will detect only MIBs for which there are discovered MIB objects. If there are certain MIBs, eg. TRAP-only MIBs you want to add to the simulation, you can force them with this option.

• `Static Simulation (--static)`

This option modifies the simulation for the MIB objects under the specified root to a "static" simulation, which executes much faster (twice as fast) than the basic simulation. It can be used when the basic simulation is not fast enough, and you don't care what values are returned from the simulation. The values assigned to the simulation are hard-coded, rather than through a [Value Space](#) lookup.

- The `sysUpTime` object is not affected by the `--static` option.
- Both table object and instance processing are done as usual. The `.tab` files are generated as usual.
- The `INDEX` objects will continue to use "table.index".
- All other table objects will always simulate the corresponding values from the first recorded instance of the tabular column.
- Since no [Value Space](#) lookup is done, the `.var` files are not generated.

- COUNTER objects will advance at the rate of the first recorded object instance. This rate cannot be changed during the simulation.

In terms of simulation language, the SIMULATE clauses are as follows:

- SIMULATE { uniform_per_tu (r,tu) } for all counter objects, where r and tu are determined from the first object instance (column) in the table. The "uniform_per_tu" function is made "constant_per_tu" if a "constant" simulation is selected.
- SIMULATE { v } for all other objects, where v is determined from the first object instance (column) of the table.

• Fast Simulation (--fast)

This option modifies the simulation for the MIB objects under the specified root to a "fast" simulation, which executes much faster than the static simulation. It can be used when the basic simulation is not fast enough, and you don't care what values are returned from the simulation. The values assigned to the simulation are hard-coded, rather than through a [Value Space](#) lookup.

- The `sysUpTime` object is not affected by the --fast option.
- Both table object and instance processing are done as usual. The .tab files are generated as usual.
- The INDEX objects will continue to use "table.index".
- Since no [Value Space](#) lookup is done, the .var files are not generated.
- COUNTER objects will advance at the constant rate of 1/second. This rate cannot be changed during the simulation.
- All other objects take on the value 1, except for OID typed values which take on the value 0.0.

In terms of simulation language, the SIMULATE clauses are as follows:

- SIMULATE { constant (1) } for all counter objects
- SIMULATE { "1" } for all other objects

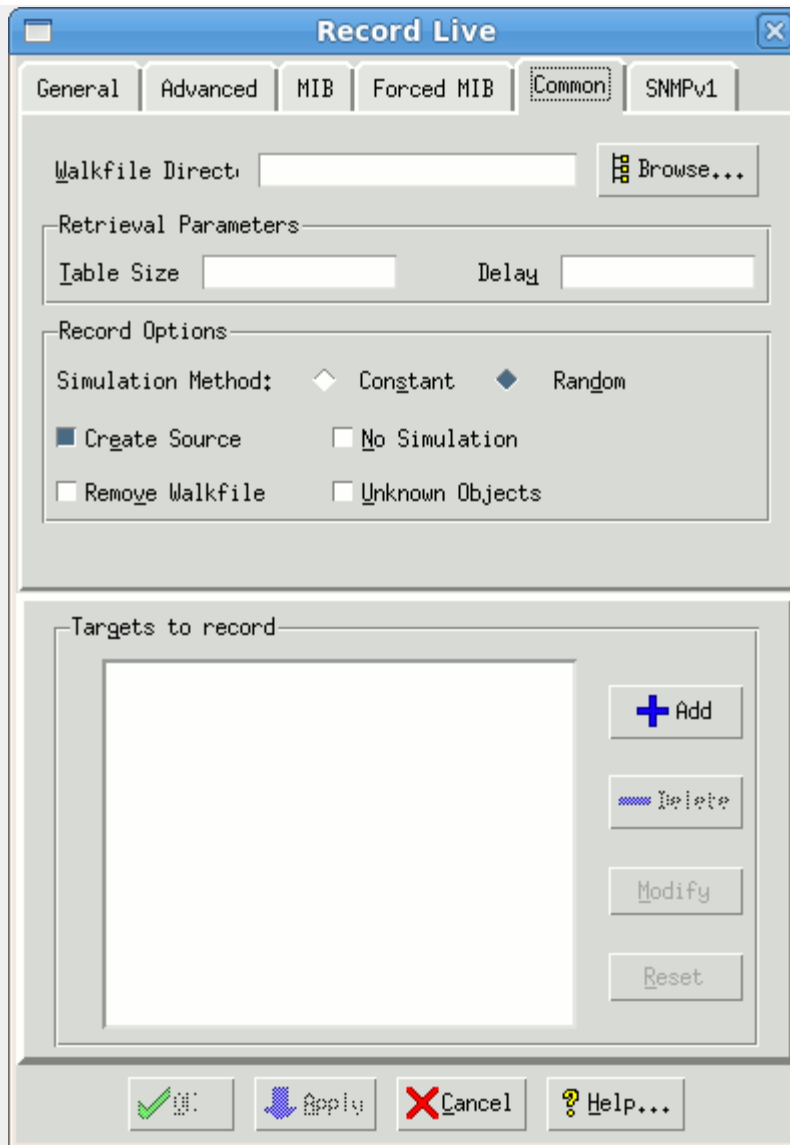


Figure - Common Tab

The following options in the `common` tab apply to all targets:

- **Walkfile Directory(--walkfiledir)**

This options lets you save the walkfile in a directory other than the default (`walks/` subdirectory in your private data area).

- **Table Size(--maxtab)**

By default, the Recorder will retrieve all entries for each table. This can take a long time for huge or dynamically increasing tables. Selecting this option limits the size of large tables to the specified number of table entries. It applies both at discovery time, where only the selected number of entries are traversed, and at simulation time, where only the selected number of entries are simulated. This is useful for devices with large tables, such as RMON tables, etc.

- **Delay (--delay)**

This option slows down the retrieval of variables in the discovery phase in order not to overwhelm the target device with SNMP requests. The delay in milliseconds is introduced between the receipt of the response and the sending of the next request. (By default this option is disabled.)

- **Simulation Method (--method)**

This option allows three choices: random, constant or linear, to specify which of three simulation methods to use for counter simulations. By default, the Recorder simulates every MIB object of type Counter with a rate with a uniformly-distributed "random" variable using the following expression

```
SIMULATE { uniform_per_tu (lookup ("r"), lookup ("tu")) }
```

You can use the "constant" option to generate a constant rate simulation with the following expression

```
SIMULATE { constant_per_tu (lookup ("r"), lookup ("tu")) }
```

This is equivalent to the function $f(t) = r/tu * t$

You can use the "linear" option to generate a linear simulation with the following expression

```
SIMULATE { lookup ("v") + constant_per_tu (lookup ("r"), lookup ("tu")) }
```

This is equivalent to the linear function $f(t) = v + r/tu * t$

This method is more flexible than the constant method, but will simulate 33% slower, because of the 1 extra lookup.

Also see [Language Reference](#).

• Create Source (--source)

Selecting this option generates the simulation source file so that you can change the simulation later on. (By default this option is disabled.)

• No Simulation (--nosim)

Selecting this option skips the simulation phase and only generates a walk file. This is useful for example if you are at a remote site, and want to generate the simulation later. (By default this option is disabled.)

• Remove Walkfile (--nokeep)

Use this option to remove the walk file upon completion. This prevents walk files from accumulating in your directory. (By default this option is disabled.)

• Exact Time (--exact)

Enabling this option causes the Recorder to query sysUptime more frequently than every 10 seconds. This provides more accurate extrapolations (based on time) for Counter objects. (By default this option is disabled.)

• No symbolic information (--noinfo)

This options prevents the Recorder from putting symbolic information in the walkfile created during discovery phase of recording. Otherwise, the output walkfile has one commented line of symbolic information for each object exported, e.g.

```
# sysDescr: read-only OCTET STRING (SIZE (0..255))
```

(By default this option is disabled.)

• Tablewalk (--tablewalk)

This options tells the Recorder how to optimize tablewalk traversal and allows 3 choices: none, row and bulk.

By default, the Recorder will use GETBULK when possible (in versions 2c and 3). In version 1, the Recorder uses a proprietary multi-variable per-PDU row traversal to optimize the number of requests. This behavior can be overridden with the --tablewalk option.

The option none disables any optimizations and will do inefficient single-variable per-PDU GETNEXT traversal.

The option row forces multi-variable per-PDU row traversal.

The option bulk forces GETBULK traversal.



Figure - SNMPv1 Tab

The following options in the `SNMPv1` tab apply to each target using the SNMPv1 protocol:

- **Community String (`--community`)**

The default community string used is "public" -- if the device uses another read community string, set it in this option.

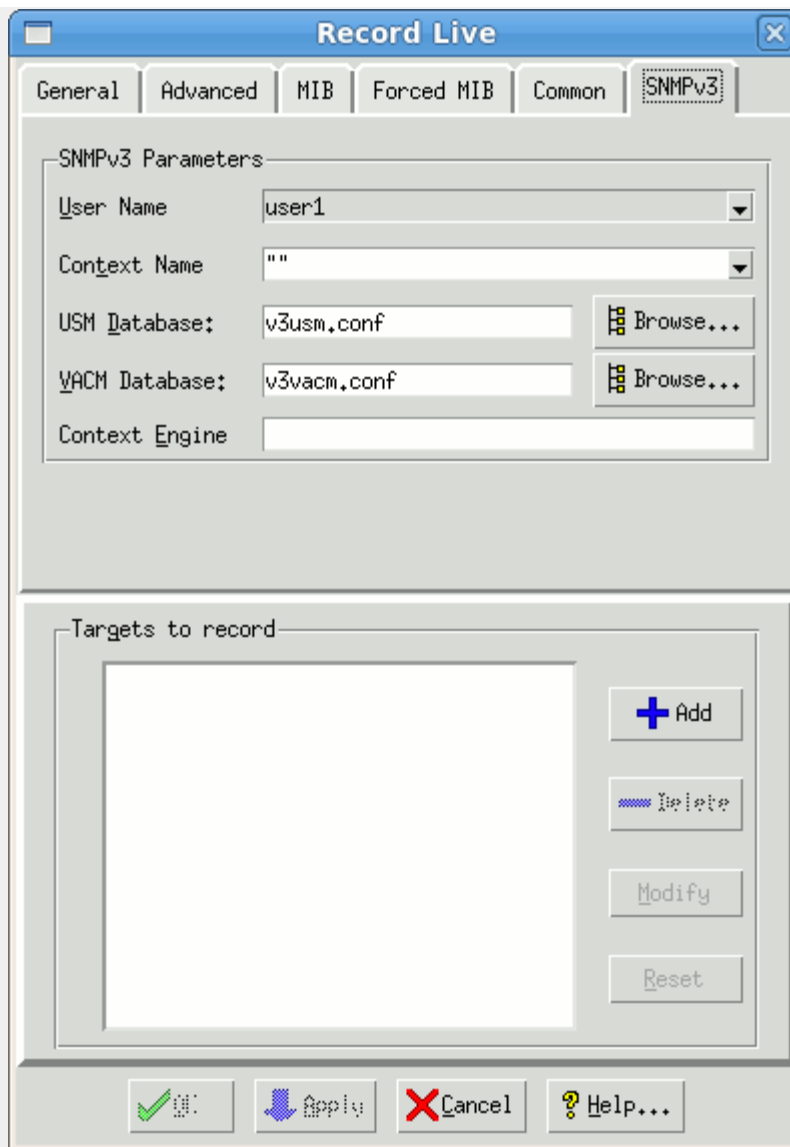


Figure - SNMPv3 Tab

The following options in the `SNMPv3` tab apply to each target using the SNMPv3 protocol:

- **User Name(--user)**

The authentication and privacy parameters are determined from the user name, context name and the USM and VACM Database files.

- **Context Name(--context)**

The authentication and privacy parameters are determined from the user name, context name and the USM and VACM Database files.

- **USM Database(--usmFile)**

The authentication and privacy parameters are determined from the user name, context name and the USM and VACM Database files. The USM Database file is optional, and will use by default `config/v3usm.conf`. You can also browse for the file by clicking `Browse...`

- **Engine ID(--engineid)**

Rather than discovering the engine ID (default), this parameter can specify it.

- **Context Engine ID(--contextengineid)**

Rather than discovering the context engine ID (default), this parameter can specify it.

File Mode

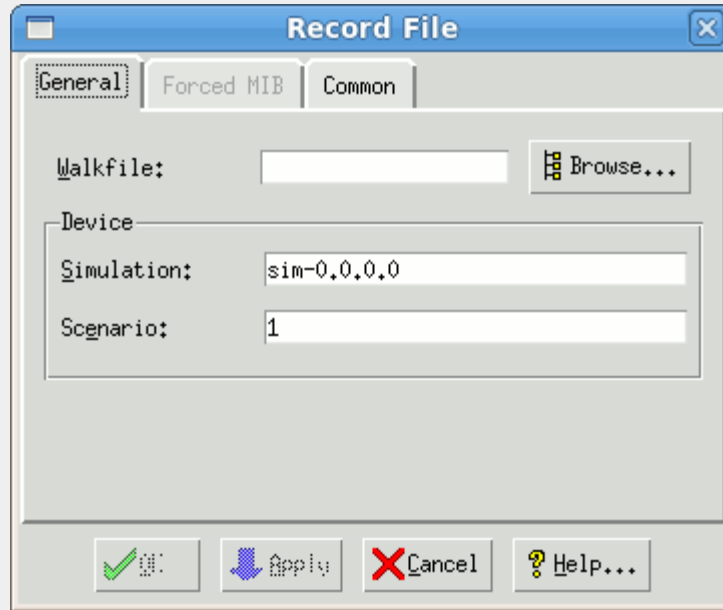


Figure - Record File Dialog

In this off-line simulation method a previously captured data file is fed to the Recorder to simulate a device. As such there is no discovery phase, just a simulation phase. The following options in the General tab are mandatory:

• Walkfile (--file)

The file contains the previously captured walk of a MIB. This walk can be the output of a third party MIB browsing tool or of a previous live capture from the Recorder. The format of each line in the file needs to be of the form

```
MIB-object-instance: value
```

where MIB-object-instance can be in dot-notation or a symbolic name. Since not every browsing tool outputs this format, we provide filters (in the common/subdirectory) for some of the common tools.

• Simulation (--simulation)

This specifies the simulation name of the simulation to create. The dialog provides a default simulation name that you can change. You can create any number of simulations for a particular device type. Using an already existing simulation causes the old simulation values to be overwritten with the new.

• Scenario (--scenario)

This specifies a scenario name. The dialog provides a default scenario name that you can change. Using an already existing scenario causes the old simulation values to be overwritten with the new.

You can force the inclusion of certain MIBs through the list in the Forced MIBs tab.

• Forced MIBs(--force)

By default, the Recorder will detect only MIBs for which there are discovered MIB objects. If there are certain MIBs, eg. TRAP-only MIBs you want to add to the simulation, you can force them with this option.

The following advanced options in the Common tab can be set:

• Table Size(--maxtab)

By default, the Recorder will retrieve all entries for each table. This can take a long time for huge or dynamically increasing tables. Selecting this option limits the size of large tables to the specified number of table entries. It applies both at discovery time, where only the selected number of entries are traversed, and at simulation time, where only the selected number of entries are simulated. This is useful for devices with large tables, such as RMON tables, etc.

• **Line(--line)**

By default, the Recorder will start at the first line of the walkfile. You can start at a different line with this command line option.

• **Simulation Method (--method)**

This option allows two choices: random or constant, to specify which of two simulation methods to use for counter simulations. By default, the Recorder simulates every MIB object of type Counter as a uniformly-distributed random variable with the following expression

```
SIMULATE { uniform_per_tu (lookup ("r"), lookup ("tu")) }
```

You can use the constant option to generate a constant simulation with the following expression

```
SIMULATE { constant_per_tu (lookup ("r"), lookup ("tu")) }
```

[See Language Reference.](#)

• **Create Source (--source)**

Selecting this option generates the simulation source file so that you can change the simulation later on. (By default this option is disabled.)

Other Options

The following options apply to both Live mode and File mode:

• **Overwrite device entry (--overwrite)**

Selecting this option overwrites the entry in the device configuration file (config/dev.cfg), even if the set of MIBs is different. When you record a device, an entry is added to dev.cfg, which you can then select in MIMICView. This option is useful if you are recording a device multiple times, each time with a newly compiled MIB. This is the normal procedure to add a new device to MIMIC, where the set of MIBs supported by the device is unknown. (By default, this option is turned off, causing a new entry to be added to dev.cfg.)

• **Simulate unknown MIB objects (--unknown)**

By default, if the Recorder encounters an unknown MIB object, ie. one for which no MIB definition exists in the compiled MIBs, then it will skip creating a simulation for this object, and will print an error message. The count error messages is displayed at the end, providing an overall measure of the correctness of the simulation.

The walkfile saved by the MIMIC Recorder will contain syntax information for unknown objects with a commented line of the form:

```
# UNKNOWN: read-only syntax
```

NOTE: any 3rd party walkfile format which does not have type information cannot be processed for unknown objects.

Enabling this option will cause the Recorder to attempt to guess the syntax of the unknown MIB objects, and will try to create a simulation for them. It does this by parsing the syntax information saved for unknown objects and will try to derive correct indexing through proprietary heuristics.

Since no MIB object naming information is known, the Recorder assigns names to the objects of the form `unknown-dashed-OID`, `table-dashed-OID`, `entry-dashed-OID`, where `dashed-OID` is the dotted OID notation, where the dot is substituted with a dash to conform with SMI rules. These MIB object names will not be used when responding to SNMP requests, but will be displayed when queried through MIMIC clients (MIMICView or MIMICShell).

In addition, a MIB is created containing derived (guessed) definitions for the unknown objects. The MIB has the name `UNKNOWN/simulation-scenario-MIB`, where `simulation` and `scenario` are the simulation and scenario names. It is compiled in, so that future recordings will take this definition.

• **Creating a Walkfile with mib2walk**

If you need to create a simulation from scratch, you can use the **mib2walk** utility provided with MIMIC. This utility takes a list of MIB names, and outputs the template of a walkfile that you can feed to the Recorder.

NOTE: the [Simulation Wizard](#) provides a user-friendly front-end to mib2walk.

For example:

```
mib2walk RFC1213-MIB IF-MIB DIAL-CONTROL-MIB xylan/XYLAN-ATM-MIB > walkfiletemplate
```

The walkfile template will contain two lines per MIB object: a comment containing object name, type and other information; and a line containing the OID of the object.

You can use this template to create your simulation by

- Creating instances of tabular objects. You can copy and paste an OID line as many times as necessary, and provide the necessary instance subids.
- Assigning values to the MIB object instances

You then feed the modified walkfile to the MIMIC Recorder to create the simulation.

The MIMIC Recorder is designed to aid in this process by allowing you to implement the simulation incrementally. To skip processing the remaining walkfile, just add a line with

```
exit
```

in it. Thus you can incrementally implement more and more of your simulation, and just move the exit point forward.

Note that counter objects depend on the notion of time to be able to extrapolate the rate for the rate-based simulation. Thus, if you want to simulate counters, you will need to define a line that sets sysUpTime at the beginning of your walkfile, such as

```
1.3.6.1.2.1.1.3.0: 100
```

Note that sysUpTime is counted in TimeTicks (1/100 sec), thus 1 sec is equivalent to 100 TimeTicks.

Walkfile Format

This section defines the format of the walkfile understood by the MIMIC Recorder. It is similar to most common walkfile formats and there is a [conversion utility](#) and conversion scripts in the common/ directory to convert from popular formats to the MIMIC format.

Each variable-value binding is represented by one or more lines with the following single-line or multi-line formats:

```
[OID]: [value]
```

or

```
[OID]: [value]  
[value continuation]
```

For example:

These are sample MIMIC walkfile lines

```
1.3.6.1.2.1.1.1.0: Cisco Systems WS-C5000  
1.3.6.1.2.1.1.2.0: 1.3.6.1.4.1.9.5.7  
1.3.6.1.2.1.1.3.0: 107d 10:25:09.99
```

This shows a multi-line hex value

```
1.3.6.1.4.1.9.5.1.3.1.1.15.5: \x02 01 01 01 01 01 01 01 01 01 01 01 01 01 01  
01 01 01 01 01 01 01 01 01
```

Notice that a variable-value binding line starts with a blank character. All other lines (except value continuation lines) are interpreted as comments. Value continuation lines are useful for long hexadecimal values. There is a blank character between the ":" and the value.

The [OID] field can be numeric or symbolic, but numeric OIDs are preferred.

The [value] field has the following formats for the different types:

- INTEGER, Counter: a number
- OCTET STRING: a character string, or a hexadecimal string that starts with \x and contains 2 hex-digits per byte separated by white space.
- OBJECT IDENTIFIER: a numeric OID
- Timeticks: [days]d [hours]:[minutes]:[seconds].[ticks]
- IpAddress: dot notation

In addition, each line containing a new MIB object will have a comment with the SYNTAX for that MIB object. This is not only for documentation, but for unknown objects allows better simulation. For details, [see above](#).

<< Previous Section Next Section >>



MIMIC Compiler Guide

Table of Contents

- [Overview](#)
- [Compiler Reference](#)
 - [Importing a MIB](#)
 - [Compiling a MIB](#)
 - [Editing a MIB](#)
 - [Importing a MIB](#)
 - [Creating a Custom Simulation](#)
 - [Compiling a Simulation](#)
 - [Editing a Simulation](#)
 - [Creating a Scenario](#)
- [MIMIC Simulation Language Reference](#)
 - [High-level Syntax](#)
 - [Types](#)
 - [Constants](#)
 - [Arithmetic Functions](#)
 - [Nested Expressions](#)
 - [System Functions](#)
 - [Conditional Operator](#)
 - [Value Space Access](#)
 - [Variable Store Access](#)
 - [Counter Statistics](#)
 - [MIB Tables](#)
 - [Information about current Object](#)
 - [Relationship between Objects](#)
 - [Trap Generation](#)

Overview

The MIMIC Compiler is used for two purposes:

- To import new MIBs into MIMIC. This allows MIMIC to recognize new [device types](#). MIMIC supplies more than 1000 [pre-compiled standard and enterprise MIBs](#) for devices from the leading network companies. If the MIB for the device that you are trying to simulate is not included, you can import it into MIMIC by compiling the MIB source file with the Compiler;
- To compile simulations. Simulations are created by amending the MIB file with the [SIMULATE clause](#).

Compiler Reference

Importing a MIB

Before compiling a new MIB, you need to import the source file into the MIMIC area.

1. In MIMICView, use [MIB->Import...](#) to invoke the `Import MIB` dialog. The source file is the file containing the definition of the MIB in ASN.1 syntax following the standard SNMP Structure of Management Information (SMI) specification. All MIB source files known to MIMIC are in the `mibs/` subdirectory.
2. Type the absolute path of the MIB source file that you want to import in the `MIB File` field, or use the `File Browser` dialog with the `Browse...` button. The MIB source file can be in any directory on your system.
3. To import a standard MIB, leave the enterprise name in the `Enterprise` field blank. To import an enterprise-specific MIB, type an enterprise name in the `Enterprise` field. For simplicity, use a single-word, lower-case, name.
4. If you want to save it as a different name in MIMIC, type the new name in the `Save As` field.
5. Pressing the `OK` button copies the file into the appropriate place in the MIMIC area. From then on the MIB source file is accessible by MIMIC.

Compiling a MIB

1. Use [MIB->Compile...](#) to pop up the `Compile MIB` dialog.
2. Type the enterprise name of the MIB in the `Enterprise` field and the MIB file name in the `MIB File` field, or browse with the `Browse...` button.
3. Click `Cancel` to dismiss the dialog, or click `OK` to begin the compilation in a separate [log window](#).

Here is a sample log. If the last line says "done", the compile was successful, otherwise it did not complete and you need to fix the shown errors.

```
INFO 06/18.14:41:40 - mimicom 2.2 #1 (sparc) of Mon Jun 15 18:16:02 EDT 1998
```

```
INFO 06/18.14:41:40 - Evaluation license, expires 7/18/1998
INFO 06/18.14:41:40 - loading mib : RFC1213-MIB
INFO 06/18.14:41:48 - done
```

Editing a MIB

In case the MIB compilation has errors, you can edit the MIB file.

1. Use `MIB->Edit...` A File Browser dialog is automatically opened.
2. Select the path to the MIB file.
3. Click `OK` to open the built-in MIMIC File Editor. (You can invoke the editor of your choice if you set the `MIMIC_EDITOR` environment variable prior to starting MIMICView.)

Creating a Custom Simulation

The easiest way to create a simulation is to use the [MIMIC Recorder](#) to record an existing device. If you cannot record a device (e.g., if you don't have one available, or it is under development), then you can create a basic simulation with [mib2walk](#). But, if you want to simulate a device with capabilities beyond the basic simulation that the Recorder outputs, you need to customize it manually as follows.

1. Create a subdirectory under the `sim/` directory in the MIMIC area. Give the simulation a simple name, such as the ones that are already there.
 2. Create a simulation file for each MIB you intend to simulate in that subdirectory. The simulation file contains the simulation expression for each MIB object. As detailed [below](#) there can be two formats for the simulation file.
- SMI with added `SIMULATE` statements. This consists of the original MIB file with a `SIMULATE` clause for each MIB object. This is useful when you start with the original MIB file and just edit it to add `SIMULATE` expressions, but it can become rather lengthy.

- minimal format. This format deletes the unnecessary clauses for each object definition (`SYNTAX`, `ACCESS`, `STATUS`, `DESCRIPTION`). Look at the existing simulations for conventions to follow. You can even copy over the files of existing simulations as a starting point. Its simplified syntax is:

```
mib-name SIMULATION ::=
list-of-object-simulations
```

where **list-of-object-simulations** is one or more object simulations of the form

```
object-name OBJECT-TYPE
SIMULATE simulation-expression
```

Alternatively, you can create a minimal simulation file from the original MIB file as follows:

1. cut the relevant lines from the original MIB files, eg.
`grep OBJECT-TYPE original-MIB-file > simulation-file`
2. edit the **simulation-file** to add the header line, and add a `SIMULATE` clause for each object.

Compiling a Simulation

To compile a simulation:

1. Use `Simulation->Compile...` to invoke the `Compile MIB Extensions` dialog.
 2. Type the simulation name in the `Simulation` field.
 3. Type the MIB extensions file name in the `MIB Extensions File` field, or browse for it by clicking `Browse...`
 4. Click `Cancel` to dismiss this dialog.
 5. Press `OK` to begin the compilation in a separate [log window](#).
- Here is a sample log. If the last line says "done", then it was successful, otherwise the compile did not complete and you need to fix the shown errors.

```
INFO 06/18.14:41:40 - mimicom 2.2 #1 (sparc) of Mon Jun 15 18:16:02 EDT 1998
INFO 06/18.14:41:40 - Evaluation license, expires 7/18/1998
INFO 06/18.14:41:40 - parsing MIB Extension
INFO 06/18.14:41:40 - loading mib : RFC1213-MIB
INFO 06/18.14:41:48 - done
```

Editing a Simulation

In case the simulation compilation has errors, you can edit the simulation file.

1. Use `Simulation->Edit...` A File Browser dialog is automatically opened.
2. Select the path to the simulation file.
3. Click `OK` to open the built-in MIMIC File Editor. (You can invoke the editor of your choice if you set the `MIMIC_EDITOR` environment variable prior to starting MIMICView.)

Creating a Scenario

Once you have created a simulation, you need to create a scenario for it. The scenario defines the MIB object instances and values. A simulation can have multiple scenarios.

The internal data files for a scenario for a particular simulation are in a directory in the `data/sim/` subdirectory, under each MIB with the name of the scenario. This directory contains files for each table and MIB object to be simulated.

- The format of the .tab files is a list of instances, one per line.
- The format of the .var files is
instance variable value

This determines the value for the specified variable for the specified instance. For example, if you were to edit ifInOctets.var as follows, you would basically be setting the rate of instances 1 and 2 to 10 and 50 (respectively):

```
1 r 10
2 r 50
```

MIMIC Simulation Language Reference

High-level Syntax

The MIMIC MIB extension language is a declarative language based on the C programming language. It specifies a simulation expression to be executed when a particular instance of a MIB object is accessed.

NOTE: The main thing to remember is that MIMIC does not execute simulation expressions unless there is an access to the MIB object.

A simulation takes on the form of a `SIMULATE` clause for each object in the MIB of the syntax:

```
SIMULATE {expression}
```

where `expression` is an expression in the MIMIC language. A value returned by MIMIC in response to a request for a MIB object instance is effectively a function of the form

value-returned = function (expression, time)

The simulation expression can be embedded into (a copy of) an existing MIB file. Each object definition is augmented with a `SIMULATE` clause. In this case, the file would contain something like

```
sysUpTime OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory
SIMULATE {expression}
DESCRIPTION
...
```

Alternatively, a separate file can be created containing just simulation expressions, such as:

```
sysUpTime OBJECT-TYPE
SIMULATE {expression}
...
```

Types

Each expression in the MIMIC language returns a value with one of the types in the SNMP Structure of Management Information (SMI) specification:

- INTEGER
- OCTET-STRING
- OBJECT-ID
- Counter
- Gauge
- IpAddress

Constants

Constant expressions return the same value at any point in the simulation.

Decimal, hexadecimal, octal, binary INTEGER constants

INTEGER constants can be specified in base 10, 16 and 8.

Syntax:
[1-9][0-9]* for decimal constants
0[0-7] for octal constants
0x[0-9a-fA-F]+ for hexadecimal constants

Examples:
1500
0700
0x700

Enumerated INTEGER Constants

You can use enumerated symbol names just as in the SMI.

Examples:
ethernet-csmacd

Octet Strings

Octets strings can be defined both in ASCII (enclosed in double quotes), and binary (hex) format.

Examples:
"this is a string"
"\x08 00 2b 12 34 56"

Object Identifiers

Object identifiers can be specified in dot notation, or by any part of the sequence of sub-identifier names.

Examples:
1.3.1.6.1.2.1
system.sysDescr
enterprises.886

IpAddress

IP addresses need to be specified in dot notation or as 4 hexadecimal bytes.

Examples:
192.9.200.71
\x C0 09 C8 47

Arithmetic Functions

MIMIC supports the basic complement of integer arithmetic functions:

- Addition +
- Subtraction -
- Multiplication *
- Division /
- Mod %

MIMIC will do type propagation based on the return value of the SIMULATE expression. For example, if the SIMULATE is done for a Counter64 MIB object, then 64-bit arithmetic is performed automatically, else 32-bit arithmetic is performed.

These functions do explicit typecasts:

- INTEGER counter32(number)
Converts the number to a 32-bit value.
- INTEGER counter64(number)
Converts the number to a 64-bit value.
- STRING string_cast(number)
Converts the number to a string value.

Nested Expressions

Nested expressions should be parenthesized, to avoid misunderstandings in operator precedence.

System Functions

This is a set of typed functions that return system-related information:

- INTEGER agent_no()
Returns the agent instance identifier.
- INTEGER agent_addr()
Returns the agent instance IP address.
- INTEGER sysuptime()
Returns an INTEGER containing the time in ticks (100ths of seconds) since the agent booted. This time will be maintained on a per agent instance basis.
- INTEGER random(integer low, integer high)
Returns a random value between (and including) low and high.

Conditional Operator

The conditional operator will allow decisions to be made in your simulation. Its syntax is just like in C: conditional-expression ? true-

expression : false-expression The conditional expression is evaluated, and either of the true-expression or false-expression is executed depending on whether the conditional expression evaluates to true or false at this particular time.

Examples:

```
ifStatus OBJECT-TYPE
SIMULATE { sysuptime () > 8640000 ? down : up }
```

This simulation expression makes an interface be up for one day, after which it will appear down.

Value Space Access

These operation allows indexed access of configurable data in the [MIMIC Value Space](#). It allows run-time configuration of simulations for individual agent instances. The lookup function returns the OCTET-STRING for a given variable for the specified object instance. The signature is

- OCTET-STRING lookup (const char* variable_name)

Examples:

This simulation expression returns a system description value directly from the Value Space:

```
sysDescr OBJECT-TYPE
SIMULATE { lookup ("v" ) }
```

The problem with the above is that a variable does not need to be set in the Value Space. Since it is adaptive, a default value will be returned, but if the lookup function is part of an expression, it will fail. For this reason, you can check the existence of the variable with the exists function, which returns 1 if it exists, 0 otherwise. Those return values can be used in the [conditional operator](#). The signature is

- boolean exists (const char* variable_name)

Examples:

This simulation expression returns a default value if not set.

```
sysDescr OBJECT-TYPE
SIMULATE { exists ("v" ) ? "not set" : lookup ("v" ) }
```

The set function allows modification of value space variables. It sets the specified variable to the specified value, and returns it. The signature is

- OCTET-STRING set (const char* variable_name, const char* value)

Examples:

This simulation expression auto-increments a Gauge MIB object tcpCurrEstab in the range between min and max:

```
tcpCurrEstab OBJECT-TYPE
SIMULATE { (exists ("_retval") == "0") ? set ( "_retval", lookup ("min")) : ((set ("_retval", (lookup ("_retval") + 1)) > lookup ("max"))
? set ("_retval", lookup ("min")) : lookup ("_retval")) }
```

If _retval is not set, then it is initialized with min. Else, _retval is incremented, and if larger than max, rolls back to min.

Variable Store Access

These operations allow access to the global and per-agent [Variable Store](#). This effectively allows even cloned agents (which share Value Space) to individualize values through the per-agent variable store.

- integer global_store_exists (const char* variable_name)
- OCTET-STRING global_store_get (const char* variable_name)
- OCTET-STRING global_store_set (const char* variable_name, const char* value, integer persist)
- integer agent_store_exists (const char* variable_name)
- OCTET-STRING agent_store_get (const char* variable_name)
- OCTET-STRING agent_store_set (const char* variable_name, const char* value, integer persist)

Examples:

This simulation expression returns a system description value directly from the agent store variable "sysDescr":

```
sysDescr OBJECT-TYPE
SIMULATE { agent_store_get ("sysDescr") }
```

The problem is that a store variable may not exist, causing the simulation to result in error. This expression checks for existence first, and returns the value from the agent store, else from the value space as in the default simulation.

```
sysDescr OBJECT-TYPE
SIMULATE { (agent_store_exists ("sysDescr") == 1 ) ? agent_store_get ("sysDescr") : lookup ("v" ) }
```

Counter Statistics

The following group of functions allows flow-based simulation of MIB objects of type Counter:

- Counter uniform(integer rate)
Returns the current value of a uniformly distributed random variable with the specified average rate per second.

- Counter `uniform_per_tu(integer rate, integer timeunit)`
Returns the current value of a uniformly distributed random variable with the specified average rate per given timeunit in seconds. This is to simulate counters with low rates. When timeunit is 1, this function is the same as `uniform()`.
- Counter `constant(integer rate)`
Returns the current value of a constant variable with the specified average rate per second.
- Counter `constant_per_tu(integer rate, integer timeunit)`
Returns the current value of a constant variable with the specified average rate per given timeunit in seconds. This is to simulate counters with low rates. When timeunit is 1, this function is the same as `constant()`.

Examples:

This simulation expression sets the rate of incoming packets to 100 per minute as a random variable with uniform distribution:

```
ifInUcastPkts OBJECT-TYPE
SIMULATE { uniform_per_tu (100, 60) }
```

or more flexibly, to allow run-time configuration of both parameters via variables `r` (for rate) and `tu` (for timeunit):

```
SIMULATE { uniform_per_tu (lookup ("r"), lookup ("tu")) }
```

MIB Tables

This group of operations is related to MIB tables and the INDEX clause.

- `table.range (minindex, maxindex, stepindex)`
Used in the table object with INDEX clause, it defines this table as a static table with a range of index values as specified by the minindex, maxindex and stepindex arguments. This function is used to define tables with a "mostly" contiguous range of indices. The minindex argument specifies the first index in the table, the maxindex argument the last index, and the stepindex the "distance" between each consecutive index.
- `table.static`
Used in the table object with INDEX clause, it defines this table as a static table with indices specified in an index file. This function is used to define tables with non-contiguous indices.
The index file lives in the scenario directory with the same name as the INDEX object, with `.tab` appended.
You can add and remove entries to this type of table at run time.
- OBJECT-ID `table.index (int i)`
Returns the OID containing the i-th index component of the index OID. This is useful for simulating the objects which are contained in the INDEX clause of a table. The compiler will make this function the default for all objects which are specified in a INDEX clause. Also, this function can be used for special casing in value simulations.

Examples:

The following simulation expression creates an interfaces table from 1 to 9, with odd indices.

```
ifEntry OBJECT-TYPE
SIMULATE { table.range (1, 10, 2) }
```

The following simulation expression creates an interfaces table that is configured at run-time from an index file. The index file lives in the scenario directory, and has the name `ifEntry.tab`.

```
ifEntry OBJECT-TYPE
SIMULATE { table.static }
```

For example, if the `ifEntry.tab` file contains

```
5
10
11
```

then only interfaces 5, 10 and 11 will be simulated.

This simulation expression returns a run-time configurable interface type for all interfaces but the second one, for which it returns the constant `ethernet-csmacd`.

```
ifType OBJECT-TYPE
SIMULATE { table.index(1) == 2 ? ethernet-csmacd : lookup ("v") }
```

Information about current Object

If you want to get information about the currently accessed MIB object, you use the operations

- OCTET-STRING `oid()`
Returns the OID of the currently accessed MIB object without the instance.
- OCTET-STRING `object()`
Returns the name of the currently accessed MIB object.
- OCTET-STRING `instance()`
Returns the instance of the currently accessed MIB object.

Relationships Between Objects

This operation implements relationships between MIB object instances.

- `value (const char* object)`
Returns the value of the specified object with the same index as the object that is being simulated. This can currently only be used with objects in the same table.

Examples:

This simulation expression makes the error rate 10 % of the incoming packets:

```
ifInErrors OBJECT-TYPE
SIMULATE { (value ("ifInUcastPkts") * 10) / 100 }
```

or, more efficiently:

```
ifInErrors OBJECT-TYPE
SIMULATE { value ("ifInUcastPkts") / 10 }
```

or, more flexibly, to allow run-time configuration of the error rate:

```
SIMULATE { (value ("ifInUcastPkts") * lookup ("error-rate")) / 100 }
```

Trap Generation **QuickStart**

This operation simulates trap PDU generation.

- `void trap_periodic (rate, time_unit, cutoff_time)`
This operation causes periodic trap PDUs to be generated for a specific trap. `rate` and `time_unit` defined the frequency of the generated traps. The `cutoff_time` argument specifies a time (in seconds) relative to now at which trap generation is stopped. This allows a precise number of traps to be generated for a precise amount of time. The default simulation expression for traps is

```
SIMULATE {trap_periodic (lookup "r", lookup "tu", lookup "c")}
```

which causes the arguments to be looked up in the value space.

MIB variable values to be sent with the generated traps can be stored in the value space and are used by all trap generation functions.

Examples:

For the `linkDown` or `linkUp` traps, if you set the variable `ifIndex` to a value, that value will be sent with any of these generated traps.

[<< Previous Section](#) [Next Section >>](#)



MIMIC Wizards Guide

Preface

The MIMIC Wizards are easy-to-use GUI front-ends to the most common or most difficult tasks in MIMIC.

Table of Contents

- o MIB Wizard Reference
 - Overview
 - Select MIBs To Import
 - Select Existing MIB Files To Compile



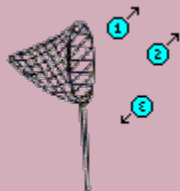
- o Discovery Wizard Reference
 - Overview
 - Discovery Wizard Sessions
 - Discovery Method
 - IP Addresses
 - Filters to Limit Discovery
 - SNMP Parameters
 - Recorder Options
 - Select Walkfiles
 - Discovered Targets
 - Recorded Targets
 - Create Lab Configuration
 - Interrupted Session



- o Snapshot Wizard Reference
 - Overview
 - Snapshot Option
 - Simulation Option
 - Choose a Live Target
 - Recorder Options
 - Snapshot Constraints
 - Take Snapshots
 - Create Change Script
 - Apply Change Script Manually



- o Trap Wizard Reference
 - Overview
 - Series Options
 - Simulation Options
 - Choose a Live Target
 - Recorder Options
 - Trap Capture Options
 - Capture Trap Series
 - Edit Trap Series
 - Play Back Testing



- o Simulation Wizard Reference
 - Overview
 - Select Walkfile To Load
 - Select MIBs
 - Specify Default Values
 - Edit Object Simulations
 - Select Walkfile To Save

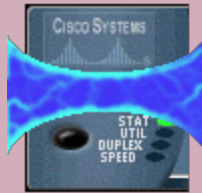
- Select Simulation To Create



- CLI Wizard Reference
 - Overview
 - Method Choice
 - Recording Parameters
 - Discover Commands
 - Issue Commands



- NetFlow Wizard Reference
 - Overview
 - Create or modify
 - Create Config file using PCAP file
 - Modify Config File



- sFlow Wizard Reference
 - Overview
 - Create or modify
 - Create Config file using PCAP file
 - Modify Config File



- IPMI Wizard Reference
 - Overview
 - Create or modify
 - Record Live Session
 - Simulation Options
 - Change IPMI Values



- Web Wizard Reference
 - Overview
 - Create Web Service Simulation
 - Recording Parameters
 - Record PCAP
 - Configurable Values

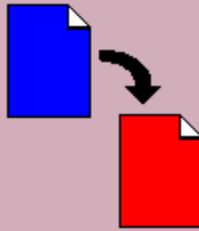


- Tutorial Wizard Reference
 - Overview
 - MIBs
 - Create Simulations
 - Run Simulations
 - Customize Simulations

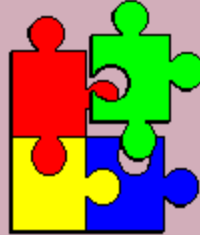


- Configuration Wizard Reference

- Overview
- MIMICView Configuration
- MIMIC Log Configuration
- Java Configuration
- E-mail Configuration
- Profiling Configuration



- Update Wizard Reference
 - Introduction
 - Select Package
 - Download Images
 - Prepare For Installation
 - Extract Files
 - Move Files



- Diagnostic Wizard Reference
 - Introduction
 - Select Configuration Files
 - Select MIB Files
 - Select Simulation Files
 - Select Walkfiles
 - Select Snapshot Files
 - Select Discovery Files
 - Select Trap Series Files
 - Select Log Files
 - Review Selected Files
 - Send Files



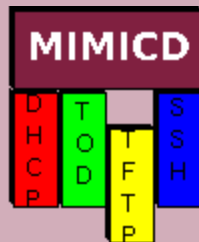
- Sanity Wizard Reference
 - Introduction
 - Select Items
 - Detection Results



- Performance Wizard Reference
 - Introduction
 - System Information
 - Setup Test
 - Performance Results
 - Send Files



- Protocol Wizard Reference
 - Introduction
 - Select Protocols
 - Supply License Keys
 - Results Installing/Uninstalling Protocols



1. MIB Wizard Reference

Overview

The MIB Wizard is designed to easily import and compile new MIBs for use with MIMIC. It is a user-friendly, graphical front-end to the [MIB](#)

compilation process and MIMIC Compiler.

The MIB Wizard lets you select a set of MIBs to import, which together with a set of existing MIBs will be compiled. The Wizard resolves all dependencies among the MIBs and compiles them in the correct order.



Select MIBs To Import

In this dialog, select the MIBs you want to import into MIMIC.

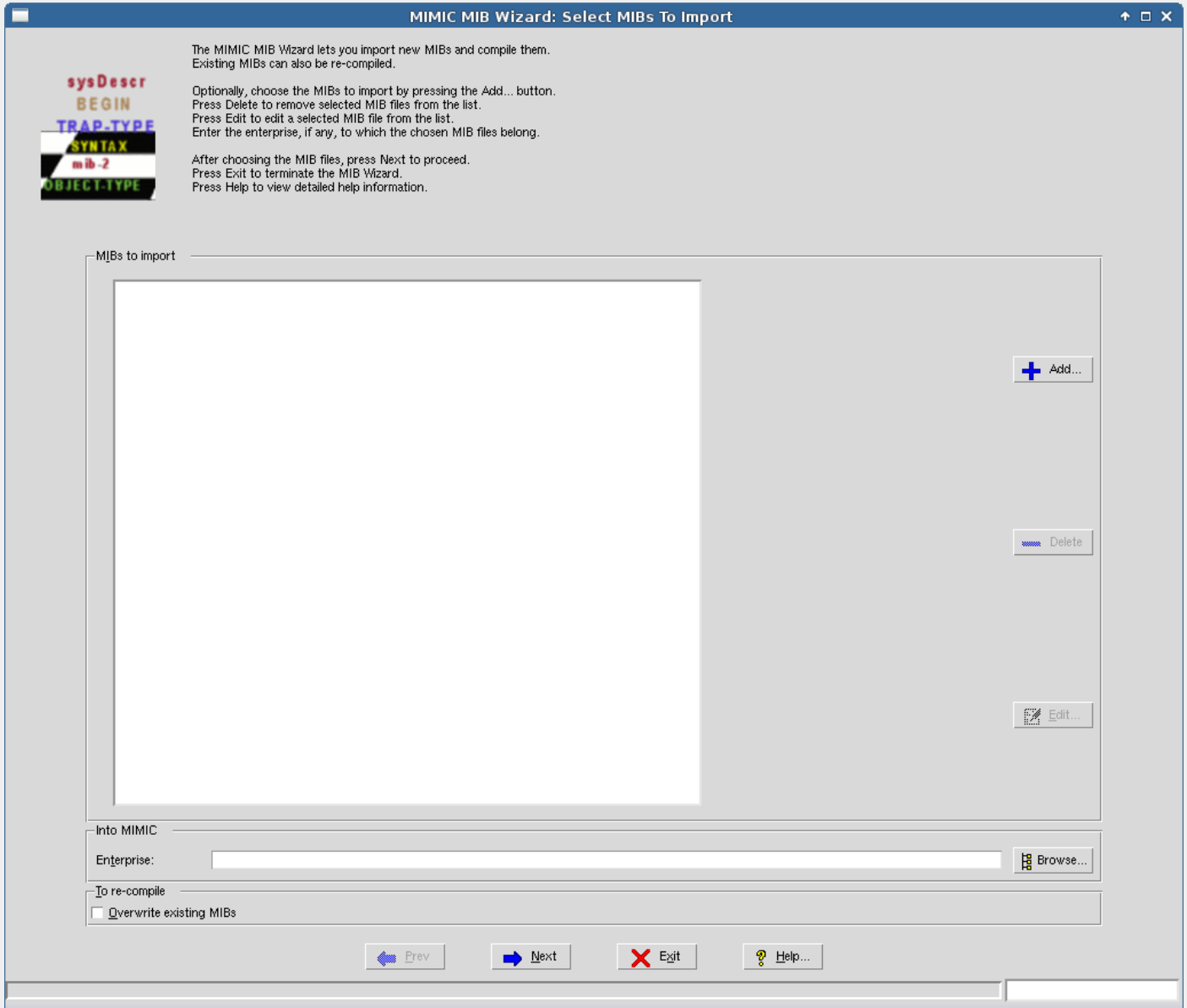


Figure 1: MIB Wizard: Select MIBs

The file selection dialog displayed by the MIB File Browse... button lets you select one or more MIB files to import.

Pick an enterprise name in the Enterprise field. This should be a short name for the enterprise of your enterprise-specific MIBs. For example, you can browse the existing enterprise names with the Browse... button.

By default, the MIB Wizard will automatically skip over MIBs that are already compiled into MIMIC. The `overwrite existing MIBs` field can be left unchecked, unless you indeed want to overwrite MIBs that were already compiled into MIMIC.

You need to click <<Add to add the MIBs to the list.

You can edit the list of MIBs further with the Delete button, and even edit MIB files with the Edit button.

Select Existing MIB Files To Compile

In this dialog, select the existing (imported into MIMIC) MIBs you want to recompile. You need to do this if a previous compile failed. This allows for an iterative process until the MIB compiles correctly.

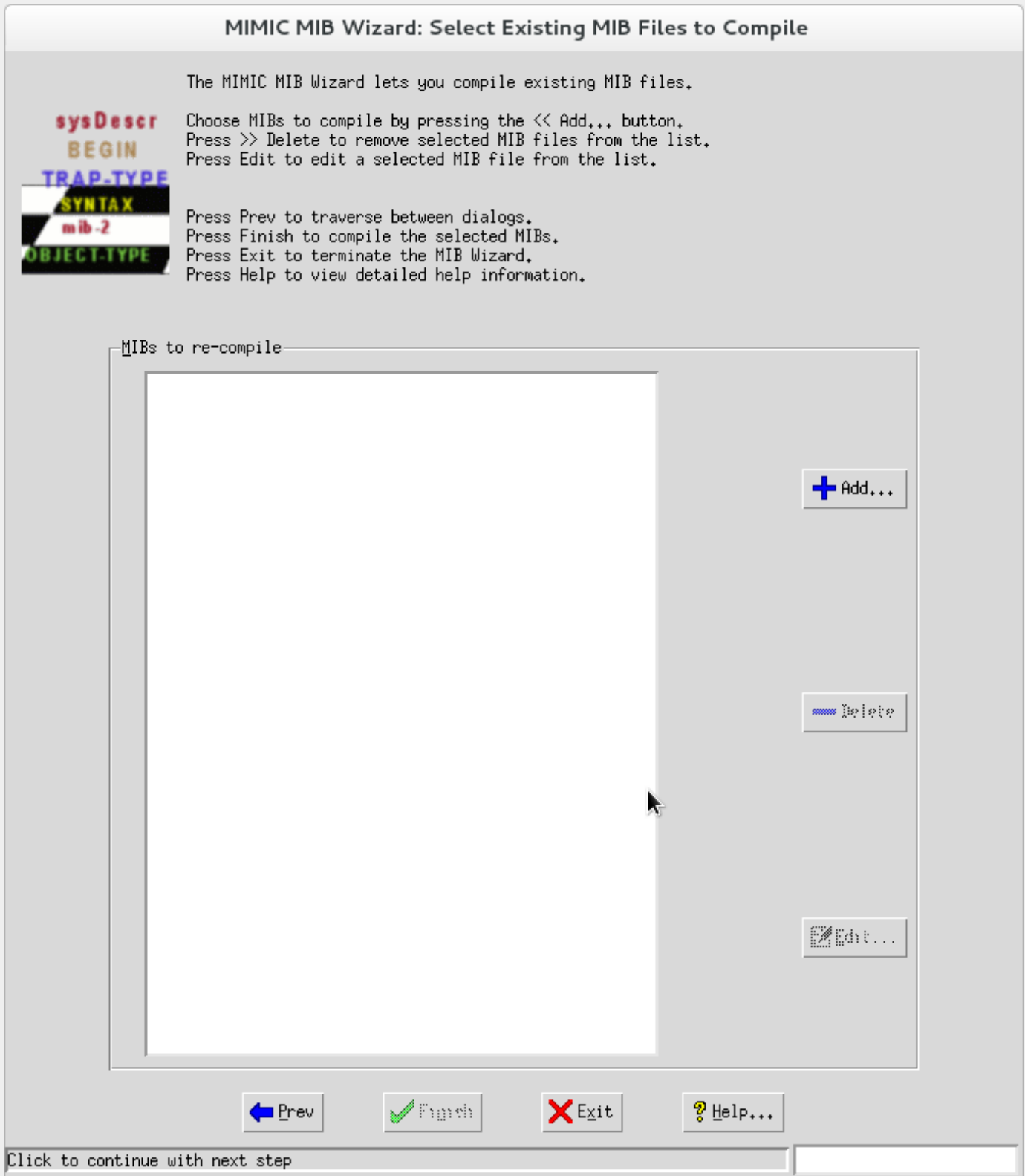


Figure 2: MIB Wizard: Existing MIBs

4. Discovery Wizard Reference

Overview

The Discovery Wizard allows you to create a network simulation by discovering SNMP Agents on a network and record them as simulations which can be played back later. It is a front-end to the [MIMIC Recorder](#), designed to record large networks.

In addition to recording from a live network, it can create network simulations from a group of previously recorded walkfiles.



Discovery Wizard Sessions

In this dialog, you can create a new session or select an existing session that you wish to use. The Discovery Wizard allows you to interrupt and continue discovery sessions. A new session will be useful to run the discovery at a remote site without creating the simulation, then continue and create the simulation at your local site by selecting that existing session.

Create New Session: perform a fresh discovery and record the simulation

Select Existing Session: resume the discovery or recreate the simulation

You can annotate your discovery in the `comments` field.

MIMIC Discovery Wizard: Select Session



The MIMIC Discovery Wizard helps you to discover SNMP agents on a network and record them as simulations. Those simulations can later be assigned to agents in the MIMIC Simulator.

Create New Session: perform a fresh discovery and record the simulations


Select Existing Session: resume the discovery or recreate the simulations

Press Next to proceed to the next step.
Press Exit to terminate the Discovery Wizard.
Press Help to view detailed help information.

Select Session

- Create new session
- Select existing session

Session name:

 Browse...

Session Description

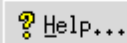
Comments:

I

 Prev

 Next

 Exit

 Help...

Specify a session name.

Figure 3: Discovery Wizard: Select Session

Discovery Method

In this dialog, select the discovery method. The Discovery Wizard supports three methods to generate the network simulation.

MIMIC Discovery Wizard: Discovery Method



Choose the method of discovery that you wish to use.

Method 1 : Scan for targets on the live network in a range of given ip addresses;

Method 2 : Use the given seed addresses and discover on the live network using the specified connectivity MIB table(s);

Method 3 : Record from previously collected walk files.

After choosing the discovery method:
Press Next to proceed to the next step.
Press Prev to start over.
Press Exit to terminate the Discovery Wizard.
Press Help to view detailed help information.

Discovery Method

- Method 1 : Traverse IP address ranges on live network
- Method 2 : Use seed address ranges on live network
- Method 3 : Import existing walkfiles

Seed record Options

Connectivity

- | | |
|--|---|
| <input checked="" type="checkbox"/> Use ARP table | <input checked="" type="checkbox"/> Use IP address table |
| <input checked="" type="checkbox"/> Use ROUTE table | <input checked="" type="checkbox"/> Use TCP connections table |
| <input checked="" type="checkbox"/> Use CDP cache | <input checked="" type="checkbox"/> Use LLDP Remote Data |
| <input checked="" type="checkbox"/> Use OSPF Neighbors | |

Filter

- Record IP aliases

← Prev

Next →

✖ Exit

? Help...

Click to continue with next step

Figure 4: Discovery Wizard: Discovery Methods

Choose **method 1** Traverse IP address ranges on live network, if you wish to discover SNMP Agents on the live network within specified ranges of IP addresses. This is useful to limit the discovery to a very specific set of addresses.

Choose **method 2** Use seed address ranges on live network, if you wish to discover SNMP Agents on the live network from the tables maintained by special "seed" agents (routers, hubs, etc) within the specified ranges of IP addresses. You may select any combination of the ARP, ROUTE, IP Address, TCP Connections, CDP Cache, LLDP Remote and OSPF tables. Discovery will traverse those MIB tables, and try to probe addresses pointed to by these tables. This method is useful if you have a large network, and only know addresses of the main routers.

NOTE: the TCP Connections table could be very large, and point to many systems to continue discovery.

NOTE: by default, the wizard will detect IP aliases of devices (multiple network interfaces), and will record only one instance of the agent through one of the interfaces. This will generate a correct simulation, assuming that the data presented through each of the interfaces is the same. Some devices present different MIBs through different interfaces, if so, you can select the option Record IP Aliases to record agents on all interfaces.

Choose **method 3** Import existing walkfiles, if you wish to create the network simulation from previously recorded walkfiles. This allows you to import walkfiles into MIMIC and generates a network simulation from them.


IP Addresses

In this dialog, enter the IP address ranges. If the Discovery method of "IP Address ranges" was chosen, these ranges will be used to discover the actual SNMP Agents. If the Discovery method of "Seed address ranges" was chosen, these ranges will be used to discover the seed agents. The chosen tables from these discovered seed targets will then be used for discovering the actual SNMP Agents.

MIMIC Discovery Wizard: Enter Seed Addresses

Use the <<Add button to add a new seed address range to the list.
Use the >>Delete button to delete any address range already added.

Press Next or Prev to traverse between dialogs.
Press Exit to terminate the Discovery Wizard.
Press Help to view detailed help information.



Target Addresses

Seed address: to: or

Parallel Processing

Simultaneous ping:

Simultaneous probe:

Specify the starting address .

Figure 5: Discovery Wizard: Seed Addresses

Rather than inputting the addresses by hand, you can feed a file containing the addresses in the From File... dialog. The file needs to have one address or one address range per line.

In this dialog, you may also choose the number of simultaneous Pings to use for the discovery process.

Filters to Limit Discovery

In this step you may optionally specify Include and Exclude patterns. These patterns will be used to determine if an IP address from the specified ranges will be included or excluded.

MIMIC Discovery Wizard: Filters to Limit Discovery



You can specify filters to include or exclude patterns of addresses. If no patterns are defined, all discovered targets will be recorded. The user-defined filter script is run on every discovered target.

Use the <<Add button to add a pattern to include or exclude address. Use the >>Delete button to delete existing include or exclude pattern(s).

Press Next or Prev to traverse between dialogs. Press Exit to terminate the Discovery Wizard. Press Help to view detailed help information.

Include Filters

String:

Exclude Filters

String:

Filter script:

Click to browse filter script.

Figure 6: Discovery Wizard: Filters

The patterns can use the special characters "*" and "?". The "*" would match 1, 2 or 3 digits. The "?" would match a single digit.

The filter script lets you define a TCL script, which will dynamically be loaded into the Wizard to do advanced filtering of discovered devices. This lets you do incremental discovery, for example. MIMIC ships with sample filter scripts (scripts/discwiz_filter*.mtcl).

SNMP Parameters

In this dialog, you may enter the SNMP parameters to use.

MIMIC Discovery Wizard: SNMP Parameters

Enter SNMP parameters.

Use the <<Add button to add a port or community to the lists.
Use the >>Delete button to delete an existing port or community.

Press Next or Prev to traverse between dialogs.
Press Exit to terminate the Discovery Wizard.
Press Help to view detailed help information.



General | **SNMPv1/2c**

Timeout (msecs) Retries

Ports

Port:

Version

v1 v2c v3

Click to continue with next step

Figure 7: Discovery Wizard: SNMP Parameters

In the General tab you may change the Timeout (in msec) and Retries. You may specify any number of ports, which will be probed in turn for each target, until one of them returns a valid response. You can select any of the supported SNMP versions, which will be used in turn for each of the selected ports for each target.

In the SNMPv1/v2c tab you specify the read community names to try. The default community is "public".

In the SNMPv3 tab you specify the user names and contexts to try.

Recorder Options

In this dialog, you may specify the options to be used with the [MIMIC Recorder](#).

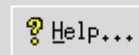
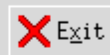
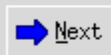
MIMIC Discovery Wizard: Recorder Options

Customize recording using the MIMIC Recorder options.



Press Next to start discovery or Prev to go back.
Press Exit to terminate the Discovery Wizard.
Press Interrupt to interrupt discovery and proceed to next step.
Press Help to view detailed help information.

MIB Parameters	
Exclude :	<input type="text"/>
Start:	<input type="text"/>
Root:	<input type="text"/>
Max table:	<input type="text" value="500"/>
Record Options	
Simulation method:	<input type="radio"/> Constant <input checked="" type="radio"/> Random
<input type="checkbox"/> Create Source	<input type="checkbox"/> No Simulation
Manual Intervention	
<input type="checkbox"/> Edit Topology	<input checked="" type="checkbox"/> Edit target list before recording
Simultaneous records:	<input type="text" value="10"/>



Specify start parameters.

Figure 7: Discovery Wizard: Recorder Options

Optionally, specify a start Oid. By default, the Recorder walks the entire MIB exported by the target agent. This options lets you specify the starting OID to start recording. This is useful if you want to restart a recording that previously failed for any reason.

Optionally, specify the root Oids. By default, the Recorder walks the entire MIB exported by the target agent. This options lets you specify MIB subtrees to record. This option is useful if you are interested in only a small part of the MIB.

Optionally, specify the Exclude Oids. By default, the Recorder walks the entire MIB exported by the target agent. This option is useful if you want to exclude parts of the MIB from being simulated. For example, some MIB tables (such as RMON history tables) can be extremely dynamic (and large). Other tables implement proprietary retrieval schemes (e.g., getTable). Some agents even crash when you access certain parts of the MIB. This option allows to skip these MIB subtrees.

Optionally, specify the Max Table size. Selecting this option limits the size of large tables to the specified number of table entries. It applies both at discovery

time, where only the selected number of entries are traversed, and at simulation time, where only the selected number of entries are simulated. This is useful for agents with large tables, such as RMON tables, etc.

Choose the `Simulation Method`. This option allows two choices: random or constant, to specify which of two simulation methods to use for counter simulations.

Optionally, choose to `Create Source`. Selecting this option generates the simulation source file so that you can change the simulation later on.

Selecting the `No Simulation` option skips the simulation phase and only generates a walk file. This is useful for example if you are at a remote site, and want to generate the simulation later.

Selecting the `Edit target list before recording` allows you edit the list of targets before actually performing the recording.

When you press `Next>` the discovery starts. A `Log Windows` displays results of the discovery and the progress bar shows the progress.

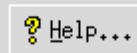
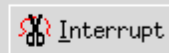
MIMIC Discovery Wizard: Discovery Progress

This dialog shows current statistics of the discovery process.



Press `Next` to start discovery or `Prev` to go back.
Press `Exit` to terminate the Discovery Wizard.
Press `Interrupt` to interrupt discovery and proceed to next step.
Press `Help` to view detailed help information.

```
Number of total targets: 272  
Number of pinged targets: 17  
Number of probed targets: 11  
Number of discovered devices: 0
```



Determining SNMP access of 10.19.43.247...

4%

Figure 8: Discovery Wizard: Progress

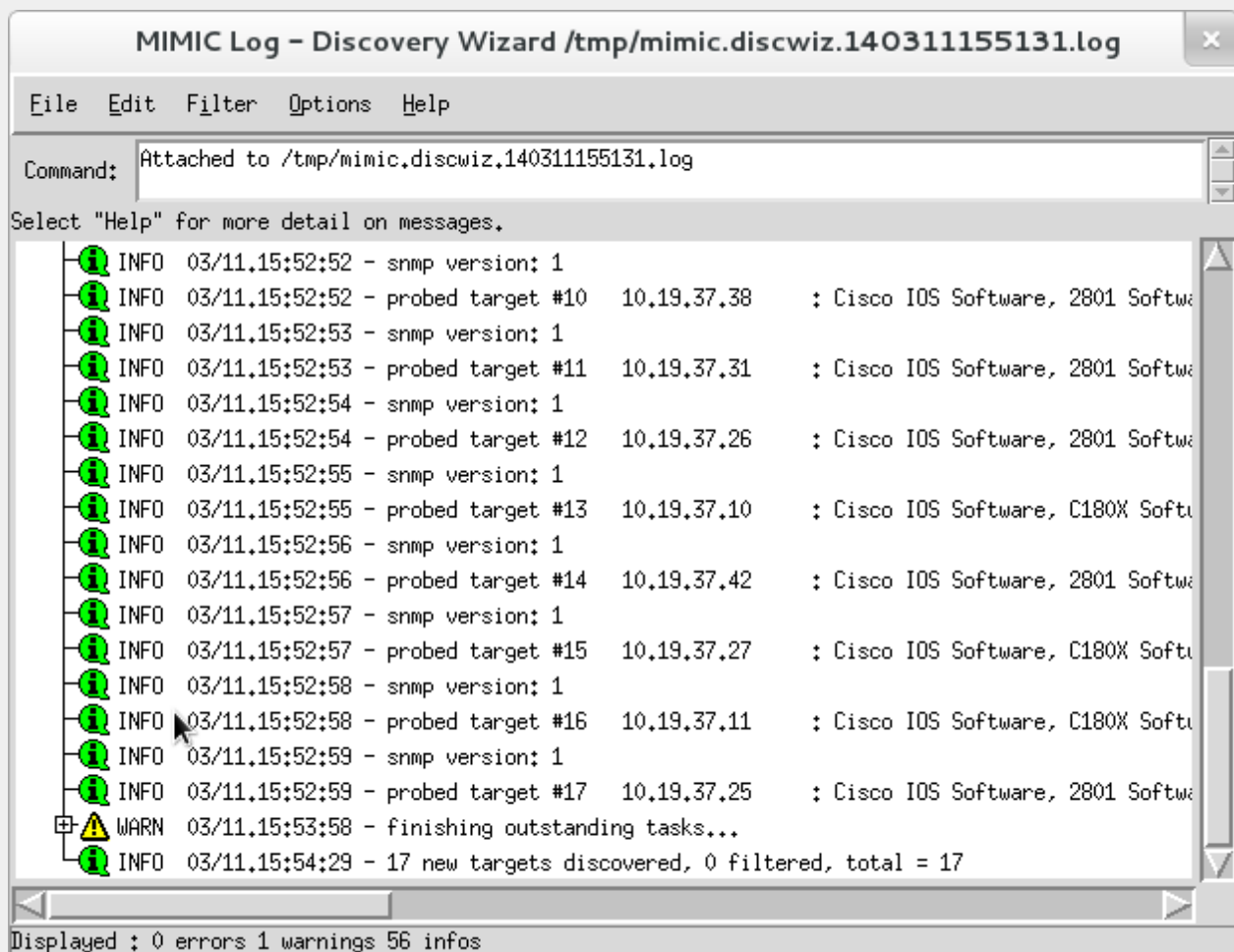


Figure 9: Discovery Wizard: Log

Select Walkfiles

This dialog allows you to specify a group of previously recorded walkfiles to import into the Discovery Wizard in lieu of live recording of devices.

Discovered Targets

In this dialog, the discovered SNMP Agents are displayed in a listbox. You may, optionally, delete some of them.

MIMIC Discovery Wizard: Record Targets

Edit list of discovered targets.

Use the >>Delete button to delete selected target(s) from the list of discovered targets.

Press Next to start recording.
Press Exit to terminate the Discovery Wizard.
Press Interrupt to interrupt the recording.
Press Help to view detailed help information.



Targets: 17 discovered, 0 filtered, 17 shown

# [+]	Address	Description	Status
1	10.19.37.15	Cisco IOS Software, C180X Software (C180X-ADVIPSERV	to be recorded
2	10.19.37.43	Cisco IOS Software, 2801 Software (C2801-IPBASEK9-M	to be recorded
3	10.19.37.16	Cisco IOS Software, 2801 Software (C2801-SPSERVICES	to be recorded
4	10.19.37.28	Cisco IOS Software, 2801 Software (C2801-SPSERVICES	to be recorded
5	10.19.37.29	Cisco IOS Software, 2801 Software (C2801-SPSERVICES	to be recorded
6	10.19.37.35	Cisco IOS Software, C180X Software (C180X-ADVIPSERV	to be recorded
7	10.19.37.20	Cisco IOS Software, 2801 Software (C2801-IPBASEK9-M	to be recorded
8	10.19.37.44	Cisco IOS Software, 2801 Software (C2801-ADVIPSERVI	to be recorded
9	10.19.37.30	Cisco IOS Software, 2801 Software (C2801-IPBASEK9-M	to be recorded
10	10.19.37.38	Cisco IOS Software, 2801 Software (C2801-IPBASEK9-M	to be recorded
11	10.19.37.31	Cisco IOS Software, 2801 Software (C2801-IPBASEK9-M	to be recorded
12	10.19.37.26	Cisco IOS Software, 2801 Software (C2801-IPBASEK9-M	to be recorded
13	10.19.37.10	Cisco IOS Software, C180X Software (C180X-ADVIPSERV	to be recorded
14	10.19.37.42	Cisco IOS Software, 2801 Software (C2801-ADVIPSERVI	to be recorded
15	10.19.37.27	Cisco IOS Software, C180X Software (C180X-ADVIPSERV	to be recorded
16	10.19.37.11	Cisco IOS Software, C180X Software (C180X-BROADBAND	to be recorded
17	10.19.37.25	Cisco IOS Software, 2801 Software (C2801-SPSERVICES	to be recorded

Delete

Prev

Next

Exit

Help...

Figure 10: Discovery Wizard: Discovered Targets

To record the discovered SNMP Agents, press Next>. The Log Window displays the results of each recording.

In real-time, the status column shows the current status of each target, one of to be recorded, recording or recorded.

Recorded Targets

The Discovery Wizard allows to record agents at one site, then resume the simulation creation at another. This is accomplished by selecting No Simulation in the original session, then resuming it. In this dialog, the recorded SNMP Agents in a resumed session are displayed in a listbox. You may, optionally, exclude some of them by deleting them from the list. Simulations will not be regenerated for excluded targets.

To specify recorder options for regenerating simulations, press Next>.

Create Lab Configuration


Once the targets are recorded, they are categorized into the [device catalog](#) in a new category under Devices -> Discovered with the same name as the discovery session named [above](#).

Then you can save the recorded lab configuration so that you can later load it into the Simulator.

Interrupted Session

In this dialog, the summary of user inputs for the last interrupted session as well as state of the engine at which the engine was interrupted is shown.


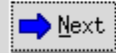


MIMIC Discovery Wizard: Interrupted Session



The selected session was interrupted.
Following are the details of the last session

Press Next to resume interrupted Session.
Press Prev to select another Session.
Press Exit to terminate the Discovery Wizard.
Press Help to view detailed help information.

```
State of the Engine =probing
Discovery Method    =2
Seed Address Range=10,19,37,1-10,19,37,100
Include pattern    =
Exclude pattern    =
Filter Script      =
Protocol List      =SNMPv1 SNMPv2c
Community List     =public
Username List      ={}
Context Name List  ={}
Port List          =161
```

 Prev Next Exit Help...

Click to continue with next step

Figure 11: Discovery Wizard: Interrupted Session

The wizard continuously checkpoints its state, so that on any interruption it can continue where it left off, as if it had never been interrupted.

Snapshot Wizard Reference

Overview

The Snapshot Wizard makes it easy to take multiple snapshots of a device over time and automatically change the simulation to account for changes. This is specially useful if you take an initial snapshot, make some changes on your device (reconfigure it, such as take cards in/out of service, make operational changes such as higher loads, etc), then take another snapshot after the change. The Wizard will automatically detect the changes (different MIB object values, new/deleted table entries, different counter rates, etc), and create timer scripts to change the simulation. In addition to manual snapshots (like a camera click), you can also take periodic snapshots, such as every hour, to capture long-term trends.



Snapshot Option

You can create a new snapshot series, or add new snapshots to an existing series. To create a new snapshot, you need to give it a new name that you can remember, preferably a short word or phrase, like `test1`, or `cisco5k-under-stress`.

Snapshot Wizard: Snapshot Option



The MIMIC Snapshot Wizard lets you record a device over time.

Choose the operating method:

- create a new snapshot and simulate a device;
- modify an existing snapshot and simulate a device.

Press Next to proceed to the next step.

Press Exit to terminate the Snapshot Wizard.


Press Help to view detailed help information.

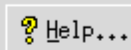
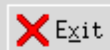
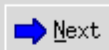
Select Snapshot

Create a new snapshot

Modify an existing snapshot

Snapshot:

 Browse...




Specify a Snapshot Name

Figure 12: Snapshot Wizard: Option

Simulation Option

You can create a new simulation, apply the snapshots to an existing simulation, or copy an existing simulation and apply the snapshots to it. To create a new simulation, you need to give it a new name that you can remember, preferably a short word or phrase, like `test1`, or `cisco5k`.

Snapshot Wizard: Simulation Option



Choose the method of simulation;

- create a new simulation;
- apply snapshot to a prerecorded simulation;
- copy an existing simulation and apply the snapshots to it

Press Next to proceed to the next step.
Press Exit to terminate the Snapshot Wizard.
Press Help to view detailed help information.

Select Simulation

- Create a new Simulation
- Apply to an existing simulation
- Copy an existing simulation

Simulation:

Specify a Simulation Name

Figure 13: Snapshot Wizard: Simulation

Choose a Live Target

Specify the mandatory target parameters to be recorded. See the [MIMIC Recorder Guide](#) for more details.

Snapshot Wizard: Choose a Live Target



Specify the target device IP-address, port, SNMP version and the related SNMP parameters. These parameters are needed to be able to poll SNMP information using the MIMIC Recorder.

Press Next to proceed to the next step.
Press Exit to terminate the Snapshot Wizard.
Press Help to view detailed help information.

General **SNMPv2c**

Target

Address:

Port:

SNMP: v1 v2c v3

Device

Simulation:

Scenario:

Prev

Next

Exit

Help...

Enter an IP address of target device to poll SNMP information

Figure 14: Snapshot Wizard: Target

Recorder Options

These optional parameters modify the default recording. See the [MIMIC Recorder Guide](#) for more details.

Snapshot Wizard: Recorder Options



Optionally provide additional input to be used to record the device and create the base simulation.

Press Next to proceed to the next step.
Press Exit to terminate the Snapshot Wizard.
Press Help to view detailed help information.

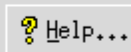
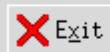
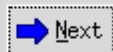
MIB Parameters

Exclude:	<input type="text"/>
Start:	<input type="text"/>
Root:	<input type="text"/>
Max table:	<input type="text"/>

Record Options

Simulation method: Constant Random

Create source



Click to continue with next step

Figure 15: Snapshot Wizard: Recorder

Once you click Next, the MIMIC Recorder will record the target. To continue, dismiss the log viewer window that pops up.

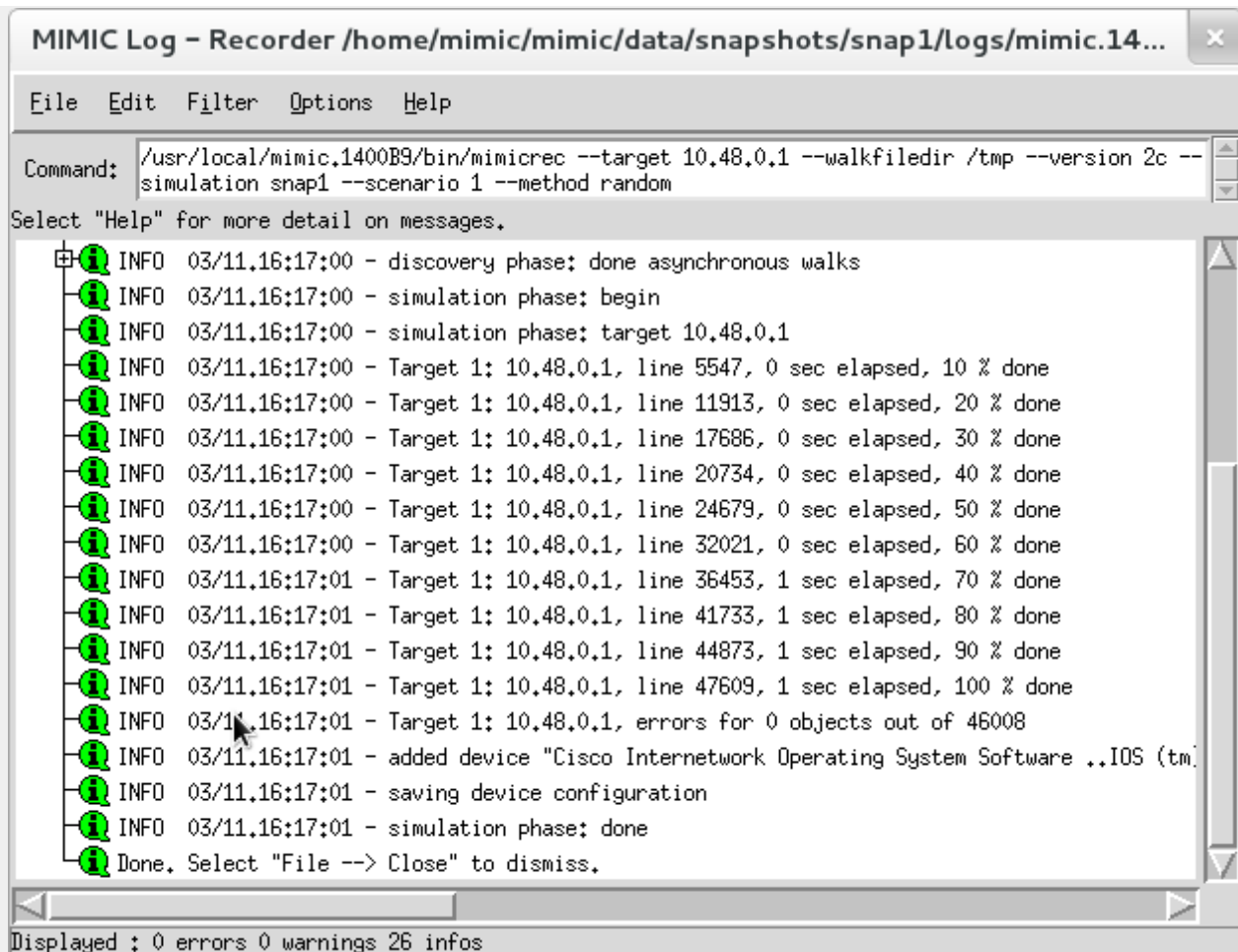


Figure 16: Snapshot Wizard: Log

Snapshot Constraints

From here on you will take snapshots of the device at different times. But, your snapshots may not need to record the entire device, if you know that the changes will be confined to a particular area. In this dialog you can impose constraints on the snapshots.

Snapshot Wizard: Snapshot Constraints

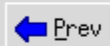


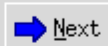
Optionally provide additional constraints to be used for limiting the subsequent snapshot size.

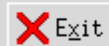
Press Next to proceed to the next step.
Press Exit to terminate the Snapshot Wizard.
Press Help to view detailed help information.

MIB Parameters

Exclude:	<input type="text"/>
Start:	<input type="text"/>
Root:	<input type="text" value="ifTable"/>
Max table:	<input type="text" value="500"/>

 Prev

 Next

 Exit

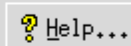
 Help...

Figure 17: Snapshot Wizard: Constraints

These optional constraints further restrict all subsequent snapshots. If the snapshot is of the entire MIB, that is very expensive both in terms of the amount of data, and the computing to detect differences. It is usually better to concentrate on MIB subtrees (Root), and/or exclude large MIB subtrees.

Take Snapshots

You can take an immediate snapshot or schedule periodic snapshots.

Snapshot Wizard: Take Snapshots



Use the following options to take one snapshot at a time, or schedule a series of snapshots with specified interval and iterations.

Press Record to take snapshot(s).
Press Interrupt to stop snapshot(s).

Press Next to the next step.
Press Exit to terminate the Snapshot Wizard.
Press Help to view detailed help information.

Snapshot Type

Immediate snapshot
 Scheduled snapshots

Start Time

Current: Tue Mar 11 04:28:53 PM EDT 2014
 Relative: seconds
 Absolute: : : HH:MM:SS

Take Snapshots

Iterations:
Interval: seconds

Comment
this is a sample snapshot

Record...

Specify the comment

Figure 18: Snapshot Wizard: Take Snapshots

If you select `Immediate Snapshot`, when you press the `Record...` button the snapshot will happen immediately. This is equivalent to pressing the snapshot button on a camera. Optionally, you can input an informative `Comment` for your snapshot.

When you choose `Scheduled Snapshots`, you can use the `Start Time` frame and `Take Snapshots` frame to schedule your periodic snapshots.

Create Change Script

This dialog allows you to edit the sequence of snapshots you took. Once you click `Finish`, a script will be created in the simulation directory to transition the simulation through the desired changes.

You need to select at least 2 walkfiles to create change script(s).

Snapshot Wizard: Creating Change Script



Select walkfiles that you want to simulate in the simulation. The wizard will create change script(s) from each selected walkfile(s) to the next one.

Press Finish to create change scripts(s) (May take a while).
Press Exit to terminate the Snapshot Wizard.
Press Help to view detailed help information.

Walkfiles

- snap1
 - walkfile 0
 - time: will not be simulated
 - comment: this is an immediate snapshot
 - walkfile 1
 - time: will not be simulated
 - comment: this is a sample snapshot
 - walkfile 2
 - time: start
 - comment: initial snapshot
 - walkfile 3
 - time: + 16 seconds
 - comment: we took a network card out of service

Edit...

Delete

Prev Finish Exit Help...

Select walkfiles to simulate in the simulation

Figure 19: Snapshot Wizard: Create Script

Apply Change Script Manually

By default, after you apply a snapshot series to a simulation device, all changes occur automatically in the background at the same pace as the recorded device. The change script(s) are change.{0,1,2...}.mtcl file(s) in the same device simulation directory, like these,

```
$ ls data/sim/procurve530/
BRIDGE-MIB
change.0.mtcl
change.1.mtcl
dev.cfg
HOST-RESOURCES-MIB
hp-HP-PROCURVE-WLAN-AP-MIB
hp-HP-PROCURVE-WLAN-AR-MIB
hp-HP-PROCURVE-WLAN-SYSTEM-MIB
hp-HP-WLAN-SFLOW-EXTENSIONS-MIB
IF-MIB
invoke.0.mtcl
invoke.1.mtcl
invoke.2.mtcl
IPV6-MIB
microsoft-LanMgr-Mib-II-MIB
OLD
RFC1213-MIB
SFLOW-MIB
SNMP-FRAMEWORK-MIB
SNMP-MPD-MIB
SNMP-USER-BASED-SM-MIB
SNMPv2-MIB
SNMP-VIEW-BASED-ACM-MIB
start+snapshot.mtcl
stop+snapshot.mtcl
TCP-MIB
```

IP-FORWARD-MIB

UDP-MIB

\$

To manually apply the changes at your own pace, you can use MIMICShell to invoke the change script(s), such as this,

```
$ ./mimicsh --nogui --agent 4 --script ~/mimic/data/sim/procurve530/change.1.mtcl
```

You can also copy the change.*.mtcl script(s) to the private area's scripts directory and use the Agent ->Script window to invoke the change script(s).

Trap Wizard Reference

Overview

The Trap Wizard allows you to capture a series of traps from a target device and creates scripts that will play them back in the Simulator. This consists of 2 distinct recording processes: first to record the MIB of the live target device to create the base simulation, then to capture the generated traps from the device.

While the initial MIB recording is a one-time process, the capture of traps can be done multiple times. You could for example cause your device to generate a sequence of traps for a certain reason (eg. some abnormal condition), verify this is what you captured, then generate another sequence for the restoration to normal operations.

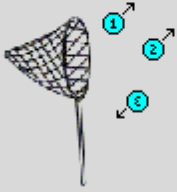
You can further use the MIMICView [Agent ->Generate Traps...](#) dialog to manually generate any trap for a group of agents.



Series Options

The end-result of the Trap Wizard is a `Trap Series` that will be played back in a simulation. This playback is implemented as a group of [Timer Scripts](#) that will be invoked upon starting an agent that runs the simulation.

Trap Wizard: Series Options



The MIMIC Trap Wizard allows you to create a trap series by capturing trap sequences generated by your real device and configuring a simulation to generate these traps.

Choose the operating method:

- start a fresh trap series regardless of the past;
- modify an existing trap series.

Press Next to proceed to the next step.

Press Exit to terminate the Trap Wizard.

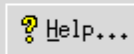
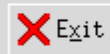
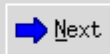
Press Help to view detailed help information.

Select Trap Series

Create a new trap series

Modify an existing trap series

Trap Series:



Specify a trap series name

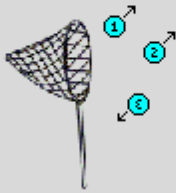
Figure 20: Trap Wizard: Options

This dialog lets you start a fresh series or modify an existing one. To create a new series, you need to give it a new name that you can remember, preferably a short word or phrase, like `test1`, or `cisco5k-booting`.

Simulation Options

The trap series will be applied to a simulation. This dialog lets you specify a fresh simulation, an existing one, or a copy of an existing one. If you create a new simulation, the [MIMIC Recorder](#) will be invoked to create it. If you copy an existing simulation, the original is not touched.

Trap Wizard: Simulation Options



You can apply the trap series to an existing simulation or record a new simulation.

Choose the method of simulation:

- record a device and create a new simulation;
- apply the trap series to an existing simulation;
- copy an existing simulation and use it.

Press Next to proceed to the next step.


Press Exit to terminate the Trap Wizard.


Press Help to view detailed help information.

Select Simulation


- Create a new Simulation
- Apply to an existing simulation
- Copy an existing simulation


Simulation Name:

 Browse...

 Prev

 Next

 Exit

 Help...

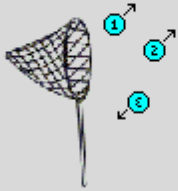
Specify a name for the simulation

Figure 21: Trap Wizard: Simulation

Choose a Live Target

If you selected to create a new simulation, then this dialog lets you specify the mandatory target parameters, such as its address, version of SNMP, etc. These parameters will be used by the [MIMIC Recorder](#) to record the target.

Trap Wizard: Choose a Live Target



Specify the target device IP-address, port, SNMP version and the related SNMP parameters. These parameters are needed to be able to poll SNMP information using the MIMIC Recorder.

Press Next to proceed to the next step.
Press Exit to terminate the Trap Wizard.
Press Help to view detailed help information.

General | **SNMPv2c**

Target

Address:

Port:

SNMP: v1 v2c v3

Device

Simulation:

Scenario:

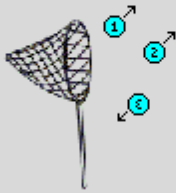
Specify a target address for traps

Figure 22: Trap Wizard: Target

Recorder Options

These optional parameters modify the recording of the MIB. See the [MIMIC Recorder Guide](#) for more details.

Trap Wizard: Recorder Options



Optionally provide additional input to be used to record the device and create the simulation.

Press Next to proceed to the next step.
Press Exit to terminate the Trap Wizard.
Press Help to view detailed help information.

MIB Parameters

Exclude:

Start:

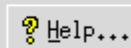
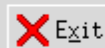
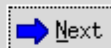
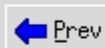
Root:

Max table:

Recorder Options

Simulation Method: Constant Random

Create Source



Click to continue with next step

Figure 23: Trap Wizard: Recorder

Once you click Next, the MIMIC Recorder will record the target. Dismiss the Log Windows to continue.

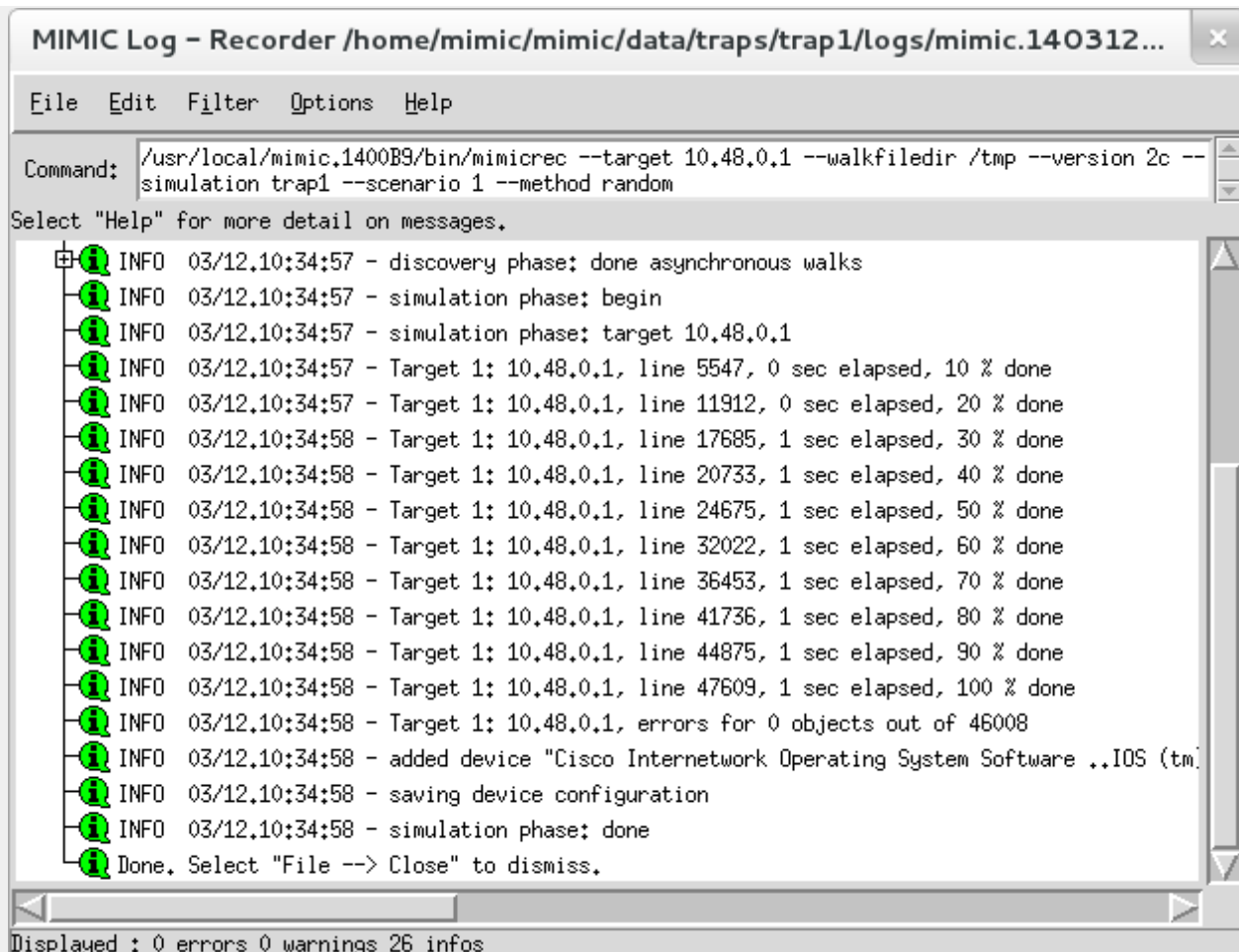


Figure 24: Trap Wizard: Log

Trap Capture Options

You can select to capture traps live, by invoking the [Trap Recorder](#) with these optional parameters to control the capture of traps. See [Record Traps](#) for more details.

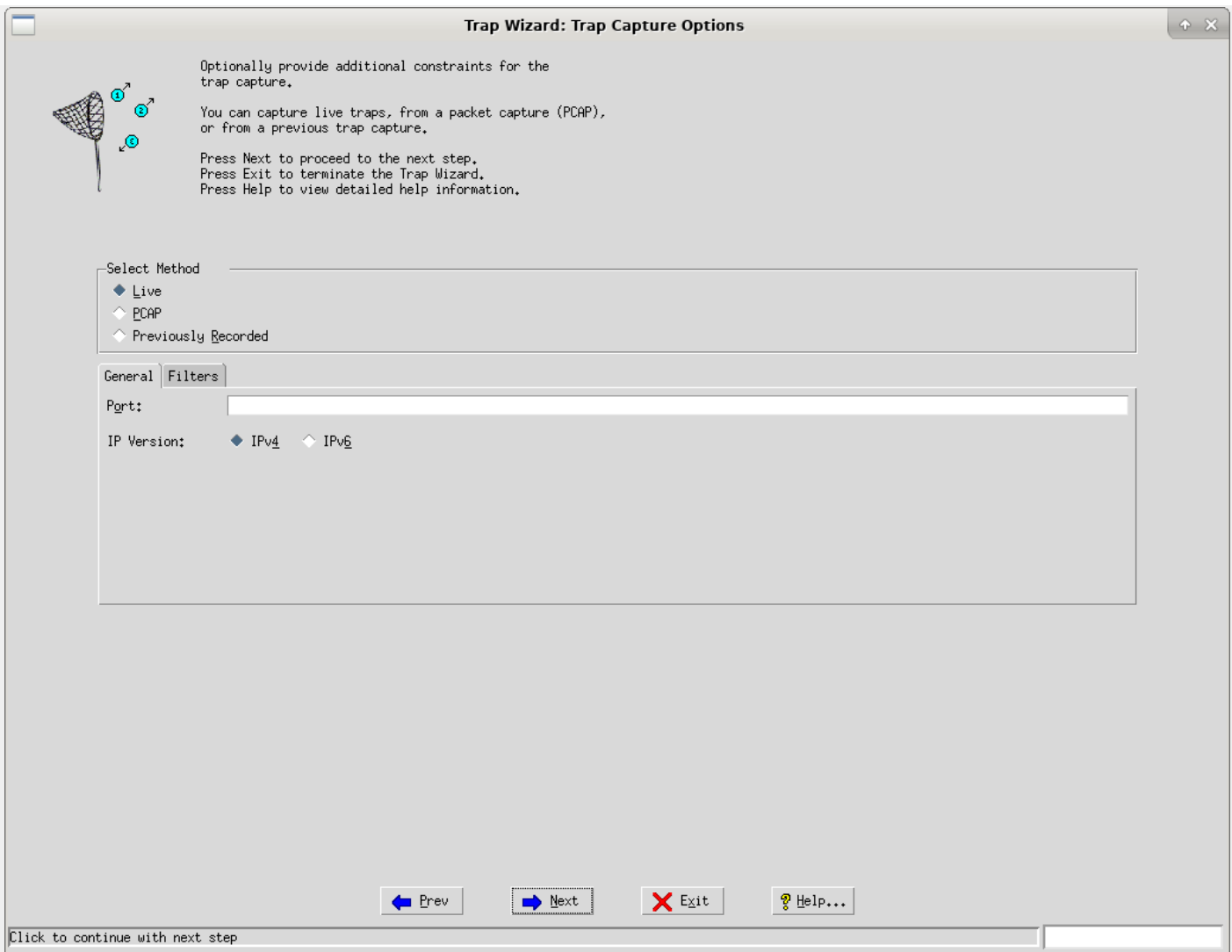


Figure 25: Trap Wizard: Capture

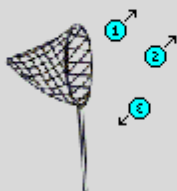
Instead of a live capture you can record from a PCAP file that contains captured TRAP messages. Through this wizard, all TRAP messages in this PCAP are recorded, but the `trapper` tool has filtering options. If you want to filter certain traps only, then you need to either use the `trapper` from the command line, or filter your PCAP in a utility like `wireshark`.

Otherwise, if you previously recorded traps with the `Trap Recorder`, then the wizard can read the `Trap Recorder` output. This way, you can deploy the `Trap Recorder` remotely, capture traps, and import them back into the MIMIC installation.

Capture Trap Series

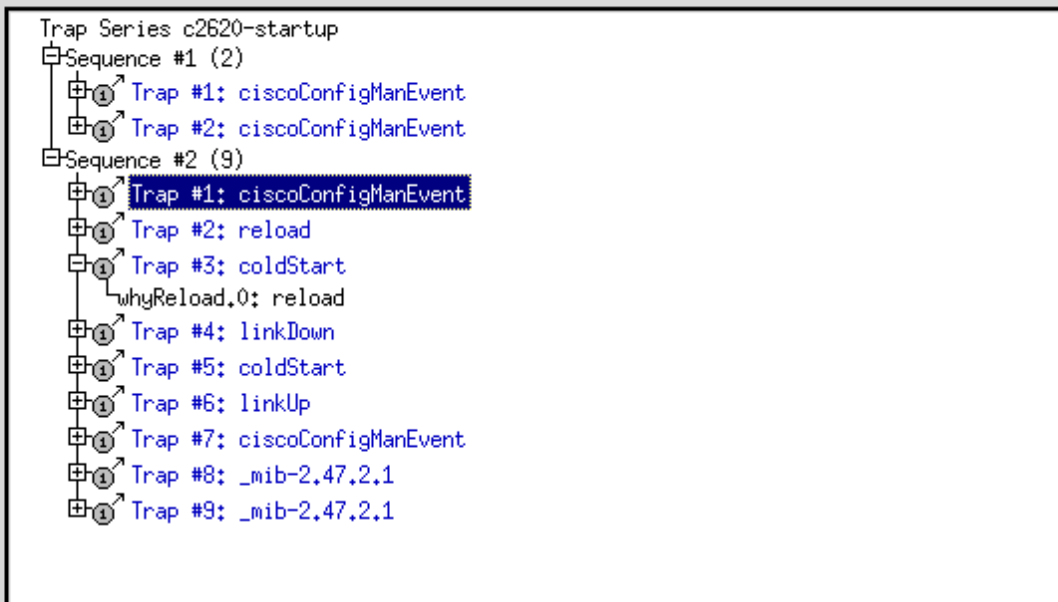
This dialog is the heart of the wizard, and lets you capture multiple sequences of traps for this series. The capture can be delayed, and its duration controlled. When you press the `Capture...` button, then the capture starts and captured traps are displayed in real-time. The capture stops when you press the `Interrupt` button or when the `Duration` expires.

Trap Wizard: Capture Trap Series



You can capture multiple sequences for this series.
Delay the capture by setting "Delay" and/or control
the duration of the capture by setting "Duration".

Press Capture to capture a trap sequence.
Press Interrupt to stop capture.
Press Next to parse capture logs and display series editor.
Press Exit to terminate the Trap Wizard.
Press Help to view detailed help information.



Capture Trap Series

Delay (secs):

Duration (secs):

Comment

Capture...

Interrupt

Prev

Next

Exit

Help...

Click to select

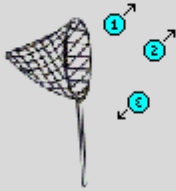
Figure 26: Trap Wizard: Series

The traps displayed in this dialog are all the captured traps, whether deleted or not. You can delete entire sequences / individual traps in the next dialog, and select / deselect them to be generated.

Edit Trap Series

Once you are done with the capture, you can edit the trap series that you want to simulate with this dialog.

Trap Wizard: Edit Series



Edit the trap sequences to customize the generation of traps. You can reorder the sequences or traps with the "Up" and "Down" buttons, edit their parameters with "Edit..." or delete them with "Delete".

The wizard will then create script(s) to generate each selected trap in the Simulator.

Press Next to create script(s).
Press Exit to terminate the Trap Wizard.
Press Help to view detailed help information.

Trap Series

- Trap Series: c2620-startup (2)
 - Sequence #2 (9)
 - Time: +0 msec
 - Comment:
 - Traps (9)
 - Trap #3: coldStart
 - Object: 0.1.3.6.1.2.1.11.0
 - Time: +8 seconds
 - Version: 1
 - Bindings
 - Parameters
 - Trap #4: linkDown
 - Object: 0.1.3.6.1.2.1.11.2
 - Time: +8 seconds 790 msec
 - Version: 1
 - Bindings
 - Parameters
 - Trap #5: coldStart
 - Object: 0.1.3.6.1.2.1.11.0
 - Time: +8 seconds 800 msec
 - Version: 1
 - Bindings
 - Parameters
 - Trap #6: linkUp
 - Object: 0.1.3.6.1.2.1.11.3
 - Time: +6 seconds 80 msec

Buttons: Up, Down, Edit..., Delete

Buttons: Prev, Next, Exit, Help...

Click to continue with next step

Figure 27: Trap Wizard: Edit Series

You can select the sequences and individual traps that you want the wizard to generate scripts for. Click on a node with a checkbox to select/deselect the item (and it's children). For example, selecting a sequence selects all the traps in the sequence.

You can delete sequences that you don't want to consider in the future, eg. if the sequence is empty. Click on a sequence, and press the >>Delete button.

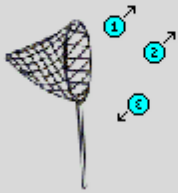
You can edit parameters of sequences (Time, Comment) and traps (Time, Version). Click on the node, then press the edit button to change the value. Changed items are shown in red.

The up and down buttons move the selected trap up or down in the series, thus changing the order of the generated traps.

Play Back Testing

When the scripts are created, you can play back the traps on your simulated device.

Trap Wizard: Play Back Testing



Now you can configure an agent with the simulation "trap1" and play back the trap sequence.

After you configure the agent and start the agent the trap sequence will, by default, play back automatically.

You can click Prev to modify the trap sequence and then come back here to test again. Otherwise, click Finish to close the Trap Wizard.

Press Exit to terminate the Trap Wizard.
Press Help to view detailed help information.



Figure 28: Trap Wizard: Play Back

When you start an agent with that simulation, you can control the generation of the traps with the [Agent->Trap Series...](#) dialog.

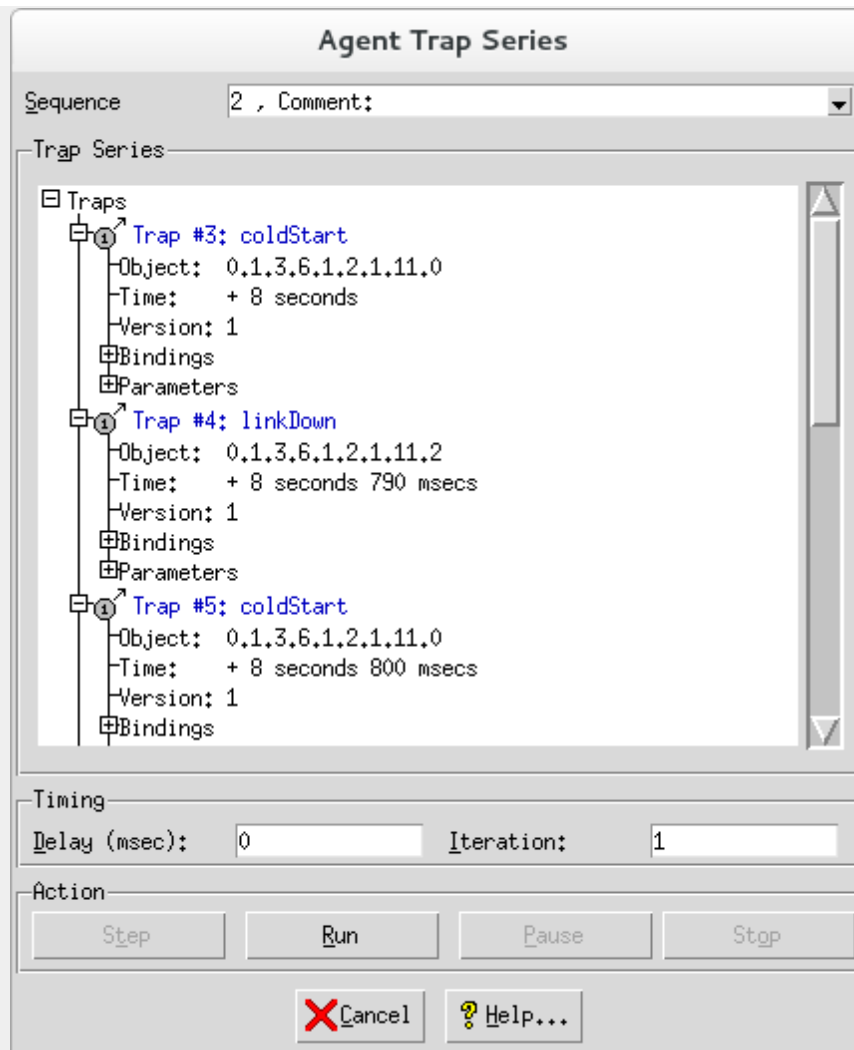


Figure 29: Trap Wizard: Run

Simulation Wizard Reference

Overview

The Simulation Wizard is designed to easily create a basic simulation from a set of walkfiles and MIBs. It is a user-friendly, graphical front-end to the [mib2walk](#) utility, and the [Record from File](#) functionality.

Through a graphical interface, you define instances and values for each of the MIB objects and save them in a walkfile, which is fed to the MIMIC Recorder to create the simulation.



Select Walkfile To Load

By loading an optional walkfile, you seed the simulation with instances and values of your MIB objects. This walkfile can be from a previous live recording with the [MIMIC Recorder](#), or saved by the Simulation Wizard.

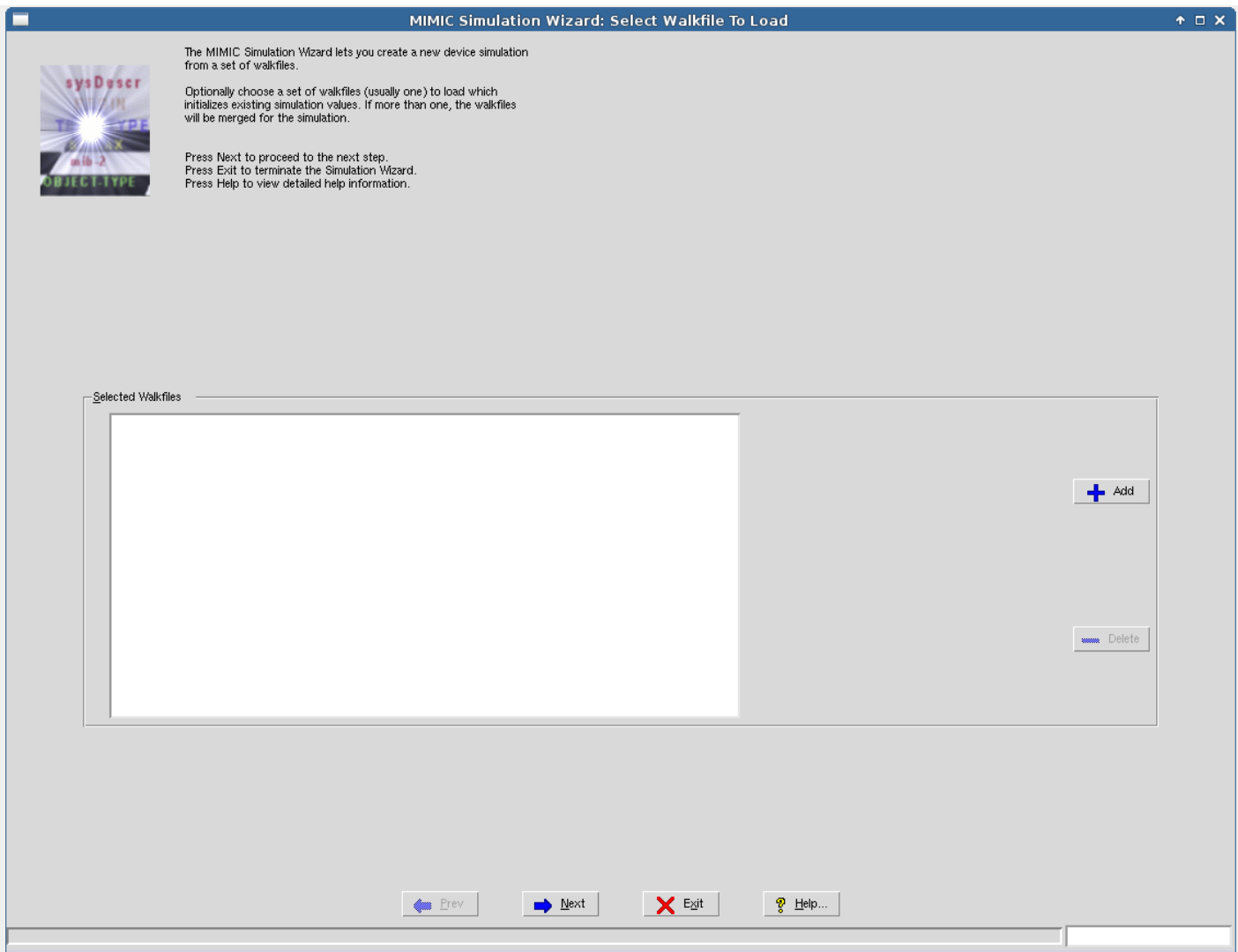


Figure 30: Simulation Wizard: Walkfile

If you load more than one walkfile, they will be merged. This is useful if you have walkfile fragments for distinct (disjoint) parts of your MIB.

If you don't have a walkfile, you will have to specify instances and values for MIB objects you want to simulate.

Select MIBs

In this dialog, select the MIBs you want in your simulation. If you loaded a walkfile in the previous dialog, the set of MIBs is detected from that walkfile.

MIMIC Simulation Wizard: Select MIBs



Choose the MIBs you wish to use by pressing the Add... button. You may remove a selected MIB by pressing the Delete button.

Press Next or Prev to traverse between dialogs. Press Exit to terminate the Simulation Wizard. Press Help to view detailed help information.

Select MIBs

RFC1213-MIB	
SNMPv2-MIB	
IF-MIB	
EtherLike-MIB	
FDDI-SMT73-MIB	
ISDN-MIB	
DS3-MIB	
FRAME-RELAY-DTE-MIB	
RS-232-MIB	
APPLETALK-MIB	
OSPF-MIB	
BGP4-MIB	
RMON2-MIB	
SMON-MIB	
HC-RMON-MIB	
BRIDGE-MIB	
SNA-NAU-MIB	
ATM-MIB	
MIP-MIB	
ENTITY-MIB	

+ Add...

Delete

← Prev Next → × Exit ? Help...

Click to continue with next step

Figure 31: Simulation Wizard: MIBs

You can incrementally add more MIBs to your simulation. Start with a small set, e.g., just RFC1213-MIB. Then add more MIBs as you complete the simulation.

Specify Default Values

The Simulation Wizard can seed all objects for which you don't specify values with default values. To accomplish this, check off the Use Default Values button. Then enter the default values you want for each of the types of MIB objects. These values will be used only if the object does not already have a DEFVAL clause defined in the MIB.

For tables, a single instance will be created with an index whose value is determined by the default value of its type.

MIMIC Simulation Wizard: Specify User Default Values



Optionally, specify default values for instances and objects where simulation values do not exist in the loaded walkfile. To specify defaults, check the box and enter values for the various types. For tables with no instances, one single instance will be created by using these same default values.

Press Next or Prev to traverse between dialogs.
Press Exit to terminate the Simulation Wizard.
Press Help to view detailed help information.

Use Default Values

Integrals Others

Integrals

- INTEGER = :	11
- Counter = :	22
- Counter64 = :	55
- Gauge = :	33
- Timetick = :	44
- BITSTR = :	21

← Prev

Next →

✖ Exit

? Help...

Select to specify default values for objects

Figure 32: Simulation Wizard: Defaults

Edit Object Simulations

In this step you instantiate MIB objects and successively assign them values.

MIMIC Simulation Wizard: Edit Object Simulations



Select an object in the object list to edit its simulation value.
Press Apply to make the value effective.
Press Details... to view its details in the MIB source file.

For tables, you need to add or delete instances.

Press Next or Prev to traverse between dialogs.
Press Exit to terminate the Simulation Wizard.
Press Help to view detailed help information.

Objects

- sysDescr
- sysObjectID
- sysUpTime
- sysContact
- sysName
- sysLocation
- sysServices
- sysORLastChange
- sysOREntry
 - ifNumber
 - ifEntry
 - ifIndex
 - ifDescr
 - ifType
 - ifMtu
 - ifSpeed
 - ifPhysAddress
 - ifAdminStatus
 - ifOperStatus
 - ifLastChange
 - ifInOctets
 - ifInUcastPkts
 - ifInNUcastPkts

Object

Name : sysDescr

OID : 1.3.6.1.2.1.1.1.0

Property

Access : read-only

Syntax :

Value : Cisco Internetwork Operating System Softwar

Apply Details...

Prev Next Exit Help...

Enter the value for selected object

Figure 33: Simulation Wizard: Edit

For all tables, you need to instantiate rows in the table. For all MIB objects that are not Counters, MIMIC creates a simulation that returns the value that you set here. For Counter objects, MIMIC generates a rate-based simulation, based on the value you give to the object, and the current value of sysUpTime.0:

$rate = value(object) / value(sysUpTime.0)$

Select Walkfile To Save

We recommend that you save the walkfile, since you can later load it again into the Wizard. This walkfile is saved in the walks/ directory in your private area.

MIMIC Simulation Wizard: Select Walkfile To Save





Optionally, save the current simulations as a walkfile. This saved walkfile may be loaded in future Simulation Wizard sessions.


Press Next or Prev to traverse between dialogs.
Press Exit to terminate the Simulation Wizard.
Press Help to view detailed help information.


Save Walkfile


Walkfile:

 Browse...

 Prev

 Next

 Exit

 Help...

Click to continue with next step

Figure 34: Simulation Wizard: Save

Select Simulation to Create

In this final step, you specify the new device simulation to create.

MIMIC Simulation Wizard: Select Simulation To Create

Create a Simulation. This lets you assign it to an agent.



Type in the Simulation and Scenario names, or press Browse.
Short, easy-to-remember names are recommended.
Choose either a Random or Constant simulation method.
The MIMIC Recorder processes the walkfile and logs its output.

Press Prev to traverse between dialogs.
Press Finish to complete the simulation.
Press Exit to terminate the Simulation Wizard.
Press Help to view detailed help information.

Device

Simulation:

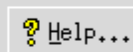
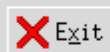
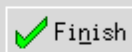
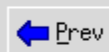
Scenario:

Record Options

Simulation method: Constant Random

Create source

Force Mibs



Click to continue with next step

Figure 34: Simulation Wizard: Simulation

Fill in any `Simulation` name, such as `test`, or a short name for the target you are recording, or your first name. Use any `Scenario` name, such as a small number, e.g., `1`.

These names will be significant once you start doing advanced tasks. Pick easy names to remember.

Choose a `constant` simulation method for regression tests: Counter objects will increment at a constant rate, causing predictable values. Use a `random` simulation method for realistic simulations: Counter objects will increment at the specified rate, but with a minor random fudge factor. Values in this type of simulation are not predictable.

1. CLI Wizard Reference

Overview

The CLI Wizard helps you to create a command line interface (CLI) Simulation from a real device with a command line interface, such as Cisco IOS, Juniper JUNOS, etc. It combines and integrates the functionality of the [IOS Explorer](#) and the [IOS Recorder](#). The wizard creates a CLI simulation to be loaded into the [MIMIC Telnet Protocol Module](#). Instead of requiring a third party NMS application to issue commands, it

discovers the commands on the device dynamically, which are then recorded.

The results of the wizard will be a [rule file](#) that maps CLI commands to responses. Each command detected will result in a rule entry and its corresponding response .mtcl file. It will also dump a transcript of the commands and responses for future processing.



Method Choice

Choose to record from a live device, a previously captured PCAP file, or from a transcript of a telnet session.

MIMIC CLI Wizard : Method Choice

The CLI Wizard helps create a command-line (CLI) Simulation.

Choose the method to use to create a CLI simulation.

- Discover the device and then record.
- Record the device from a packet capture (PCAP) file.
- Record the device from a transcript of an interactive session.

Press Next to proceed to the next step.
Press Exit to terminate the CLI Wizard.
Press Help to view detailed help information.

Simulation Method

- Record Live
- Record PCAP
- Record Transcript

← Prev Next → ✖ Exit ? Help...

Figure 44: CLI Wizard: Method

Recording Parameters

This dialog specifies the parameters for live recording or from PCAP. For the latter, you need to specify the PCAP In File below.

MIMIC CLI Wizard: Recording Parameters



This dialog allows to provide all the parameters for recording the device.

The parameters in this page are mandatory.

Press Next or Prev to traverse between dialogs.

Press Exit to terminate the CLI Wizard.

Press Help to view detailed help information.

General Mode Advanced Server Others

Address:

Login

Interactive Login

Username:

Password:

In File:

Out File:

CLI Record Options

Append Norepeat

click to browse for rule files

Figure 45: CLI Wizard: Parameters

The following General parameters have to be specified:

- **Address**
The IP Address of the device to be recorded. This is mandatory.
- **Interactive Login**
This indicates you want to discover in interactive mode. A simple terminal dialog is presented. You will need to interactively enter login information. This is useful if you need to go through terminal servers, or other frontends before reaching your target device. You are done when you are logged into the device and you see its command prompt and can press Next.
- **Username**
The username to use to login to the device.
- **Password**

The password to login to the device.

- **In File**
This is only used in Method 2 Record From PCAP.
- **Out File**
The Rule file to be created. It is recommended that this be in its own folder, ie. a name of the form `YOUR-FOLDER/YOUR-RULE` where `YOUR-FOLDER` is your folder name, and `YOUR-RULE` is your rule name. The Wizard will put all related files for your CLI simulation in that folder.
- **Append**
To append to the existing rule file.
- **Norepeat**
To add rule without repeat

Optionally, you can also specify these Mode parameters:

- **Modes**
Select to discover different modes.
- **Enable Password**
Enable password to enter `enable` mode.
- **Interface Command**
Interface command to enter `config-if` mode.
- **Router Command**
Router command to enter `config-router` mode.
- **Line Command**
Line command to enter `config-line` mode
- **Command1 ... Command5**
Commands to discover other modes.
- **Prompt1 ... Prompt5**
Corresponding prompts to expect in other modes.

Optionally, you can also specify these Advanced parameters:

- **Template Rules**
TBD

Optionally, you can also specify these Telnet server parameters:

- **Port**
The telnet port to use (default 23).
- **Login Prompt**
The login prompt to be detected, if it is non-standard. The IOS Recorder uses patterns like the login prompt ("Username: "), password prompt ("Password: ") and login failure message ("Login incorrect") to construct the connection welcome message. These default values can be overridden by setting the `Login Prompt`, `Password Prompt` and `Login Failure Message` configurables in this tab.
- **Password Prompt**
- **Login Failure Message**

In the others tab you can specify these parameters:

CLI Discovery Options

- **Discovery Depth**
How many subcommand levels to traverse.
- **Timeout**
Timeout to wait for response.
- **Ignore Output Redirect Commands**
This options lets you ignore output redirections. The `Output Redirect Symbol` field specifies a space-separated list of redirect symbols.
- **Include Permutational Commands**

If you are recording a live device, pressing `Next` goes to the [next step](#).

If you are recording from a PCAP file, pressing `Finish` will process any Telnet requests and responses in that PCAP for the specified server address and create the rule file in the `Out File` you specified above.

This dialog allows you supply the recording parameters from a transcript.

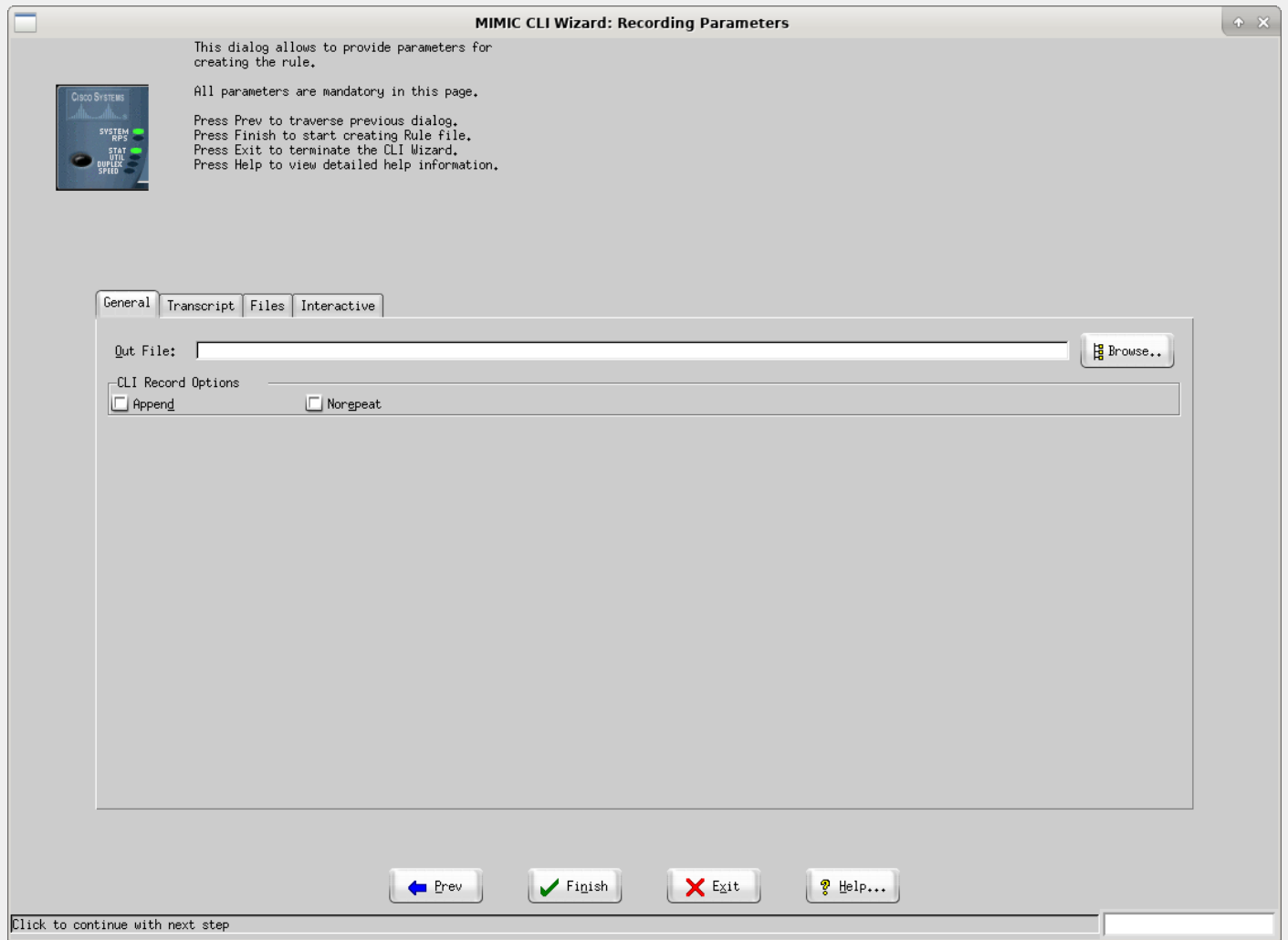


Figure 45b: CLI Wizard: Record Transcript

The following **General** parameters have to be specified:

- **Out File**

The Rule file to be created. It is recommended that this be in its own folder, ie. a name of the form `YOUR-FOLDER/YOUR-RULE` where `YOUR-FOLDER` is your folder name, and `YOUR-RULE` is your rule name. The Wizard will put all related files for your CLI simulation in that folder.

The **Transcript** tab allows you to interactively enter (eg. copy / paste) snippets of Telnet interaction (command / response) to **Add** to the **Command List**.

The **Files** tab lets you do the same by reading transcript files.

In the **Interactive** tab you can manually add commands and responses.

When you click **Finish**, the rule file will be generated for the commands in the cumulative **Command List**.

Discover Commands

The user has the ability to either completely discover the device by specifying no commands to be included and no commands to be excluded. In the first dialog you can restrict the discovery by specifying the commands to be included and commands to be excluded. Appropriately only included commands will be discovered and excluded commands will be ignored.

MIMIC CLI Wizard: Discover Commands



The MIMIC CLI Wizard allows the user to auto discover CLI commands supported by the target device.

Optionally specify the commands to be included or excluded. If you want to issue commands without auto discovery then append "\$" (eg., show running-config\$) in the included commands.

Use the +Add button to add a command to be included or excluded. Use the -Delete button to delete command to be included or excluded.

After specifying commands, Press Next to proceed to the next step. Press Exit to terminate command discovery. Press Help to view detailed help information.

Commands to include

Include Commands: From File...

Commands to exclude

Exclude Commands: From File...

connect
configure
disable
disconnect
enable password
enable secret
end
exit
logout
no enable password

Click to browse for files containing commands to be excluded:

Figure 46: CLI Wizard: Discover

Issue Commands

Once the commands are discovered, this dialog allows the commands to be issued after editing them by providing required arguments. The checklist shows all the commands with a checkbox indicating the command to be issued. Commands in red indicate they were excluded from the discovery process. The argument of the command appear as a leaf in the tree. When selected it allows the user to add the arguments through the graphical interface, which becomes the issued command.

MIMIC CLI Wizard: Issue Commands



This dialog allows the user to choose discovered commands to issue to the target devices and specify their parameters.

Set the commands to be issued by checking the checkbox.
(NOTE: To avoid unwanted results on the target device please select only the commands that are safe to issue)

Press Prev to go to previous dialog or Finish to issue commands.
Press Interrupt to interrupt discovery and proceed to next step.
Press Exit to terminate command discovery.
Press Help to view detailed help information.

Discovering commands in test> ...

Discovered Commands

- top
 - [- access-enable
 - host
 - timeout
 - [
 - access-profile
 - ignore-sanity-checks
 - merge
 - replace
 - clear
 - aaa
 - cache
 - filterserver
 - acl
 - [WORD]
 - connect
 - disable
 - disconnect
 - enable
 - [
 - exit

Argument

Change argument:

Discovering 1 of 1 of show ip helper-address Async ..

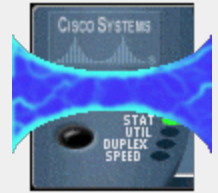
Figure 47: CLI Wizard: Issue

All the checked commands will be issued to the device when you click on the Finish button.

1. NetFlow Wizard Reference

Overview

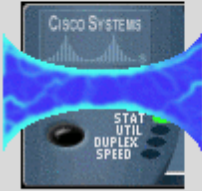
The NetFlow Wizard allows you to easily create a NetFlow Simulation from a real Cisco device by capturing a packet capture of the generated flows. Additionally, once you have created a simulation, the wizard allows you to modify it to suit your needs.



Create or modify

In this initial dialog you can choose to create a fresh simulation from a live NetFlow Exporter or modify an existing simulation.

NetFlow Wizard: Create or modify



The MIMIC NetFlow Wizard assists you in creating or customizing a NetFlow simulation. You can easily create a NetFlow simulation with a packet capture (PCAP) from a real NetFlow Exporter. To customize an existing NetFlow simulation, you can add templates of interesting flows and modify parameters.

Choose the operating method:

- create a new NetFlow configuration;
- modify an existing NetFlow configuration.

Press Next to proceed to the next step.
Press Exit to terminate the NetFlow Wizard.
Press Help to view detailed help information.

Select NetFlow

Create a new NetFlow Config

Modify an existing NetFlow Config

NetFlow:

← PrevNext →✖ Exit? Help...

Figure 48: NetFlow Wizard: Method

Create Config file using PCAP file

In this dialog you can enter the parameters to the `netflowrec` utility to create a fresh simulation from a packet capture (PCAP) of a real NetFlow Exporter. Check the [netflowrec documentation](#) for details.

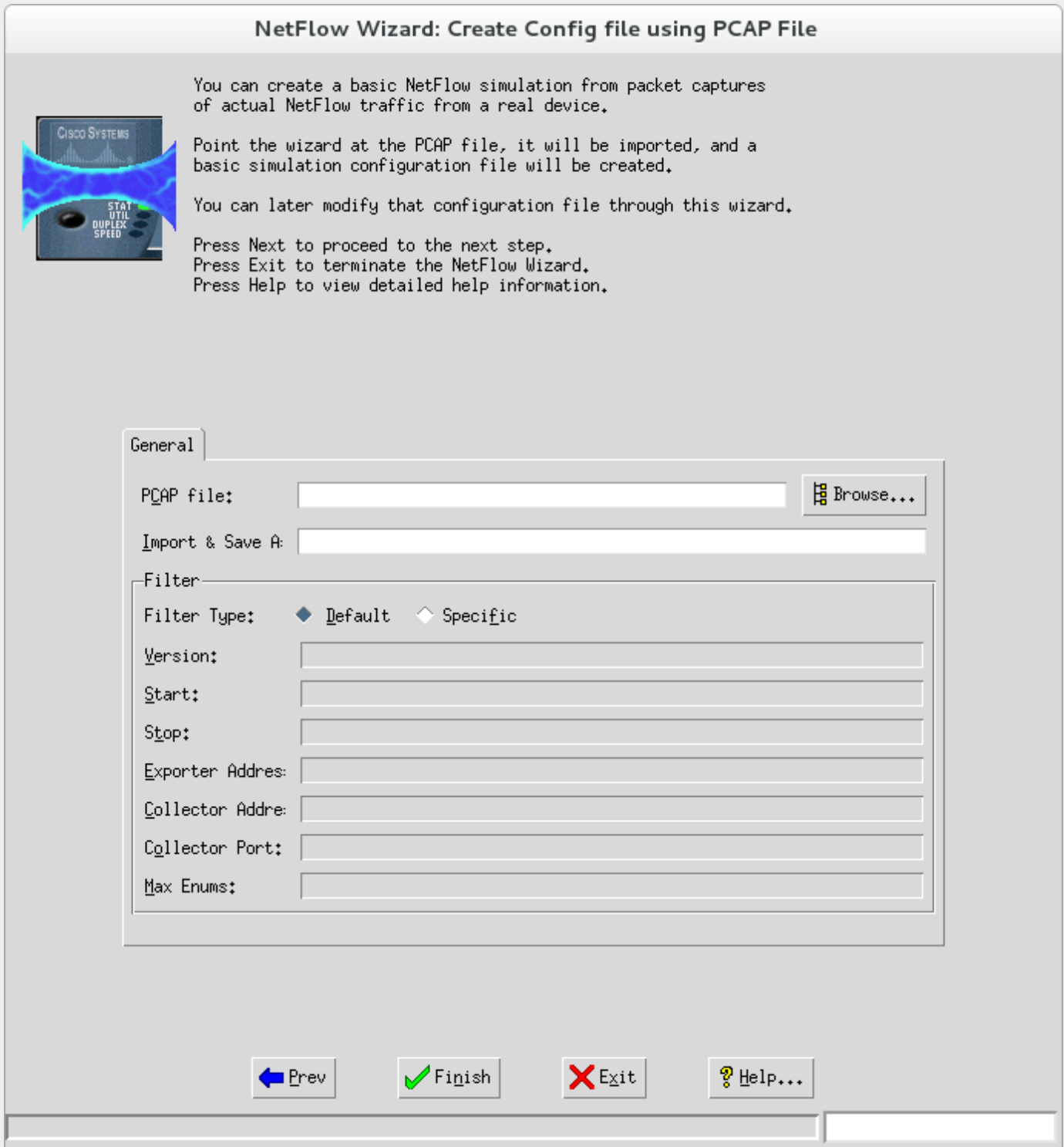


Figure 49: NetFlow Wizard: PCAP

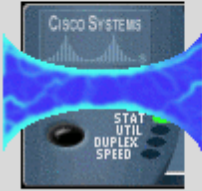
Modify Config File

This dialog lets you modify the previously selected configuration file. The left pane shows the sections of the [NetFlow configuration](#).

The right pane shows the flows and their values.

NetFlow Wizard: Modify Config File NFA_ASA.cfg

This dialog lets you modify the NetFlow configuration file. You can change parameters, add/delete/edit flowsets, and add/delete/edit flows.



Press Next to save the changes.
Press Exit to terminate the NetFlow Wizard.
Press Help to view detailed help information.

NFA_ASA.cfg

- globals
 - configurables
 - Flow:256 (selected)
 - Flow:257
 - Flow:258
 - Flow:259
 - Flow:260
 - Flow:261
 - Flow:262
 - Flow:263
 - Flow:264
 - Flow:265
 - Flow:266
 - Flow:267
 - Flow:268
 - definitions
 - Flow:256
 - Field:NF_F_CONN_ID (selected)
 - Field:IPV4_SRC_ADDR
 - Field:L4_SRC_PORT
 - Field:INPUT_SNMP
 - Field:IPV4_DST_ADDR
 - Field:L4_DST_PORT
 - Field:OUTPUT_SNMP
 - Field:PROTOCOL

Flow:256

- dfs_num_rec : 1
- Flow:268
 - uid : 12
 - id : 268
 - dfs_count : 0
 - dfs_num_rec : 1
- DEFINITIONS
 - Flow:256
 - uid : 0
 - id : 256
 - scope_count : 0
 - field_count : 21
 - Field:NF_F_CONN_ID (selected)
 - length : 4
 - type : RANGE
 - min : 0
 - max : 4282908673
 - sequential : 0
 - Field:IPV4_SRC_ADDR
 - length : 4
 - type : RANGE
 - bidirectional : 1
 - min : 0,0,0,0
 - max : 255,195,0,2

+ Add - Delete

< Prev Next > X Exit ? Help...

Figure 50: NetFlow Wizard: Modify

1. sFlow Wizard Reference

Overview

The sFlow Wizard allows you to easily create a sFlow Simulation from a Cisco device by capturing a packet capture of the generated flows. Additionally, once you have created a simulation, the wizard allows you to modify it to suit your needs.



In this initial dialog you can choose to create a fresh simulation from a live sFlow Exporter or modify an existing simulation.

sFlow Wizard: Create or modify



The MIMIC sFlow Wizard assists you in creating or customizing a sFlow simulation. You can easily create a sFlow simulation with a packet capture (PCAP) from a real sFlow Exporter. To customize an existing sFlow simulation, you can add templates of interesting flows and modify parameters

- Choose the operating method:
- create a new sFlow configuration;
 - modify an existing sFlow configuration.

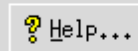
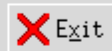
Press Next to proceed to the next step.
Press Exit to terminate the sFlow Wizard.
Press Help to view detailed help information.

Select sFlow

Create a new sFlow Config from pcap

Modify an existing sFlow Config

sFlow:



Specify a sFlow Config File e.g. test.cfg

Figure 51: sFlow Wizard: Method

sFlow Wizard: Create Config file using PCAP File

You can create a basic sFlow simulation from packet captures of actual sFlow traffic from a real device.

Point the wizard at the PCAP file, it will be imported, and a basic simulation configuration file will be created.



You can later modify that configuration file through this wizard.

Press Next to proceed to the next step.
Press Exit to terminate the sFlow Wizard.
Press Help to view detailed help information.

General

PCAP file:

Import As:

Record Type: Default Specific

Filter

Exporter Address:

Collector Address:

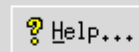
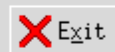
Collector Port:

Start:

Stop:

Exclude:

Sequential



Click to continue with next step

Figure 52: sFlow Wizard: Create from PCAP

Modify Config File

This dialog lets you modify the previously selected configuration file. The left pane shows the sections of the sFlow configuration.

The right pane shows the samples and their values.

sFlow Wizard: Modify Config File multiple_flow_sflow.cfg

This dialog lets you modify the sFlow configuration file. You can change parameters, add/delete/edit flowsets, and add/delete/edit flows.



Press Next to traverse next dialog.
Press Exit to terminate the sFlow Wizard.
Press Help to view detailed help information.

multiple_flow_sflow.cfg

- includes
 - struct_sflow_v5.inc
- sample_set
 - flow_samples
 - extended_switch
 - sampler_header
 - flow_samples
 - extended_switch
 - sampler_header

multiple_flow_sflow.cfg

- comment :
- includes
 - struct_sflow_v5.inc
- SAMPLE_SET
 - sub_agent_id : 0
 - sys_uptime_offset : 12234
 - frame_sequence : 47983
 - flow_sequence : 1156
 - counter_sequence :
- FLOW_SAMPLES
 - data_source : 21
 - sample_pool : 18939904
 - sampling_rate : 16384
 - drops : 0
 - input_interface : 21
 - output_interface : 2
- Sample:EXTENDED_SWITCH
 - Field:src_vlan
 - type : value
 - value : 555
 - Field:src_priority
 - type : value
 - value : 0
 - Field:dst_vlan
 - type : value
 - value : 555
 - Field:dst_priority
 - type : value
 - value : 0
- Sample:SAMPLED_HEADER
 - Field:protocol
 - type : value

Buttons: + Add, - Delete, Up, Down, Prev, Next, Exit, Help...

Figure 53: sFlow Wizard: Modify

Overview

The IPMI Wizard allows you to easily create a IPMI Simulation by capturing a session between a IPMI device and a management app. Additionally, once you have created a simulation, the wizard allows you to modify it to suit your needs.



Create or modify

In this initial dialog you can choose to create an IPMI simulation from a live IPMI session, record from an existing walkfile, or modify an existing simulation at runtime.

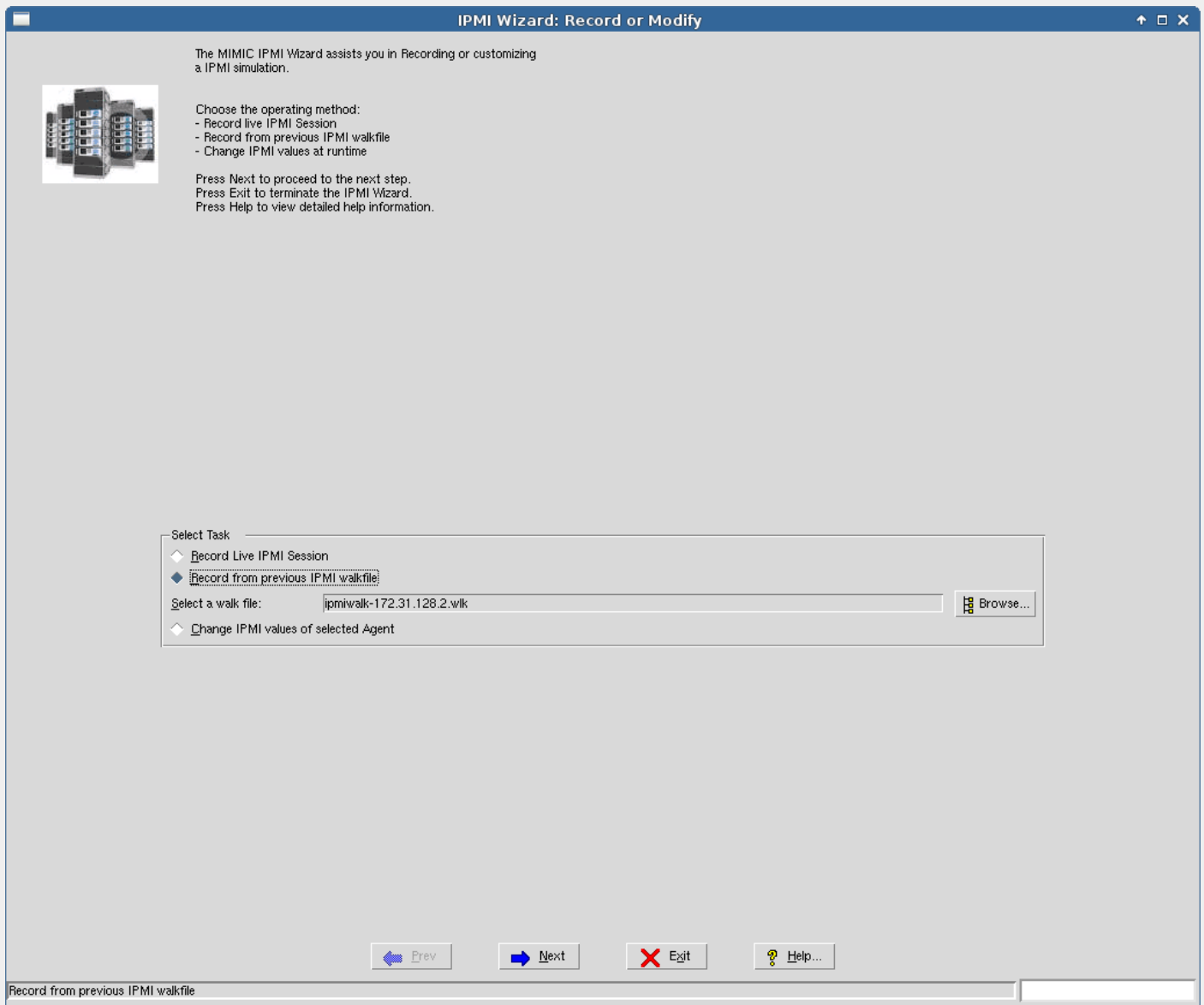


Figure 56: IPMI Wizard: Method

Record Live Session

To record the live session, the `ipmirec` tool implements a proxy server between the IPMI management app and the IPMI device, storing the requests and responses in an IPMI walkfile.

IPMI Wizard: Record Live Session

This IPMI Wizard assists you in recording a live IPMI session.

Please specify the target info and the related IPMI parameters.



Press Next to proceed to the next step.
Press Exit to terminate the IPMI Wizard.
Press Help to view detailed help information.

Mandatory Optional

Target Host Info

Target Host:

Authentication Code:

IPMI v2.0

The following commands apply only to IPMI v2.0

Password:

Proxy Cipher:

Target Cipher:

Click to continue with next step

Figure 57: IPMI Wizard: Live

The mandatory options are the target host IP address and the authentication code.

Simulation Options

You can apply the IPMI simulation to a new SNMP simulation that will be recorded from the live device, to an existing simulation that you created previously, or copy an existing simulation so that you don't modify it.

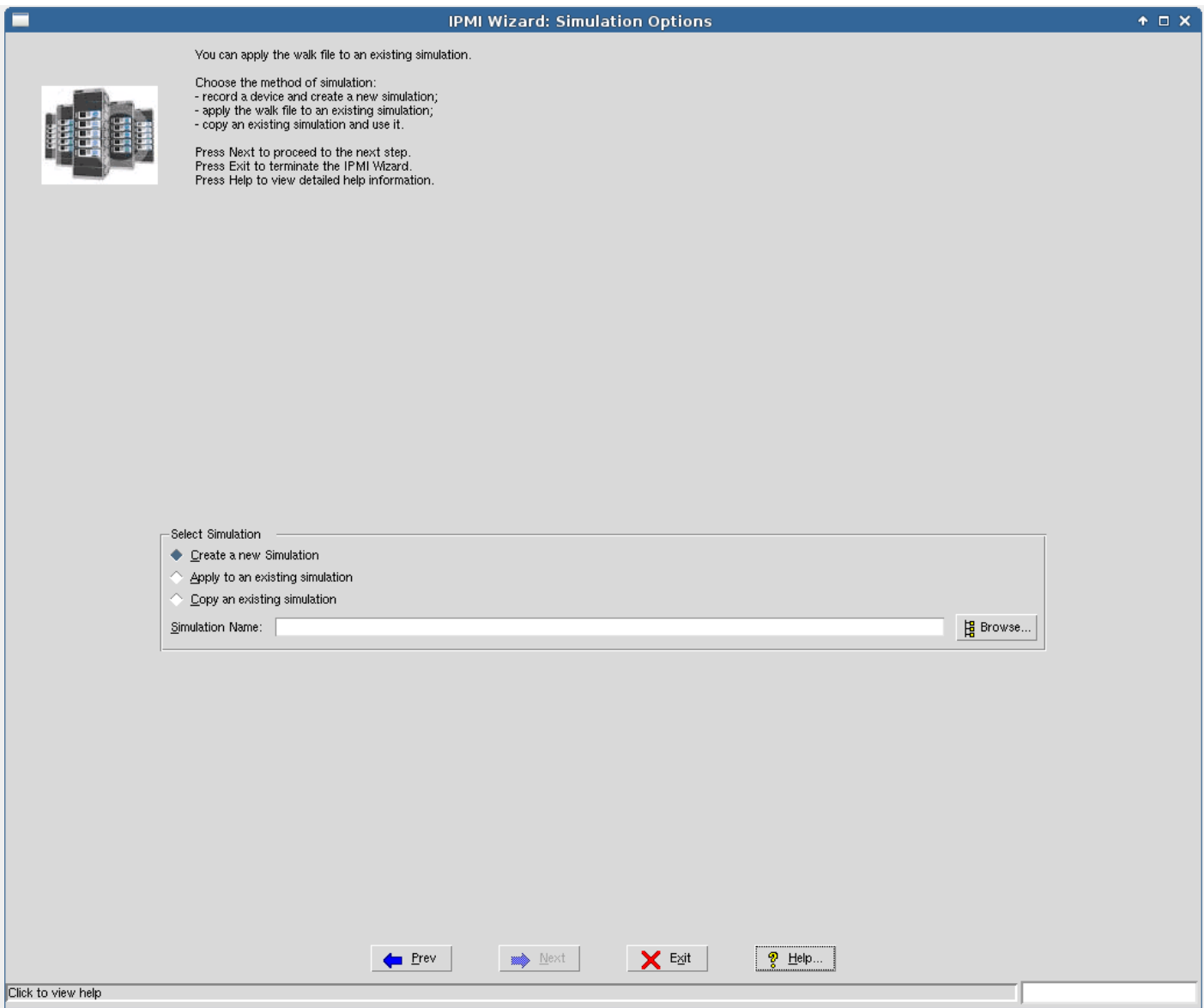


Figure 58: IPMI Wizard: Simulation Options

Pressing Next will present the Create IPMI Simulation dialog to summarize the selections. Pressing Next again will actually create the simulation.

Change IPMI Values

After you select a running agent with an IPMI simulation, this dialog allows you to change the simulation at runtime. You can modify any of the field values in responses for any of the supported requests, and add or delete instances to requests.

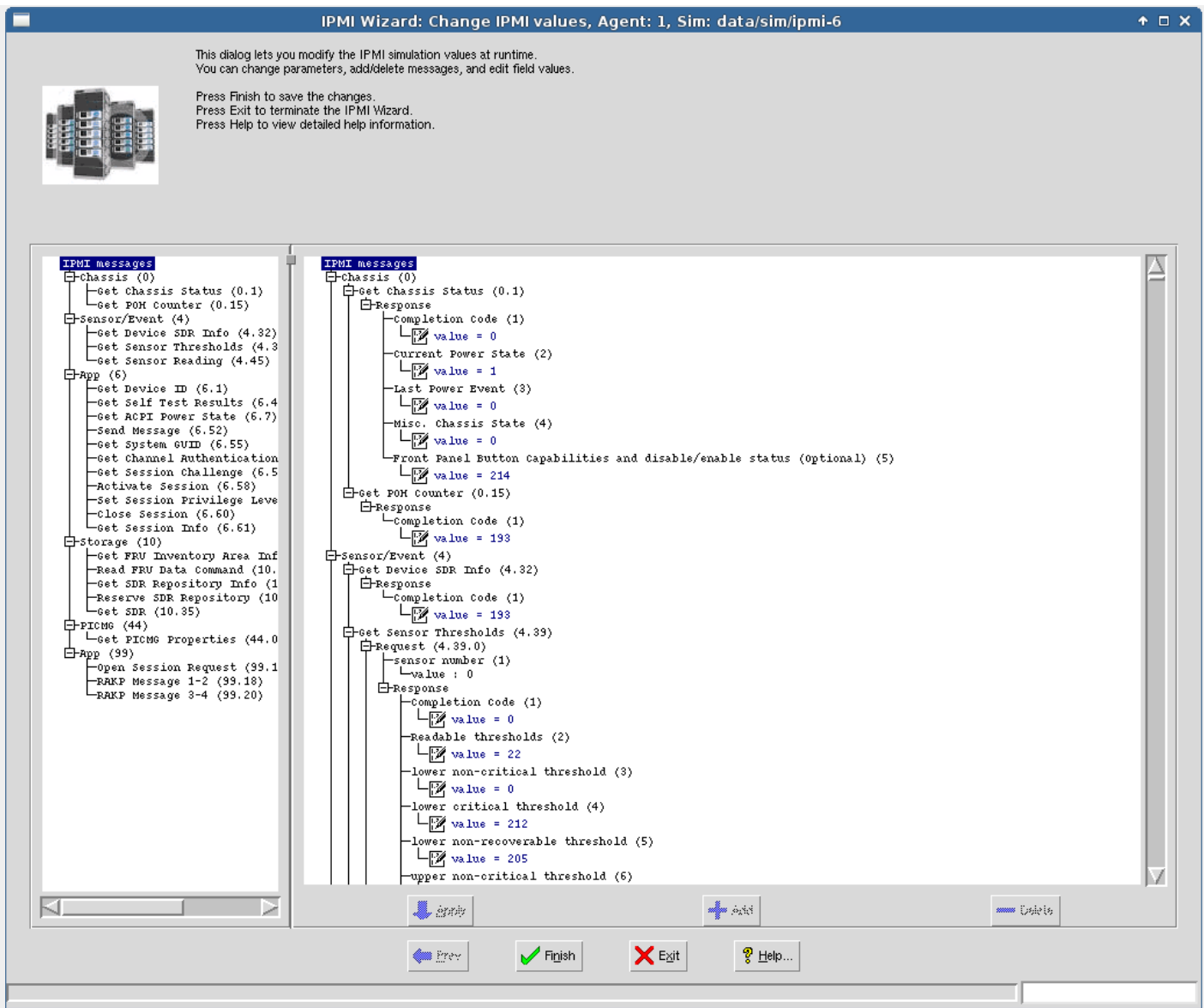


Figure 59: IPMI Wizard: Modify

1. Web Wizard Reference

Overview

The Web Wizard allows you to easily create a Web Services Simulation by capturing a session / conversation between a Web Server and client. Additionally, once you have created a simulation, the wizard allows you to modify it at run-time to suit your needs.



Create Web Service Simulation

Unless you already have recorded before, you start by recording a new conversation between a web client and a server.

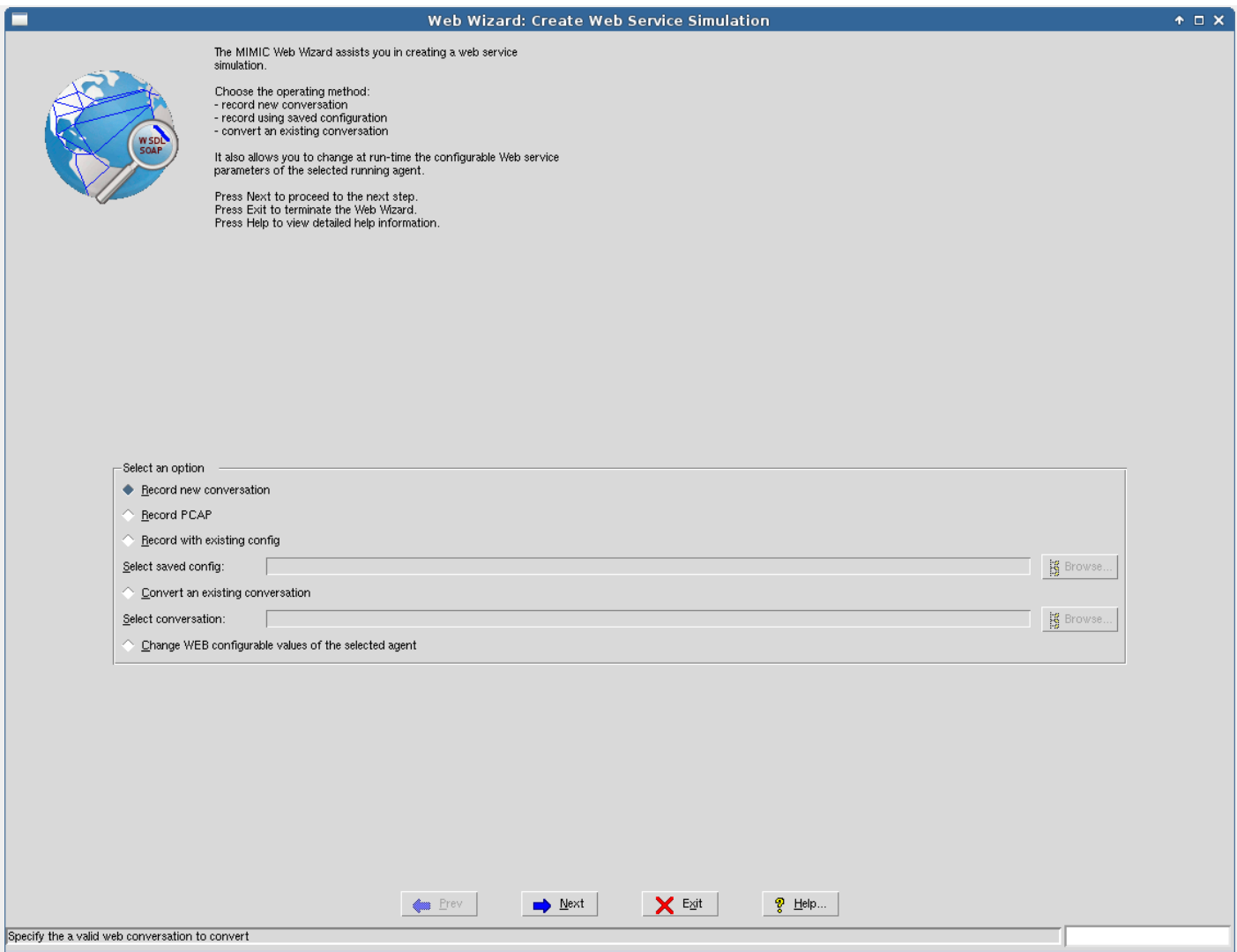


Figure 60: Web Wizard: Create

If you have never recorded before, then you have 2 choices to record a web service session.

First, select **Record new conversation** to record a live conversation between your application and your web service. The **Recording Parameters** dialog will be displayed when pressing **Next** allowing you to specify the parameters of the live session to record.

Second, if you have a packet capture (PCAP) that you recorded, you can use **Record PCAP** to record it. The **Record PCAP** dialog will allow you to record from the packet capture. This allows you to record sessions with a packet capture tool like **Wireshark** where you cannot record live.

Alternatively, use **Record with existing config** if you have recorded before and saved the recording configuration. The **Recording Parameters** dialog will be displayed when pressing **Next** of the previously recorded session, allowing you to record again.

Select **Convert an existing conversation** to convert an existing session that was recorded to a Web Services simulation.

Finally, if you want to change a WEB simulation while it is running, select **Change WEB configurable values of the selected agent**. The **Change WEB Values** dialog will be displayed when pressing **Next**.

Recording Parameters

To record a conversation, specify the conversation file name, and the web server you want to record. The HTTP port is usually port 80, and the SSL port is usually 443, although you should verify these before you record.

Web Wizard: Recording Parameters



You can record a web conversation by placing this tool between your WEB client and server.

Enter the mandatory parameters to configure and start the recorder in the next dialog.
You may even set the optional parameters else continue with defaults.

You can later convert this recorded file through this wizard in order to create its equivalent simulation files.

Press Prev to select any other option.
Press Next to proceed to the next step.
Press Exit to terminate the Web Wizard.
Press Help to view detailed help information.

Mandatory Optional

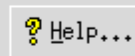
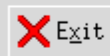
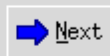
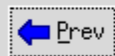
Save Conversation As

Target Web Service

Destination IP:

Destination Port:

Destination SSL Port



Click to repeat previous step

Figure 61: Web Wizard: Record

When you press Next, the Recorder Settings dialog appears, allowing you to start a recording.

Record PCAP

Specify the PCAP file to record, and if there are multiple WEB conversations in it, you can pick the one between a specific server and a client, or between specific packet numbers.

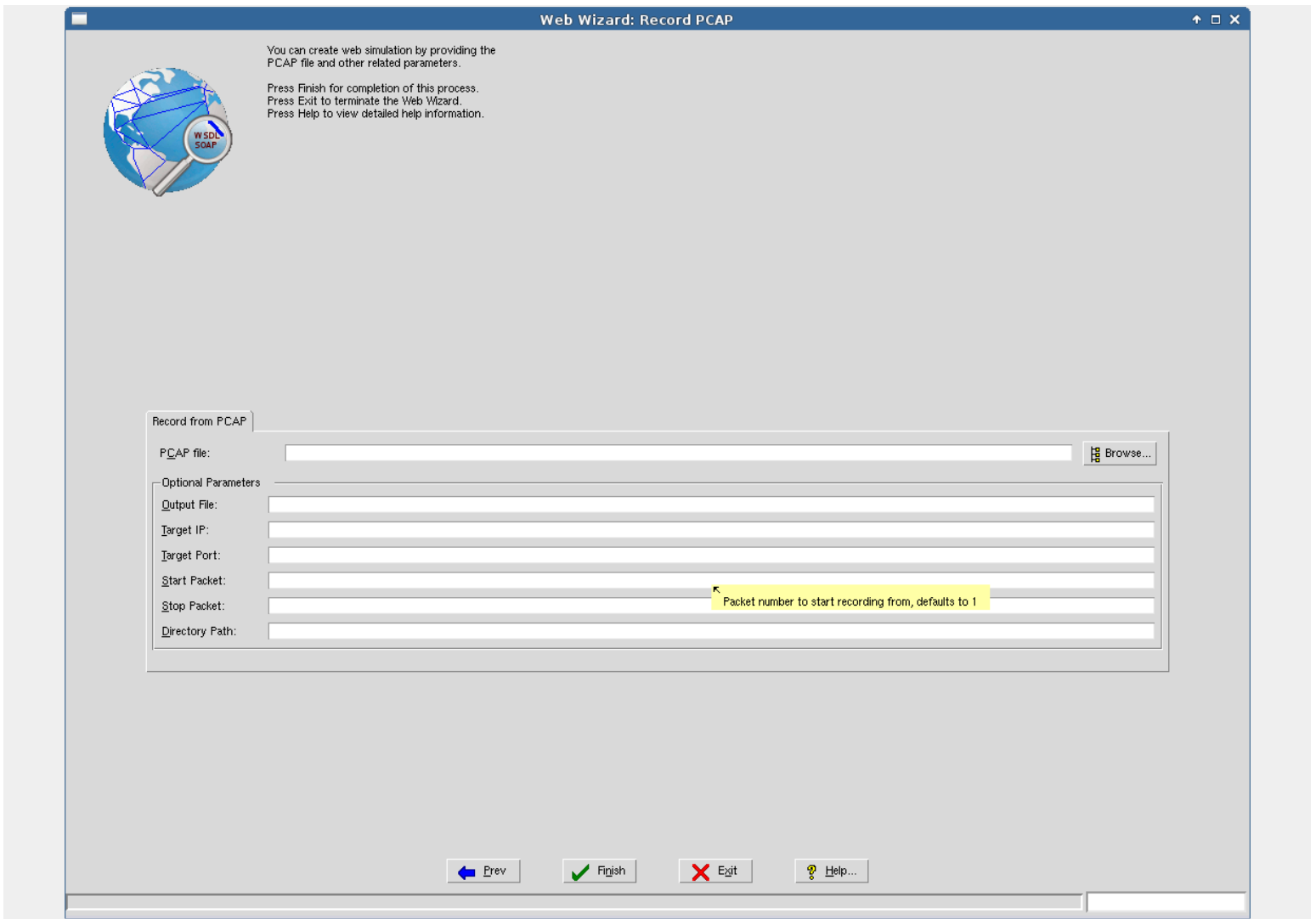


Figure 62: Web Wizard: Record PCAP

Configurable Values

WEB module simulations can implement configurable values, which can be changed at run-time in real-time on a running agent.

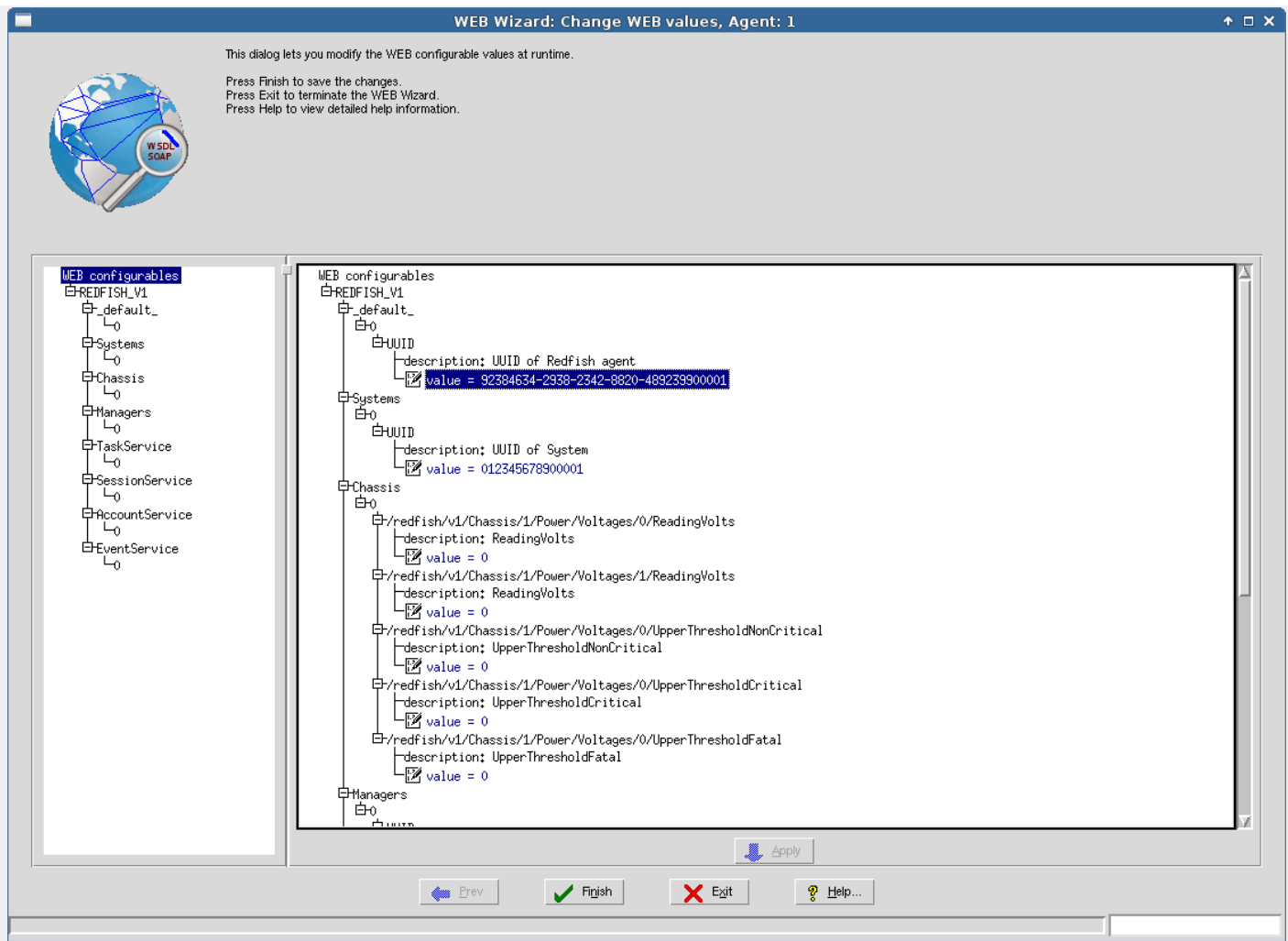



Figure 63: Web Wizard: Configurable Values

1. Tutorial Wizard Reference

Overview

The MIMIC Tutorial Wizard guides you through the MIMIC software. The process to run SNMP simulations is described in the [QuickStart manual](#). This wizard will follow that process.

As a short-cut, you may want to watch the videos  in the [QuickStart Tutorial](#).

The process we follow here is the most efficient to get you going. The Tutorial Wizard will invoke the existing tools to accomplish each task. You can at any point invoke the tools manually.

The 4 main steps covered here are:

1. [setup MIBs needed by the simulations](#)
2. [create simulations](#)
3. [run simulations](#)
4. [customize simulations](#)

You can skip any of the steps, for example if you just want to run the simulation that ships with MIMIC, go to [step 3](#). MIMIC is designed so you can easily come back to an earlier step later. The wizard lets you do this, in effect exploring the steps and the tools necessary to complete them.



MIBs

The first step in the process is to make sure you have all the MIBs you need for your simulations. Ultimately, you will need a MIB only if your management application accesses the objects in that MIB.

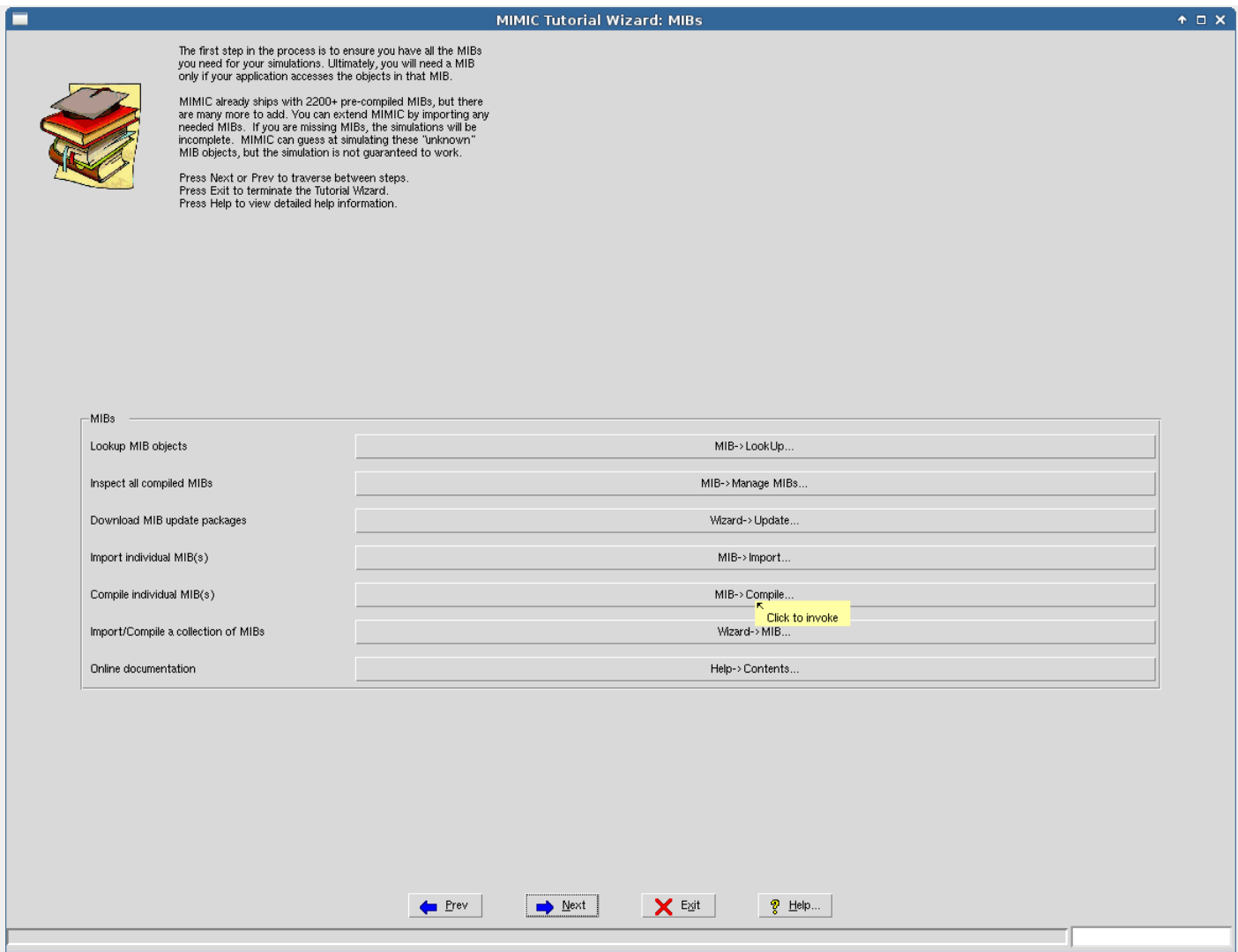


Figure 70: Tutorial Wizard: MIBs

MIMIC already ships with 2100+ [precompiled MIB files](#), but there are many more. You can extend MIMIC by importing any needed MIBs. If you are missing MIBs, the simulations will be incomplete. MIMIC can guess at simulating these "unknown" MIB objects, but the simulation is not guaranteed to work.

Remember that you can get more help at any time by clicking the `help...` button.

If you are not sure whether MIMIC knows about your required MIB, you can lookup objects, to see whether they are known. When you click the `MIB->Lookup...` button, the corresponding `Lookup...` menu item in the MIB menu will be invoked.

You can inspect all the MIBs known to MIMIC by clicking the `MIB->Directory...` button. The MIBs that come pre-compiled with MIMIC are shown in **blue**, while the MIBs that you compiled are shown in **red**. This is due to the difference between the [shared and private data areas](#).

If the MIBs you require are not already known, you can extend MIMIC in various ways. First, you want to see if Gambit supplies them in the optional updates that you can download from the Web. To do this, use the [Update Wizard](#). Initially, install only patches necessary from the `Patches` section, then look at the `MIB Library` section to install required MIBs. You can install MIBs even if they are not required. The overhead on MIMIC is minimal, because the software is designed to efficiently handle a large amount of MIBs. If you will be looking at the details of MIBs, it may be useful to download the optional "MIB source files" update.

If the MIBs you require are not in any of the updates in the Update Wizard, then you can import and compile them yourself. To do this, you need the MIB source files (in standard SMI format). You first import the MIB source files into MIMIC with `MIB->Import...`, then compile them with `MIB->Compile...`. You can look at a flash demo of this task at the [MIMIC demo page](#).

If you have a number of MIBs to add to MIMIC, the `MIB Wizard` will make this job much easier, resolving any dependencies among them automatically. You can look at a flash demo of this task at the [MIMIC demo page](#).

Create Simulations

The second step is to create a simulation of a device (which as a short-cut we call "device"). This step assigns MIBs to the device, and specifies values to be returned for each MIB object instance. For example, we need to specify what string to return for the `sysDescr` object.

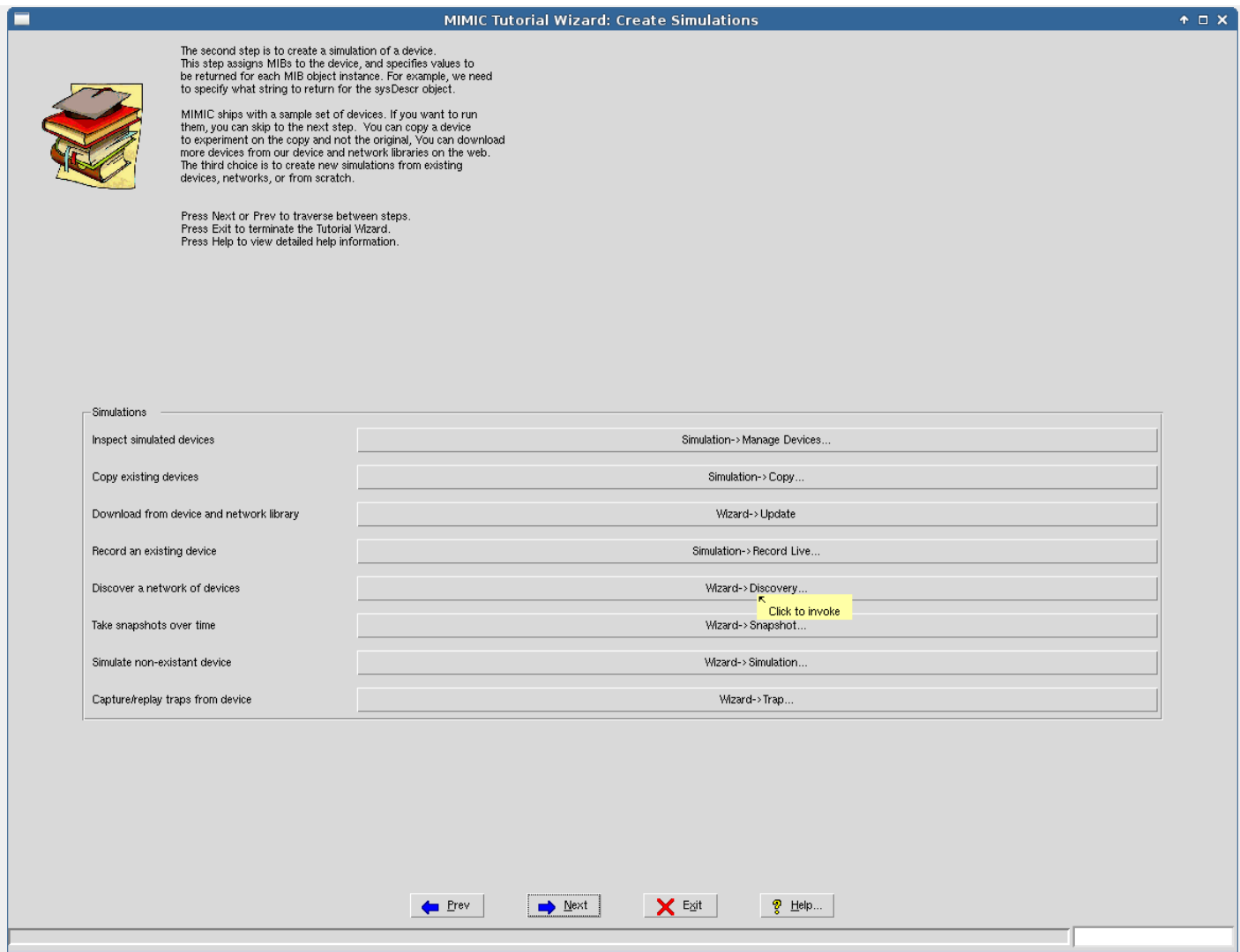


Figure 71: Tutorial Wizard: Simulations

MIMIC ships with a sample set of devices. If you want to use them, you can skip to the next step. To inspect the currently known set of devices, use the [Simulation->Devices...](#) button.

Else, you can download more devices from our device and network libraries on the web with the [Update Wizard](#). Look at the [Device Library](#) and [Network Library](#) sections. You can browse for the available devices from several manufacturers, and complete networks. Expand a node, and double-click to get extra info. To install them, most of them require you to have installed the permanent license keys, but the `CISCO VLAN` network is available for evaluators.

The third choice is to create new simulations from existing devices, networks, or from scratch. To record a small group of existing devices, use the [Simulation->Record Live...](#) button. You can look at a flash demo of this task at the [MIMIC demo page](#).

To discover a larger network of interconnected devices, use the [Discovery Wizard](#). You can look at a flash demo of this task at the [MIMIC demo page](#).

The [Snapshot Wizard](#). allows to take snapshots from an existing device, so that the simulation changes just like the device does.

If you don't have a device to record, you can use the [Simulation Wizard](#). to create the simulation. This involves specifying values for all relevant MIB objects.

The [Trap Wizard](#). lets you capture traps from a real device, which can then be replayed from one of the simulations.

Run Simulations

This step configures MIMIC to run simulations.

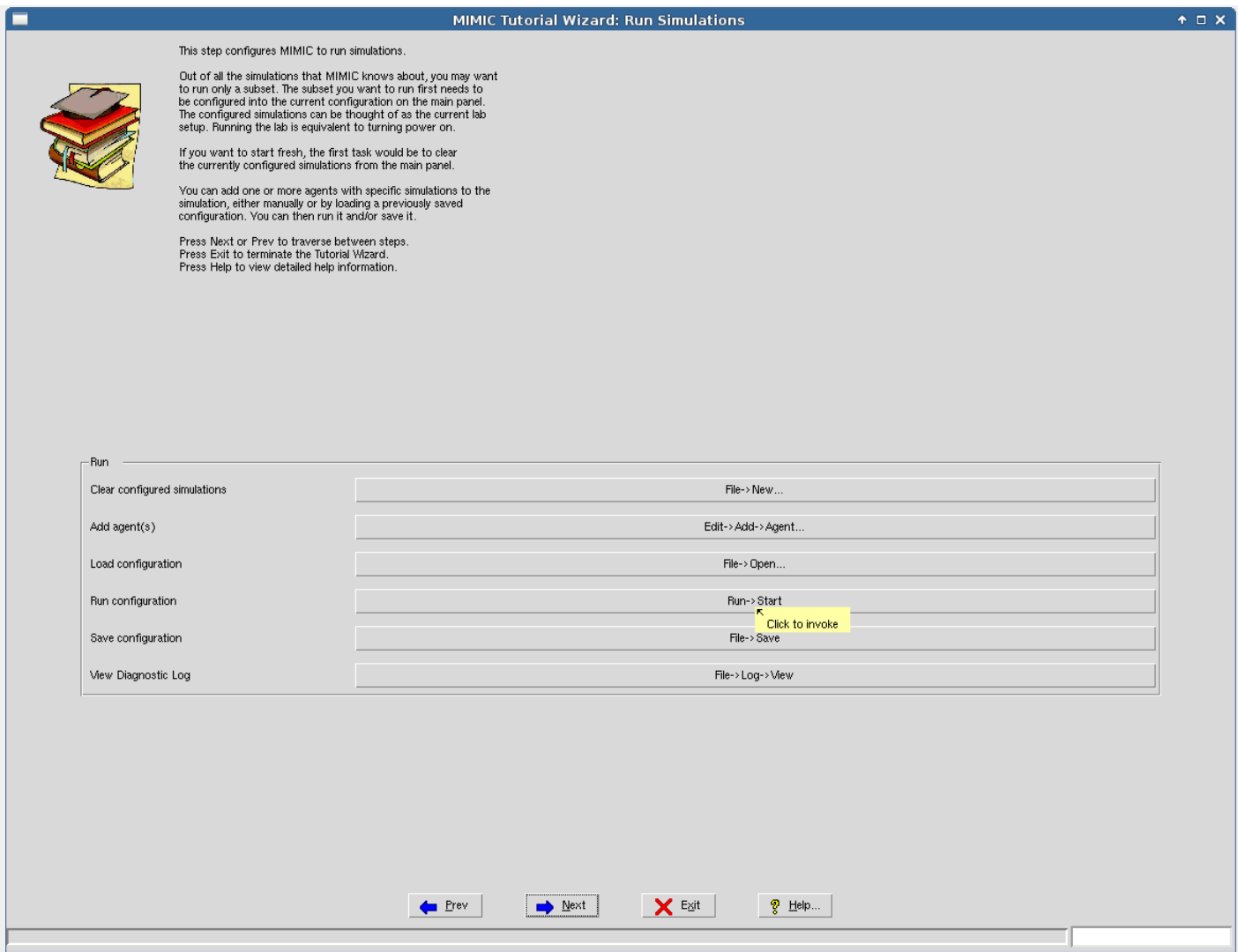


Figure 72: Tutorial Wizard: Run Simulations

Out of all the simulations that MIMIC knows about, you may want to run only a subset. The subset you want to run first needs to be configured into the current configuration on the main panel. The configured simulations can be thought of as the current lab setup. Running the lab is equivalent to turning power on.

If you want to start fresh, the first task would be to clear the currently configured simulations from the main panel.

You can add one or more agents with specific simulations to the simulation, either manually or by loading a previously saved configuration. You can then run it and/or save it.

The diagnostic log lets you troubleshoot the MIMIC simulations.

Customize Simulations

MIMIC allows you to change simulations at run-time, dynamically. You can control these changes interactively through MIMICView, or via scripts in the supported APIs (Tcl, Perl, Java, C++).

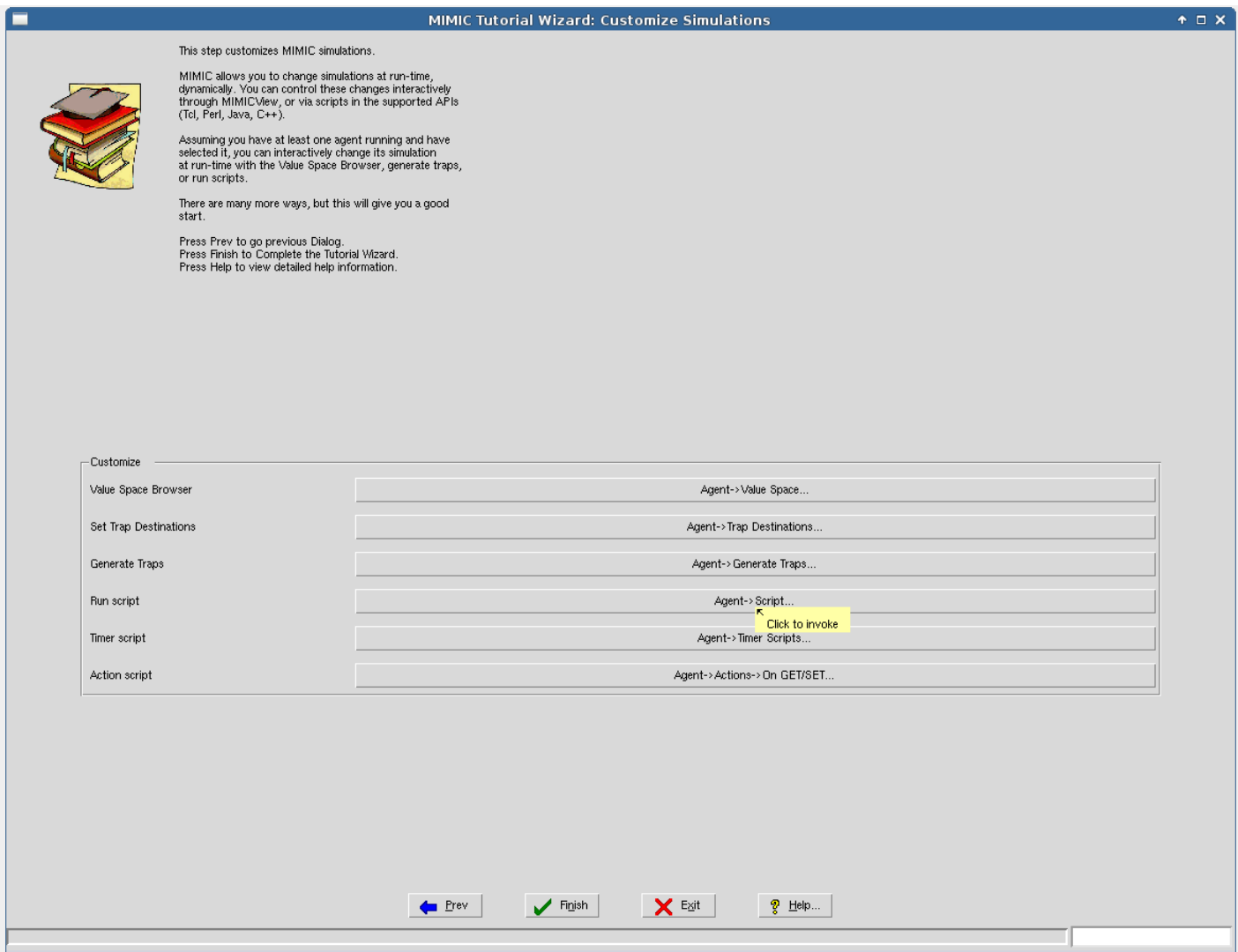


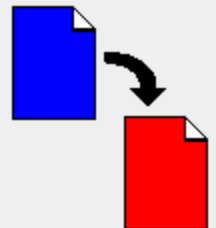
Figure 73: Tutorial Wizard: Customize Simulations

Assuming you have at least one agent running and have selected it, you can interactively change its simulation at run-time with the Value Space Browser, generate traps, or run scripts.

4. Configuration Wizard Reference

Overview

The Configuration Wizard lets you tailor the MIMIC environment.



MIMICView Configuration

Each of the tabs lets you configure parameters in MIMICView:

- Update Notification
Turn [Automatic Update Notification](#) on or off.

We recommend to enable notification, unless you are not connected to the Internet.

Proxy configuration is only needed if you require a proxy to access the Internet. MIMIC Update Wizard works through [SOCKS proxy servers](#) with the help of the [socat relay utility](#) which is an optional installable on newer versions of Fedora Linux. For details see this [FAQ](#) entry.

- Colors
You can choose colors for various aspects of the GUI. You can re-configure other aspects of the user interface with the MIMICView resource file (see also the

FAQ section).

- Help Viewer

Pick your favorite help viewer for viewing the context-sensitive online documentation. By default, on Windows the Microsoft HTML Help Control (HH.exe) is used, on Unix the first popular HTML browser it can find.

- Agent

You can toggle the address label of the agent icon to show its IP address or resolved fully qualified domain name (FQDN).

NOTE: currently the FQDN will only show after starting the agent.

You can toggle the description label of the agent icon to show its name (as shown in Simulation -> Devices) or simulation,MIB,scenario triplets.

You can configure the status message hint of the agent icon to show any information that the display subsystem knows about. By default it shows

```
Agent $id, address $_host_name
```

- Maps

The Max agents in root map configurable limits the number of agents in the root map. Once the limit is surpassed, agents are automatically placed in submaps in batches of this configurable. Eg. if this number is 100, if the number of agents added is more than that, then submaps will be created with 100 agents each. Any leftover agents will be placed in the root map.

This results in better GUI performance for large lab configurations.

- Log

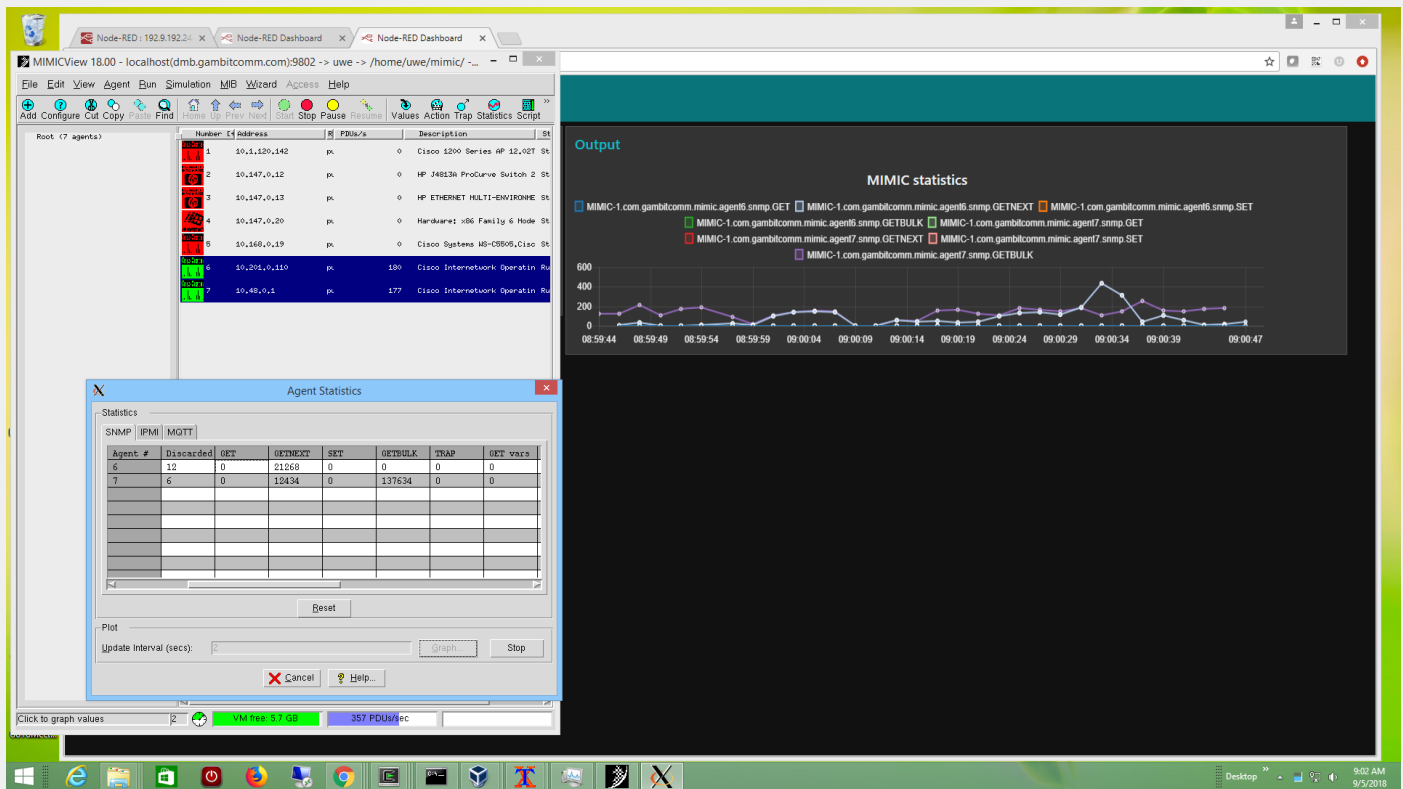
Lets you turn on/off the automatic switching of the Simulator Log. If enabled, then the Log is switched to a new file every midnight. This is useful for management of large logs.

- Graph

These configurables are for graphing agent statistics in the Agent -> Statistics dialog. You can change from Built-in to Graphite to integrate with Graphite or Grafana . The following fields can be set:

- CARBON_ADDR specifies the IP address of the Graphite server. This is required.
- CARBON_PORT specifies the TCP port of the Graphite server. This is required.
- CARBON_PREFIX set the prefix for the statistics. This is useful if you have multiple MIMIC instances that are exporting agent statistics.

For example, this shows agent PDU statistics for agents 6 and 7 for an MIMIC instance named MIMIC-1.



MIMIC Log Configuration

You can limit the number of lines to be shown in the log windows, by selecting the Limited button. Enter the number of lines to be displayed in the Lines to log text field.

You can store new log files in a different Log Directory.

Java Configuration

MIMIC needs to know about the Java Runtime Environment (JRE) installed on your machine. It attempts to detect it in the default paths, but you may have to point it to the correct location (bin/ subdirectory of the JRE install area).

E-mail Configuration

Certain MIMIC components ([Diagnostic Wizard](#), [profile reporting](#)) need to know how to send information back to Gambit. Here you configure your mail gateway.

Profiling Configuration

You can enable/disable profile reporting, and how often you want it to happen.

i. Update Wizard Reference

Introduction

The Update Wizard makes it easier to update to newer versions of MIMIC over the Internet, or to install optional software from CD-ROM. It automatically notifies you of new software updates, and presents a friendly front-end to install them.

NOTE: in order to work, MIMIC requires access to the internet, in particular to HTTP port 80 for the Gambit update servers www.gambitcomm.com and mirror2.gambitcommunications.com. If you do not have direct access, then it can also work with [a proxy](#).

If you have it [enabled](#), MIMICView will notify you of the availability of new updates of MIMIC. This occurs whenever you start MIMICView, but at most once a day. MIMICView will download the MIMIC update database, and check for applicable updates to your software.

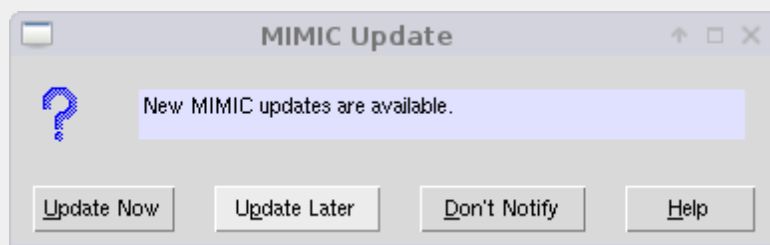


Figure 80: Update Wizard: Notify

If there are new updates, you will be notified with a dialog. This lets you either:

- update immediately, by invoking the Update Wizard.
- update later. You have to manually invoke the Update Wizard from the `Wizards` menu.
- don't notify. This turns off the automatic update notification feature just as in the [Configuration Wizard](#).

In order to use the Update Wizard, you will need write permissions to the MIMIC shared area (the directory where you installed MIMIC).

First you need to select the source of your update, either over the Internet from the Gambit Web site, or from CD-ROM. Since the CD-ROM can be at a different directory path on each system, you will need to supply it's location in the `Path` field.

If you have installed MIMIC previously, you can select the same packages that you had already installed then, to be installed for this release with `select packages from previous release of MIMIC`. This selection is done automatically and a `Previous installation` dialog pops up, then can be reviewed and modified in the next [Select Package](#) dialog.

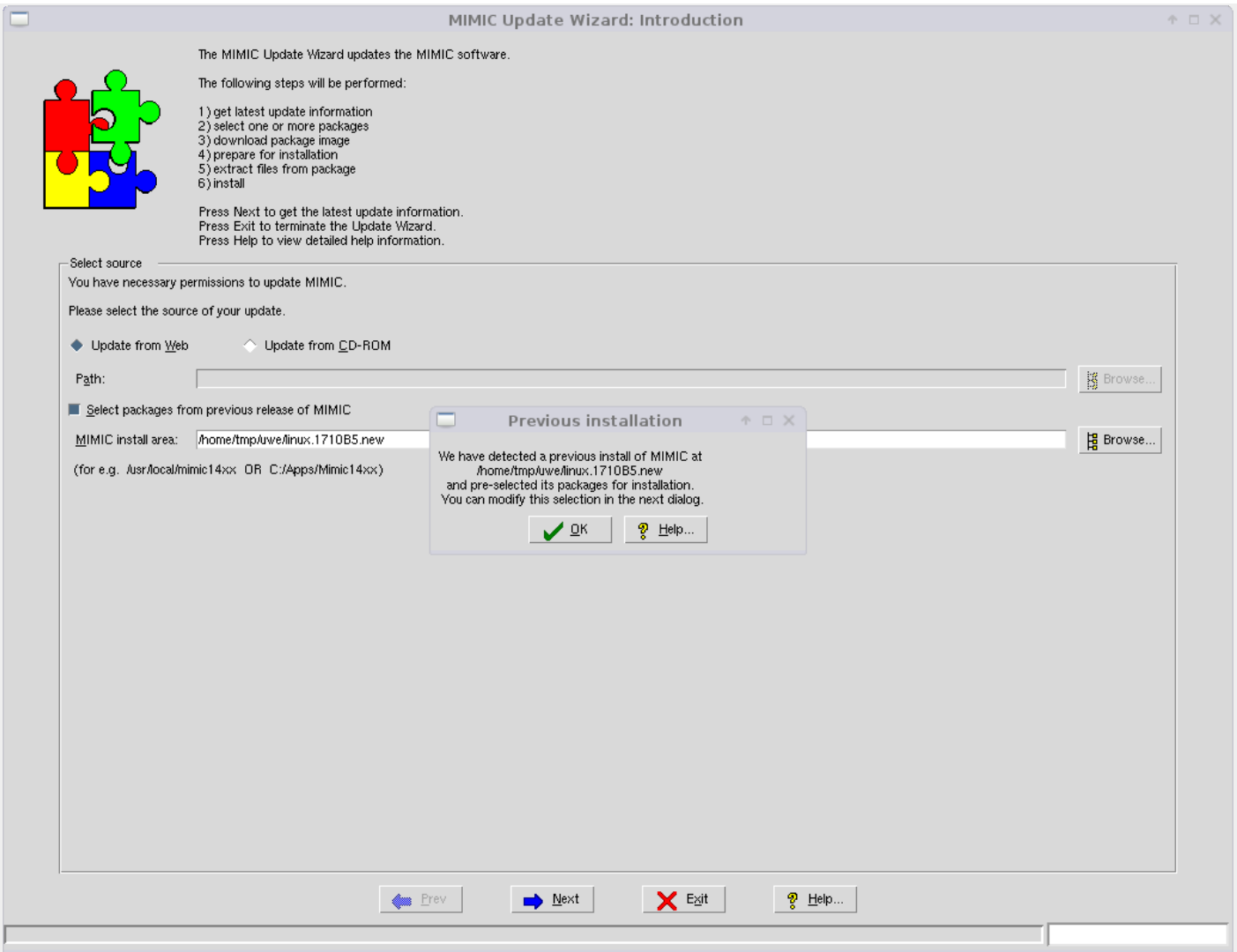


Figure 81: Update Wizard: Introduction

Select Package

A [Log Window](#) is displayed which logs all steps taken in the update.

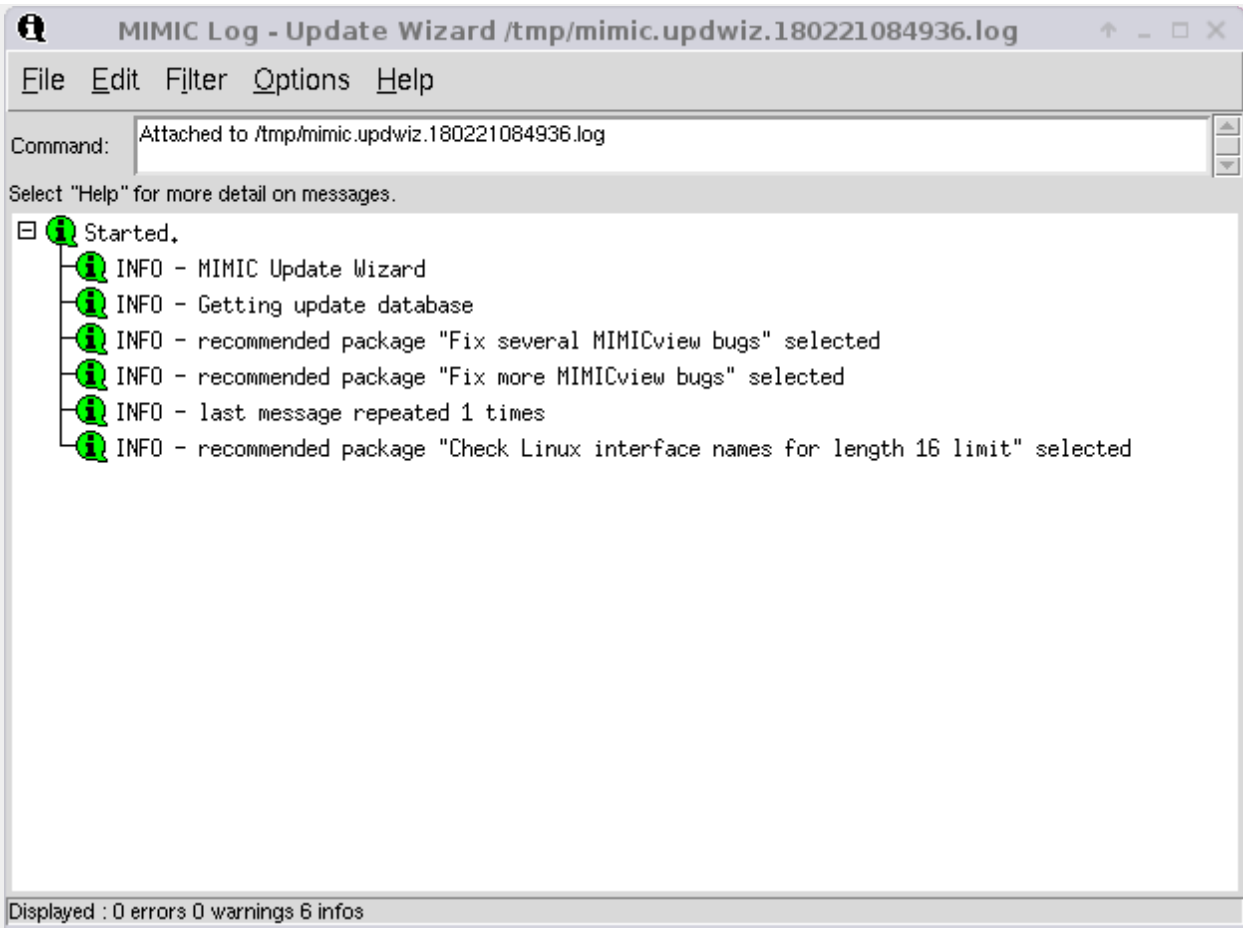


Figure 82: Update Wizard: Log

In this dialog, select one or more updates you want to install for your version of MIMIC. If there are no applicable updates, you are done.

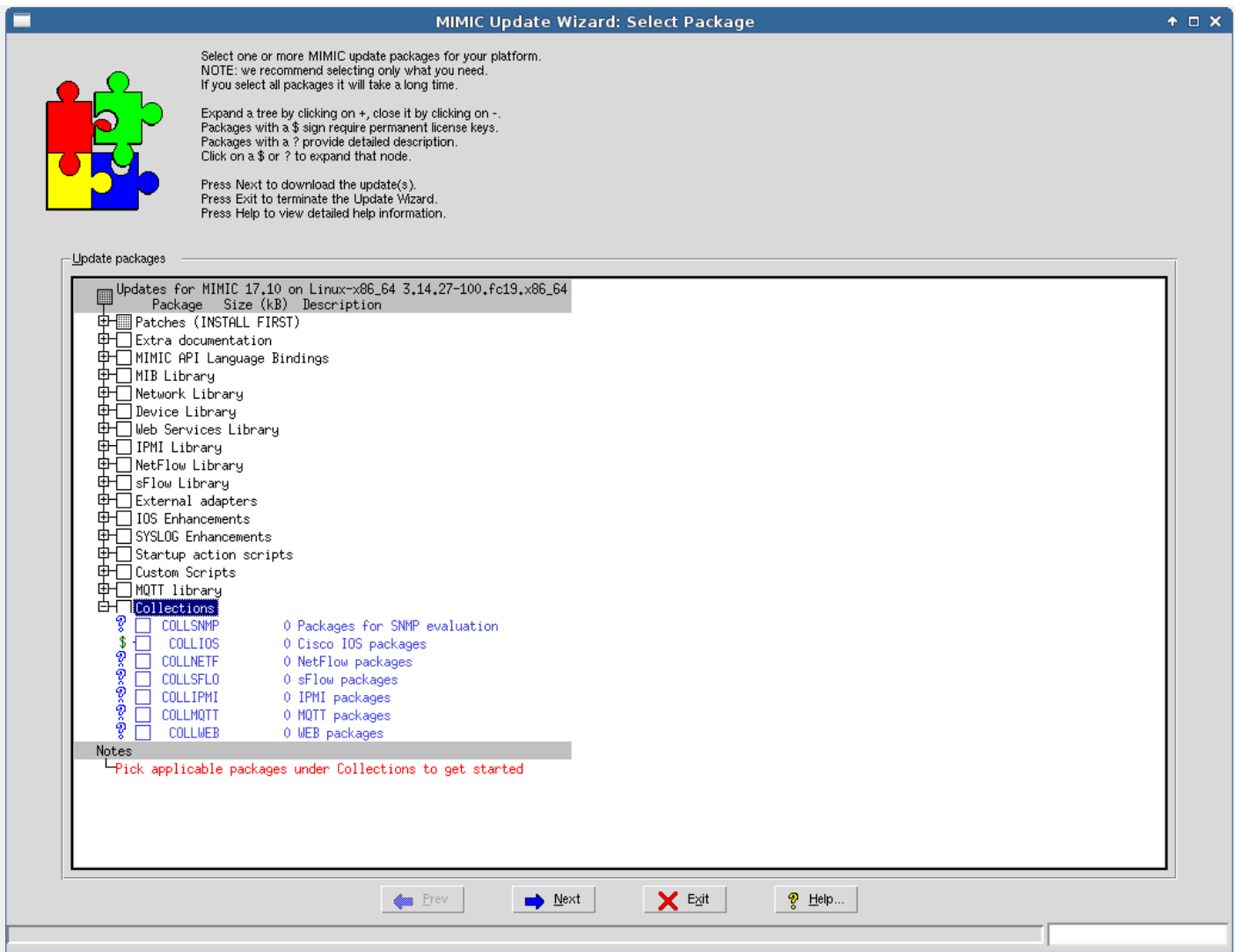


Figure 83: Update Wizard: Select

Pressing **Next** displays the Selection Summary dialog allowing you to verify your selected update package(s).

Pressing **Next** downloads the selected update package(s).

Download Images

Each update is downloaded from the Gambit Communications Web site. The download progress is displayed on the Status and Progress bars at the bottom.

If the update has already been downloaded, you are asked whether to download again.

If the extraction tools (gzip, tar) are not executable, they are also downloaded.

Pressing **Next** prepares for installation. The wizard automatically continues after some number of seconds, as displayed in the status bar.

Prepare For Installation

Certain updates need extra preparation to complete the update. This is done here. You will be prompted for any extra steps.

Pressing **Next** extracts the files. The wizard automatically continues after some number of seconds, as displayed in the status bar.

Extract Files

Files are extracted from the update. The list of files is displayed for your information.

Pressing **Next** extracts the contents of the update, and moves files to the shared area. Any existing files are backed up. The wizard automatically continues after some number of seconds, as displayed in the status bar.

Move Files

First, the contents of the update are extracted into a temporary directory. Then files are moved from the temporary directory to the shared area.

While files are being moved, the progress is displayed on the Status and Progress bars.

Diagnostic Wizard Reference

Introduction

The Diagnostic Wizard makes it easier to diagnose and report problems with MIMIC. In case of technical problems, you can use this wizard to collect information from your installation to be sent to Gambit Technical Support. You traverse a set of dialogs, each collecting different information, and the last dialog sends it after your explicit confirmation.



Select Configuration Files

This dialog lets you select the various MIMIC configuration files. You can click on the + box on a node to expand it, on the - to collapse it. You can click on a checkbox to include a file in the report. You can click on a checkbox of a category to include all files in the category. You can double-click on a file to examine its contents.

By default, relevant system and thread information is included. You can review the information with the `Details...` button.

The next 9 dialogs let you add more MIMIC files to send in your report. As a shortcut, if you want to skip the rest of the dialogs, you can select `Skip adding files and send report` and press `Next`.

Select MIB Files

This dialog lets you select your compiled MIB files. This will not send the source code of your MIBs, to prevent divulging proprietary MIB files. If you need to transfer the source code of your MIBs, Gambit support will give you instructions to transfer them manually.

Select Simulation Files

You can select simulations to be collected. Since these are quite large, select only the simulations necessary to diagnose your problem.

Select Walkfiles

This dialog lets you select your Walkfiles. You can additionally select walkfiles from other areas by pressing the "Browse..." button. Since these are quite large, select only the walkfiles necessary to diagnose your problem. You can inspect the walkfiles by double-clicking on them.

Select Snapshot Files

You can select `snapshots` to be collected. Since these are quite large, select only the snapshots necessary to diagnose your problem.

Select Discovery Files

You can select `discovery sessions` to be collected. Since these are quite large, select only the discoveries necessary to diagnose your problem.

Select Trap Series Files

You can select `trap series` to be collected. Since these are quite large, select only the trap series necessary to diagnose your problem.

Select Log Files

Any necessary log files to diagnose your problem can be selected here. The logfiles are sorted in reverse temporal order (most recent first), and show the highest severity in the file. Since these are quite large, select only the log files necessary to diagnose your problem. You can inspect the log files by double-clicking on them.

Review Selected Files

The complete list of selected items is shown, and you can review it and remove items.

Send Files

Currently, only manual and e-mail transfer mode are supported. Manual transfer just creates the compressed tar-ball with your selected files. You can transfer it to Gambit manually, eg. as an attachment in an e-mail message.

If your mail configuration is defined in the [Configuration Wizard](#) then E-mail mode can send the e-mail for you. You just have to fill in your e-mail address in the `From` field. All other fields are optional.

Sanity Wizard Reference

Introduction

The Sanity Wizard lets you check the integrity of your MIMIC data (ie. the files in your private area). You can use this wizard upon request from the Gambit Technical Support team.



Select Items

In the first dialog you select the items to be checked. By default, all checks will be performed, as indicated with the checkmarks on the items. The set of checks is broken down by

- Lab Configurations
- Device Configurations
- MIB Directory

Each check indicates the number of files and checks involved. You can select/deselect any subset of the checks. Only if there is something to check will you be able to continue to the next step.

Detection Results

This dialog shows a summary of the check results. It will indicate how many checks have been performed, and how many have passed/failed. The Log Window will show details of the checks.

i. Performance Wizard Reference

Introduction

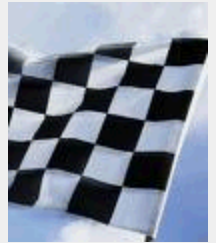
The Performance Wizard lets you check the SNMP performance of your system running MIMIC. It runs a standard test which simultaneously walks a varying number of agents with the MIMIC Recorder. This attempts to capture the performance characteristics of a typical scenario involving a network of agents being accessed by a small set of management applications.

This process is useful if you want to gauge the performance of your system in a standard way. You can also incrementally tune the system and measure whether the performance is actually improved.

The wizard will first collect system information and will allow you to identify the system.

Then it will run a series of performance tests using the MIMIC Recorder. This test lasts a couple of minutes.

Finally you can choose to report this data. By doing so, you will be able to contribute to a database of systems.



System Information

In the first dialog the wizard collects information about your system. The `Details...` button displays the collected information.

Optionally you can identify the brand, model and description of your system. For example, if you have multiple MIMIC systems of the same type, you can put the system name in the `Description` field. If you decide to share your results with Gambit, the more identifying information you provide about your system, the better our database will be.

Setup Test

The performance test consists of configuring 10 agents, and simultaneously walking subsets of them using the MIMIC Recorder.

If you want to preserve your lab configuration, save it now with the `File -> Save As...` menu command.

By default the agent addresses are 10.0.0.1 to 10.0.0.10. Optionally specify alternative addresses of the agents in this dialog, then press `Next` to start the test.

The test will run for a couple of minutes, and display its progress in a [Log Window](#).

Performance Results

This dialog shows a summary of the performance results.

Send Files

This dialog sends the files for inclusion in Gambit's database.

Currently, only manual and e-mail transfer mode are supported. Manual transfer just creates the compressed tar-ball with your selected files. You can transfer it to Gambit manually, eg. as an attachment in an e-mail message.

If your mail configuration is defined in the [Configuration Wizard](#) then E-mail mode can send the e-mail for you. You just have to fill in your e-mail address in the `From` field. All other fields are optional.

Introduction

The Protocol Wizard lets you install and uninstall optional protocol modules into MIMIC.

First, you can select the protocol modules to be installed or uninstalled. The currently installed modules are already selected in the list.

Next, you need to supply the license keys for each protocol module to be installed. Once you proceed, the modules are installed.

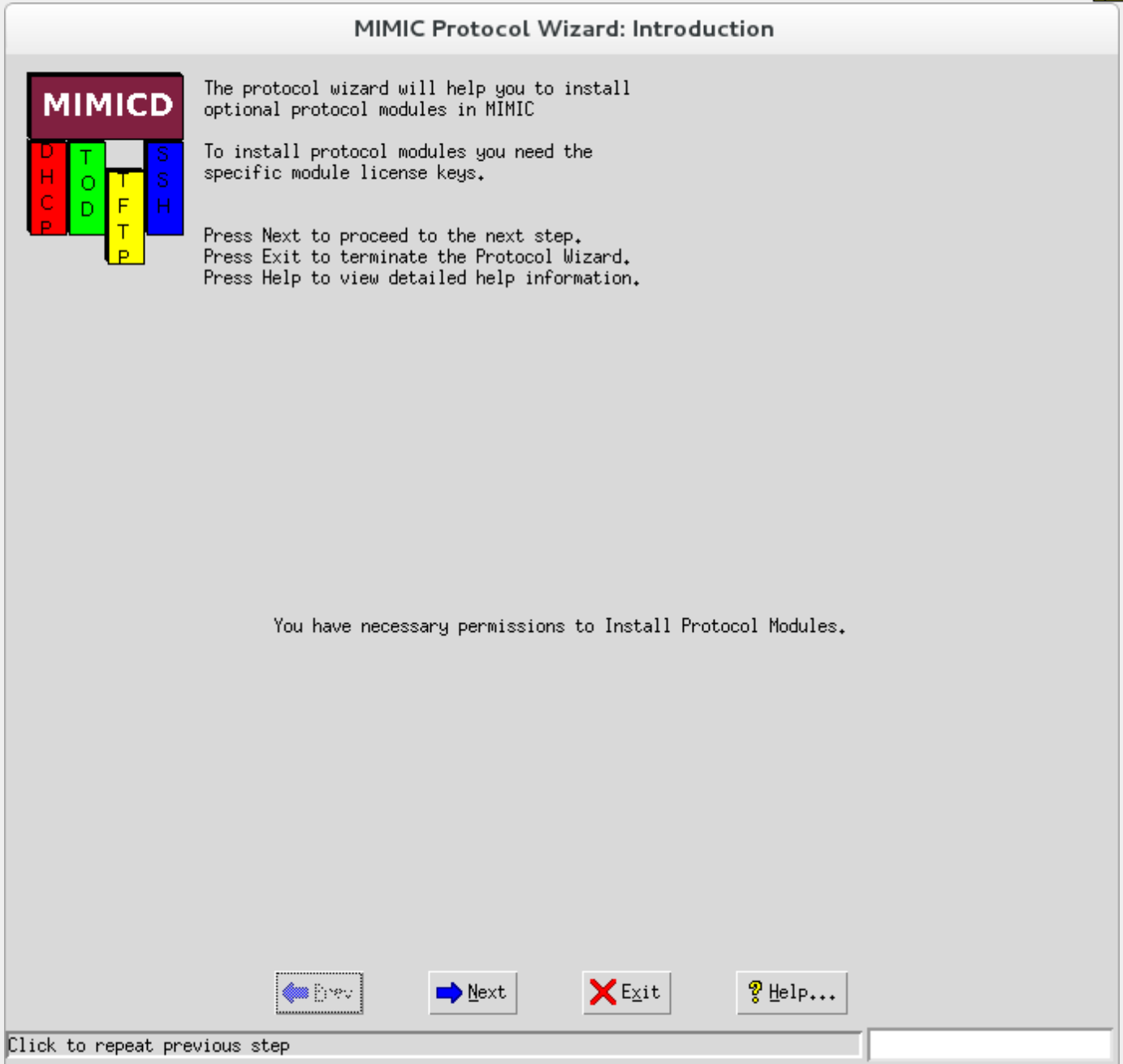
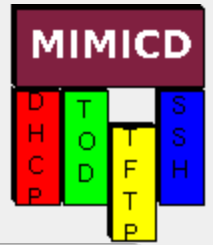


Figure 90: Protocol Wizard: Introduction

Select Protocols

In the first dialog the wizard collects information about protocol modules to be installed or uninstalled.

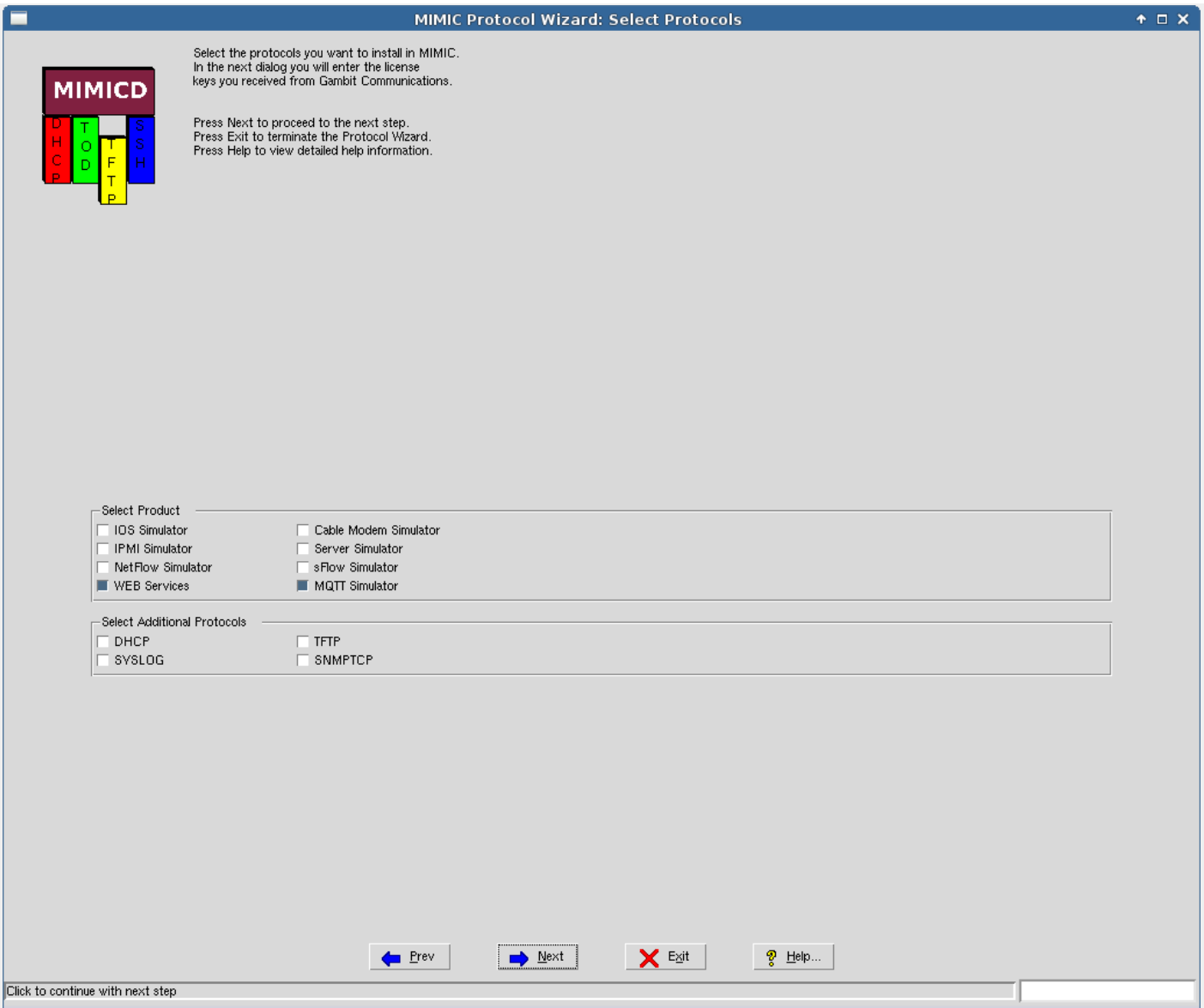
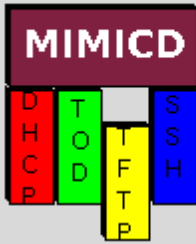


Figure 91: Protocol Wizard: Protocols

Supply License Keys

For each selected protocol which needs a license key, enter the key string you received from Gambit Communications.

MIMIC Protocol Wizard: Supply License Keys



Paste the license keys in the text box
OR
Browse the location of the license file(*.lic)
by clicking on the "From file..." button.
Press Next to proceed to the next step.
Press Exit to terminate the Protocol Wizard.
Press Help to view detailed help information.

License Keys

IOS:	<input type="text"/>	From File..
Cable:	<input type="text"/>	From File..
IPMI:	<input type="text"/>	From File..
Server:	<input type="text"/>	From File..
NetFlow:	<input type="text"/>	From File..
sFlow:	<input type="text"/>	From File..
SNMPTCP:	<input type="text"/>	From File..
WEB:	<input type="text"/>	From File..

Protocols to Install

DHCP IPMI NETFLOW SFLOW SSH SYSLOG TELNET TFTP WEB

Protocols to Uninstall

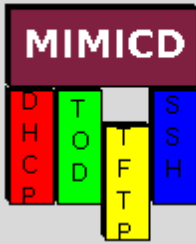
Click to continue with next step

Figure 92: Protocol Wizard: Keys

Results Installing/Uninstalling Protocols

This dialog displays the results of the install/uninstall.

MIMIC Protocol Wizard: Result Installing/Uninstalling Protocols



Result of Installing/Uninstalling Protocols


Press Finish to complete the wizard.
Press Exit to terminate the Protocol Wizard.
Press Help to view detailed help information.


Result


Installing/Uninstalling Protocols.


Successfully installed modules:
DHCP IPMI NETFLOW SFLOW SSH SYSLOG TELNET TFTP WEB

Click 'Finish' to restart the MIMIC daemon and MIMICView so
that the changes that were made will take effect

 Prev

 Finish

 Exit

 Help...

Click to continue with next step

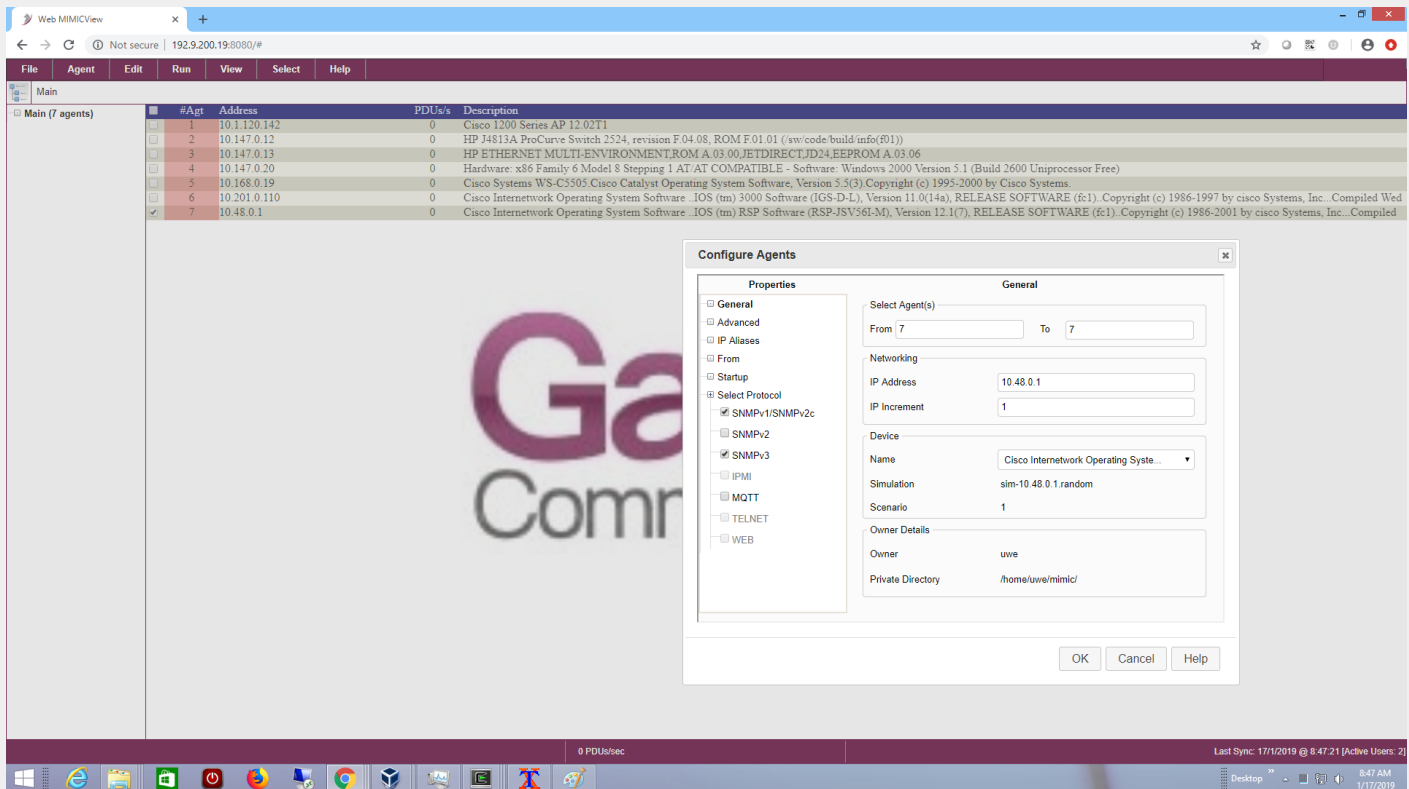
Figure 93: Protocol Wizard: Results



MIMIC WEBUI Guide

Preface

The MIMIC WEBUI is an optional browser-based web interface to control MIMIC Simulator. It is an alternative to the [MIMICview](#) desktop-based graphical user interface (GUI), or the multitude of scripting interfaces in different programming languages.



The WEBUI server is installed by the MIMIC administrator as an optional [Update Wizard](#) package, and needs to run either on the same system as the MIMICD simulator daemon, or a system that can connect to it. See [Installation](#) section below.

Once you have installed it, users can consult its [user guide](#) on how to use it to control MIMIC.

Table of Contents

- [Requirements](#)
- [Installation](#)
- [WEBUI Server](#)
- [User Guide](#)

Requirements

Linux or Windows 64-bit Java
Java 8, 9 (we currently do NOT support Java 11 SE)

Installation

Install the WEBUI packages in [Update Wizard](#).

Once the WEBUI packages have been installed on your MIMIC system, you need to run the WEBUI server. You can run it on the same system as MIMICD, but if you want more CPU power to be available to both, you should run it on a different system.

Since WEBUI is implemented as a MIMIC client serving the web-based user interface, if run on a different system, then all the same rules apply as all other

MIMIC clients, as detailed in this [FAQ entry](#). In particular, you want the MIMIC install area and private area(s) accessible at the same absolute paths on both MIMIC and WEBUI system, eg. via NFS and/or symbolic links.

i. WEBUI Server

The `webui.sh` shell script is a convenience front-end script to invoke the Java-based WEBUI server. It is designed to run in 2 modes:

- Local mode - in this mode the WEBUI server provides access to a single instance of MIMIC to local users.
- Lab mode - in this mode, the WEBUI server provides lab access to one or more instances of MIMIC to remote users, possibly over the Internet.

It takes these command line options:

- `-mh <lt;address>` | `--mimichost <lt;address>` for MIMIC host address
- `-mp <lt;port #>` | `--mimicport <lt;port #>` for MIMIC management port (default 9797)
- `-a <lt;agent-range>` | `--agents <lt;agent-range>` for range of agents (default all agents)
- `-sh <lt;address>` | `--servicehost <lt;address>` for client facing web service (default 0.0.0.0)
- `-sp <lt;port #>` | `--serviceport <lt;port #>` for client facing web service (default 8080)
- `-sd <lt;path>` | `--shared_dir <lt;path>` for mimic shared directory path
- `-td <lt;path>` | `--temp_dir <lt;path>` for mimic temporary directory path
- `-es` | `--enablesecured` to enable HTTP secured server
- `-kf <lt;filepath>` | `--keyfile <lt;filepath>` path of the keystore file (in PKCS12 format)
- `-pw <lt;password>` | `--password <lt;password>` password for the keystore file
- `-ssh <lt;address>` | `--securedservicehost <lt;address>` for client facing web service (default 0.0.0.0)
- `-ssp <lt;port #>` | `--securedserviceport <lt;port #>` for client facing web service (default 8443)
Note : Both HTTP and HTTPS can coexist in single server instance using different port numbers

Lab Mode

- `-cs` | `--cloudserver` to run in cloud based lab server mode
Admin web service is available only in cloud based lab server mode.
- `-ah <lt;address>` | `--adminhost <lt;address>` for admin web service (default 0.0.0.0)
- `-ap <lt;port #>` | `--adminport <lt;port #>` for admin web service (default 8080) to run as Browser based UI server for specified MIMIC Host
- `-sah <lt;address>` | `--securedadminhost <lt;address>` for admin web service (default 0.0.0.0)
- `-sap <lt;port #>` | `--securedadminport <lt;port #>` for admin web service (default 8443)

Saved runtime configuration

- `-ic` | `--ignoreconfig` to ignore previously saved configuration, and will not saved configuration with new value
- `-rc` | `--resetconfig` use with `-ic` to overwrite save config with new values

Once the server is running, you can restrict it's CPU usage by setting it's [CPU affinity](#) on Linux, eg. like this

```
[root@fc28 ~]# ps ax | grep java
1304 pts/0    Sl      0:11 java -cp ../../akka/*:../Apache-Oro/*:webui.jar WebUI.WebUIServer
-sd ../../ -mh 192.9.192.210 -mp 9797

[root@fc28 ~]# taskset -c -p 0 1304
pid 1304's current affinity list: 0,1
pid 1304's new affinity list: 0
```

ii. User Guide

The WEBUI server contains the usual `help` buttons and menus. For more details see its [user guide](#).



MIMIC Virtual Lab Online Documentation


[<< Previous Section](#) [Next Section>>](#)

1. [Table of Contents](#)

2. [Release Notes](#)

3. [Windows Installation Instructions](#)

4. [Quick Start Guide](#)

This button  is used throughout the documentation as a shortcut into the Quick Start Guide.

5. [Virtual Lab User Guide](#)

6. [Virtual Lab and JRE](#)

7. [Appendix A: Virtual Lab IOS Commands](#)

8. [Appendix B: Empty](#)

9. [Appendix C: Common Error Messages](#)

10. [Appendix D: Frequently Asked Questions](#)

This documentation is best viewed with a commercial web browser such as Opera, Arena, Mozilla or Internet Explorer.

[<< Previous Section](#) [Next Section>>](#)

Copyright © 2002-2016 Gambit Communications, Inc. All Rights Reserved.

MIMIC is a registered trademark of Gambit Communications, Inc.

All other product and brand names are trademarks or registered trademarks of their respective holders.



MIMIC Utilities Guide

Preface

The MIMIC Utilities are command-line companion programs for specialized tasks.

Table of Contents

- [grapher](#)
- [oidinfo](#)
- [walkfile converters](#)

grapher - a generic graphing tool

The Grapher lets you plot a interactive XY Line graph for more than one set of data to visualize the change over a period of time.

The grapher may be invoked using the following command line options:

- `--type graph/barchart`
Specify the type of pictorial representation to be used to depict data. Currently only "graph" option may be used to represent data as XY Line graph.
- `--series *`
Specify the characteristics of a series with this option. Color may be any one of following green, blue, red or yellow. Symbol may be diamond, circle, triangle or square and the label is brief description about the series which used in legend for the series in the graph window.
- `--source`
Specify the filename for which the grapher reads the data to be plotted. The file has text in the following format for each data point in the series.

```
cmd series-id x-axis-value y-axis-value
e.g PLOT 1 0 2
(0,2) is to be plotted on series 1 in the graph.
```

Currently two commands are supported PLOT and EXIT. EXIT command is without any arguments.

The following options are optional.

- `--scroll`
This option may be used to display an horizontal scrollbar in the graph window.
- `--xmin n`
Specifies the minimum limit of x axis. Any x coordinate of data point less than n is not displayed.
- `--xmax n` Specifies the maximum limit of x axis. Any x coordinate of data point greater than n is not displayed.
- `--ymin n`
Specifies the minimum limit of y axis. Any y coordinate of data point less than n is not displayed.
- `--ymax n`
Specifies the maximum limit of y axis. Any y coordinate of data point greater than n is not displayed.

oidinfo - display OID information

This utility gives information about a MIB object either by name or OID. The information returned is in internal format, but can be useful to learn more about a MIB object.

Examples:

```
ultra5[1556] oidinfo ifOperStatus
INFO 08/25.08:47:22 - NAME ifOperStatus = 1.3.6.1.2.1.2.2.1.8
INFO 08/25.08:47:22 - MIB = IF-MIB
parent = ,      subid = 8, name = ifOperStatus, access = 1
type = 3
1, up
2, down
3, testing
4, unknown
5, dormant
```

```
ultra5[1557] oidinfo 1.3.6.1.2.1.2.2.1.8
INFO 08/25.08:47:41 - OID 1.3.6.1.2.1.2.2.1.8 = ifOperStatus
INFO 08/25.08:47:41 - MIB = IF-MIB
parent = ,      subid = 8, name = ifOperStatus, access = 1
type = 3
1, up
2, down
3, testing
4, unknown
5, dormant
```

walkfile converters

The convertwalk utility converts walkfiles from 3rd party packages into a format that the MIMIC Recorder understands. MIMICView invokes this utility from the Walkfile Importer dialog wherever walkfiles are accessed, eg. in the Simulation->Record dialog.

Additionally, we provide some obsolete shell scripts to convert files. These scripts convert walkfiles from 3rd party packages into a format that the MIMIC Recorder understands. The scripts live in the common/ directory.

The following formats can be converted:

- HP Openview

```
wazoom[265] ./hpov.csh
Usage: ./hpov.csh infile outfile [start]
```

- HP Netmetrix

```
wazoom[266] ./netmetrix.csh
Usage: ./netmetrix.csh infile outfile [start]
```

- UCD snmpwalk

```
wazoom[267] ./ucd.csh
Usage: ./ucd.csh infile outfile [start]
```

- NetHealth

```
wazoom[267] ./nethealth.csh
Usage: ./nethealth.csh infile outfile [start]
```



[<< Previous Section](#) [Next Section >>](#)

MIMIC Tcl API Guide

For historical reasons, the Tcl command set is documented in the [Simulator Guide](#), [MIMICShell Reference](#) section.

[<< Previous Section](#) [Next Section >>](#)



[<< Previous Section](#) [Next Section >>](#)

MIMIC Java API Guide

Table of Contents

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)

Requirements

The MIMIC Java API requires the use of the [Java\(TM\) JDK 7](#), [OpenJDK 11](#), or newer. We have tested up to Java 13 on both Linux and Windows.

Class Reference

Click [here](#) for the generated reference documentation.

Examples

- If you prefer, you can create a .jar that contains all the .class files of the API with

```
jar cf MIMIC.jar Mimic/*class Utils/*class
```

- The `TestApp.java` file contains a sample test program which exercises all the classes of the MIMIC Java API.

To compile:

```
cd java/test
javac -classpath .. TestApp.java
```

To run (assuming MIMIC is running on this machine):

```
java -classpath ../../juds/juds-0.9.jar TestApp
```

NOTE: if you want to pass environment variables into a Java application, you have to explicitly specify them with the `-D` command line option to the Java virtual machine, eg.

```
java -classpath ../../juds/juds-0.9.jar -DMIMIC_PRIV_DIR=some-path TestApp
```

- The [MimicApplet.html](#) web page contains invocation of a simple applet, that uses the MIMIC Java API to display information about MIMIC running on a system that you can specify. (NOTE: If you are connecting to a remote system, you will need to modify your security policy as detailed below.)

Point your browser to load this .html file. The browser will let you load the needed plug-in. This has been tested with both Netscape 4.7x and Internet Explorer 5.x on all supported platforms.

If you are using the applet to monitor remote systems, A change needs to be made in a file called `java.policy` for the Java Runtime Engine (JRE) which is being used by the browser. The JRE is the plugin that is downloaded for Java1.2 if not already installed. On Windows, the default location is

```
c:\Program Files\JavaSoft\Jre\lib\security\
```

Add a line to allow a connection to the MIMIC port:

```
permission java.net.SocketPermission "*:9797", "connect";
```

- The [Script Generator](#) will generate Java code from MIMICView dialogs.

[<< Previous Section](#) [Next Section >>](#)



MIMIC Perl API Guide

Table of Contents

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)

Requirements

The MIMIC Perl API requires the use of the [Perl](#) 5.8.8 or later. Perl is already installed on Solaris and Linux platforms, but will have to be installed on Windows platforms. The MIMIC Perl programs expect the perl executable to be accessible through `/usr/local/bin/perl`, so you would need to create a symbolic link to the installed executable. MIMIC has been tested with Perl 5.8.8 and later on all supported platforms.

NOTE: Recent versions of Perl have removed ".", the current working directory, from the @INC include path. If you get these errors:

```
Can't locate Mimic.pm in @INC (you may need to install the Mimic module) (@INC contains:
/usr/local/lib64/perl5 /usr/local/share/perl5 /usr/lib64/perl5/vendor_perl
/usr/share/perl5/vendor_perl /usr/lib64/perl5 /usr/share/perl5) at TestApp.pl line 9.
BEGIN failed--compilation aborted at TestApp.pl line 9.
```

then just supply the `-I .` command line option to Perl.

Class Reference

The Perl API is modelled after the [Java API](#). All classes and methods have the same name as in Java, except that Perl programming constructs are used. Click [here](#) for the generated Java reference documentation.

Examples

- The `mimicsh-perl` program is an interactive Perl shell with the MIMIC API built-in. Type "help" for online help.

```
./mimicsh-perl
MIMIC Perl API shell
Copyright (c) 2001-2014 Gambit Communications, Inc.
Type "help;" for online help.
mimicsh-perl> help
MIMIC Perl API shell:
The syntax of the Perl API is based upon the Java API.
For details, look at the Java API Guide in the online
documentation.
Example:
mimicsh-perl> $client = new Mimic::Client();
mimicsh-perl> $session = $client->open_session("localhost", 9797);
mimicsh-perl> $agent = $session->get_agent(2);
mimicsh-perl> $valuespace = $agent->get_valuespace();
mimicsh-perl> $sysDescr = $valuespace->get_value( "sysDescr", "0", "v" );
mimicsh-perl> print $sysDescr, "\n";
```

- The [Script Generator](#) will generate Perl scripts from MIMICView dialogs.



MIMIC C++ API Guide

Table of Contents

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)
- [Actions](#)

Requirements

The MIMIC C++ API requires the use of Gnu C++ 4.8.5 on the Linux platforms, or Visual Studio 2012 on Windows. If compiling on Visual Studio 2013 on Windows, you can enable "2012 mode" as documented [here](#) where it says

You can use Visual Studio 2013 to open a C++ project that was created in Visual Studio 2012 or Visual Studio 2010 SP1. If you want to use the Visual Studio 2013 build environment to build a project that was created in Visual Studio 2012, you must have both versions of Visual Studio installed on the same computer.

The user needs to use the header and library files shipped with a MIMIC installation to build a C++ application or action binary. All required API files are shipped under the "cpp" directory in the installation. The structure and contents of this directory is as follows :

```
[ - ] cpp
  [ - ] inc
    COMMON
      mimic_action.hh
      mimic_agent.hh
      mimic_api.hh
      mimic_apiutils.hh
      mimic_client.hh
      mimic_errorinfo.hh
      mimic_exports.hh
      mimic_helper.hh
      mimic_session.hh
      mimic_valuespace.hh
  [ - ] lib
    WINDOWS
      libcppapi.lib
      libcppapi_internal.lib
      libgci.lib
      libwin.lib
      pthreads.lib
    SOLARIS
      libcompat.a
      libcppapi.a
      libcppapi_internal.a
      libgci_pthreads.a
    LINUX
      libcppapi.a
      libcppapi_internal.a
      libgci_pthreads.a
  [ - ] src
    WINDOWS
      Build-action-vc6.bat
      Build-app-vc6.bat
    UNIX
      Makefile-testaction
      Makefile-testapp
    COMMON
      entryStatus.cc
      ifAdminStatus.cc
      mimic_helper.cc
      rowStatus.cc
      start.cc
      testaction.cc
      testapp.cc
      testtimer.cc
      timefilter.cc
```

Sample "Unix Makefiles" or "Windows batchfiles" are shipped for building a typical C++ client application or a C++ action. Also sample source code is provided to serve as a easy starting point.

Class Reference

The C++ API is modelled after the [Java API](#). All classes and methods have the same name as in Java, except that C++ programming constructs are used. Click [here](#) for the generated Java reference documentation.

Examples

- The `cpp/src/testapp.cc` program is a sample test program.

```
% make -f Makefile-testapp
g++ -g -I../inc -DDEBUG -D_REENTRANT -c -o testapp.o testapp.cc
g++ -o testapp testapp.o ../lib/libcppapi.a ../lib/libgci_threads.a -lpthread

% cd ..

% src/testapp
TestApp: starting...
test_session_init : Session : 0x80aa6c0
test_session_init over
TestApp: config file = agent.cfg
TestApp: max agents = 0
TestApp: last agent = 6
TestApp: version = 5.40
TestApp: num clients = 2
TestApp: configured agents = [1, 2, 3, 4, 5, 6]
TestApp: active data :
TestApp: configured changed agents = [1, 2, 3, 4, 5, 6]
TestApp: state changes :
  agent = 1   state = 2
  agent = 2   state = 2
  agent = 3   state = 2
  agent = 4   state = 2
  agent = 5   state = 2
  agent = 6   state = 2
...
```

- The [Script Generator](#) will generate C++ code from MIMICView dialogs.

Actions

The C++ actions need to be compiled into a dynamically loadable library (.so on Unix, .dll on Windows). It also needs to export a specific interface that can be invoked by the simulator. This requires the code to be structured in a certain manner, which is illustrated in the following sample code:

```
#include "mimic_action.hh"

#ifdef WIN32
__declspec( dllexport )
#endif
extern "C"
int
process_action_message (
    int msg_id,
    void* msg_data
)
{
    switch( msg_id )
    {
        // on load only
        case ACTION_MODULE_REGISTER:
        {
            action_module_register_t *reg_data =
                (action_module_register_t*)msg_data;

            // verify version for the simulator
            reg_data->version.sprintf (
                ACTION_BUILD_ID
            );

            return ACTION_SUCCESS;
        }

        // on load only
        case ACTION_MODULE_INIT:
        {
            action_module_init_t *init_data =
                (action_module_init_t*)msg_data;

            //...

            return ACTION_SUCCESS;
        }

        // on unload only
    }
}
```

```

case ACTION_MODULE_UNINIT:
{
    action_module_uninit_t *uninit_data =
        (action_module_uninit_t*)msg_data;

    //...

    return ACTION_SUCCESS;
}

// on every invocation
case ACTION_MODULE_RUN:
{
    action_module_run_t *run_data =
        (action_module_run_t*)msg_data;

    // access globals
    printf ( "SharedDir = %s\n",
            run_data->globals->get("gSharedDir")
        );

    // ...

    return ACTION_SUCCESS;
}
}

return ACTION_SUCCESS; // default success
}

```

As per the code segment above, the DLL is required to export a well-known function "process_action_message" which takes an integer and an opaque pointer as arguments. This is invoked by the simulator for various events.

The REGISTER and LOAD are called during loading of the action, UNINIT is called during unloading of the action and RUN is called for each invocation of the action.

The header "cpp/inc/mimic_action.hh" defines function prototypes as well as data structures required to implement the DLL. Some sample implementations are also available in `cpp/src/testaction.cc`, `cpp/src/entryStatus.cc`, `cpp/src/rowStatus.cc`, `cpp/src/start.cc`, `cpp/src/timefilter.cc`.

[<< Previous Section](#) [Next Section >>](#)



[<< Previous Section](#) [Next Section >>](#)

MIMIC Python API Guide

Table of Contents

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)
- [Jython](#)

Requirements

The MIMIC Python API is an optional [Update Wizard](#) package, and requires the use of the [Python](#) 2.7 or later. Python is already installed on Linux platforms, but will have to be installed on the other platforms. MIMIC has been tested with Python 2.7.3 and 3.3.2 on Linux, and Python 2.7 and 3.5.1 on Windows.

Class Reference

The Python API is modelled after the [Java API](#). All classes and methods have the same name as in Java, except that Python programming constructs are used. Click [here](#) for the generated Java reference documentation.

Examples

These example programs with source code show different usage of the Python API to control MIMIC:

- The [TestApp.py](#) program is a small test program to exercise the MIMIC API.
- The [tkdemo.py](#) sample program uses the [TkInter](#) GUI framework to implement the equivalent of the [tkdemo.tcl](#) Tcl script.
- The [mimicdthreads.py](#) client displays MIMICD thread statistics to determine performance of MIMICD threads. You can download it with [Update Wizard](#).
- Another optional [Update Wizard](#) patch is this [MIMICview](#) clone implemented in Python.

Jython

- The Python API has been tested against Jython using version 2.2.1 on both Windows Vista and Linux (Fedora Core 8).
- A performance test of the Python API in a Jython environment shows a significant loss in response time as compared to a native Python environment.

[<< Previous Section](#) [Next Section >>](#)



MIMIC PHP API Guide

Table of Contents

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)

Requirements

The MIMIC PHP API is an optional [Update Wizard](#) package, and requires the use of the [PHP](#) 5.5 or later. PHP is already installed on recent Linux platforms, but will have to be installed on the other platforms. MIMIC has been tested with PHP 5.6 on Linux.

Class Reference

The PHP API is modelled after the [Java API](#). All classes and methods have the same name as in Java, except that PHP programming constructs are used. Click [here](#) for the generated Java reference documentation.

Examples

- The [testapp.php](#) program is a small test program to exercise the MIMIC PHP API just like in other language bindings. Invoke as follows (assuming MIMIC is running on the local host):

```
% php testapp.php

****Starting PHP TestApp****
TestApp: test_get:
get_host: localhost
get_port: 9797
get_user: mimicuser
get_number: 1
get_privdir: /home/mimicuser/mimic/
get_version: 16.00
get_cfgfile: test-agent.cfg
get_configured_list: 1 2 3 4 5 6 7
get_active_list:
get_active_data_list:
get_changed_config_list: 1 2 3 4 5 6 7
get_changed_state_list: {1 2} {2 2} {3 2} {4 2} {5 2} {6 2} {7 2}
get_log:
get_interfaces: {p4p1}
get_cfg_file_changed: 1
get_sim_privdir: /home/uwe/mimic/
get_protocols: snmpv1,snmpv2c,snmpv2,snmpv3,NETFLOW,TELNET
get_product: 29
get_netaddr: 192.9.200.14
get_netdev:
...
```

- The [phpinfo.php](#) program is a small diagnostic program to print PHP information for your PHP environment. If any problems with PHP, run

```
% php phpinfo.php
```

and send us the output with your problem report.

- The [MIMICView-PHP](#) GUI is a package downloadable with the [Update Wizard](#) and provides a web-based interface to MIMIC.



MIMIC Javascript API Guide

Table of Contents

- [Requirements](#)
- [Class Reference](#)
- [Javascript API Server](#)
- [Examples](#)

Requirements

The MIMIC Javascript API exports 2 separate paradigms:

- a synchronous API
- an asynchronous API based on [Mozilla Fetch](#)

Both APIs require a [Javascript](#) enabled browser to be used. The browser acts as a client to the Javascript API Server detailed [below](#).

Class Reference

Synchronous API

The synchronous Javascript API is modelled after the [Java API](#) with these differences:

- All the `Session` class methods implementing global management commands are grouped under `mimic_` and have prefix `mimic_` followed by the `Session` class method name, e.g.,

```
mimic_get_max()
```

- All the `Agent` class methods are grouped under `mimic_agent_` and have prefix `mimic_agent_` followed by the `Agent` class method name, e.g.

```
mimic_agent_get_host()
```

- All the `Valuespace` class methods are grouped under `mimic_agent_valuespace_` and have prefix `mimic_agent_valuespace_` followed by the `Valuespace` class method name, e.g.

```
mimic_agent_valuespace_get_info()
```

The Javascript API commands can be written in two additional formats:

- The equivalent of `mimic_` is `mimic.` or `m.`, thus we can also write the global commands as `mimic.` or `m.` followed by the Java method name, e.g.

```
mimic.get_max()  
m.get_max()
```

- The equivalent of `mimic_agent_` is `mimic.agent.` and `m.a.`, thus we can also write the Agent commands as `mimic.agent.` or `m.a.` followed by the Java method name, e.g.

```
mimic.agent.get_host()  
m.a.get_host()
```

- The equivalent of `mimic_agent_valuespace` is `mimic.agent.valuespace` and `m.a.vs`, thus we can also write the Valuespace commands as `mimic.agent.valuespace` or `m.a.vs` followed by the Java method name, e.g.

```
mimic.agent.valuespace.get_info()  
m.a.vs.get_info()
```

Asynchronous API

The asynchronous Javascript API is modelled after the synchronous methods above, but every method is of type [async](#).

Javascript API Server

The Javascript API Server needs to be running in order to execute the Javascript API commands. The steps to start the Javascript API Server are as follows:

- Open the command prompt on your machine and set the current working directory to the folder where the `jsapiserver.jar` file is located in your machine, by default in the `java/jsapiserver/` folder of your MIMIC install area..

- Then type the command

```
java -jar jsapiserver.jar --cors
```

to start the Javascript API Server.

- Usage is given below:

```
Usage: java -jar jsapiserver.jar
--host | -h <ip-addr>      ip address to listen on for web server
--port | -p <port>        port to listen on for web server
--mimichost | -mh <ip-addr> address of mimic host
--mimicport | -mp <port>   port to connect to on mimic host
--cors                    add Access-Control-Allow-Origin header to response
--verbose | -v            enable verbose mode
```

Examples

Synchronous API

- The `testapp.js` program is a small test program to exercise the synchronous MIMIC Javascript API.

You can invoke it by clicking on the link to [testapp.html](#) which includes the `testapp.js`.

If you get the error message

```
No response from server location 'http://127.0.0.1:8080'
```

then you need to run the [JS API Server](#) as detailed above, and point the `testapp.html` URL to it by appending `?jsapiserver=YOUR-SERVER&jsapiport=YOUR-PORT` where `YOUR-SERVER` is the IP address of the system where you are running the server and `YOUR-PORT` is the port.

Once it works you'll get the output like this:

```
MIMIC JS API Server location set to: http://YOUR-SERVER:YOUR-PORT
Connection to MIMICD version 19.00 is active
test_3510 : test_3510: session: cfgfile =
test_session_get : config file =
test_session_get : version = 19.00
test_session_get : max agents = 100000
test_session_get : protocols = snmpv1,snmpv2c,snmpv2,snmpv3,MQTT,NETFLOW,SYSLOG
test_session_get : last agent = 12400
test_session_get : clients = 20
test_session_get : configured list = 1,2,3,4,5,6,7,8,9,10
test_session_get : active list =
test_session_get : changed config agents = 1,2,3,4,5,6,7,8,9,10
test_session_get : interfaces = {p4pl}
test_session_get : cfg file changed = 1
test_session_get : privdir = /home/uwe/mimic/
test_session_get : product = 29
test_session_get : net address = 192.9.200.14
test_session_get : net dev =
test_session_get : expiration = ok
test_session_get : licensng = "{MIMIC Simulator Universal v19.00
built Jul 19 2018 10:46:47
Copyright (c) 1997-2018 Gambit Communications, Inc.
Evaluation license, expires 8/26/2019
Running 64-bit
machine x86_64
OS Linux 3.14.27-100.fc19.x86_64
4 CPUs}"
...
```

- The [MIMICview-JS](#) GUI is a package downloadable with the [Update Wizard](#) and provides a web-based interface to MIMIC.
- The open-source [node-red-contrib-mimic](#) package for Node-RED on Github integrates MIMIC with Node-RED via the Javascript API.

Asynchronous API

- The `testapp-async.js` program is a small test program to exercise the asynchronous MIMIC Javascript API.

You can invoke it by clicking on the link to [testappasync.html](#) which includes the `testapp-async.js`.

If you get the error message

```
Error from server location 'http://127.0.0.1:8080': Error: TypeError: NetworkError when
attempting to fetch resource.
```

then you need to run the [JS API Server](#) as detailed above, and point the `testapp.html` URL to it by appending `?jsapiserver=YOUR-SERVER&jsapiport=YOUR-PORT` where `YOUR-SERVER` is the IP address of the system where you are running the server and `YOUR-PORT` is the port.

Once it works you'll get the output like this:

```
MIMIC JS API Server location set to: http://YOUR-SERVER:YOUR-PORT
Connection to MIMICD is active
test_3510 : test_3510: session: cfgfile =
test_session_get : config file =
test_session_get : max agents = 100000
test_session_get : last agent = 10
test_session_get : version = 18
test_session_get : num clients = 2
test_session_get : configured agents = 1,2,3,4,5,6,7,8,9,10
test_session_get : configured changed agents = 1,2,3,4,5,6,7,8,9,10
test_session_get : log file =
test_session_get : interfaces = {p4p1}
test_session_get : protocols =snmpv1,snmpv2c,snmpv2,snmpv3,MQTT,NETFLOW,SYSLOG
test_session_get : product = 29
test_session_get : net address = 192.9.200.14
test_session_get : net dev =
test_session_get : expiration = ok
test_session_get : licensing = "{MIMIC Simulator Universal v18.00
built Jul 19 2018 10:46:47
Copyright (c) 1997-2018 Gambit Communications, Inc.
Evaluation license, expires 8/26/2018
Running 64-bit
machine x86_64
OS Linux 3.14.27-100.fc19.x86_64
4 CPUs}"
...
```

[<< Previous Section](#) [Next Section >>](#)



MIMIC Go API Guide

Table of Contents

- [Requirements](#)
- [Class Reference](#)
- [Examples](#)

Requirements

The MIMIC Go API is an optional [Update Wizard](#) package, and requires the use of a recent version of the [Go language](#).

MIMIC has been tested with Go version 1.5.4 on Linux and version 1.13.5 on Windows.

1. Make sure you have Go installed, eg.

```
$go version
go version go1.5.4 linux/amd64
```

2. Set your `GOPATH` environment variable to the `golang` subdirectory under your MIMIC install area, eg. if MIMIC is installed under the default `/usr/local/mimic`, in a bash:

```
$export GOPATH=/usr/local/mimic/golang
or on Windows
```

```
$export GOPATH=c:/apps/mimic1920B5/golang
```

3. Install required packages:

```
$go get github.com/pborman/getopt/v2
```

Class Reference

The Go API is modelled after the [Java API](#). All classes and methods have the same name as in Java, except that Go programming constructs are used. Click [here](#) for the generated Java reference documentation.

Examples

- The `testapp.go` program is a small test program to exercise the MIMIC API. Run it as follows:

```
$go run testapp.go
2019/12/06 15:46:49 TestApp: starting...
2019/12/06 15:46:49 client dump
2019/12/06 15:46:49 number of sessions 1
2019/12/06 15:46:49 session dump
2019/12/06 15:46:49 host 127.0.0.1
2019/12/06 15:46:49 port 9797
2019/12/06 15:46:49 sock 0xc000010010
2019/12/06 15:46:49 INFO: correctly prevented session.Connect from leaking connections
2019/12/06 15:46:49 TestApp: config file = agent.cfg
2019/12/06 15:46:49 TestApp: max agents = 20000
2019/12/06 15:46:49 TestApp: last agent = 13900
2019/12/06 15:46:49 TestApp: version = 19.20
2019/12/06 15:46:49 TestApp: num clients = 1
...
```

- The `mimicd-threads` program is a small diagnostic program to display sorted MIMICD threads CPU usage via the MIMIC API. The source code is in [golang/src/mimicd-threads.go](#) under the MIMIC install area.



MIMIC OpenAPI Guide

Table of Contents

- [Requirements](#)
- [API Reference](#)
- [Examples](#)

Requirements

The MIMIC OpenAPI is an optional [Update Wizard](#) package, which implements a [OpenAPI](#) based REST API to control MIMIC.

In order to export the MIMIC OpenAPI, you have to run the `mimic-openapi-daemon` from the `linux` directory of the MIMIC installation as follows:

```
$ ./mimic-openapi-daemon
Thu Jan 23 14:30:20 EST 2020 - Only plain HTTP is working because tcl-TLS is not available in the system

Thu Jan 23 14:30:20 EST 2020 - Opened socket sock24ceb20 for port 8080
```

If you want to export the OpenAPI via HTTPS, then you also need to download the optional `TLSTCL` package with Update Wizard.

API Reference

See the definition on [SwaggerHub](#)

Examples

- The `TestOpenAPI.java` program is a small test program to exercise the MIMIC API.
- You can use any command-line HTTP client such as [curl](#). Eg. if the default `agent.cfg` lab configuration is loaded:

```
$ curl -u lab:lab123 -X GET http://127.0.0.1:8080/mimic/get/max
20000
$ curl -u lab:lab123 -X GET --insecure https://127.0.0.1:8443/mimic/get/max
20000
$ curl -u lab:lab123 -X GET http://127.0.0.1:8080/mimic/get/last
7
$ curl -u lab:lab123 -X GET http://127.0.0.1:8080/mimic/get/cfgfile
agent.cfg
$ curl -u lab:lab123 -X GET http://127.0.0.1:8080/mimic/agent/1/get/host
10.1.120.142
$ curl -u lab:lab123 -X GET http://127.0.0.1:8080/mimic/agent/7/get/mibs 2> /dev/null \
| python -mjson.tool
[
  {
    "device": "sim-10.48.0.1.random",
    "mib": "RFC1213-MIB",
    "scenario": "1"
  },
  {
    "device": "sim-10.48.0.1.random",
    "mib": "SNMPv2-MIB",
    "scenario": "1"
  },
  {
    "device": "sim-10.48.0.1.random",
    "mib": "IF-MIB",
    "scenario": "1"
  },
  {
    "device": "sim-10.48.0.1.random",
    "mib": "EtherLike-MIB",
    "scenario": "1"
  },
  // removed 285 lines of MIBs for brevity
  {
    "device": "sim-10.48.0.1.random",
    "mib": "SNMP-NOTIFICATION-MIB",
    "scenario": "1"
  }
]
```

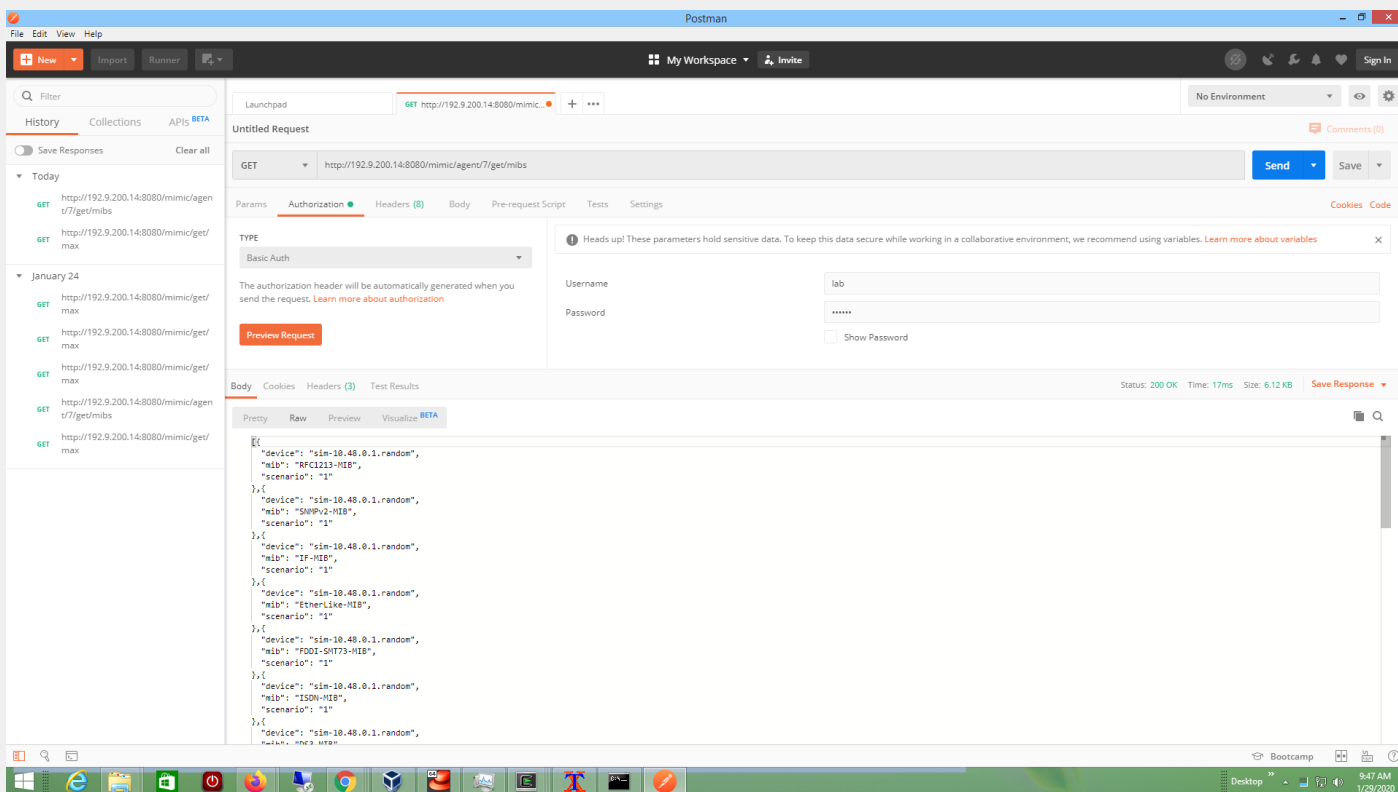


```

    },
    {
      "device": "sim-10.48.0.1.random",
      "mib": "SNMP-USER-BASED-SM-MIB",
      "scenario": "1"
    },
    {
      "device": "sim-10.48.0.1.random",
      "mib": "SNMP-VIEW-BASED-ACM-MIB",
      "scenario": "1"
    },
    {
      "device": "sim-10.48.0.1.random",
      "mib": "cisco/CISCOTRAP-MIB",
      "scenario": "1"
    },
    {
      "device": "sim-10.48.0.1.random",
      "mib": "cisco/OLD-CISCO-TCP-MIB",
      "scenario": "1"
    }
  ]
}

```

- The **Postman** API Client, can be used to do simple queries.





MIMIC Cable Modem Simulator

Table of Contents

- [Overview](#)
- [Implementation](#)
- [Installation](#)
- [MIBs](#)
- [Actions](#)
- [Key Exchange](#)



Overview

MIMIC Cable Modem Simulator is designed to simulate cable modem installations from an OSS standpoint which are [DOCSIS](#) (Data-over-Cable Service Interface Specification) compliant and optionally support [PacketCable](#) specifications from CableLabs.

A DOCSIS compliant cable modem is ready for data communications between an end-user and the service-provider. The DOCSIS standards define a set of standard protocols for startup initialization as well as for steady state monitoring of the cable modem. These standards consists of various IP based protocol components which are needed to interact with the Operations Support System (OSS) at the service-provider's end. The following components can be simulated using MIMIC to approximate a cable modem end device :

1. SNMP agent [see SP-OSSv1.1-I02-000714 : Section 2]: cable modem units are required to support version 1 and 3 of this widely popular network management protocol in accordance with RFC-1157 and RFC-2570ff. SNMP is used for variety of things like usage and performance monitoring, billing, fault detection and analysis, etc.
2. DHCP client [see SP-RFiv1.1-I05-000714 : Section 9.2.6, Appendix D]: cable modems are required to dynamically obtain their IP Address from the service-provider's OSS center in accordance with RFC-2131. MIMIC can also act as a DHCP relay.
3. TFTP client [see SP-RFiv1.1-I05-000714 : Section 9.2.8, Appendix C & D]: cable modems are required to dynamically download their configuration information from the service-provider's OSS center in accordance with RFC-1350 and RFC-2349.
4. ToD client [see SP-RFiv1.1-I05-000714 : Section 9.2.7]: cable modems are required to get the current Time of the day from the service-provider's OSS center in accordance with RFC-868.

Implementation

Depending on the user's requirements, there are two basic implementation strategies for creating CM and CMTS simulations with MIMIC. In the first case, the devices are running in a steady-state and need only to respond to SNMP stimulus from a manager or managers. This is the simpler of the cases; a recording of a live device can be used as is. The only configuration which should be necessary will consist of setting up the SNMP access if SNMPv3 is in use. In the case of some DOCSIS 1.0 devices, even this will not be necessary, assuming the default community string access is acceptable.

In the second case, the CM startup sequence is of interest. This will include receiving an IP address via DHCP, downloading one or more configuration files via TFTP, and syncing the device time through use of the ToD protocol, and sending traps or informs to signal successful device initialization. Additionally, implementation-dependent factors may require SNMPv3 initialization via Diffie-Hellman (DH) key exchange, Kerberos ticketing, or other steps. These considerations are examined in more detail below.

In either case, the first required step is to create a device simulation. The methods used to do this are outlined in the [MIMIC Quick Start Guide](#). The basic choices are either to record a live device that is attached to the network using the [MIMIC Recorder](#), to create a simulation from an SNMP walkfile using [Simulation->Record File...](#), or to build the simulation using the [MIMIC Simulation Wizard](#). The Simulation Wizard allows the user to specify which MIBs the device will contain, optionally to load some object values by including a walkfile, and either to enter the rest manually or to use default values. Users can alternatively use a preconfigured MIMIC simulation provided by Gambit or a colleague who uses MIMIC. A sample generic DOCSIS 1.0 CM simulation with the associated scripts is available upon request.

After a simulation has been created, it may be added to the simulation in the GUI as an agent instance using the MIMICView [Edit->Add->Agent](#) menu option. For the first case, the agent is now ready for use. If SNMPv3 is to be utilized, this can be specified during the add agent dialog by checking the associated checkbox on the tool's Advanced tab. Other desired configuration parameters for SNMPv3 may include edits to the configuration files v3usm.conf and v3vacm.conf to enter desired access modifications. These files are contained in the config directory of the MIMIC installation area. The comments in the file give usage information.

This document provides an overview of the process of creating and running CableModem simulations in MIMIC. Simulations of CMTSs are similar, but do not require the same initialization as CM's do to be authentic in detail. Many of the same device MIBs are used. Consideration of many detailed subjects, such as providing unique hardware addresses for use in the DHCP request have been omitted in the interest of conciseness. Customers with support agreements are invited to direct further questions to support@gambitcomm.com.

Installation

A specific MIMIC licensing option, the Cable Modem license, allows access to the protocol modules required to simulate a DOCSIS-compliant or PacketCable-compliant cable modem or CMTS. They are currently

- [DHCP](#)
- [TFTP](#)
- [ToD](#)

Protocol module installation is required prior to first-time use of the cable modem features. This consists in copying the dynamic library from the bin/dynamic/optional directory to the bin/dynamic directory. Please consult the Installation section of each of the Protocol Module Guides for details.

The protocol-specific modules for DHCP, TOD, and TFTP can be enabled for an agent simulation via check boxes found on the Advanced tab of the [Edit ->Configure](#) dialog.

For the protocol-specific modules to work, appropriately configured servers (DHCP server, TFTP server, ToD server) to support these functions are assumed. For example, the DHCP offers returned from the DHCP server are required to contain properly formatted information required for device initialization.

MIBs

MIMIC ships with the DOCSIS RFC mibs pre-compiled. Other MIBs, for example the MCNS and BPIPLUS mibs, have not yet attained RFC status, and are not shipped. These can be downloaded from the [IETF draft standards page](#) and compiled as needed.

Actions

For the simulation to initialize properly according to DOCSIS specifications, an agent startup action script must be created and copied into the simulation directory for the agent. This script should be written to accomplish tasks related to device initialization which are required at startup.

One task will be to parse the DHCP offer for TFTP server and initialization file.

This data will then be used to launch a TFTP session. Other implementation-specific requirements can also be addressed.

Additionally, a TFTP completion script is required parse the configuration file downloaded, set the appropriate mib values, and to send configuration informs/traps once the simulation has completed initialization. This can be copied into the scripts directly and launched from the start.mtcl script.

Key Exchange

MIMIC implements a set of commands for Diffie-Helman key generation and PBKDF2 to support Diffie-Helman Key Ignition and DH Key Change.

This is made available in the form of MIMIC commands to be used in action scripts, so that the TFTP completion script can generate the authentication key and privacy key.

Diffie-Helman key generation commands

This function generates a random private key and computes the public based on that and returns both the private and the public keys.

```
mimic crypt DH_generate_keys OAKLEY1 | OAKLEY2
or
mimic crypt DH_generate_keys prime generator
Example:
```

```
set key [mimic crypt DH_generate_keys OAKLEY2]
set priv_key [lindex $key 0]
set pub_key [lindex $key 1]
```

Diffie-Helman shared key compute commands

This function takes the agent's private key, manager's public key and either OAKLEY1 or OAKLEY2 or DH Prime and DH generator as input and returns back the shared key.

```
mimic crypt DH_compute_shared_key local-private-key remote-public-key OAKLEY1 | OAKLEY2
or
mimic crypt DH_compute_shared_key local-private-key remote-public-key prime generator
Example:
```

```
set shared_key [mimic crypt DH_compute_shared_key [list $priv_key] [list $remote_public_key] OAKLEY2]
```

PBKDF2 key generation command

This function takes shared key generated by the previous command as input and generates both the privacy and authentication key.

```
mimic crypt PBKDF2_generate_keys shared-key
Example:
```

```
set pass [mimic crypt PBKDF2_generate_keys [list $shared_key]]
set privacy [lindex pass 0]
set auth_key [lindex pass 1]
```

USM key generation command

This command generates the private key and the public key for the agent then computes the shared based on the managers public key passed in as parameter. Then it uses the shared key to generate both the privacy and authentication key. This command takes manager public key and either OAKLEY1 or OAKLEY2 or dh prime and dh generator as input. The output are privacy, authentication, private and public keys.

```
mimic crypt USM_generate_keys remote-public-key OAKLEY1 | OAKLEY2
or
mimic crypt USM_generate_keys remote-public-key prime generator
Example:

set pass [mimic crypt usm_generate_keys [list $remote_public_key] OAKLEY2]
set privacy [lindex pass 0]
set auth_key [lindex pass 1]
set priv_key [lindex pass 2]
set pub_key [lindex pass 3]
```

Licensing information

This product includes software developed by the [OpenSSL Project](#) for use in the OpenSSL Toolkit with the following [copyrights](#).

[<< Previous Section](#) [Next Section >>](#)



MIMIC DHCP Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using DHCP from MIMICView](#)
- [Using DHCP from MIMICShell](#)
- [Compatibility](#)

Overview

The MIMIC Dynamic Host Configuration Protocol (DHCP) Protocol Module allows assigning addresses to agents dynamically from a DHCP server according to the DHCPv4 protocol defined in [RFC 2131](#), at the startup time of an agent instance.

Installation

DHCP client support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 10.00, you can use the [Protocol Wizard](#) to install the DHCP module. If you prefer to enable DHCP by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the DHCP library (`dhcp.dll` on Windows, `dhcp.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the DHCP module was properly loaded :
INFO - DHCP : Loaded protocol from < path-to-DLL >
INFO - DHCP (CableModem) v7.00 : Individual license #2345

Once DHCP is loaded, any agent instance configured to support the DHCP protocol will request an IP address from the DHCP server at agent start time.

Using DHCP from MIMICView

If the DHCP module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display DHCP as an additional checkbox in the `Advanced` pane along with the `SNMP` protocols. On selecting the checkbox a new `DHCP` pane will appear.

This DHCP configuration pane lets the user configure the `Hardware Address` used by the DHCP module while negotiating the IP address lease. In case the user leaves this field to be blank, the module auto generates unique hardware addresses based on the agent's id. (e.g. 00:00:00:00:00:01, 00:00:00:00:00:02, ...)

In order to send additional options in the DHCP DISCOVER and REQUEST messages, you can use the `Additional Options` field. This is a list of options of the form

```
OPT1,LEN1,VALUE1;OPT2,LEN2,VALUE2;.....;OPTN,LENN,VALUEN[;]
```

where an `OPT` value can be 1 to 254 and `LEN` can be 1 to 255. `VALUE` can be either an ASCII string or hexadecimal string (eg. `\xFF FF` or `\xFF:FF` or `\xFF-FF`)

An action script can be configured to execute when an agent gets the IP address from the DHCP server (`BOUND` state action script). This is useful to react to lease and re-lease of IP addresses. The agent may want to start a TFTP transfer once an address is leased, or if the lease is renewed.

The script will be executed with the following global variables defined:

- **gCurrentAgent**
This variable holds the agent that initiated the DHCP exchange.
- **gCurrentState**
This variable holds the current state of the DHCP exchange (for now always will be `BOUND`).
- **gPreviousState**
This variable holds the previous state of the DHCP exchange (one of `REQUESTING`, `RENEWING` or `REBINDING`).
- **gParams**
Provides the DHCP parameters, same as those returned from the `mimic agent protocol msg DHCP params` command.

NOTE: During agent startup, the bound action script might be executed even before the agent transitioned to running state. So any command dependent on the agent's state could fail.

In the case of DOCSIS Cable Modem simulations, it is assumed that the DHCP server is configured to provide values for `siaddr` and `file` DHCP options, so that the agent will be able to retrieve follow-on configuration parameters to be used for the TFTP configuration file download. Retrieval of these settings is accomplished through use of a MIMIC agent startup action script `start.mtcl`, which is placed in the `data/sim/SIMULATION-NAME` directory corresponding to the simulated device in use. This file, created by the user, will run when the agent is started. It should contain lines resembling the following:

```
#
# parse the values received in the DHCP offer
#
set dhcp_param_list [mimic agent protocol msg DHCP params]
puts "Configured via DHCP at IP [mimic agent get host]"

# load server parameter
set x [ lsearch $dhcp_param_list siaddr* ]
puts " server_address = $x"
set server_listvar [ lindex $dhcp_param_list $x ]
set server [ lindex $server_listvar 1 ]

#load file parameter
set y [ lsearch $dhcp_param_list file* ]
puts " file_name = $y "
set file_listvar [ lindex $dhcp_param_list $y ]
set file [ lindex $file_listvar 1 ]
This will cause the agent to parse the received DHCP offer and load variables with the parameters required for the TFTP portion of the configuration cycle. Another section below lists further commands associated with TFTP startup, which will also be included in the start.mtcl agent startup action script file.
```

When the agent is started, successful completion of the DHCP portion will be indicated in the log by entries resembling this:

```
INFO 08/09.10:19:32 - agent 3 configured at 192.9.200.211,161.
INFO 08/09.10:19:33 - agent 3 loading
INFO 08/09.10:19:38 - agent 3 loaded device
Configured via DHCP at IP 192.9.200.211
server_address = 192.9.200.13
file_name = modem.bin
```

Using DHCP from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to use the DHCP functionality. Here is a synopsis:

- **mimic protocol msg DHCP get args**

This command lets the user gather the list of arguments required and their particulars. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg DHCP get args
{{hwaddr} {Hardware Address} {macaddress} {} {optional} {}}
{{server} {Server Address} {string} {} {optional} {}}
{{relay} {Relay Address} {string} {} {optional} {}}
{{classid} {Class Identifier} {string} {} {optional} {}}
{{clientid} {Client Identifier} {string} {} {optional} {}}
{{add_options} {Additional Options} {string} {} {optional} {}}
{{script} {Action Script} {file} {scripts} {*.mtcl} {MIMIC Tcl scripts}
{edit yes} {new yes}} {*.tcl} {Tcl script files} {edit yes} {new yes}}
{*.so} {C/C++ DLL files} {edit no} {new no}} - both {optional} {}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent set protocol snmpv1,DHCP
```

- **mimic agent protocol msg DHCP get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg DHCP get config
{hwaddr=} {classid=} {add_options=} {script=}
```

- **mimic agent protocol msg DHCP set config [config]**

This command lets the user change the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg DHCP set config {hwaddr=aa:bb:cc:dd:ee:ff}
{add_options=67,4,boot} {script=dhcp_action.mtcl} "{classid=some string}"
mimicsh> mimic agent protocol msg DHCP get config
{hwaddr=aa:bb:cc:dd:ee:ff} {classid=some string} {add_options=67,4,boot} {script=dhcp_action.mtcl}
```

- **mimic agent protocol msg DHCP get trace**

mimic agent protocol msg DHCP set trace [0 or 1]

This command lets the user change the DHCP tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 9
```

```
mimicsh> mimic agent protocol msg DHCP get trace
0
mimicsh> mimic agent protocol msg DHCP set trace 1

mimicsh> mimic agent protocol msg DHCP get trace
1
and the log would show:
```

```
INFO - agent 9 trace enabled for DHCP
INFO - agent 9, sent DHCP-DISCOVER msg (broadcast)
INFO - agent 9, received DHCP OFFER from 192.9.200.1.
INFO - agent 9, sent DHCP-REQUEST msg (broadcast)
INFO - agent 9, received DHCP ACK from 192.9.200.1.
INFO - agent 9, REQUESTING: obtained lease for address=192.9.200.235. lease_time=43200, renew_time=21600, rebind_time=37800
INFO - agent 9, BOUND: calling agent_start_pre_cb()
INFO - agent 9 configured at 192.9.200.235,161.
INFO - agent 9 loading
INFO - agent 9 loaded device
```

- **mimic protocol msg DHCP get stats_hdr**
mimic agent protocol msg DHCP get statistics
Returns DHCP statistics information:

- a list of statistic headers, and
- current statistics values for the specified client.

In order, the statistic values are:

- Number of DHCPDISCOVER messages sent
- Number of DHCPPOFFER messages received
- Number of DHCPREQUEST messages sent
- Number of DHCPACK messages received
- Number of DHCPNAK messages received
- Number of DHCPDECLINE messages received
- Number of DHCPRELEASE messages sent
- Number of DHCPINFORM messages sent
- Number of timeouts

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg DHCP get stats_hdr
{{discover} {DHCP-DISCOVER}} {{offer} {DHCP-OFFER}} {{request} {DHCP-REQUEST}}
{{ack} {DHCP-ACK}} {{nak} {DHCP-NAK}} {{decline} {DHCP-DECLINE}}
{{release} {DHCP-RELEASE}} {{inform} {DHCP-INFORM}} {{timeout} {DHCP-TIMEOUT}}

mimicsh> mimic agent protocol msg DHCP get statistics
1 1 1 1 0 0 1 0 0
```

- **mimic agent protocol msg DHCP params**
This command allows the user to retrieve parameters configured by the server in its DHCP-OFFER message. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg DHCP params
{htype 1} {hlen 6} {ciaddr 0.0.0.0} {yiaddr 192.9.200.201} {siaddr 192.9.200.13}
{giaddr 0.0.0.0} {chaddr 00 00 00 00 00 01} {sname } {file myconfig.dat}
{options {53 1 {5}} {54 4 {192 9 200 39}} {1 4 {255 255 255 0}}
{3 4 {192 9 200 22}} {28 4 {192 9 200 255}} {51 4 {0 0 0 10}}}
```

Compatibility

The DHCP protocol module has been tested for compatibility with a variety of DHCP servers. It works with the DHCP server included in Solaris 8, Windows Server 2003, RedHat Linux 7.3 and newer, and with the version 3.0 server in the [Internet Software Consortium](#) DHCP distribution.

The version 2.0 ISC DHCP server has bugs which prevent it from working with MIMIC. This server is included in RedHat Linux 6.x



MIMIC TFTP Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using TFTP from MIMICView](#)
- [Using TFTP from MIMICShell](#)
- [Compatibility](#)

Overview

The MIMIC TFTP Protocol Module is an optional facility that enables client TFTP file downloads (sessions) either through MIMICView at the startup time of an agent instance, or through MIMICshell commands at both startup and running state of an agent. Each of these downloads (sessions) are identified by a unique identifier called session-id.

MIMIC TFTP gives you control whether you want to save the downloaded data in a disk file, process it through an action script, or discard it upon transfer.

Installation

TFTP client support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 10.00, you can use the [Protocol Wizard](#) to install the TFTP module. If you prefer to enable TFTP by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the TFTP library (tftp.dll on Windows, tftp.so on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the TFTP module was properly loaded :
INFO - TFTP : Loaded protocol from < path-to-DLL >
INFO - TFTP (IOS) v7.00 : Individual license #2345

Once TFTP is loaded, any agent instance configured to support the TFTP protocol will be able to do TFTP transfers from a TFTP server.

Using TFTP from MIMICView

If the TFTP module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display TFTP as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new TFTP pane will appear.

This TFTP configuration pane lets the user configure the parameters for a TFTP session:

- **Server**
This mandatory parameter specifies the TFTP Server IP address, either as "dot-value" notation (e.g., 192.9.200.1), or as a hostname (e.g., gambit), or fully qualified domainname (e.g., gambit.gambitcomm.com) provided that they can be resolved to an address (via /etc/hosts, Yellow Pages or DNS).
- **Port**
The TFTP client talks to the standard TFTP port 69 on the TFTP server by default. If the server is configured to use a non-standard port, you can specify it in this option.
- **Client**
This optional parameter specifies the local IP address, either as "dot-value" notation (e.g., 192.9.200.1), or as a hostname (e.g., gambit), or fully qualified domainname (e.g., gambit.gambitcomm.com) provided that they can be resolved to an address (via /etc/hosts, Yellow Pages or DNS). This address can be a previously configured IP alias, or a temporary address for the duration of the transfer. By default, if it is left empty the main address of the agent is used.
- **Source File**
This mandatory parameter specifies the file to be retrieved from the server. This is a server-dependent file path, eg. on Unix a relative path off /tftpboot.
- **Destination File**
This optional parameter specifies a file path to save the downloaded data in. The file path is relative to the directory specified in the environment variable MIMIC_TMPDIR, or /tmp/. If the action script parameters is set below, but the destination file is not set, then its name will be auto generated. MIMIC TFTP will not allow more than one session to share the same destination filename (to prevent corruption of data).
- **Action Script**
If this optional parameter is set, then upon completion of the data transfer, either successfully or unsuccessfully, an action script will be called with the following global variables as input:

- gCurrentAgent
- gSessionId
- gStatus = [success | error | timeout | interrupted]
- gModeOfOperation = [read | write]
- gServer
- gServerPort
- gSrcFile
- gDstFile
- gModeOfTransfer = [ascii | bin]
- gTimeout
- gRetries
- gCache
- gBuffer = data or error message
- gBytes
- gErrorCode
- gTransferTime

If the action script was not set then the data will be discarded.

- Transfer Mode
By default, the transfer mode is "ascii". Else, if you want binary data to be transferred, use the mode "raw". If you want data to be converted to a space-separated hexadecimal representation, use "binary".
- Timeout
This optional parameter specifies the time to wait for a response between successive retransmits in seconds. The default is 5 seconds.
- Retries
This optional parameter specifies the number of retransmits. The default is 5.
- Trace
This parameter enables tracing.
- Blocks to cache (0 - 10)
If this parameter is set to non zero blocks then the data will be buffered in memory. If the data grows beyond a chosen limit then the buffered data will be moved to the destination file. If no destination file name is set an auto generated temporary file name will be used as the destination file name. The subsequent write will be done on this file.

If the mandatory parameters are supplied, the agent will automatically initiate a TFTP transfer upon starting.

In the case of a DOCSIS CM simulation, when DHCP has completed, the TFTP protocol will take over. To configure TFTP parameters for the agent using the information received from the DHCP offer and loaded into variables above, insert the following into the start.mtcl agent startup action script:

```
#
# configure agent for tftp, set server, filename params
set session_id [mimic agent protocol msg TFTP session read $server $file]
puts stderr " session_id = $session_id "
mimic agent protocol msg TFTP $session_id set dstfile myfile_$session_id
mimic agent protocol msg TFTP $session_id set script tftp-rfis-appndx-d-2.1.mtcl
mimic agent protocol msg TFTP $session_id set cache 0
# mimic agent protocol msg TFTP $session_id set mode binary
mimic agent protocol msg TFTP $session_id start
# mimic timer script add dhcp_timer.mtcl -1 [mimic agent assigned]
The script referred to above, tftp-rfis-appndx-d-2.1.mtcl, is a MIMIC TFTP completion script which will run following completion of the file download of the device configuration file. It's purpose is to parse the received configuration file and set the simulated device's operating parameters accordingly.
```

The tftp-rfis-appndx-d-2.1.mtcl script is available by request from Gambit, either separately or as part of the DOCSIS sample simulation. Many, but not all defined DOCSIS config file TLV's (configuration settings in Type-Length-Value format) are included in this script, along with procedures for parsing different encoding types. This file is user-editable if TLV's not defined in the script are required.

Finally, the "timer script add" statement which is commented out at the end of the start.mtcl fragment above refers to a script which would be used to launch traps or informs following successful completion of the initialization cycle.

Using TFTP from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to use the TFTP functionality. Here is a synopsis:

- **mimic protocol msg TFTP get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg TFTP get args
{{server} {Server} {string} {} {optional} {}}
{{port} {Port} {integer} {} {optional} {69}}
{{client} {Client} {string} {} {optional} {}}
{{srcfile} {Source File} {string} {} {optional} {}}
{{dstfile} {Destination File} {string} {} {optional} {}}
{{script} {Action Script} {file} {scripts} {{*.mtcl {MIMIC Tcl scripts}
  {edit yes} {new yes}} {{*.tcl {Tcl script files} {edit yes} {new yes}}
  {{*.so {C/C++ DLL files} {edit no} {new no}}} - both} {optional} {}}
{{mode} {Transfer Mode} {string} {} {optional} {ascii}}
{{timeout} {Timeout (sec)} {integer} {} {optional} {5}}
{{retries} {Retries} {integer} {} {optional} {5}}
{{trace} {Trace} {string} {} {optional} {off}}
{{cache} {Blocks to cache (0 - 10)} {integer} {} {optional} {4}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent set protocol snmpv1,TFTP
mimicsh> mimic agent get protocol
snmpv1,TFTP
```

- **mimic agent protocol msg TFTP get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg TFTP get config
{server=dmb.gambitcomm.com} {port=69} {client=} {srcfile=five-m} {dstfile=uwe123}
{script=} {mode=ascii} {timeout=5} {retries=5} {trace=off} {cache=4}
```

- **mimic agent protocol msg TFTP set config [config]**

This command lets the user change the current argument settings of all TFTP sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg TFTP set config {server=genesis.gambitcomm.com}

mimicsh> mimic agent protocol msg TFTP get config
{server=genesis.gambitcomm.com} {port=69} {client=} {srcfile=five-m} {dstfile=uwe123}
{script=} {mode=ascii} {timeout=5} {retries=5} {trace=off} {cache=4}
```

- **mimic agent protocol msg TFTP get trace**

- **mimic agent protocol msg TFTP set trace [0 or 1]**

This command lets the user change the TFTP tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 1

mimicsh> mimic agent protocol msg TFTP get trace
0
mimicsh> mimic agent protocol msg TFTP set trace 1

mimicsh> mimic agent protocol msg TFTP get trace
1
```

and the log would show:

```
INFO - agent 9 trace enabled for TFTP
INFO - agent 9 configured at 10.0.0.9,161.
INFO - agent 9 loading
INFO - agent 9 loaded device
INFO - TFTP: starting "read" session 5 for agent 9
INFO - TFTP [AGT=9,SES=5]: sent RRQ <file=foo.10, mode=netascii>
INFO - TFTP [AGT=9,SES=5]: received ERROR <code=1, msg= File not found
ERROR - TFTP [AGT=9,SES=5]: 7911790
- Error code 101: File not found
```

- **mimic protocol msg TFTP get stats_hdr**

- **mimic agent protocol msg TFTP get statistics**

Returns TFTP statistics information:

- a list of statistic headers, and
- current statistics values for the specified client.

In order, the statistic values are:

- Total number of TFTP packets sent.
- Total number of TFTP packets received.
- Total number of TFTP packets discarded.
- Total number of retransmits.
- Total number of timeouts.
- Total number of interrupted transfers.
- Total number of errors.
- Total number of completed transfers.

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg TFTP get stats_hdr
{{pktSnt} {PktsSent}} {{pktRcvd} {PktsRcvd}} {{pktDisc} {PktsDiscarded}}
{{rexmt} {Retransmits}} {{timeout} {TimeOuts}} {{inter} {Interrupted}}
{{errRcvd} {Errors}} {{cmptd} {Completed}}
```

```
mimicsh> mimic agent protocol msg TFTP get statistics
1 0 0 0 0 1 0
```

- **mimic agent protocol msg TFTP session read server srcfile**
mimic agent protocol msg TFTP session write server srcfile

TFTP client sessions are created by the `session` command by passing the mode of operation (read or write), the server address (ip address or host name) and the source file name as arguments. This command returns a non zero session-id if the session got created successfully. A sample exchange for this command would be:

```
mimicsh> set sessionid [mimic agent protocol msg TFTP session read server srcfile]
5
```

or

```
mimicsh> set sessionid [mimic agent protocol msg TFTP session write server srcfile]
5
```

In write mode the source file (srcfile) will be read from the simulation directory for the agent (ie. data/sim/[sim-name]). Many TFTP servers allow TFTP put (write) operation only if a file with the same name exists. If you want to write to a different file on the server, then use

```
mimic agent protocol msg TFTP $sessionid set dstfile filename
```

- **mimic agent protocol msg TFTP session-id get parameter**

The user can get the attribute values of the session at any time by specifying a valid session id and parameter name. The get command will fail with an error message "Invalid TFTP session id", if the session id is invalid. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg TFTP 5 get server
genesis.gambitcomm.com
mimicsh> mimic agent protocol msg TFTP 5 get port
69
```

- **mimic agent protocol msg TFTP session-id set parameter**

The user can set the attribute values of each session by specifying a valid session id, attribute name and the value. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg TFTP 5 set port 79

mimicsh> mimic agent protocol msg TFTP 5 get port
79
```

The users can set values to its attributes before the start command is issued. Any set command after start fails with an error message "TFTP session started".

If the session id is invalid then the start command fails with an error message "Invalid TFTP session id".

- **mimic agent protocol msg TFTP session-id start**

The TFTP data transfer can be started by start command, provided all the mandatory attributes are set. A sample exchange for this command would be:

```
mimic agent protocol msg TFTP 5 start
```

A start command would fail for the following reasons i) if the session is already started. ii) if the session id is not valid.

Note: If no destination file is specified the TFTP client handles it differently for read and write. In case of write, the destination file name will be substituted by source file name. But in case of read the client would auto-generate a temporary file name and use it as destination file name.

- **mimic agent protocol msg TFTP session-id status**

Basically this command is to query the status of client session. This command return a space separated string of type:value list. This command will fail with an error message "Invalid session id" if the session id is invalid. A sample exchange for this command would be:

```
mimic agent protocol msg TFTP 5 status
state:running bytes:1628672 in:3.3 seconds
```

- **mimic agent protocol msg TFTP session-id stop**

The TFTP data transfer (after start) can be terminated by stop command with a valid session-id. This command will also start the action script by setting the global variable `gStatus = interrupted`. If the stop command is called before start, then it will just close the TFTP session. A sample exchange for this

command would be:

```
mimic agent protocol msg TFTP 5 stop
```

Compatibility

This section details interoperability with some widely available TFTP servers.

[Tftpd32](#) [Fedora Linux](#) [WinAgents](#)

Tftpd32

Download Url	Version	Size
http://tftpd32.jounin.net/	2.80	173 Kb

Installation and usage

1. Extract the tftpd32.280.zip file.
2. Run tftpd32.exe from the Extracted Location.
3. Click settings, Change the base directory to a location of your choice.
4. Click Ok, to confirm changes.
5. Click "TFTP server" tab to watch transaction logs.

Uninstallation

1. Execute the uninst.exe file.

Fedora Linux

TFTP daemon comes with Fedora Linux distribution.

Download Url	Version	Size
http://fedora.redhat.com/	3	(depending on installation)

Installation and usage

1. To start TFTP daemon with a system boot, set the "disable" field in configuration file /etc/xinetd.d/tftp to "no".
2. The default directory for saved files is /tftpboot .
3. To allow saving a file to the server, by default, the file name must already exist. And its permission must allow all users to write.
4. To allow downloading a file from the server, by default, the file must allow all users to read.

Uninstallation

1. Your choice of uninstallation procedure in linux.

WinAgents

Download Url	Version	Size
http://www.winagents.com/	3.0	1.66MB

Installation and usage

1. Run the tftsetup.exe file.
2. Use the "TFTP Service Configuration" Start menu item to choose a location for saving files.
3. Click Ok, to confirm changes.
4. Use the "Service Manager" Start menu item to start or stop TFTP service.

Uninstallation

1. Use the "Uninstall TFTP Service" Start menu item.

[<< Previous Section](#) [Next Section >>](#)



MIMIC Time-Of-Day Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using TOD from MIMICView](#)
- [Using TOD from MIMICShell](#)

Overview

The MIMIC Time-Of-Day (TOD) Protocol Module is an optional facility that enables retrieving the current time of day through the Time Service ([RFC 868](#)) from a Time Server, either through MIMICView at the startup time of an agent instance, or through MIMICShell commands at both startup and running state of an agent.

Installation

Time-Of-Day (TOD) client support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 10.00, you can use the [Protocol Wizard](#) to install the TOD module as part of the [Cable Modem Simulator](#). If you prefer to enable TOD by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the TOD library (tod.dll on Windows, tod.so on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the TOD module was properly loaded :
INFO - TOD : Loaded protocol from < path-to-DLL >
INFO - TOD (CableModem) v7.00 : Individual license #2345

Once TOD is loaded, any agent instance configured to support the TOD protocol will be able to retrieve the current time of day from the Time Server.

Using TOD from MIMICView

If the TOD module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display TOD as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new TOD pane will appear.

This TOD configuration pane lets the user configure the parameters for a TOD retrieval:

- **Server**
This mandatory parameter specifies the Time Server IP address, either as "dot-value" notation (e.g., 192.9.200.1), or as a hostname (e.g., gambit), or fully qualified domainname (e.g., gambit.gambitcomm.com) provided that they can be resolved to an address (via /etc/hosts, Yellow Pages or DNS).
- **Port**
The TOD client talks to the standard TOD port 37 on the Time Server by default. If the server is configured to use a non-standard port, you can specify it in this option.
- **Action Script**
If this optional parameter is set, then upon completion of the time of day request, either successfully or unsuccessfully, an action script will be called with the following global variables as input:
 - gCurrentAgent
 - gStatus = [success | error | timeout]
 - gServer
 - gServerPort
 - gTimeOfDay - can be parsed using TCL clock command
 - gBuffer - date or error message
 - gTimeout
 - gRetries

- gErrorCode
- Timeout
This optional parameter specifies the time to wait for a response between successive retransmits in seconds. The default is 5 seconds.
- Retries
This optional parameter specifies the number of retransmits. The default is 5.

If the mandatory parameters are supplied, the agent will automatically initiate a TOD request upon starting.

Using TOD from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to use the TOD functionality. Here is a synopsis:

- **mimic protocol msg TOD get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg TOD get args
{{server} {Server} {string} {} {optional} {}}
{{port} {Port} {integer} {} {optional} {37}}
{{script} {Action Script} {file} {scripts} {{*.mtcl {MIMIC Tcl scripts}
{edit yes} {new yes}} {*.tcl {Tcl script files} {edit yes} {new yes}}
{*.so {C/C++ DLL files} {edit no} {new no}}} - both} {optional} {}}
{{timeout} {Timeout (sec)} {integer} {} {optional} {5}}
{{retries} {Retries} {integer} {} {optional} {5}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent set protocol snmpv1,TOD
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,TOD
```

- **mimic agent protocol msg TOD get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg TOD get config
{server=} {port=37} {script=} {timeout=5} {retries=5}
```

- **mimic agent protocol msg TOD set config config**

This command lets the user change the current argument settings of all TOD sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg TOD set config {server=genesis.gambitcomm.com}

mimicsh> mimic agent protocol msg TOD get config
{server=genesis.gambitcomm.com} {port=37} {script=} {timeout=5} {retries=5}
```

- **mimic agent protocol msg TOD get trace**

- **mimic agent protocol msg TOD set trace [0 or 1]**

This command lets the user change the TOD tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 9

mimicsh> mimic agent protocol msg TOD get trace
0
mimicsh> mimic agent protocol msg TOD set trace 1

mimicsh> mimic agent protocol msg TOD get trace
1
```

and the log would show:

```
INFO 10/01.11:00:12 - agent 9 trace enabled for TOD
INFO 10/01.11:00:20 - agent 9 configured at 10.0.0.9,161.
INFO 10/01.11:00:20 - agent 9 loading
INFO 10/01.11:00:21 - agent 9 loaded device
INFO 10/01.11:00:21 - TOD [AGT=9]: sending time request
INFO 10/01.11:00:21 - TOD [AGT=9]: time recieved
```

- **mimic protocol msg TOD get stats_hdr**

- **mimic agent protocol msg TOD get statistics**

Returns TOD statistics information:

- a list of statistic headers, and
- current statistics values for the specified server.

In order, the statistic values are:

- Total number of TOD packets sent.
- Total number of TOD packets received.

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg TOD get stats_hdr  
{pktSnt} {PktsSent} {{pktRcvd} {PktsRcvd}}
```

```
mimicsh> mimic agent protocol msg TOD get statistics  
1 1
```

- **mimic agent protocol msg TOD gettime server `server-address` [`port port`] [`script script-name`] [`timeout seconds` [`retries`] `times`]**
This command retrieves the current time of day from the specified server (mandatory) with the specified attributes (optional).

[<< Previous Section](#) [Next Section >>](#)



MIMIC IOS Simulator Guide

Table of Contents

- [Overview](#)
- [Implementation](#)
- [Installation](#)
- [IOS Explorer](#)
- [IOS Recorder](#)

Overview

Cisco's Network Management System (NMS) software such as [Cisco DSL Manager](#) (CDM), [Service Connection Manager](#) (SCM) and [Cisco Network Order Manager](#) (CNOM) use SNMP and/or the IOS (Internetwork Operating System) command line interface (CLI) to communicate with and manage Cisco devices. MIMIC supports both SNMP and IOS CLI simulation to enable a realistic simulation for any management application that uses the IOS CLI.

Implementation

The easiest way of implementing a new IOS simulation is to use the [CLI Wizard](#) to record a session between a management application and a device running IOS. The resulting "basic" simulation will give verbatim responses to requests that were captured in the recording. For example, the `show clock` would return the same response regardless of the time of day.

For more advanced simulations, with dynamic responses, you need to write [Telnet rules](#) with Tcl scripts. The [MIMIC Virtual Lab](#) implements such advanced simulations. They can be loaded into MIMIC with the [Update Wizard](#).

Installation

A specific MIMIC licensing option, the IOS license, allows access to the protocol modules required to simulate a Cisco IOS entity. They are currently

- [Telnet](#)
- [TFTP](#)
- [SYSLOG](#)
- [SSH](#)

Protocol module installation is required prior to first-time use of the IOS features. This consists in copying the dynamic library from the bin/dynamic/optional directory to the bin/dynamic directory. Please consult the Installation section of each of the Protocol Module Guides for details.

The protocol-specific modules for Telnet and/or TFTP can be enabled for an agent simulation via check boxes found on the Advanced tab of the [Edit ->Configure](#) dialog.

IOS Explorer

Overview

IOS Explorer (`iosdisc`) is a stand-alone utility of the [CLI Wizard](#) functionality to automatically discover IOS commands supported by Cisco devices. This utility works in conjunction with [IOS Recorder](#) to create an IOS simulation. Instead of requiring a third party NMS application to issue IOS commands, `iosdisc` discovers the IOS commands on the device dynamically, which are recorded by the IOS Recorder.

The protocol capture needs to be launched prior to initiating the discovery process to capture maximum traffic, as documented [below](#).

Auto discover commands

The user has the ability to either completely discover the device by specifying no commands to be included and no commands to be excluded. In the first dialog you can restrict the discovery by specifying the commands to be included and commands to be excluded. Appropriately only included commands will be discovered and excluded commands will be ignored.

Issue discovered commands

Once the IOS commands are discovered, this dialog allows the commands to be issued after editing them by providing required arguments. The checklist shows all the commands with a checkbox indicating the command to be issued. Commands in red indicate they were excluded from the discovery process. The argument of the command appear as a leaf in the tree. When selected it allows the user to add the arguments through the graphical interface, which becomes the issued command. All the checked commands will be issued to the device when you click on the `Finish` button.

IOS Explorer can be invoked with following command line options:

• **--address ip-address**

mandatory command line argument to specify the device supporting IOS commands and a telnet session.

• **--out filename**

mandatory command line argument to specify the output file where the rules are to be stored. This file will be written under the `scripts/telnet` directory.

• **--password password**

mandatory command line argument to specify the password to be used by the user to login to the device.

• **[--port port-number]**

optional argument if the IOS device uses a non standard port.

• **[--username user-name]**

optional command line argument to specify the user initiating a telnet session to the device.

• **[--exclude cmd-list]**

optional argument to specify the commands to be excluded from the discovery process.

• **[--include cmd-list]**

optional argument to indicate the IOS commands to be discovered if the user chooses partial discovery of the device.

• **[--loginprompt prompt]**

[--passwordprompt prompt]

optional arguments to indicate the login and password prompt patterns. IOS Explorer uses patterns like the login prompt ("Username: ") and password prompt ("Password: ") to construct the connection welcome message.

• **[--timeout time]**

optional argument to specify timeout to wait for response from device in seconds.

• **[--depth level]**

optional argument to specify the maximum command level to discover. For example, if `show ip` is the command to be discovered, all subcommands of `show ip` will be traversed. If a depth level of 2 is specified, only the `show ip` command is shown.

• **[--notemplate]**

optional argument to disable template rules lookup. The templates are for well-known commands like `exit`.

• **[--template filename]**

optional argument to specify a different template file from the default.

• **[--norepeat]**

optional argument to ignore repeated requests. The captured data might contain some request more than once. The response for those duplicate request might change for every invocation, those responses are stored in a `mtcl` file and sent to the user in the order they were captured. This enables the user to simulate play back mode. This feature is disabled using the `--norepeat` option, in which case only the first response is used.

• **[--device type]**

optional argument to indicate a different type of CLI. The default is `cisco`, but currently we also know about `juniper` and `riverstone` type devices.

• **[--noredirect symbol]**

optional argument to ignore output redirect commands with the specified symbol.

4. IOS Recorder

The IOS Recorder (`iosrec`) is a stand-alone utility to record Cisco IOS sessions and create basic IOS simulations.

The IOS Recorder uses `tshark` (version 1.10 or newer) (this used to be `tethereal`) to record one or more sessions between a management application (or telnet client) and an IOS device.

The `tshark` package needs to be installed and the `config/iosrec.cfg` file needs to point to the directory path of the installed `tethereal` / `tshark` program.

Here is a sample `config/iosrec.cfg` configuration file:

```
# iosrec configuration file.
tethereal_path = /usr/local/bin
#network_latency = 20
#login_failure_msg = % Login invalid
#login_prompt = Username:
#passwd_prompt = Password:
#rule_template = library.rul
```

Although live recording of IOS transactions will be part of IOS Recorder in the future, the initial version of IOS Recorder depends on third party packet capture tools (e.g. `ethereal`, `tethereal`, `snoop`, `tcpdump`, `Sniffer`) for data acquisition. The captured live traffic protocol data is dumped by the packet capture tool into a binary file that needs to be fed to the IOS Recorder (with the `--in` command line option).

The IOS Recorder identifies connect events, requests and responses by looking at certain sequences of patterns in the captured data. (Technically, `iosrec` uses the TCP flags, the login prompt, password prompt, login failure message as patterns.)

Since it uses sequences of patterns to identify commands and create the rules for the IOS Simulator, it is highly recommended to start `iosrec` before starting a telnet session to the device, ie. one should not record in the middle of a session, eg. after you get login prompt or password prompt, or after typing half of a request.

The IOS Recorder supports the following command line options:

• `--server server-address`

mandatory command line argument to filter the captured data from the specified IOS server.

• `[--client client-address]`

optional argument that will enable the user to generate rules from the traffic between two nodes (if the captured data has traffic from multiple client systems).

• `[--port server-port]`

should be used if the IOS server uses a non standard port.

• `--out rulesdb`

mandatory command line option that specifies the rule file name. The rule file will be stored in the `scripts/telnet` directory.

• `--in capture-file`

• `[--norepeat]`

The captured data might contain some request more than once. The response for those duplicate request might change for every invocation, those responses are stored in a `mtcl` file and sent to the user in the order they were captured. This enables the user to simulate play back mode. This feature is disabled using the `--norepeat` option, in which case only the first response is used.

• `[--append]`

this option enables incremental rule file creation, ie. the new rules are appended to the existing rule file. If `--append` option is used along with `--norepeat` option, the `norepeat` option will be applied only to the newly generated rules.

• `[--template template-file]`

The template library file contains custom rules, which if they match the command will be placed in the rules file, ignoring the response seen by the recorder. This allows to create advanced rules for common IOS commands. This option overrides the default template library file `scripts/telnet/library.rul`.

• `[--notemplate]`

disables lookup in the template library.

› **[--exclude request]***

to exclude specific request(s)

› **[--delay]**

to calculate the delay between request and response. The difference between the response and request includes the network latency. To account only for the command latency, you can set the `network_latency` configuration variable in `config/iosrec.cfg` to eliminate the network delay. The unit is milliseconds.

› **[--loginprompt prompt]**

IOS Recorder uses patterns like the login prompt ("Username: "), password prompt ("Password: ") and login failure message ("Login incorrect") to construct the connection welcome message. These default values can be overridden by setting the `login_prompt`, `passwd_prompt` and `login_failure_msg` configurables in `config/iosrec.cfg` respectively, or by this and the following two command line options.

› **[--passwordprompt prompt]**

Set the password prompt.

› **[--loginfailuremsg msg]**

Set the login failure message.

› **[--resolvename]**

if name resolution is enabled the server and client address can be host names.

"Cisco" and "IOS" are registered trademarks of [Cisco Systems Inc.](#)

[<< Previous Section](#) [Next Section >>](#)



MIMIC TELNET Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using TELNET from MIMICView](#)
- [Using TELNET from MIMICShell](#)
- [TELNET Rules File](#)
- [Global Tcl Variables](#)
- [Events](#)
- [Connection Variable Store](#)
- [TELNET User Database](#)
- [TELNET Key Mappings](#)
- [TELNET Configuration file](#)

1. Overview

The MIMIC TELNET Protocol Module is an optional facility that enables a command line management interface to device simulations on top of the Telnet protocol (RFC 854, 855). This interface is used for [Cisco IOS](#), JUNOS or TL/1 command simulation.

2. Installation

TELNET server support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 10.00, you can use the [Protocol Wizard](#) to install the Telnet module as part of the [IOS Simulator](#). If you prefer to enable Telnet by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the TELNET shared library (telnet.dll on Windows, telnet.so on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the TELNET module was properly loaded :
INFO - TELNET : Loaded protocol from < path-to-DLL >
INFO - TELNET (IOS) v7.00 : Individual license #2345

Once TELNET is loaded, any agent instance configured to support the TELNET protocol will be able to act as a TELNET server.

3. Using TELNET from MIMICView

If the TELNET module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display TELNET as an additional checkbox in the Advanced pane along with the SNMP protocols. On selecting the checkbox a new TELNET pane will appear.

This TELNET configuration pane lets the user configure the parameters for a TELNET session:

- **Port**
This optional parameter specifies the port at which the server will be listening. The default is the standard port 23.
- **Rule File**
This mandatory parameter specifies the rules that govern the TELNET sessions to this agent. The rule file is in `scripts/telnet/`, and in effect implements a command-line interface. Each rule consists of a regular expression that is matched against the typed request (command), and a response that is shown. For more complicated rules, a MIMICshell script can be run. A sample rule file is shipped as `sample.rul`. Details can be found [below](#).
- **Prompt**
This optional parameter specifies the prompt to be displayed by the server. If not specified, a default prompt is shown.
- **Paging Prompt**
This optional parameter specifies the paging prompt to be displayed by the server when a command output is longer than the terminal length. If not specified, a default paging prompt is shown.
- **User DB**
This optional parameter specifies the user database. This database is used for login and access control. Details can be found [below](#).
- **Key Map File**
This optional parameter specifies the key mappings. This database is used for keyboard shortcuts. Details can be found [below](#).

If the mandatory parameters are supplied, the agent will automatically execute a TELNET server upon starting. A message in the log of the form

```
INFO 11/28.09:53:42 - TELNET server started for agent 2
```

indicates that the Telnet simulation is running.

Using TELNET from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the TELNET functionality. Here is a synopsis:

- **mimic protocol msg TELNET get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg TELNET get args
{{port} {Port} {integer} {} {optional} {23}}
{{rule} {Rule File} {file} {scripts/telnet {*.rul {Rule database files}
{edit yes} {new yes}}} - both} {mandatory} {sample.rul}}
{{prompt} {Prompt} {string} {} {optional} {}}
{{paging_prompt} {Paging Prompt} {string} {} {optional} { --More--}}
{{userdb} {User DB} {file} {scripts/telnet {*.db {User database files}
{edit yes} {new yes}}} - both} {optional} {sampleuser.db}}
{{keymap} {Key Map File} {file} {scripts/telnet {*.kmap {Key mapping files}
{edit yes} {new yes}}} - both} {optional} {sample.kmap}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,TELNET
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,TELNET
mimicsh> mimic agent get protocol
snmpv1,TELNET
```

- **mimic agent protocol msg TELNET get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg TELNET get config
{port=23} {rule=sample.rul} {prompt=} {paging_prompt=} {userdb=sampleuser.db}
{keymap=sample.kmap}
```

- **mimic agent protocol msg TELNET set config [config]**

This command lets the user change the current argument settings of all TELNET sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg TELNET get config
{port=23} {rule=sample.rul} {prompt=} {paging_prompt=} {userdb=sampleuser.db}
{keymap=sample.kmap}

mimicsh> mimic agent protocol msg TELNET set config {prompt=IOS>}

mimicsh> mimic agent protocol msg TELNET get config
{port=23} {rule=sample.rul} {prompt=IOS>} {paging_prompt=} {userdb=sampleuser.db}
{keymap=sample.kmap}
```

- **mimic agent protocol msg TELNET get trace**
mimic agent protocol msg TELNET set trace [0 or 1]

This command lets the user change the TELNET tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 1

mimicsh> mimic agent protocol msg TELNET get trace
0
mimicsh> mimic agent protocol msg TELNET set trace 1

mimicsh> mimic agent protocol msg TELNET get trace
1
```

and the log would show:

```
INFO - agent 1 trace enabled for TELNET
INFO - TELNET [AGT=1]: server started
INFO - TELNET[AGT=1,CON=2]: connection request from [10.1.120.142,41851]
INFO - TELNET[AGT=1,CON=2]: sending 3 byte(s) data in pkt #1 pkt to [10.1.120.142,41851]
INFO - FF FB 01 ...
INFO - TELNET[AGT=1,CON=2]: sending 3 byte(s) data in pkt #2 pkt to [10.1.120.142,41851]
INFO - FF FB 03 ...
INFO - TELNET[AGT=1,CON=2]: sending 3 byte(s) data in pkt #3 pkt to [10.1.120.142,41851]
```

```

INFO - FF FD 1F ...
INFO - TELNET[AGT=1,CON=2]: received 18 byte(s) data in pkt #3 from [10.1.120.142,41851]
INFO - FF FD 01 FF FD 03 FF FB 1F FF FA 1F .....
INFO - 00 50 00 18 FF F0 .P....
...

```

- **mimic protocol msg TELNET get stats_hdr**
mimic agent protocol msg TELNET get statistics
Returns TELNET statistics information:

- a list of statistic headers, and
- current statistics values for the specified server.

In order, the statistic values are:

- Total number of TELNET connections.
- Total number of TELNET disconnects.
- Total number of TELNET TCP packets sent.
- Total number of TELNET TCP packets received.
- Total number of TELNET command requests.
- Total number of TELNET command responses.
- Total number of unrecognized TELNET commands.
- Total number of TELNET script errors.

A sample exchange for these commands would be:

```

mimicsh> mimic protocol msg TELNET get stats_hdr
{{connect} {Connect}} {{disconnect} {Disconnect}} {{pktSnt} {PktsSent}}
{{pktRcvd} {PktsRcvd}} {{request} {CmdRequest}} {{response} {Response}}
{{norule} {No Matching Rule }} {{script_err} {Script Error}}

mimicsh> mimic agent protocol msg TELNET get statistics
9032775 9032772 162591536 39691090 8960770 0 0 1123

```

- **mimic agent protocol msg TELNET server get state**
Returns 1 if the TELNET server can accept connections otherwise 0.
- **mimic agent protocol msg TELNET server get rulesdb**
Returns the rules data base file name.
- **mimic agent protocol msg TELNET server get userdb**
Returns the user data base file name.
- **mimic agent protocol msg TELNET server get keymap**
Returns the keymap file name.
- **mimic agent protocol msg TELNET server get users**
Retrieves the user information from the user database: a list of configured user entries, each with its user-name, password flag (set to 1 for no password), list of groups and optional password in the following format

```

{ user-name1 0|1 {group-list1} [password1] }
{ user-name2 0|1 {group-list2} [password2] }
...
...

{ usern 0|1 {group listn} [passwordn] }

```

- **mimic agent protocol msg TELNET server get connections**
Retrieves the currently connected sessions in a list of connection identifiers that can be used with the [mimic agent protocol msg TELNET connection](#) commands.

```

foreach conn [mimic agent protocol msg TELNET server get connections] {
    mimic agent protocol msg TELNET connection logon $conn hidden "{}"
    mimic agent protocol msg TELNET connection request $conn syslog
    mimic agent protocol msg TELNET connection request $conn exit
}

```

- **mimic agent protocol msg TELNET connection logon conn-id user password**
Changes the current logon, so that (hidden) commands for a different access mode can be run.
- **mimic agent protocol msg TELNET connection request conn-id command**
Executes the command as if it had been typed by the user. This is the way to execute asynchronous commands.
- **mimic agent protocol msg TELNET connection signal conn-id signal-name**
This command triggers the asynchronous signal event with the specified signal name. See [below](#).
- **mimic agent protocol msg TELNET ipalias enable ipaddress[,port]**
mimic agent protocol msg TELNET ipalias disable ipaddress[,port]
By default, the MIMIC TELNET server listens on all the IP addresses (aliases) that are configured for an agent before the TELNET service is started. While the agent is running, TELNET service can be enabled and disabled on an IP alias using the above commands.
- **mimic agent protocol msg TELNET ipalias isenabled ipaddress[,port]**
This command returns 1 if the IP alias is enabled, else 0.

- **mimic agent protocol msg TELNET ipalias list**

This command returns the list of enabled IP aliases for this Telnet server.

TELNET Rules File

Overview

The rules database is a ASCII text file that is associated with a TELNET server instance, and defines the behavior of the Telnet simulation. The loader reads in the file and sources any Tcl scripts as needed. The contents of the file are used to initialize data structures that will then be used to return a response string (output) for a given request string (user input). In the absence of a match an error is returned.

The rules database is located in the `scripts/telnet/` directory in the MIMIC directory hierarchy. Any custom Tcl scripts also need to be in this same directory.

NOTE: rules and script files are cached, and modifications to them will only be reloaded by the parser when the rule file modification time is newer than what is in the cache.

Format

The rules database is made of a series of "rule" blocks. A typical rules database will look something like:

```
version = 5.10
rule = {
  request = ^show memory$
  response = 10485760 bytes (10MB)
}
rule = {
  request = ^show processes$
  mapping = tcl_proc
  response = custom_responses.mtcl:show_processes
  delay = 25
}
rule = {
  request = ^config*
  mapping = tcl_proc
  response = custom_responses.mtcl:configure
  delay = 15
}
rule = {
  request = ^(sh|sho|show)\s(clo|cloc|clock)+
  mapping = tcl_expr
  response = clock format [clock seconds] -format "%H:%M:%S %Z %a %b %d %Y\r\n"
  delay = 25
}
```

In addition special blocks called "event" can also be defined in this file. These events will be hooked to pre-defined events during the course of a TELNET session. e.g.

```
event = {
  request = connect
  mapping = tcl_expr
  response = set str "Welcome to MIMIC IOS simulation.\r\n"
  response = append str " Server: $gCurrentConnLocAddr, $gCurrentConnLocPort\r\n"
  response = mts_puts $gCurrentConnId $str
}
```

Matching

A rule will match the request clause with a user command in one of 3 ways:

- regular expression match

This matching is selected with the `matching = regexp` clause. It is also the default if no `matching` clause is present in the rule. For example, a rule containing

```
request = ^(sh|sho|show)\s(clo|cloc|clock)+
```

with no other `matching` clauses will match any command that the user types in that starts with `show cloc`, including any combination of abbreviations `sh` and `sho`, or `clo` and `cloc`.

- string comparison

This matching is selected with the `matching = strcmp` clause. This causes a straight string comparison between the command typed in by the user and the string in the `request` clause. For example, a rule containing

```
request = where
mapping = strcmp
```

will match only the command `where`. This is really just a shorter form of the regular expression

```
request = ^where$
```

- case insensitive string comparison

This matching is selected with the `matching = stricmp` clause. This causes a case insensitive string comparison between the command typed in by the user and the string in the `request` clause.

The first rule in the file which matches the command will be used. Thus it is better to put more specific matches at the beginning, more general matches later in

the rule file.

Mappings

There are three distinct types of mappings between requests and responses that will be supported by the rules database. These are as follows :

- **map to string**
In this form a match of the request string will result in a hardcoded response string to be returned. This is the default, if no `mapping` clause is provided in the rule.
- **map to Tcl expression**
This mapping is selected with the `mapping = tcl_expr` clause. In this form the regular expression is used to match the request string. A match will then result in execution of a specified Tcl expression. The resulting value is sent back as a response.
- **map to Tcl procedure**
This mapping is selected with the `mapping = tcl_proc` clause. In this form the regular expression is used to match the request string. A match will then result in invocation of a specified Tcl procedure. The response string will specify the Tcl file to be sourced and the Tcl proc to be executed. The resulting return value is sent back as response.

Pipe

If a pipe symbol is detected in the command line, the parser will try to match the command before the pipe (through the expressions in the rules file) and feed its output to the command matched after the pipe symbol, again through the rules in the rule file (where commands can be marked to be matched only on the right hand side with the

```
pipe = consumer
```

statement). The sample `ios.rul` shows this for the IOS `include`, `exclude` and `begin` commands. The default pipe symbol is `|`, which you can override as detailed [below](#).

Global Tcl Variables

Any Tcl expression evaluated as well as any Tcl procedure invoked will have a predefined set of global variables passed to it which identify the current environment.

- **gCurrentRequest**
This variable holds the command completed and normalized (space separated) user request string which was received on behalf of the user or the name of the current event.
- **gOriginalRequest**
This variable holds the user request string as it was sent by the client (without normalization and command completion) or the name of the current event.
- **gCurrentAgent**
This variable holds the agent that the current server belongs to.
- **gCurrentSrvrId**
This variable holds the current server's identifier.
- **gCurrentConnId**
This variable holds the current connection's identifier.
- **gCurrentConnLocAddr**
This variable holds the current connection's local address.
- **gCurrentConnLocPort**
This variable holds the current connection's local port.
- **gCurrentConnRemAddr**
This variable holds the current connection's peer address.
- **gCurrentConnRemPort**
This variable holds the current connection's peer port.
- **gCurrentRulesDB**
This variable holds the current rules database name.
- **gSkip**
On return, this variable indicates to the command parser what to do after processing this matching rule. A return value of 1 skips to the next matching rule without returning the output result. A return value of 2 skips caching the rule. A return value of 3 skips processing of any pipe consumer commands.

Delay

An optional delay parameter can be associated with a "rule" block which allows a user to prescribe a certain latency for the response. This delay is in milliseconds. The user of the rules database (TELNET DLL) is responsible to use this parameter to simulate the latency.

Access

This optional access parameter can be used to specify a list of users that have access to a matching "rule" block. If this list is empty or not specified then any user can access the rule. This parameter can be used to restrict access to privileged command sets.

Built-in Commands

A set of built-in MIMIC Telnet Simulator (MTS) commands prefixed by "mts_" allows the modeller to invoke the API supported by the TELNET module.

Login sessions

- **mts_exit conn-id**

This command will allow the caller to close the connection identified by the conn-id parameter. This is useful for the user to map arbitrary keywords to close the current connection. e.g.

```
rule = {
  request = ^exit$|^logout$
  mapping = tcl_expr
  response = mts_exit $gCurrentConnId
}
```

- **mts_setprompt conn-id**

This command will allow the caller to change the prompt for connection identified by conn-id parameter. e.g.

```
rule = {
  mapping = tcl_expr
  request = ^setprompt*
  response = eval mts_setprompt $gCurrentConnId [lrange $gCurrentRequest 1 end]
}
```

- **mts_getprompt conn-id**

This command will enable the caller to retrieve the prompt for connection identified by conn-id parameter.

- **mts_logon sub-cmd conn-id args**

This command will allow the caller to invoke the API of the user-login subsystem embedded in each server. The following sub-commands will be supported :

- **mts_logon set conn-id attribute value**

The MTS framework enables the user to configure the login subsystem through a configuration file. This subsystem can also be fine tuned on per connection basis using the following attributes.

This command enables the callers to fine tune the user-login subsystem to their need. The following attributes are supported

- **callback** - A Tcl procedure or expression that will be called upon completion of user-login subsystem. This will override the value set using logon_callback variable in the configuration file.
The callback procedure will be called by setting all the global variables mentioned [above](#) as well as these other global variables:
 - gCurrentLogonStatus -- status of login. It will be set to "SUCCESS" if the logon is successful otherwise set to "FAILURE".
 - gCurrentAuthRetries -- number of failed attempts
 - gCurrentMaxAuthRetries -- allowed failed Attempts
 - gCurrentAuthUser -- the last user name attempted
- **retries** - Number of unsuccessfull login attempts allowed. This will override the value set using logon_retries variable in the configuration file
- **login_timeout** - Login idle timeout in seconds. This will override the value set using idle_timeout_login variable in the configuration file
- **login_prompt** - Login prompt (default is "login: "). This will override the value set using login_prompt variable in the configuration file
- **password_prompt** - Password prompt (default is "passwd: "). This will override the value set using password_prompt variable in the configuration file
- **failure_msg** - The message to be sent on every unsuccessful attempt. This will override the value set using login_failure_msg variable in the configuration file
- **timeout** - terminal idle timeout in seconds. This will override the value set using idle_timeout variable in the configuration file

- **mts_logon init conn-id [user] [password]**

This command allows the caller to enter into a logon transaction. If the user and password is provided then the client is not prompted for any information. If password is not provided then it is prompted. If user and password are not provided then both are prompted in that order. On completion (SUCCESS|FAILURE) of the login attempt the optional Tcl procedure or expression is invoked if specified by "mts_logon set callback" command.

- **mts_logon done conn-id user**

This command allows the caller to logoff as a particular user in the context of the specified connection.

- **mts_logon list conn-id**

This command lists the users that are logged in for the specified connection. The caller can parse this list to determine if appropriate logon has been done.

Example: Here is a typical usage:

```
event = {
  request = connect
  mapping = tcl_proc
  response = custom_responses.mtcl:on_connect
}
event = {
  request = idle
  mapping = tcl_expr
  response = mts_exit $::gCurrentConnId
}
rule = {
  request = enable
  mapping = tcl_expr
  response = mts_logon init $gCurrentConnId enable
}
rule = {
  request = ^config*
  mapping tcl_proc
  response = custom_responses.mtcl:config
  access = enable
}

proc on_connect {} {

  set ret_msg "Welcome to MIMIC IOS simulator"

  global gCurrentConnId
  mts_logon config callback logon_callback
  mts_logon init $gCurrentConnId

  return $ret_msg
}

proc logon_callback {} {

  if { [mts_logon list] == "" } {

    global gCurrentConnId
    mts_puts $gCurrentConnId "% Login invalid"

    mts_exit $gCurrentConnId
  }
}
```

Terminal

- **mts_get terminal conn-id length**
returns configured terminal length, default is 23.
- **mts_set terminal conn-id length length**
sets the terminal length. Paging can be disabled by setting the length to 0.

Key Control

These are built-in actions for the key map configuration file to change the behavior of keys. They support functionality such as command-line editing, history, hotkeys, etc. The working principle of the command line interface is that keys are accumulated in a command buffer (ie. the "line"), until a special key (usually the ENTER key) causes the command buffer to be passed to the command parser. Within the command buffer there is a "insertion point", which is where keys will be inserted. By default, the insertion point is at the end of the buffer, but it can be moved with the key commands. The insertion point is marked specially on the terminal, usually through a blinking cursor.

- **mts_key_insert**
This command inserts the key into the command buffer. If no argument, then the key sequence that triggered this command is inserted into the buffer. This behaviour is the default for all unmodified key bindings. With an argument the key sequence in the argument is inserted instead.
- **mts_key_done**
This command causes the parser to start parsing the command buffer.
- **mts_key_echo_newline**
This causes a new line to be echoed.
- **mts_key_rubout**
This removes the preceding character in the command buffer.
- **mts_key_move_char_backward**
This moves the insertion point back one position. Once at the beginning of the line, the insertion point cannot be moved further.
- **mts_key_move_char_forward**
This moves the insertion point forward one position. Once at the end of the line, the insertion point cannot be moved further.

This command combined with `mts_key_rubout` can be used to ignore a key.

- **mts_key_move_line_start**
This moves the insertion point to the beginning of the line.
- **mts_key_move_line_end**
This moves the insertion point to the end of the line.
- **mts_key_kill_word_backward**
This command removes the preceding word, ie. a sequence of non-blank characters delimited by blank spaces.
- **mts_key_kill_line_backward**
This command removes the preceding characters to the beginning of the line.
- **mts_key_kill_line_forward**
This command removes the following characters to the end of the line.

Command Completion

- **mts_cmd_buffer conn-id set string**
mts_cmd_buffer conn-id append string
mts_cmd_buffer conn-id get
mts_cmd_buffer conn-id reset
These commands control the command buffer. You can set a string, append a string, get and reset the buffer.
- **mts_cmd_complete conn-id command**
This command allows a user to pass in an incomplete command string and peek at the completed result if an unambiguous resolution is possible. This behaviour is built into the rule-execution code by default.
- **mts_cmd_hint conn-id command**
This command allows the user to pass in an incomplete command string and list possible completion alternatives. This is tied into the "\t" hotkey currently.
- **mts_sub_cmds conn-id command**
This command allows the user to pass in a command string and list possible subcommands that would follow the same to the next level. This would be useful for writing dynamic help.

Command History

- **mts_get_cmd_history conn-id**
returns a TCL list of command strings in the history buffer. This is used for the `show history` IOS command.
- **mts_key_get_prev_history**
mts_key_get_next_history
return the previous/next command in the history buffer. They are usually tied to keys in the [key mapping](#).
- **mts_set history conn-id length length**
sets the history buffer length. If the new length is less than the number of commands stored in the history, the oldest commands are lost. See the IOS `terminal history size` command implementation for a use of this command.
- **mts_inherit_cmd_history conn-id**
inherits the history buffer from the calling user. This is used in the IOS `enable` logon callback handler to share history buffer in user EXEC and privileged EXEC modes.

Responses

The response string displayed in a Telnet session is manipulated by the `response` primitive in rules files. Normally, this response is static, in that it will be displayed with the string returned from a rule. The following commands dynamically manipulate the response buffer for more advanced control of the response string (eg. in callback routines).

- **mts_response set conn-id string**
Sets the response to the specified string.
- **mts_response append conn-id string**
Appends the specified string to the response.
- **mts_response clear conn-id**
Clears the response string.
- **mts_response get conn-id**
Gets the response string.
- **mts_response flush conn-id**
Flushes the response string and displays it in the telnet session.
- **mts_puts conn-id string**
Synchronously display the string. This is independent of the response buffer.

Events

The Telnet framework supports asynchronous event handling for advanced processing of rules.

When an event occurs, the event listener will execute the callback procedure registered for it. Events are executed one at a time in the order they are received.

The event callback is invoked with all the global variables that are passed to the original request (where the event was registered). Along with those global variables some additional global variables are also passed to the callback procedure. They are:

- **gCurrentEventId**
event id, generated by the system
- **gCurrentEventType**
Will be either DATA, TIMER or SIGNAL
- **gCurrentEventValue**
gCurrentEventId will have a signal name for signal event, key or line for a data type event and time quantum for timer type events.
- **gCurrentEventData**
For a timer type event this variable will hold the time (in milliseconds since UCT) when the event occurred.

For a data type event this variable will hold the data sent by the client. If the data has any embedded unprintable character then it will be converted into standard mimic format (\xff ff ff).

Currently, the following 2 built-in events are supported:

- **connect**
This event occurs when a connection is first established to the telnet server.
- **idle**
This event occurs when there is a idle timeout on the connection.

These commands register and deregister user-defined events:

- **mts_event register conn-id timer msec callback [arguments]**
This command allows the caller to register a timer event and an associated callback for the connection specified by conn-id. On successful registration it returns an event-id. The values of timer event-id are prefixed with "tim_". Each command request can have 10 outstanding timer event, which can be increased upto 500 through the `max_timer_event` variable in `telnet.cfg`. This command takes optional arguments that will be passed as argument list while invoking the callback procedure. When the registered time quantum msec expires, the timer event will be cleared and the callback will be invoked.

Timer event registration will result in an error when it exceeds the configured limit.

The returned event-id can be used to cancel the event.

- **mts_event register conn-id data char|line callback [arguments]**
This command allows the caller to register a callback procedure to be invoked when the client sends either a single character or a line of characters (delimited by "\r\n") for the connection identified by conn-id parameter. This command takes optional arguments that would be passed in to the callback procedure as argument list. The caller can have only one outstanding data event registration. The data event registration will be cancelled as soon as the data event occurs. This command return an event identifier on success. The event identifier is always prefixed with "dat_" to indicate that the event is a data event.

Attempting to register a data event will result in an error, when there is already an outstanding data event registration.

The returned event-id can be used to cancel the event.

- **mts_event register conn-id signal signal-name callback [arguments]**
This command allows the caller to register a signal using a case sensitive unique signal identifier `signal-name` and an associated callback. It returns a unique event-id prefixed with "sig_".

Signals are triggered either by scripts (action, completion, etc) or manually through the mimic shell using the "telnet" protocol command (defined [above](#)). The caller is allowed to register a maximum of 10 signals, which can be configured through the `max_signal` configurable in `telnet.cfg`. When a signal is triggered its registration will be cancelled and the associated callback procedure will be called with all the optional arguments as argument list.

Any attempt to register a signal with an outstanding (not yet triggered) signal name will result in error.

The returned event-id can be used to cancel the event.

- **mts_event cancel conn-id event-id**
This command cancels a pending event registration identified by event-id. An attempt to cancel an unregistered event results in error.
- **mts_event list conn-id**
This command will return a TCL list of outstanding (not yet serviced) registered events in the following format:

```
{
  { event_id1, type1, value1, callback1, {arg_list1} } ,
  { event_id2, type2, value2 ,callback2, {arg_list2} }
  .....
  .....
  { event_idn, typen, valuen ,callbackn, {arg_listn} }
```

```
}
```

Connection Variable Store

The Connection Variable Store is a volatile data storage facility (similar to the [MIMIC Variable Store](#)) that resides within the context (or scope) of each connection and is active for the duration of its associated connection. Variables can be used to store connection-specific state without interfering with other connections. These variables are cleaned up when the connection is closed.

The Connection Variable Store facility can be accessed through the `mts_store` built-in command.

- `mts_store set conn-id var value`
`mts_store append conn-id var value`
These commands allow the creation of a new variable `var`, or changing an existing variable, with the value `val`. The `append` sub-command will append the value `val` to an existing variable, or create a new one. The `set` sub-command will overwrite an existing variable, or create a new one.
- `mts_store unset conn-id var`
Deletes a currently defined variable `var`.
- `mts_store get conn-id var`
Fetches the value associated with a variable `var`. The value will be returned as a string (like all Tcl values).
- `mts_store exists conn-id var`
This command can be used as a predicate to ascertain the existence of a given variable `var`. It returns "1" if the variable exists, else "0".
- `mts_store list conn-id`
This command will return the list of variables in the said scope. The list will be a Tcl format list with curly braces "{}" around each list element. These elements in turn are space separated.

TELNET User Database

The user database contains entries for each user that has access to the simulation. The username and password will be authenticated at login. In addition, each user belongs to one or more access groups, with which access to commands can be controlled (with the `access` parameter). Here is a sample file:

```
version = 5.20
user = {
username = lab
password = lab123
}

user = {
username = admin
password = admin123
group = admin config
}

user = {
username = enable
password = enable123
group = enable
}
```

TELNET Key Mappings

The key mapping file defines keyboard shortcuts for commands or special functionality. Here is a sample file:

```
version = 5.40

keymap = {
# IOS help hot key
key = ?
action = mts_key_insert
action = mts_key_echo_newline
action = mts_key_done
}

keymap = {
# IOS command completion hot key
key = \t
action = mts_key_cmd_completion
}

keymap = {
# up arrow
key = \M-[A
action = mts_key_get_prev_history
}

keymap = {
# down arrow
key = \M-[B
action = mts_key_get_next_history
}
```

```
}
```

1. TELNET Configuration file

The optional Telnet configuration file `config/telnet.cfg` overrides some hard-coded parameters in the Telnet server. It is only necessary to customize some of the default prompts and error messages. Here is a sample:

```
# this is a sample telnet.cfg file to configure advanced options

# port - Telnet server port (default 23)
port=2323

# maximum simultaneous clients per agent (default 3)
# this does not place a limit on the number of simultaneous clients,
# but an upper limit on resources over all the agents. That upper limit
# is max_agents * simultaneous_clients .
#simultaneous_clients = 10

# tcp_nodelay - control TCP_NODELAY, ie. enable/disable Van Jacobson TCP/IP
# optimizations (default is 1, ie. optimization off)
# for small number of telnet sessions the default behavior achieves better
# response time, at the cost of utilization
#tcp_nodelay=1

#prompt=

# rulesdb - default rules database
#rulesdb=sample.rul

# userdb - default user database
#userdb=sampleuser.db

login_prompt=Username:

#password_prompt=

# logon_failure_msg - error message on logon failure
#logon_failure_msg=

# noaccess_msg - error message on access failure
#noaccess_msg=$r: Permission denied\r\n

# ruleslookup_failure - error message on rules lookup failure
#ruleslookup_failure_msg=$r: Command not found\r\n

# echo - controls echoing: on (default), off, negotiated
#echo=on

# term_keymap_file - key handling in raw mode
#term_keymap_file=t11.kmap

# pipe symbol(s)
#pipe = |
```

[<< Previous Section](#) [Next Section >>](#)



MIMIC SSH Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using SSH from MIMICView](#)
- [Using SSH from MIMICShell](#)
- [Licensing Information](#)

1. Overview

The MIMIC SSH Protocol Module is an optional facility that enables a command line management interface to device simulations on top of the [secure shell protocol \(ssh\)](#). Either protocol versions 1.5 and 2.0 are supported in this bilingual server. This interface is used for [Cisco IOS](#) or JUNOS command simulation.

2. Installation

SSH server support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 10.00, you can use the [Protocol Wizard](#) to install the SSH module as part of the [IOS Simulator](#). If you prefer to enable SSH by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- First install the [Telnet module](#) on which the SSH module depends.
- Copy the SSH shared library (ssh.dll on Windows, ssh.so on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the SSH module was properly loaded :
INFO - Loaded protocol [SSH] from < path-to-DLL >

Once SSH is loaded, any agent instance configured to support the SSH protocol will be able to act as a SSH server.

NOTE: On Solaris 10, you need to install the optional [openssl](#) software. We have tried the package at [SunFreeware](#) and it works.

NOTE: SSH depends on openssl which uses RAND number generation device /dev/urandom and /dev/random. Older versions of Solaris don't carry these device by default.

In that case SSH would show the below error.

```
ERROR 06/24.10:57:32 ssh_threads.cc:166 - SSH [AGT=1]: cannot start server
06/24.10:57:32 ssh_server.cc:54 - host key generation failed
06/24.10:57:32 ssh_key.cc:42 - You need to read the OpenSSL FAQ,
http://www.openssl.org/support/faq.html
```

Excerpt from FAQ

If you are using Solaris 8, you can add /dev/urandom and /dev/random devices by installing patch 112438 (Sparc) or 112439 (x86), which are available via the Patchfinder at <http://sunsolve.sun.com> (Solaris 9 includes these devices by default). For /dev/random support for earlier Solaris versions, search for "/dev/random" at SunSolve.

3. Using SSH from MIMICView

If the SSH module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display SSH as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new `SSH` pane will appear.

This SSH configuration pane lets the user configure the parameters for a SSH session:

- **Port**
This optional parameter specifies the port at which the server will be listening. The default is the standard port 22.

NOTE: If you are running a SSH server on your MIMIC host, by default it will prevent starting simulated SSH servers on the standard port. You can either configure your simulated SSH servers on a different port, or instruct the host SSH server to only listen on a single IP address, eg. on Linux change the `ListenAddress` in `/etc/ssh/sshd_config`.

Version

This optional parameter specifies the version supported. If set to 1.5, then only that version is accepted. If set to 2.0, then only that version is accepted. If set to the default 1.99, then either version is accepted.

If the mandatory parameters are supplied, the agent will automatically execute a SSH server upon starting. A message in the log of the form

```
INFO 11/28.09:53:42 - SSH server started for agent 2
indicates that the SSH simulation is running.
```

Using SSH from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the SSH functionality. Here is a synopsis:

- **mimic protocol msg SSH get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg SSH get args
{{port} {Port} {integer} {} {optional} {22}}
{{version} {Version} {string} {} {optional} {1.99}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,SSH
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,SSH
mimicsh> mimic agent get protocol
snmpv1,SSH
```

- **mimic agent protocol msg SSH get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg SSH get config
{port=22} {version=}
```

- **mimic agent protocol msg SSH set config [config]**

This command lets the user change the current argument settings of all SSH sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg SSH get config
{port=22} {version=}

mimicsh> mimic agent protocol msg SSH set config {port=8122}

mimicsh> mimic agent protocol msg SSH get config
{port=8122} {version=}
```

- **mimic protocol msg SSH get stats_hdr**

- **mimic agent protocol msg SSH get statistics**

Returns SSH statistics information:

- a list of statistic headers, and
- current statistics values for the specified server.

In order, the statistic values are:

- Total number of SSH connections.
- Total number of SSH disconnects.
- Total number of SSH packets sent.
- Total number of SSH packets received.
- Total number of SSH command requests.
- Total number of SSH command responses.

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg SSH get stats_hdr
{{connect} {Connect}} {{disconnect} {Disconnect}} {{pktSnt} {PktsSent}}
{{pktRcvd} {PktsRcvd}} {{request} {CmdRequest}} {{response} {Response}}

mimicsh> mimic agent protocol msg SSH get statistics
2 1 0 14 0 0
```

- **mimic agent protocol msg SSH ipalias enable ipaddress[,port]**

mimic agent protocol msg SSH ipalias disable ipaddress[,port]

By default, the MIMIC SSH server listens on all the IP addresses (aliases) that are configured for an agent before the SSH service is started. While the agent is running, SSH service can be enabled and disabled on an IP alias using the above commands.

- **mimic agent protocol msg SSH ipalias isenabled ipaddress[,port]**

This command returns 1 if the IP alias is enabled, else 0.

- **mimic agent protocol msg SSH ipalias list**

This command returns the list of enabled IP aliases for this SSH server.

Licensing Information

This implementation is based on code from [OpenSSH](#) which contains the enclosed [license](#).

This product includes software developed by the [OpenSSL Project](#) for use in the OpenSSL Toolkit with the following [copyrights](#).

[<< Previous Section](#) [Next Section >>](#)



MIMIC SYSLOG Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using SYSLOG from MIMICView](#)
- [Using SYSLOG from MIMICShell](#)
- [Compatibility](#)

1. Overview

The MIMIC SYSLOG Protocol Module is an optional facility that enables the BSD SYSLOG protocol ([RFC 3164](#)). This facility is used for transmission of event notification messages across networks, most notably for [Cisco IOS](#).

2. Installation

SYSLOG support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 10.00, you can use the [Protocol Wizard](#) to install the SYSLOG module. If you prefer to enable SYSLOG by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the SYSLOG shared library (`syslog.dll` on Windows, `syslog.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the SYSLOG module was properly loaded :
INFO - SYSLOG : Loaded protocol from < path-to-DLL >
INFO - SYSLOG v7.00

Once SYSLOG is loaded, any agent instance configured to support the SYSLOG protocol will be able to send SYSLOG events to a SYSLOG server.

3. Using SYSLOG from MIMICView

If the SYSLOG module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display SYSLOG as an additional checkbox in the Advanced pane along with the SNMP protocols. On selecting the checkbox a new SYSLOG pane will appear.

This SYSLOG configuration pane lets the user configure the parameters for a SYSLOG session:

- **Server**
This optional parameter specifies the SYSLOG server(s) to send messages to. If more than one, they need to be space separated.
- **Server Port**
This optional parameter specifies the SYSLOG server well-known port. The default is the standard port 514.
- **Local Port**
This mandatory parameter specifies the local port to use. The default is 0, which means to auto-assign a port number.

The remaining configurables are ignored and need to be set with the `mimic agent protocol msg SYSLOG set attr` command below.

4. Using SYSLOG from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the SYSLOG functionality. Here is a synopsis:

- **mimic protocol msg SYSLOG get args**
This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg SYSLOG get args
{{server} {Server} {string} {} {optional} {}}
{{serverport} {Server Port} {string} {} {optional} {514}}
{{client} {Client} {string} {} {optional} {}}
{{localport} {Local Port} {integer} {} {optional} {}}
{{sequence} {Sequence} {integer} {} {optional} {}}
{{separator} {Separator} {string} {} {optional} {}}
{{timestamp} {Timestamp} {string} {} {optional} {}}
```

```
{{hostname} {Hostname} {string} {} {optional} {}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,SYSLOG
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,SYSLOG
mimicsh> mimic agent get protocol
snmpv1,SYSLOG
```

- **mimic agent protocol msg SYSLOG state**

This command lets the user query the state of the SYSLOG agent. This is particularly useful at agent startup time to wait for SYSLOG startup. A sample usage for this command would be:

```
if { [mimic agent protocol msg SYSLOG state] != "up" } {
    # do necessary to wait until SYSLOG is up
}
```

- **mimic agent protocol msg SYSLOG get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg SYSLOG get config
{server=} {serverport=514} {client=} {localport=0} {sequence=} {separator=} {timestamp=} {hostname=}
```

- **mimic agent protocol msg SYSLOG set config [config]**

This command lets the user change the current argument settings of all SYSLOG sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg SYSLOG get config
{server=} {serverport=514} {client=} {localport=0} {sequence=} {separator=} {timestamp=} {hostname=}

mimicsh> mimic agent protocol msg SYSLOG set config localport=514

mimicsh> mimic agent protocol msg SYSLOG get config
{serverport=514} {localport=514}
```

- **mimic agent protocol msg SYSLOG get trace**

- **mimic agent protocol msg SYSLOG set trace [0 or 1]**

This command lets the user change the SYSLOG tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 9

mimicsh> mimic agent protocol msg SYSLOG get trace
0
mimicsh> mimic agent protocol msg SYSLOG set trace 1

mimicsh> mimic agent protocol msg SYSLOG get trace
1
```

and the log would show:

```
INFO - agent 9 trace enabled for SYSLOG
INFO - agent 9 configured at 10.0.0.9,161.
INFO - agent 9 loading
INFO - agent 9 loaded device
Agent 9 invoking start+syslog.mtcl
INFO - SYSLOG[AGT=9]: sent to [192.9.200.71,514] - <3>Oct 01 11:23:41 cisco-7513
Cisco Internetwork Operating System Software
IOS (tm) RSP Software (RSP-JSV56I-M), Version 12.1(7), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Fri 23-Feb-01 05:14 by kellythw
```

- **mimic protocol msg SYSLOG get stats_hdr**

- **mimic agent protocol msg SYSLOG get statistics**

Returns SYSLOG statistics information:

- a list of statistic headers, and
- current statistics values for the specified client.

In order, the statistic values are:

- Total number of SYSLOG packets sent.
- Total number of SYSLOG packets discarded.

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg SYSLOG get stats_hdr
{{pktSnt} {PktsSent}} {{pktDisc} {PktsDiscarded}}

mimicsh> mimic agent protocol msg SYSLOG get statistics
1 0
```

- **mimic agent protocol msg SYSLOG send pri msg**

This command sends a SYSLOG event to the server(s) with the specified PRI and MSG fields. It can only be used after starting the agent.

By default, the format of the message will adhere to the [standard RFC 3164](#). The standard format of the message is

```
>PRI[<>SEQ<>SEP<]>TIMESTAMP<>SEP[<>HOSTNAME<>SEP<]>MSG<
where fields between [] are optional, and between >< are substituted as follows:
```

- PRI is the PRI field of RFC 3164
- SEQ is an optional sequence number (used by Cisco IOS)
- SEP is a separator, " " by default
- TIMESTAMP is the TIMESTAMP field of RFC 3164
- HOSTNAME is the HOSTNAME field of RFC 3164
- MSG is the MSG field of RFC 3164

However, this can be modified with the following commands.

- **mimic agent protocol msg SYSLOG set attr value**

where attr is one of the following:

- server - a space-separated list of server,port
- client - the source address for messages. Must be one of the IP aliases of this agent
- sequence - the starting sequence number, if any. By default, none.
- separator - the separator in the HEADER field, by default a space
- timestamp - either "clock" (default) for real-time clock, "sysUpTime" for timestamp based on the value of sysUpTime.0, or any other string.
- hostname - the hostname, by default the value of sysName.0

to control the above fields. A sample exchange for these commands would be (eg. to create an identical message to that of Cisco IOS):

```
mimicsh> mimic agent protocol msg SYSLOG set server \{192.9.200.70 192.9.200.84\}

mimicsh> mimic agent protocol msg SYSLOG get server
192.9.200.70,514 192.9.200.84,514

mimicsh> mimic agent protocol msg SYSLOG set sequence 32

mimicsh> mimic agent protocol msg SYSLOG set separator \{: \}

mimicsh> mimic agent protocol msg SYSLOG set hostname \{\}

mimicsh> mimic agent protocol msg SYSLOG set timestamp sysUpTime

mimicsh> mimic agent protocol msg SYSLOG send 189 %SYS-5-CONFIG_I: Configured from console
by vty0 (192.9.200.76)

mimicsh> mimic protocol msg SYSLOG get stats_hdr
{{pktSnt} {PktsSent}} {{pktDisc} {PktsDiscarded}}

mimicsh> mimic agent protocol msg SYSLOG get statistics
1 0

mimicsh> mimic agent protocol msg SYSLOG send 187 "% LINK-5-CHANGED: Interface Ethernet2/1,
changed state to administratively down"

mimicsh> mimic agent protocol msg SYSLOG get statistics
2 0 0 0 0 0 0
```

Compatibility

The Unix syslog servers work out of the box with the MIMIC SYSLOG module, provided that the syslog server is setup to receive network messages.

This section details further interoperability with some widely available Windows syslog servers.

[Tftpd32](#) [Kiwi Syslog](#) [WinSyslog](#)

[Tftpd32](#)

Download Url	Version	Size
--------------	---------	------

http://tftpd32.jounin.net/	2.80	173 Kb
---	------	--------

Installation and usage

1. Extract the tftpd32.280.zip file.
2. Run tftpd32.exe from the Extracted Location.
3. Click settings, Change the base directory to a location of your choice.
4. Check enable the "Save syslog messages" if you wish to save the syslog messages to a file.
5. Click Ok, to confirm changes.
6. Click "Syslog server" tab.

Uninstallation

1. Execute the uninst.exe file.

[Kiwi Syslog](#)

Download Url	Version	Size
http://www.kiwisyslog.com/	7.1.4	4.36 Mb, 4.56 Mb

Kiwi Syslog daemon comes in two versions : Standard Version and Service Version. The Kiwi_Syslogd.exe i.e. Standard version runs as a daemon while Kiwi_Syslogd_Service.exe i.e. Service version can be installed to run the application as a windows service.

Installation and usage

1. To install the Kiwi Syslog execute the Kiwi_Syslogd.exe Or Kiwi_Syslogd_Service.exe.
2. Kiwi can now log messages, but if you wish to configure advanced options follow steps 2 through 5 else you can continue with instruction for Vlab.
3. To Change the files location where syslog messages will be saved, click File->Setup.
4. Specify whether you want the messages to be saved to file under the Rules->Default->Action Tree.
5. You can also specify TCP, UDP, SNMP logging under the inputs tree.
6. Press Ok to confirm changes.

Uninstallation

1. In the Control Panel->Add Remove Programs
2. Select Kiwi Syslog Deamon->press Remove
3. Follow instructions if any to remove the program

[WinSyslog](#)

Download Url	Version	Size
http://www.winsyslog.com/en/Download/	6	10.7 Mb

Installation and usage

1. To install the WinSyslog execute the wnsyslog6.exe and follow the instructions.
2. Execute the WINSyslogClient.exe client application from the location where you have installed it.
3. Execute the interactive syslog server (InteractiveSyslogServer.exe)
4. Click the start logging button to log syslog messages.

Uninstallation

1. In the Control Panel->Add Remove Programs
2. Select WinSysLog->press Remove
3. Follow instructions if any to remove the program

<< [Previous Section](#) [Next Section](#) >>



MIMIC IPMI Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using IPMI from MIMICView](#)
- [Using IPMI from MIMICShell](#)
- [Message Dictionary](#)
- [IPMI Proxy](#)
- [IPMI Recorder](#)
- [Interoperability](#)
- [Limitations](#)

Overview

The MIMIC IPMI Protocol Module is an optional facility that enables the Intelligent Platform Management Interface (IPMI) Remote Management Control Protocol (RMCP) (full specifications at the official [website](#)). The MIMIC IPMI module simulates the LAN interface and currently implements both the IPMI v1.5 and 2.0 specifications.

Installation

IPMI support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 10.00, you can use the [Protocol Wizard](#) to install the IPMI module (select `IPMI Simulator`). If you prefer to enable IPMI by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the IPMI shared library (`ipmi.dll` on Windows, `ipmi.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the IPMI module was properly loaded :
INFO - IPMI : Loaded protocol from < path-to-DLL >
INFO - IPMI v8.00

Once IPMI is loaded, any agent instance configured to support the IPMI protocol will be able to accept IPMI requests from an IPMI application on the "IPMI over LAN" interface.

Using IPMI from MIMICView

If the IPMI module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display IPMI as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new `IPMI` pane will appear.

This IPMI configuration pane lets the user configure the parameters for a IPMI session:

- **Primary Port**
This optional parameter specifies an alternate IPMI primary port. The default is the standard port 623.
- **Secure Port**
This optional parameter specifies an alternate secure port to use. The default is the standard port 664.
- **Version**
This optional parameter specifies the version of the protocol to use, 2.0 or 1.5. The default is 2.0.

Using IPMI from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the IPMI functionality. Here is a synopsis:

- **mimic protocol msg IPMI get args**
This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg IPMI get args
{{primary_port}} {Primary Port} {integer} {} {optional} {623}}
{{secure_port}} {Secure Port} {integer} {} {optional} {664}}
```

```
{{version} {Version} {string} {} {optional} {2.0}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,IPMI
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,IPMI
mimicsh> mimic agent get protocol
snmpv1,IPMI
```

- **mimic agent protocol msg IPMI get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg IPMI get config
{primary_port=623} {secure_port=664} {version=2.0}
```

- **mimic agent protocol msg IPMI set config [config]**

This command lets the user change the current argument settings of all IPMI sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg IPMI get config
{primary_port=623} {secure_port=664} {version=2.0}

mimicsh> mimic agent protocol msg IPMI set config primary_port=2623

mimicsh> mimic agent protocol msg IPMI get config
{primary_port=2623} {secure_port=664} {version=2.0}
```

- **mimic agent protocol msg IPMI get trace**

- **mimic agent protocol msg IPMI set trace [0 or 1]**

This command lets the user change the IPMI tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 9

mimicsh> mimic agent protocol msg IPMI get trace
0
mimicsh> mimic agent protocol msg IPMI set trace 1

mimicsh> mimic agent protocol msg IPMI get trace
1
```

and the log would show:

```
INFO - agent 9 trace enabled for IPMI
INFO - agent 9 decoded IPMI request
INFO - Message 56, Get Channel Authentication Capabilities
INFO - SID 0x0 = 0
INFO - Field 1 "Channel Number", len 1-1
INFO - Value: \x8e
INFO - 1..... = "get IPMI v2.0+ extended data"
INFO - ...1110 = "retrieve information for channel this request was issued on"
INFO - Field 2 "Requested Maximum Privilege Level", len 1-1
INFO - Value: \x04 = "Administrator level"
INFO - agent 9 simulated IPMI response
INFO - Message 56, Get Channel Authentication Capabilities
INFO - SID 0x0 = 0
INFO - Field 1 "Completion Code", len 1-1
INFO - Value: \x00 = "UNKNOWN"
INFO - Field 2 "Channel Number", len 1-1
INFO - Value: \x01
INFO - Field 3 "Authentication Type Support", len 1-1
INFO - Value: \x97
INFO - 1..... = "IPMI v2.0+ extended capabilities"
INFO - .0.....
INFO - ..0.....
INFO - ...1.... = "straight password / key"
INFO - ...0...
INFO - .....1.. = "MD5"
INFO - .....1. = "MD2"
INFO - .....1 = "none"
INFO - Field 4 "Authentication Type Support", len 1-1
INFO - Value: \x04
INFO - 0.....
INFO - .0.....
INFO - ..0..... = "KG is set to default (all 0's)"
INFO - ...0....
INFO - ....0...
INFO - .....1.. = "Non-null usernames enabled"
INFO - .....0.
```

```

INFO - .....0
INFO - Field 5 "Extended Capabilities", len 1-1
INFO - Value: \x0a
INFO - .....1. = "channel supports IPMI v2.0 connections"
INFO - .....0
INFO - Field 6 "OEM ID", len 3-3
INFO - Value: \x00 00 00
INFO - Field 7 "OEM auxiliary data", len 1-1
INFO - Value: \x00
...

```

- **mimic protocol msg IPMI get stats_hdr**
mimic agent protocol msg IPMI get statistics
Returns IPMI statistics information:

- a list of statistic headers, and
- current statistics values for the specified server.

In order, the statistic values are:

- Total number of connections established.
- Total number of disconnects.
- Total number of received commands.
- Total number of "too short" errors.
- Total number of "unrecognized" errors (IANA enterprise, message type, etc).
- Total number of invalid authentications.
- Total number of bad session IDs.
- Total number of authentication failures.
- Total number of SOL requests dropped (not implemented).
- Total number of OEM payloads dropped.
- Total number of bad sequence numbers.
- Total number of received responses which are dropped.
- Total number of dropped commands due to unknown NETFN.
- Total number of dropped commands due to unknown command.
- Total number of dropped commands due to bad syntax.
- Total number of dropped commands due to bad simulation.

A sample exchange for these commands would be:

```

mimicsh> mimic protocol msg IPMI get stats_hdr
{{connect} {Connect}} {{disconnect} {Disconnect}} {{received} {Received}}
{{tooshort} {TooShort}} {{unrecognized} {Unrecognized}} {{invalidauth} {InvalidAuth}}
{{badsessionid} {BadSessionID}} {{authfail} {AuthFail}} {{droppedsol} {SerialOverLAN}}
{{droppedoem} {OEMPayload}} {{badseq} {BadSequence}} {{droppedrsp} {Response}}
{{droppednetfn} {NETFN}} {{unknowncmd} {Unknown}} {{badparse} {BadParse}}
{{badsim} {SimError}}

mimicsh> mimic agent protocol msg IPMI get statistics
0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0

```

- **mimic agent protocol msg IPMI get attr**
mimic agent protocol msg IPMI set attr value
where attr is one of the following:

- working_authtype - the current authentication type
- session_id - the current session ID
- outbound_seq - the current outbound sequence number
- inbound_seq - the current inbound sequence number
- field N - the value of the N-th field in the request

These are useful in action scripts.

4. Message Dictionary

All IPMI messages known to MIMIC are defined in the message dictionary config/ipmi/messages.cfg in the uniform MIMIC configuration file syntax. For example, this file contains:

```

version = 8.00

global = {
}

# Chassis
include = {
    file = netfn_00.cfg
}

# App
include = {
    file = netfn_06.cfg
}

```



```

# Storage
include = {
    file = netfn_0a.cfg
}
and netfn_06.cfg contains
netfn = {
    id = 6
    number = \x06
    name = App
    reference = Intelligent Platform Management Interface Specification Second Generation v2.0
    Document Revision 1.0 February 12, 2004 May 5, 2005 Markup

    # Global Commands
    include = {
        file = msg_06_01.cfg
    }
    include = {
        file = msg_06_02.cfg
    }
    include = {
        file = msg_06_03.cfg
    }
    include = {
        file = msg_06_04.cfg
    }
    include = {
        file = msg_06_05.cfg
    }
    include = {
        file = msg_06_06.cfg
    }
    include = {
        file = msg_06_07.cfg
    }
    include = {
        file = msg_06_08.cfg
    }
}

# IPMI Messaging Support Commands
include = {
    file = msg_06_37.cfg
}
include = {
    file = msg_06_38.cfg
}
include = {
    file = msg_06_39.cfg
}
include = {
    file = msg_06_3a.cfg
}
...

```

The dictionary basically comprises a hierarchical tree of configuration files, one for each known message. In each message configuration file, the request and response blocks for the defined command list the fields they contain, including their types, values, etc. For example, the `msg_06_38.cfg` file contains:

```

message = {
    id = 56
    number = \x38
    name = Get Channel Authentication Capabilities
    reference = 22.13
    request = {
        field = {
            id = 1
            byte = 1
            name = Channel Number
            length = 1
            type = bits
            bit = {
                offset = 7
                on = get IPMI v2.0+ extended data
            }
            bit = {
                offset = 3-0
                enum = {
                    value = \xe
                    name = retrieve information for channel this request was
                    issued on
                }
            }
        }
        field = {
            id = 2
            byte = 2
            name = Requested Maximum Privilege Level
            length = 1
            type = enum
            enum = {
                value = 0
            }
        }
    }
}

```

```

        name = reserved
    }
    enum = {
        value = 1
        name = Callback level
    }
    enum = {
        value = 2
        name = User level
    }
    enum = {
        value = 3
        name = Operator level
    }
    enum = {
        value = 4
        name = Administrator level
    }
    enum = {
        value = 5
        name = OEM Proprietary level
    }
}
}

response = {
    field = {
        id = 1
        byte = 1
        name = Completion Code
        length = 1
        type = enum
    }
    field = {
        id = 2
        byte = 2
        name = Channel Number
        length = 1
        type = number
    }
    field = {
        id = 3
        byte = 3
        name = Authentication Type Support
        length = 1
        type = bits
        bit = {
            offset = 7
            enum = {
                value = 1
                name = IPMI v2.0+ extended capabilities
            }
            enum = {
                value = 0
                name = IPMI v1.5 support only
            }
        }
        bit = {
            offset = 6
            on = reserved
        }
        bit = {
            offset = 5
            on = OEM proprietary
        }
        bit = {
            offset = 4
            on = straight password / key
        }
        bit = {
            offset = 3
            on = reserved
        }
        bit = {
            offset = 2
            on = MD5
        }
        bit = {
            offset = 1
            on = MD2
        }
        bit = {
            offset = 0
            on = none
        }
    }
    field = {
        id = 4

```

```

        byte = 4
        name = Authentication Type Support
        length = 1
        type = bits
        bit = {
            offset = 7
            on = reserved
        }
        bit = {
            offset = 6
            on = reserved
        }
        bit = {
            offset = 5
            enum = {
                value = 1
                name = KG is set to non-zero value.
            }
            enum = {
                value = 0
                name = KG is set to default (all 0's)
            }
        }
        bit = {
            offset = 4
            on = Per-message Authentication is disabled
        }
        bit = {
            offset = 3
            on = User Level Authentication is disabled
        }
        bit = {
            offset = 2
            on = Non-null usernames enabled
        }
        bit = {
            offset = 1
            on = Null usernames enabled
        }
        bit = {
            offset = 0
            on = Anonymous Login enabled
        }
    }
    field = {
        id = 5
        byte = 5
        name = Extended Capabilities
        version = 2.0
        length = 1
        type = bits
        bit = {
            offset = 1
            on = channel supports IPMI v2.0 connections
        }
        bit = {
            offset = 0
            on = channel supports IPMI v1.5 connections
        }
    }
    field = {
        id = 6
        byte = 6
        name = OEM ID
        length = 3
        type = octets
    }
    field = {
        id = 7
        byte = 9
        name = OEM auxiliary data
        length = 1
        type = octets
    }
}
}
}

```

1. IPMI Proxy

The IPMI Proxy ([ipmiproxy](#)) is a stand-alone utility to record IPMI sessions and create basic IPMI simulations. The proxy sits between the IPMI management application and the target device, recording requests and responses into a walkfile, which can then be fed to the [IPMI Recorder](#) to create the simulation. To achieve the recording, the IPMI management applications needs to query the IPMI Proxy address instead of the target.

The following command line options are supported:

- `--target ip-address`

mandatory command line argument to specify the target device supporting IPMI commands.

- **--remote remote-port**

optional command line argument to specify the port number on the target device to send IPMI requests to. By default this will be the standard port 623.

- **--local local-port**

optional command line argument to specify the port number on proxy on which to accept IPMI requests from the management application. By default this will be the standard port 623. This option is specially useful if `ipmi-proxy` is run as non-root, because you will get a permission failure trying to open the default port 623.

- **--authcode authentication-code**

mandatory command line argument to specify the password to access the target device.

- **--transform transform-file**

optional command line argument to specify a transform file, which will transform certain messages. This is useful to force IPMI v1.5 communication, when the device and/or application support either.

- **--walkfiledir directory**

optional command line argument to specify a different walkfile directory. The default is `/tmp`.

- **--walkfile filename**

optional command line argument to specify a different walkfile filename. The default is `ipmiwalk-IP-ADDRESS`.

- **--app-cipher cipher-suite**

the cipher suite to use to communicate with this proxy. Use `--app-cipher --help` to list the available ciphers.

- **--trg-cipher cipher-suite**

the cipher suite to use to communicate with the target. Use `--trg-cipher --help` to list the available ciphers.

- **--password**

the password to use (required)

- **--kgkey**

the kgkey to use (optional)

For example, assuming the target device is at `10.0.0.1`, the following are the command lines for the different cipher suites:

```
% ./ipmi-proxy --target 10.0.0.1 --local 6623 --app-cipher 0 --password admin123
```

```
% ./ipmi-proxy --target 10.0.0.1 --local 6623 --app-cipher 1 --password admin123
```

IPMI Recorder

The IPMI Recorder (`ipmirec`) is a stand-alone utility to record IPMI sessions and create basic IPMI simulations. The recorder queries the target device, recording requests and responses into a walkfile and creating the simulation.

The IPMI Recorder can run in either live or file mode. In live mode, the Recorder interacts with a running target device on your network in real time using the IPMI protocol. In the discovery phase, the Recorder queries the target device according to the configuration file `config/ipmi/discover.cfg`, and stores the requests and responses in a walkfile. In the second simulation phase, the Recorder creates the simulation from the discovered commands.

The file mode works by reading a walkfile created previously either by hand or from a previous run of the Recorder against a Live target device.

Once data is captured from a device, the MIMIC Agent Simulator can use the simulation generated by the IPMI Recorder to simulate the device realistically. Any variation on this basic simulation can later be done for use with the Simulator.

The following command line options are supported:

- **--target ip-address**

mandatory command line argument to specify the target device to query in "live mode" via the IPMI protocol.

- **--port remote-port**

optional command line argument to specify the port number on the target device to send IPMI requests to. By default this will be the standard port 623.

- **--file walkfile**

mandatory command line argument to specify the input walkfile for "file mode" described above.

- **--simulation simulation-name**

This specifies the simulation name of the existing SNMP simulation to store the IPMI simulation.

- `--scenario scenario-name`

This specifies a scenario name. The SNMP scenario with the same name has to already exist.

- `--walkfiledir directory`

optional command line argument to specify a different walkfile directory. The default is `/tmp`.

- `--walkfile filename`

optional command line argument to specify a different walkfile filename. The default is `ipmiwalk-IP-ADDRESS`.

1. Interoperability

The IPMI modules has been tested for interoperability with [IPMITOOL 1.8.10](#). In particular, the following shows that RMCP+ cipher suites 0 through 3 interoperate:

```
% ./ipmitool -I lanplus -H 10.0.0.1 -v -C 0 -U admin -P admin123 -A password chassis status
System Power      : off
Power Overload    : false
Power Interlock   : inactive
Main Power Fault  : false
Power Control Fault : false
Power Restore Policy : always-off
Last Power Event  :
Chassis Intrusion : inactive
Front-Panel Lockout : inactive
Drive Fault       : false
Cooling/Fan Fault : false
Sleep Button Disable : not allowed
Diag Button Disable : allowed
Reset Button Disable : not allowed
Power Button Disable : allowed
Sleep Button Disabled: false
Diag Button Disabled : true
Reset Button Disabled: false
Power Button Disabled: true

% ./ipmitool -I lanplus -H 10.0.0.1 -v -C 1 -U admin -P admin123 -A password chassis status
System Power      : off
Power Overload    : false
Power Interlock   : inactive
Main Power Fault  : false
Power Control Fault : false
Power Restore Policy : always-off
Last Power Event  :
Chassis Intrusion : inactive
Front-Panel Lockout : inactive
Drive Fault       : false
Cooling/Fan Fault : false
Sleep Button Disable : not allowed
Diag Button Disable : allowed
Reset Button Disable : not allowed
Power Button Disable : allowed
Sleep Button Disabled: false
Diag Button Disabled : true
Reset Button Disabled: false
Power Button Disabled: true

% ./ipmitool -I lanplus -H 10.0.0.1 -v -C 2 -U admin -P admin123 -A password chassis status
System Power      : off
Power Overload    : false
Power Interlock   : inactive
Main Power Fault  : false
Power Control Fault : false
Power Restore Policy : always-off
Last Power Event  :
Chassis Intrusion : inactive
Front-Panel Lockout : inactive
Drive Fault       : false
Cooling/Fan Fault : false
Sleep Button Disable : not allowed
Diag Button Disable : allowed
Reset Button Disable : not allowed
Power Button Disable : allowed
Sleep Button Disabled: false
Diag Button Disabled : true
Reset Button Disabled: false
Power Button Disabled: true

% ./ipmitool -I lanplus -H 10.0.0.1 -v -C 3 -U admin -P admin123 -A password chassis status
System Power      : off
Power Overload    : false
```

```

Power Interlock      : inactive
Main Power Fault    : false
Power Control Fault  : false
Power Restore Policy : always-off
Last Power Event    :
Chassis Intrusion   : inactive
Front-Panel Lockout : inactive
Drive Fault         : false
Cooling/Fan Fault   : false
Sleep Button Disable : not allowed
Diag Button Disable : allowed
Reset Button Disable : not allowed
Power Button Disable : allowed
Sleep Button Disabled: false
Diag Button Disabled : true
Reset Button Disabled: false
Power Button Disabled: true

```

The IPMI modules has been tested for interoperability with [IPMIUTIL 2.91](#). In particular, the following shows that RMCP+ cipher suite 6 interoperates:

```

% ./ipmiutil sensor -N 10.0.0.1 -U admin -P admin123 -V 4 -T 2 -J 6
ipmiutil ver 2.91
isensor: version 2.91
Opening lanplus connection to node 10.0.0.1 ...
Connected to node 10.0.0.1

-- BMC version 2.10, IPMI version 2.0
_ID_ SDR_Type_xx ET Own Typ S_Num Sens_Description Hex & Interp Reading
0001 SDR Full 01 01 20 a 01 snum 01 Temp = 49 OK -55.00 degrees C
0002 SDR Full 01 01 20 a 01 snum 02 Temp = 4c OK -52.00 degrees C
0003 SDR Full 01 01 20 a 01 snum 05 Temp = a8 OK 40.00 degrees C
0004 SDR Full 01 01 20 a 01 snum 06 Temp = a8 OK 40.00 degrees C
0005 SDR Full 01 01 20 a 01 snum 08 Ambient Temp = 98 OK 24.00 degrees C
0006 SDR Comp 02 6f 20 a 29 snum 10 CMOS Battery = 0000 OK
0007 SDR Comp 02 6f 20 a 29 snum 11 ROMB Battery = 0000 OK
0008 SDR Comp 02 03 20 a 02 snum 12 VCORE = 0001 OK
0009 SDR Comp 02 03 20 a 02 snum 13 VCORE = 0001 OK
000a SDR Comp 02 03 20 a 02 snum 16 CPU VTT = 0001 OK
000b SDR Comp 02 03 20 a 02 snum 17 1.5V PG = 0001 OK
000c SDR Comp 02 03 20 a 02 snum 18 1.8V PG = 0001 OK
000d SDR Comp 02 03 20 a 02 snum 19 3.3V PG = 0001 OK
000e SDR Comp 02 03 20 a 02 snum 1a 5V PG = 0001 OK
000f SDR Comp 02 03 20 a 02 snum 1b 1.5V PXH PG = 0001 OK
0010 SDR Comp 02 03 20 a 02 snum 1c 5V Riser PG = 0001 OK
0011 SDR Comp 02 03 20 a 02 snum 1d Backplane PG = 0001 OK
0012 SDR Comp 02 03 20 a 02 snum 20 Linear PG = 0001 OK
0013 SDR Comp 02 03 20 a 02 snum 21 0.9V PG = 0001 OK
0014 SDR Comp 02 03 20 a 02 snum 22 0.9V Over Volt = 0001 OK
0015 SDR Comp 02 03 20 a 02 snum 23 CPU Power Fault = 0001 OK
0016 SDR Full 01 01 20 a 04 snum 30 FAN 1 RPM = 71 OK 8475.00 RPM
0017 SDR Full 01 01 20 a 04 snum 31 FAN 2 RPM = 6e OK 8250.00 RPM
0018 SDR Full 01 01 20 a 04 snum 32 FAN 3 RPM = 6f OK 8325.00 RPM
0019 SDR Full 01 01 20 a 04 snum 33 FAN 4 RPM = 6d OK 8175.00 RPM
001a SDR Full 01 01 20 a 04 snum 34 FAN 5 RPM = 38 OK 4200.00 RPM
001b SDR Full 01 01 20 a 04 snum 35 FAN 6 RPM = 38 OK 4200.00 RPM
001c SDR Comp 02 6f 20 a 25 snum 50 Presence = 0001 OK*
001d SDR Comp 02 6f 20 a 25 snum 51 Presence = 0001 OK*
001e SDR Comp 02 6f 20 a 25 snum 54 Presence = 0001 OK*
001f SDR Comp 02 6f 20 a 25 snum 55 Presence = 0001 OK*
0020 SDR Comp 02 6f 20 a 25 snum 56 Presence = 0001 OK*
0021 SDR Comp 02 6f 20 a 25 snum 57 Presence = 0001 OK*
0022 SDR Comp 02 6f 20 a 1b snum 59 DRAC5 Conn 2 Cbl = 0001 _
0023 SDR Comp 02 03 20 a 02 snum 5f PFault Fail Safe = 0000 Absent
0024 SDR Comp 02 6f 20 a 07 snum 60 Status = 0080 ProcPresent
0025 SDR Comp 02 6f 20 a 07 snum 61 Status = 0080 ProcPresent
0026 SDR Comp 02 6f 20 a 08 snum 64 Status = 0001 Present
0027 SDR Comp 02 6f 20 a 08 snum 65 Status = 0001 Present
0028 SDR Comp 02 6f 20 a 1b snum 66 Status = 0001 _
0029 SDR Comp 02 6f 20 a 15 snum 70 RAC Status = 0007 OK
002a SDR Comp 02 6f 20 a 23 snum 71 OS Watchdog = 0000 OK
002b SDR Comp 02 6f 20 a 10 snum 72 SEL = 0000 Absent
002c SDR Comp 02 6f 20 a 05 snum 73 Intrusion = 0000 OK
002d SDR Comp 02 0b 20 a 08 snum 74 PS Redundancy = 0001 Present
002e SDR Comp 02 0b 20 a 04 snum 75 Fan Redundancy = 0001 Redundant
002f SDR Comp 02 03 20 m 01 snum 76 CPU Temp Interf = 0040 NotAvailable
0030 SDR Comp 02 6f 20 a 0d snum 80 Drive = 0001 Unused Faulty
0031 SDR Comp 02 6f 20 a 1b snum 90 Cable SAS A = 0001 _
0032 SDR Comp 02 6f 20 a 1b snum 91 Cable SAS B = 0001 _
0033 SDR Full 01 01 20 a 03 snum 94 Current 1 = 00 Absent 0.00 na
0034 SDR Full 01 01 20 a 03 snum 95 Current 2 = 00 Absent 0.00 na
0035 SDR Full 01 01 20 a 02 snum 96 Voltage 1 = 00 Absent 0.00 na
0036 SDR Full 01 01 20 a 02 snum 97 Voltage 2 = 00 Absent 0.00 na
0037 SDR Full 01 01 20 a 03 snum 98 System Level = 00 Absent 0.00 na
0038 SDR Comp 02 6f 20 a c0 snum 99 Power Optimized = 0000 Absent
0039 SDR EntA 08 10 07 01 c0: 03 01 03 02 0a 01 0a 02
003a SDR IPMB 12 0e dev: 20 00 df 07 01 BMC
003b SDR IPMB 12 11 dev: 26 00 38 0b 01 DRAC 5
003c SDR FRU 11 17 dev: 20 00 80 00 07 01 System Board

```

```
003d SDR FRU 11 0f dev: 20 b0 02 00 03 01 CPU1
003e SDR FRU 11 0f dev: 20 b0 02 00 03 02 CPU2
003f SDR FRU 11 12 dev: 20 02 80 00 1a 03 Storage
0040 SDR FRU 11 0f dev: 20 03 80 00 0a 01 PS 1
0041 SDR FRU 11 0f dev: 20 04 80 00 0a 02 PS 2
0042 SDR FRU 11 12 dev: 20 05 80 00 1a 01 Storage
0043 SDR FRU 11 11 dev: 26 00 80 00 0b 01 DRAC 5
0044 SDR FRU 11 10 dev: 20 b0 02 00 10 01 Riser
0045 SDR FRU 11 14 dev: 20 b0 02 00 10 02 Sideplane
0046 SDR Comp 02 6f b1 a 0c snum 01 ECC Corr Err = 0000 Unknown
0047 SDR Comp 02 6f b1 a 0c snum 02 ECC Uncorr Err = 0000 Unknown
0048 SDR Comp 02 6f b1 a 13 snum 03 I/O Channel Chk = 0000 Unknown
ipmiutil sensor, completed successfully
```

The IPMI modules has been tested for interoperability with [FREEIPMI 1.6.3](#):

```
% ./ipmi-chassis -D LAN_2_0 -h 10.0.0.9 -I 0 -u "admin" -p "admin123" --get-chassis-status
System Power : on
Power overload : false
Interlock : inactive
Power fault : false
Power control fault : false
Power restore policy : Always off
Last Power Event : unknown
Chassis intrusion : inactive
Front panel lockout : inactive
Drive Fault : false
Cooling/fan fault : false
Power off button : enabled
Reset button : disabled
Diagnostic Interrupt button : disabled
Standby button : enabled
Power off button disable : allowed
Reset button disable : unallowed
Diagnostic interrupt button disable : allowed
Standby button disable : allowed
```

```
% ./ipmi-chassis -D LAN_2_0 -h 10.0.0.9 -I 1 -u "admin" -p "admin123" --get-chassis-status
System Power : on
Power overload : false
Interlock : inactive
Power fault : false
Power control fault : false
Power restore policy : Always off
Last Power Event : unknown
Chassis intrusion : inactive
Front panel lockout : inactive
Drive Fault : false
Cooling/fan fault : false
Power off button : enabled
Reset button : disabled
Diagnostic Interrupt button : disabled
Standby button : enabled
Power off button disable : allowed
Reset button disable : unallowed
Diagnostic interrupt button disable : allowed
Standby button disable : allowed
```

```
% ./ipmi-chassis -D LAN_2_0 -h 10.0.0.9 -I 2 -u "admin" -p "admin123" --get-chassis-status
System Power : on
Power overload : false
Interlock : inactive
Power fault : false
Power control fault : false
Power restore policy : Always off
Last Power Event : unknown
Chassis intrusion : inactive
Front panel lockout : inactive
Drive Fault : false
Cooling/fan fault : false
Power off button : enabled
Reset button : disabled
Diagnostic Interrupt button : disabled
Standby button : enabled
Power off button disable : allowed
Reset button disable : unallowed
Diagnostic interrupt button disable : allowed
Standby button disable : allowed
```

```
% ./ipmi-chassis -D LAN_2_0 -h 10.0.0.9 -I 3 -u "admin" -p "admin123" --get-chassis-status
System Power : on
Power overload : false
Interlock : inactive
Power fault : false
Power control fault : false
Power restore policy : Always off
Last Power Event : unknown
Chassis intrusion : inactive
```

```
Front panel lockout      : inactive
Drive Fault             : false
Cooling/fan fault       : false
Power off button        : enabled
Reset button            : disabled
Diagnostic Interrupt button : disabled
Standby button          : enabled
Power off button disable : allowed
Reset button disable    : unallowed
Diagnostic interrupt button disable : allowed
Standby button disable  : allowed

% ./ipmi-chassis -D LAN_2_0 -h 10.0.0.9 -I 6 -u "admin" -p "admin123" --get-chassis-status
System Power            : on
Power overload          : false
Interlock               : inactive
Power fault             : false
Power control fault     : false
Power restore policy    : Always off
Last Power Event        : unknown
Chassis intrusion       : inactive
Front panel lockout     : inactive
Drive Fault             : false
Cooling/fan fault       : false
Power off button        : enabled
Reset button            : disabled
Diagnostic Interrupt button : disabled
Standby button          : enabled
Power off button disable : allowed
Reset button disable    : unallowed
Diagnostic interrupt button disable : allowed
Standby button disable  : allowed
```

1. Limitations

The current version has these limitations:

- cipher suites 0,1,2,3,6 (section 22.15.2) are supported; others will be added in the future.

[<< Previous Section](#) [Next Section >>](#)



MIMIC Server Proxy Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using PROXY from MIMICView](#)
- [Using PROXY from MIMICShell](#)
- [Compatibility](#)

Overview

The MIMIC Server Proxy (PROXY) Protocol Module is an optional facility that enables simulating any number of existing UDP- or TCP-based servers. The module acts as a reverse proxy for any number of existing services for a specific MIMIC agent. Thus, a MIMIC agent can simulate any existing UDP- or TCP-based service. The PROXY module in effect multiplies an existing service by the number of agents that run it.

Installation

Server Proxy (PROXY) support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 10.00, you can use the [Protocol Wizard](#) to install the PROXY module (select `Server Simulator`). If you prefer to enable PROXY by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the PROXY library (`proxy.dll` on Windows, `proxy.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the PROXY module was properly loaded :

```
INFO - PROXY : Loaded protocol from < path-to-DLL >
INFO - PROXY v10.00 : Individual license #2345
```

Once PROXY is loaded, any agent instance configured to support the PROXY protocol will be able to act as a reverse proxy to an existing server.

Using PROXY from MIMICView

If the PROXY module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display PROXY as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new `PROXY` pane will appear.

This PROXY configuration pane lets the user configure the parameters for a PROXY retrieval:

- **Port**

This mandatory parameter specifies the port number at which the service can be accessed for this agent. All other configurables on the PROXY pane apply to this port only. [MIMICShell commands](#) can be used to add and configure additional PROXY ports on an agent.
- **Target**

This mandatory parameter specifies the comma-separated target server IP address and port. The IP address can be input either as "dot-value" notation (e.g., 192.9.200.1), or as a hostname (e.g., gambit), or fully qualified domainname (e.g., gambit.gambitcomm.com) provided that it can be resolved to an address (via `/etc/hosts`, Yellow Pages or DNS). The port is the numeric target server port. For example:

 - a standard Telnet server at address 192.9.200.1 would be specified as `192.9.200.1,23`
 - a standard SSH server at address 192.9.200.2 would be specified as `192.9.200.2,22`
 - a standard TFTP server at address 192.9.200.3 would be specified as `192.9.200.3,69`
 - a standard HTTP server at address 192.9.200.4 would be specified as `192.9.200.4,80`
- **Transport**

This optional parameter specifies the transport protocol for PROXY ports on the agent. The transport protocol must be either "TCP" or "UDP". If left blank, TCP is assumed.

Configure the Transport parameter to TCP if you intend to proxy TCP-based protocols such as TELNET, SSH or DNS. Configure the Transport parameter to

UDP if you intend to proxy UDP-based protocols such as DNS.

The configured transport protocol applies to all PROXY ports on the agent: the PROXY ports on the agent must be all TCP ports or all UDP ports, a mix of TCP and UDP PROXY ports cannot be configured on a single agent.

- **Maximum Connections**

For TCP PROXY ports, this optional integer parameter specifies the maximum number of simultaneous client connections that may be opened on the port. Additional connections beyond this limit will be refused.

For UDP PROXY ports, this parameter specifies the maximum number of UDP client requests that can be served simultaneously by a single port on the agent. Additional simultaneous client requests beyond this limit can cause responses to be dropped to the requesting client or to other clients using the port.

This parameter must be configured to a positive integer value. If left blank, it defaults to 10.

- **TCP_NODELAY**

This optional integer parameter specifies whether TCP_NODELAY (Nagel buffering) is applied to TCP connections to the target server. Legal values are 0 for no and 1 for yes.

- **Disconnect Delay (msec)**

Delay in milliseconds between client or target server disconnect and PROXY connection close. This configuration is used to address issues with certain TCP protocols, and can affect PROXY connection performance. Do not specify unless instructed to do so by Gambit Support.

- **Pre-Connect Script**

File name of PROXY pre-connect action script for port. If set, the specified action script is called after each client connection to the proxy port, before the target server connection is established. The action script is called with the following global variables as input:

- gCurrentAgent - Agent number
- gProxyPort - PROXY port number
- gFromAddress - Client IP address
- gFromPort - Client port number

The pre-connect action script may be used to reconfigure PROXY connection configurations, for example, to reset the target server address or the TCP_NODELAY setting on the target server connection.

If the pre-connect action script is not specified, the PROXY port configurations are used to configure the connection.

- **Client-to-Server Script**

File name of an optional PROXY client-to-server data action script. Do not specify unless instructed to do so by Gambit support.

- **Server-to-Client Script**

File name of an optional PROXY server-to-client data action script. Do not specify unless instructed to do so by Gambit support.

If the mandatory parameters are supplied, the agent will automatically serve as a reverse proxy to the target server upon starting.

Using PROXY from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to use the PROXY functionality. Here is a synopsis:

- **mimic protocol msg PROXY get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg PROXY get args
{{portno} {Port} {integer} {} {mandatory} {}}
{{target} {Target} {string} {} {mandatory} {}}
{{transport} {Transport (TCP or UDP)} {string} {} {optional} {TCP}}
{{max_connects} {Maximum Connections} {integer} {} {optional} {100}}
{{TCP_NODELAY} {TCP_NODELAY} {integer} {} {optional} {1}}
{{disconnect_delay} {Disconnect Delay (msec)} {integer} {} {optional} {0}}
{{pre_connect} {Pre-connect Script} {string} {} {optional} {}}
{{client_to_server} {Client-to-Server Script} {string} {} {optional} {}}
{{server_to_client} {Server-to-Client Script} {string} {} {optional} {}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent set protocol snmpv1,PROXY
```

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,PROXY
```

- **mimic agent protocol msg PROXY state**

This command lets the user query the state of the PROXY module. This is particularly useful at agent startup time to wait for PROXY startup. A sample usage for this command would be:

```
if { [mimic agent protocol msg PROXY state] != "up" } {
    # do necessary to wait until PROXY server is up
}
```

- **mimic agent protocol msg PROXY get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg PROXY get config
{portno=9922} {target=127.0.0.1,9923} {transport=TCP} {max_connects=100} {TCP_NODELAY=1}
{disconnect_delay=0} {pre_connect=} {client_to_server=} {server_to_client=}
```

- **mimic agent protocol msg PROXY set config config**

This command lets the user change the current argument settings of all PROXY sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg PROXY set config TCP_NODELAY=1
```

- **mimic agent protocol msg PROXY get trace**

- **mimic agent protocol msg PROXY set trace [0 or 1]**

This command lets the user change the PROXY tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 9
```

```
mimicsh> mimic agent protocol msg PROXY get trace
```

```
0
```

```
mimicsh> mimic agent protocol msg PROXY set trace 1
```

```
mimicsh> mimic agent protocol msg PROXY get trace
```

```
1
```

and the log would show:

```
INFO - PROXY [AGT=9]: PROXY server started
INFO - agent 9 trace enabled for PROXY
INFO - PROXY [AGT=9]: socket 4: connect at 10.0.0.9,2323 from 10.0.0.9,43292
INFO - PROXY [AGT=9]: port 2323: socket 5: receiving 3 data bytes from server at 10.0.0.9,60752 in pkt #1
INFO - FF FD 25 ..%
INFO - PROXY [AGT=9]: port 2323: socket 4: sending 3 data bytes to client at 10.0.0.9,2323 in pkt #2
INFO - FF FD 25 ..%
...
```

- **mimic protocol msg PROXY get stats_hdr**

- **mimic agent protocol msg PROXY get statistics**

Returns PROXY statistics information:

- a list of statistic headers, and
- current statistics values for the specified proxy.

In order, the statistic values are:

- Number of connections established
- Number of connect errors
- Number of disconnects
- Number of packets from client to server
- Number of bytes from client to server
- Number of packets from server to client
- Number of bytes from server to client

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg PROXY get stats_hdr
{{connect} {connect}} {{connectError} {connectError}} {{disconnect} {disconnect}}
{{pktsTo} {pktsTo}} {{bytesTo} {bytesTo}} {{pktsFrom} {pktsFrom}} {{bytesFrom} {bytesFrom}}
```

```
mimicsh> mimic agent protocol msg PROXY get statistics
1 0 0 24 202 26 1009
```

- **mimic agent protocol msg PROXY port add portno,target**

This command adds a proxy port with port number portno to the proxy server. This proxy port serves as a proxy to a target server at address target. target should be of the form IP address,port number. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg proxy port add 16001,192.9.200.71,23
```

- **mimic agent protocol msg PROXY port remove portno**

This command removes a proxy port with port number portno. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg proxy port remove 16001
```

- **mimic agent protocol msg PROXY port list**

This command returns a list of port numbers for configured proxy ports. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg proxy port list
{9923}
mimicsh> mimic agent protocol msg proxy port add 16001,192.9.200.71,23
mimicsh> mimic agent protocol msg proxy port add 16002,192.9.200.71,23
mimicsh> mimic agent protocol msg proxy port list
{9923 16001 16002}
```

- **mimic agent protocol msg PROXY port start portno**

mimic agent protocol msg PROXY port stop portno

mimic agent protocol msg PROXY port is_started portno

These commands start/stop and check started status of a proxy port with port number `portno`. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg proxy port is_started 16001
0
mimicsh> mimic agent protocol msg proxy port start 16001
mimicsh> mimic agent protocol msg proxy port is_started 16001
1
mimicsh> mimic agent protocol msg proxy port stop 16001
mimicsh> mimic agent protocol msg proxy port is_started 16001
0
```

- **mimic agent protocol msg PROXY port set config portno config=value**

mimic agent protocol msg PROXY port get config portno config

These commands set/get the configurable `config` for the port with port number `portno`. `config` can be one of

- **target**

target server address - IP address, portno

- **max_connects**

maximum number of proxy connections on port - number > 0. For UDP servers, this is the maximum number of simultaneous requests it will handle before some are possibly discarded.

- **TCP_NODELAY**

set TCP_NODELAY on proxy connections - 0 (false) or 1 (true)

- **disconnect_delay**

PROXY connection disconnect delay in milliseconds

- **pre_connect**

file name of PROXY pre-connect action script

- **client_to_server**

file name of PROXY client-to-server action script

- **server_to_client**

file name of PROXY server-to-client action script

A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg proxy port get config 16000 target
192.9.200.71,23
mimicsh> mimic agent protocol msg proxy port set config 16000 target=192.9.200.27,23
mimicsh> mimic agent protocol msg proxy port get config 16000 target
192.9.200.27,23
```

Compatibility

Click [here](#) for the compatibility document. If you get an error, you need to download the optional update package with the [Update Wizard](#).

<< [Previous Section](#) [Next Section](#) >>



MIMIC NETFLOW Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using NETFLOW from MIMICView](#)
- [Using NETFLOW from MIMICShell](#)
- [Recording NetFlow](#)
- [Simulation Configuration](#)
- [Compatibility](#)

1. Overview

The MIMIC NETFLOW Protocol Module is an optional facility that simulates the de-facto standard Cisco NetFlow service as detailed in [this Cisco whitepaper](#) , in [RFC 3954](#) and [RFC 5101](#) over the UDP transport protocol. Currently, the [SCTP Transport](#) and [DTLS Security](#) are not implemented.

Since it is completely configurable, MIMIC supports

- IPFIX Biflows as specified in [RFC 5103](#) via the 29305 Private Enterprise Number and the `biflowDirection` element.
- IPFIX Testing as specified in [RFC 5471](#) is mostly supported (currently 25 out of 32 tests). See [this doc](#) for details.
- IPFIX Packet Sampling (PSAMP) as specified in [RFC 5473](#) via the Common Properties Options Template Record and `commonPropertiesID` scoped element.
- the information elements in the PSAMP Information Model specified in [RFC 5477](#) numbered 301 through 311 and 313 through 338 are pre-defined in MIMIC.
- Network Secure Event Logging (NSEL) for [Adaptive Security Appliance \(ASA\)](#)
- Cisco Application Visibility and Control (AVC) NBAR, its [fields](#), [metrics](#) and [monitoring](#)
- [Performance Routing \(PfRv3\) Reporting](#)
- IPFIX Information Elements for Logging NAT Events [RFC 8158](#)
- Juniper [JFlow](#)

2. Installation

NetFlow support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 11.00, you can use the [Protocol Wizard](#) to install the NETFLOW module. If you prefer to enable NETFLOW by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the NETFLOW shared library (`netflow.dll` on Windows, `netflow.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the NETFLOW module was properly loaded :
INFO - NETFLOW : Loaded protocol from < path-to-DLL >
INFO - NETFLOW v11.00

Once NETFLOW is loaded, any agent instance configured to support the NetFlow services will be able to send NetFlow data to a NetFlow collector.

3. Using NETFLOW from MIMICView

If the NETFLOW module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display NETFLOW as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new NETFLOW pane will appear.

This NETFLOW configuration pane lets the user configure the parameters for a NETFLOW session:

- `Config file`
This mandatory parameter specifies the NETFLOW configuration file(s), each determining what NetFlow data is generated by the corresponding flow source. Multiple configuration files can be concatenated to produce flows from multiple flow sources at this IP address, eg. simulating multiple flow generators behind a NAT firewall. You will not be able to start the agent unless this parameter is set.

The configuration file is detailed [below](#). You can either edit configuration files directly, or use the [NetFlow Wizard](#).

- **Collector**
This optional parameter specifies the address of the collector. Both IPv4 and IPv6 addresses are supported. If a IPv6 collector is specified, the agent needs to have at least one [IPv6 alias](#) configured. If there is not at least one collector defined, then no flows are exported.
- **Collector Port**
This optional parameter specifies the collector port to use. The default port is 9999.
- **Bundle Flowsets**
This optional boolean parameter lets you bundle template and data flowsets into packets. The default is "0", which means to send template and data flowsets in separate packets.

Using NETFLOW from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the NETFLOW functionality. Here is a synopsis:

- **mimic protocol msg NETFLOW get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg NETFLOW get args
{{filename} {Config File} {file} {scripts/netflow {{*.cfg {Netflow config files}
{edit yes} {new yes}}} - both} {mandatory} {}}
{{collector} {Collector} {string} {} {mandatory} {}} {{collectorport} {Collector Port}
{string} {} {mandatory} {9999}}
{{bundleflowsets} {Bundle flowsets} {boolean} {0} {optional} {0}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,NETFLOW
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,NETFLOW
mimicsh> mimic agent get protocol
snmpv1,NETFLOW
```

- **mimic agent protocol msg NETFLOW get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg NETFLOW get config
{filename=} {collector=} {collectorport=9999} {bundleflowsets=0}
```

- **mimic agent protocol msg NETFLOW set config [config]**

This command lets the user change the current argument settings of all NETFLOW sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg NETFLOW get config
{filename=} {collector=} {collectorport=9999} {bundleflowsets=0}

mimicsh> mimic agent protocol msg NETFLOW set config \{collector=192.9.200.71 192.9.200.72\}

mimicsh> mimic agent protocol msg NETFLOW get config
{filename=} {collector=192.9.200.71 192.9.200.72} {collectorport=9999} {bundleflowsets=0}

mimicsh> mimic agent protocol msg NETFLOW set config \{filename=file1.cfg file2.cfg\}

mimicsh> mimic agent protocol msg NETFLOW get config
{filename=file1.cfg file2.cfg} {collector=192.9.200.71 192.9.200.72} {collectorport=9999}
{bundleflowsets=0}
```

- **mimic agent protocol msg NETFLOW get trace**

- **mimic agent protocol msg NETFLOW set trace [0 or 1]**

This command lets the user change the NETFLOW tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 1

mimicsh> mimic agent protocol msg NETFLOW get trace
0
mimicsh> mimic agent protocol msg NETFLOW set trace 1

mimicsh> mimic agent protocol msg NETFLOW get trace
1
```

and the log would show:

```
INFO 01/23.11:45:35 - agent 1 trace enabled for NETFLOW
INFO 01/23.11:45:35 - NETFLOW[AGT=1]: sent to [192.9.200.71,9999] - v9 bundled flowsets
```

```
INFO 01/23.11:45:35 - NETFLOW[AGT=1]: sent to [192.9.200.71,9999] - V9 bundled flowsets
...
```

- **mimic protocol msg NETFLOW get stats_hdr**
mimic agent protocol msg NETFLOW get statistics
Returns NETFLOW statistics information:

- a list of statistic headers, and
- current statistics values for the specified server.

In order, the statistic values are:

- Total number of NETFLOW packets sent.
- Total number of NETFLOW packets received.
- Total number of NETFLOW packets discarded.
- Total number of NETFLOW template flowsets sent.
- Total number of NETFLOW data flowsets sent.
- Total number of NETFLOW data flows sent.

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg NETFLOW get stats_hdr
{{pktSnt} {PktsSent}} {{pktRcvd} {PktsRcvd}} {{pktDisc} {PktsDiscarded}}
{{tflowsetSnt} {TempFlowsetSent}} {{dflowsetSnt} {DataFlowsetSent}}
{{dflowSnt} {DataFlowSent}}

mimicsh> mimic agent protocol msg NETFLOW get statistics
190 0 0 18 172 1720
```

- **mimic agent protocol msg NETFLOW halt**
mimic agent protocol msg NETFLOW set filename ...
mimic agent protocol msg NETFLOW set collector ...
mimic agent protocol msg NETFLOW reload
mimic agent protocol msg NETFLOW resume

This group of commands lets the user reload the configuration file for an agent without stopping it. This in effect allows dynamic reconfiguration of the flows generated by this exporter. The flow generation needs to first be halted, then the reload command reloads the configuration file, and the resume command continues flow generation.

The commands `mimic agent protocol msg NETFLOW set` allow to reconfigure certain attributes at runtime, such as the config file and/or collector(s).

- **mimic agent protocol msg NETFLOW flow list**
mimic agent protocol msg NETFLOW flow change ...

You can change flow generation parameters at runtime with this group of commands for an agent without stopping it.

The command `mimic agent protocol msg NETFLOW flow list` lists the flow generation parameters for a flow source config file, as shown in the `filename` configurable.

You can change the following global flow generation parameters:

- `tfs_interval` the template flowset interval in msec
- `dfs_interval` the data flowset interval in msec

A sample exchange for these commands would be:

```
mimicsh> mimic agent protocol msg NETFLOW flow list v9_simplest.cfg
{flow_source #0} {version 9} {count 1} {tfs_interval 10000}
{dfs_interval 1000} {num_flowsets 1}
{{{uid 0} {id 256} {tfs_length 88} {dfr_length 47} {dfs_length 476}
{dfs_count 3} {dfs_remaining 0} {dfs_num_rec 10} {dfs_pad 2} {seq 0}
{scope_count 0} {} {option_count 20}
{{field #0} {id 21} {length 4} {v_type CONSTANT} {value 3488219376}
{field #1} {id 22} {length 4} {v_type CONSTANT} {value 3488219284}
{field #2} {id 1} {length 4} {v_type SEQ} {min 2003} {max 10000}
...

mimicsh> mimic agent protocol msg NETFLOW flow change v9_simplest.cfg dfs_interval 2000

mimicsh> mimicsh> mimic agent protocol msg NETFLOW flow list v9_simplest.cfg
{flow_source #0} {version 9} {count 4294967238} {tfs_interval 10000}
{dfs_interval 2000} {num_flowsets 1} ...
```

You can change the field parameters with the command

```
mimic agent protocol msg NETFLOW flow change flowsrc-config-file flowset-uid field-num attr value
```

for the attribute `attr` with value `value` for field `field-num` in flowset `flowset-uid`.

For fields with `RANGE` or `SEQ` type simulation, the `min` and `max` attribute values can be changed. For fields with `CONSTANT` simulation, the `min` and `max` attributes are equivalent to `value`.

For example, for the flows above, this command would yield

```
mimicsh> mimic agent protocol msg NETFLOW flow change v9_simplest.cfg 0 0 min 57
```

```
mimicsh> mimic agent protocol msg NETFLOW flow list v9_simplest.cfg
{flow source #0} {version 9} {count 1} {tfs_interval 10000}
{dfs_interval 1000} {num_flowsets 1}
{{{uid 0} {id 256} {tfs_length 88} {dfr_length 47} {dfs_length 476}
{dfs_count 3} {dfs_remaining 0} {dfs_num_rec 10} {dfs_pad 2} {seq 0}
{scope_count 0} {} {option_count 20}
{{field #0} {id 21} {length 4} {v_type CONSTANT} {value 57}
{field #1} {id 22} {length 4} {v_type CONSTANT} {value 3488219284}
{field #2} {id 1} {length 4} {v_type SEQ} {min 2003} {max 10000}
...

```

Recording NetFlow

To create a default simulation, you can record from a NetFlow packet capture (PCAP) with the `netflowrec` utility. This tool work in either of 2 modes:

- in file mode, a previously captured PCAP file is read in
- in collector mode, it captures live NetFlow packets from an Exporter and saves them in a temporary PCAP file

It looks at NetFlow packets in the PCAP file, and for the first Exporter it finds, creates a simulation that attempts to cause the NetFlow module to generate equivalent flows. For each flow source (combination of UDP port and Source ID / Observation Domain ID) from the Exporter IP address a configuration file will be generated for all the flows from that source.

For better recording, here are the recommendations on how to capture Netflow packets with a packet capture program like [Wireshark](#) :

- in order to capture large, fragmented packets, it is better to filter by exporter or collector IP address than by collector port. If one does the latter, then the filter will discard subsequent fragments in fragmented IP packets, leading to missed Netflow packets in the capture.

By default, the recorder will attempt to detect client-server bi-directional flows, and will create a configuration with a limited number of flows to/from the clients to servers. Thus, to reproduce flows with more fidelity, it is better to feed a small number of flows at a time. The generated configurations can be loaded to produce the desired mix of flows in the simulation.

The [NetFlow Wizard](#) gives you a user-friendly interface in front of this tool.

This section documents the command-line options to this utility, either

- `--file pcap-file`
read NetFlow packets from the specified packet capture (PCAP) file.

or

- `--localport port`
specifying this option causes `netflowrec` to act as a Collector, reading packets from the specified port.
- `--count count`
optional argument to specify how many packets to collect. If not specified, then the collector will capture packets until interrupted (eg. with CTL-C).

and the common options

- `--out output-file`
this optional argument specifies the output file name. By default, the output file name will be the same as the PCAP file, but with the `.cfg` suffix.
- `--exporter host-address`
this optional argument specifies a host-address to filter the captured packets. Only packets from this exporter will be considered.
- `--collector host-address`
this optional argument specifies a host-address to filter the captured packets. Only packets from this collector will be considered.
- `--port port`
this optional argument specifies a port to filter the captured packets. Only packets to the specified collector port will be considered for simulation.
- `--version 5|9|10`
you can filter by NetFlow version number with this argument.
- `--start start`
- `--stop stop`
with these 2 options you can specify a range of packets to be recorded.
- `--maxenums number`
this forces a maximum number of enumerations to record, the default being 128.

The `netflowrec` tool has hardcoded information elements (fields) from [IANA](#) and uses the configuration file `scripts/netflow/netflow-fields.cfg` to reference vendor proprietary fields, and to override simulation behavior of internal fields.

For example, a section like


```
field = {
  name = VENDOR_PROP
  vendor_type = 148
  alias = NF_F_CONN_ID
  type = RANGE
}
```

defines an information element number 148 with the name `NF_F_CONN_ID` and will simulate it as an integer with the range detected in the PCAP.

The `netflow-fields.cfg` file can include other configuration files, eg.

```
includes = {
  include = fields/cisco-asa.cfg
}
```

will define the Cisco ASA NetFlow Secure Event Logging (NSEL) fields defined by this [Cisco](#) page.

Simulation Configuration

NetFlow data to be generated by the simulated exporter is specified by the [configuration file\(s\)](#) loaded into the agent instance. The [NetFlow Wizard](#) gives you a user-friendly interface to edit a NetFlow configuration file.

Common

This section documents the field definition parameters common to all versions of the NetFlow simulation.

For each field in the template, you can specify how to simulate instances of the field in successive flow records.

Parameter	Description
Field name	The name of the standard field, or a vendor proprietary field.
length	Number of bytes for this field.
type	The type of simulation: <ul style="list-style-type: none"> ○ CONSTANT - always return the specified value ○ RANGE - return a value in the range between min and max ○ SEQ - sequentially return a value from the specified range between min and max ○ ENUM - return an enumerated value ○ ACTION - return a value as computed by the specified action script ○ RANDOM - return a random value
select	How to select the next value if in a range or enumerated list. If specified as <code>SEQ</code> , then values are traversed sequentially either in the range or enumerated list, rather than randomly, which is the default.
reverse	Simulate bidirectional flow records. If this parameter is specified for any field in the record, then whenever a record is generated containing this field, a second record is automatically generated in the reverse direction. Both fields must be defined in the template. If the <code>reverse</code> parameter specifies a field name, then the reverse flow will reverse the values of both fields, maintaining the value of the other fields. If the <code>reverse</code> parameter specifies a percentage, then the reverse flow will contain a value with a multiple of the original flow, thus producing asymmetric flows.

Version 5

This section documents the **version 5** simulation configuration parameters.

- Global parameters
These apply to all the flowsets defined in this configuration file.

Configurable	Description
Comments	User editable comment about the configuration. This value does not impact the simulation and is solely intended for self-documentation.
version	NetFlow version, for version 5 see this Cisco whitepaper
sysUpTime	SysUptime of the first generated flowset. Is updated in real-time after that.
packet_count	
unix_secs	Time in seconds since 0000 UTC 1970, at which the Export Packet leaves the Exporter.
flow_sequence	Starting sequence number, incremented thereafter.

interval	Interval in milliseconds between data flowsets.
sim_count	If non-zero, specifies the number of iterations the simulation in this config file is run, else indefinitely.
num_rec	Number of flow records per flowset.

Version 9

This section documents the **version 9** simulation configuration parameters.

- Global parameters
These apply to all the flowsets defined in this configuration file.

Configurable	Description
Comments	User editable comment about the configuration. This value does not impact the simulation and is solely intended for self-documentation.
version	NetFlow version, for version 9 see RFC 3954
sysUpTime	SysUptime of the first generated flowset. Is updated in real-time after that.
UNIXSecs	Time in seconds since 0000 UTC 1970, at which the Export Packet leaves the Exporter. A value of 0 sets the current time.
SequenceNumber	Starting sequence number, incremented thereafter.
SourceID	A 32-bit value that identifies the Exporter Observation Domain. One configuration can generate flows from one SourceID.
tfs_interval	Interval in milliseconds between template flowsets.
dfs_interval	Interval in milliseconds between data flowsets.
sim_count	If non-zero, specifies the number of iterations the simulation in this config file is run, else indefinitely.

- Flowset configurables
These configurables apply to each flowset.

Parameter	Description
id	Template id for this flowset. Unique for this configuration file.
dfs_count	Number of flowsets for this flowset per iteration. If bundled, they will be in the same packet.
dfs_num_rec	Number of flow records per data flowset.

- Flowset definitions
This section defines the flow templates and how to simulate the fields.

Parameter	Description
id	Template id for this flowset. Unique for this configuration file.
scope_count	Number of scopes for this options template.
field_count	Number of fields in the flow.

- The field definitions are detailed in the common section [above](#).

Version 10

This section documents the **version 10** simulation configuration parameters.

- Global parameters
These apply to all the flowsets defined in this configuration file.

Configurable	Description
Comments	User editable comment about the configuration. This value does not impact the simulation and is solely intended for self-documentation.
version	NetFlow version, for version 10 see RFC 5101
	Time in seconds since 0000 UTC 1970, at which the Export Packet leaves the Exporter.

ExportTime	A value of 0 sets the current time.
SequenceNumber	Starting sequence number, incremented thereafter.
ObservationDomainID	A 32-bit value that identifies the Exporter Observation Domain. One configuration can generate flows from one ObservationDomainID.
tfs_interval	Interval in milliseconds between template flowsets.
dfs_interval	Interval in milliseconds between data flowsets.
sim_count	If non-zero, specifies the number of iterations the simulation in this config file is run, else indefinitely.

- The Flowset configurables and definitions are as in version 9 [above](#), the field definitions are in the common section [above](#).

Compatibility

Click [here](#) for the compatibility document. If you get an error, you need to download the optional update package with the [Update Wizard](#).

[<< Previous Section](#) [Next Section >>](#)



MIMIC SFLOW Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using SFLOW from MIMICView](#)
- [Using SFLOW from MIMICShell](#)
- [Recording sFlow](#)
- [Simulation Configuration](#)
- [Compatibility](#)

1. Overview

The MIMIC SFLOW Protocol Module is an optional facility that simulates the standard sFlow service as detailed at sFlow.org.

2. Installation

sFlow support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 12.00, you can use the [Protocol Wizard](#) to install the SFLOW module. If you prefer to enable SFLOW by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the SFLOW shared library (`sflow.dll` on Windows, `sflow.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the SFLOW module was properly loaded :
INFO - SFLOW : Loaded protocol from < path-to-DLL >
INFO - SFLOW v12.00

Once SFLOW is loaded, any agent instance configured to support the sFlow services will be able to send sFlow data to a sFlow collector.

3. Using SFLOW from MIMICView

If the SFLOW module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display SFLOW as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new `SFLOW` pane will appear.

This SFLOW configuration pane lets the user configure the parameters for a SFLOW session:

- **Config file**
This mandatory parameter specifies the SFLOW configuration file which determines what sFlow data is generated. You will not be able to start the agent unless this parameter is set.

The configuration file is detailed [below](#). You can either edit configuration files directly, or use the [sFlow Wizard](#).
- **Collector**
This optional parameter specifies the address of the collector. If there is not at least one collector defined, then no flows are exported. Multiple collectors can only be set through the scripting interface as shown below.
- **Collector Port**
This optional parameter determines the collector port to use. The default port is 6343.
- **Exports per minute**
This optional parameter specifies the frequency of samples generated.
- **Encoding (Compact/Expanded)**
This optional parameter controls the encoding of sample values.
- **Samples (Both/Flow/Counter)**
This optional parameter specifies whether to include Flow or Sample counters or Both (the default).
- **Records/Sample (All/Single/Random)**
This optional parameter controls the records per sample.
- **Samples/Datagram (All/Single/Random)**

This optional parameter allows you to specify the number of samples per datagram.

Using SFLOW from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the SFLOW functionality. Here is a synopsis:

- **mimic protocol msg SFLOW get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg SFLOW get args
{filename} {Config File} {file} {scripts/sflow {*.cfg {sflow config files}
{edit yes} {new yes}} - both} {mandatory} {}
{collector} {Collector} {string} {} {mandatory} {}
{collectorport} {Collector Port} {string} {} {mandatory} {6343}
{flows_per_min} {Exports per minute} {number} {} {optional} {6}
{encoding_type} {Encoding (Compact/Expanded)} {radio} {} {optional} {Compact}
{include_samples} {Samples (Both/Flow/Counter)} {string} {} {optional} {All}
{records_per_sample} {Records/Sample (All/Single/Random)} {string} {} {optional} {All}
{samples_per_datagram} {Samples/Datagram (All/Single/Random)} {string} {} {optional} {All}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,SFLOW
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,SFLOW
mimicsh> mimic agent get protocol
snmpv1,SFLOW
```

- **mimic agent protocol msg SFLOW get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg SFLOW get config
{filename=} {collector=} {collectorport=6343} {flows_per_min=6}
{encoding_type=Compact} {include_samples=All} {records_per_sample=All}
{samples_per_datagram=All}
```

- **mimic agent protocol msg SFLOW set config [config]**

This command lets the user change the current argument settings of all SFLOW sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg SFLOW get config
{filename=} {collector=} {collectorport=6343}

mimicsh> mimic agent protocol msg SFLOW set config \{collector=192.9.200.71 192.9.200.72\}

mimicsh> mimic agent protocol msg SFLOW get config
{filename=} {collector=192.9.200.71 192.9.200.72} {collectorport=6343}
```

- **mimic agent protocol msg SFLOW get trace**

- **mimic agent protocol msg SFLOW set trace [0 or 1]**

This command lets the user change the SFLOW tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 1

mimicsh> mimic agent protocol msg SFLOW get trace
0
mimicsh> mimic agent protocol msg SFLOW set trace 1

mimicsh> mimic agent protocol msg SFLOW get trace
1
```

and the log would show:

```
INFO 01/23.11:45:35 - agent 1 trace enabled for SFLOW
INFO 01/23.11:45:35 - SFLOW[AGT=1]: sent to [192.9.200.71,6343] - V5 datagram
...
```

- **mimic protocol msg SFLOW get stats_hdr**

- **mimic agent protocol msg SFLOW get statistics**

Returns SFLOW statistics information:

- a list of statistic headers, and
- current statistics values for the specified server.

In order, the statistic values are:

- Total number of SFLOW packets sent.
- Total number of SFLOW packets discarded.

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg SFLOW get stats_hdr
{{pktSnt} {PktsSent}} {{pktDisc} {PktsDiscarded}}

mimicsh> mimic agent protocol msg SFLOW get statistics
190 0
```

- **mimic agent protocol msg SFLOW halt**
- **mimic agent protocol msg SFLOW reload**
- **mimic agent protocol msg SFLOW resume**

This group of command lets the user reload the configuration file for an agent without stopping it. This in effect allows dynamic reconfiguration of the flows generated by this exporter. The flow generation needs to first be halted, then the reload command reloads the configuration file, and the resume command continues flow generation.

Recording sFlow

To create a default simulation, you can record from a sFlow packet capture (PCAP) with the `sflowrec` utility. This tool work in either of 2 modes:

- in file mode, a previously captured PCAP file is read in
- in collector mode, it captures live sFlow packets from an Exporter and saves them in a temporary PCAP file

It looks at sFlow packets in the PCAP file, and for the first Exporter it finds, creates a simulation that attempts to cause the sFlow module to generate equivalent flows. For each flow source (combination of UDP port and sub-agent ID) from the Exporter IP address a sample set will generated for all the flows from that source.

For better recording, here are the recommendations on how to capture sFlow packets with a packet capture program like [Wireshark](#) :

- in order to capture large, fragmented packets, it is better to filter by exporter or collector IP address than by collector port. If one does the latter, then the filter will discard subsequent fragments in fragmented IP packets, leading to missed sFlow packets in the capture.

The [sFlow Wizard](#) gives you a user-friendly interface in front of this tool.

This section documents the command-line options to this utility, either

- `--file pcap-file`
read sFlow packets from the specified packet capture (PCAP) file.

or

- `--localport port`
specifying this option causes `sflowrec` to act as a Collector, reading packets from the specified port.
- `--count count`
optional argument to specify how many packets to collect. If not specified, then the collector will capture packets until interrupted (eg. with CTL-C).

and the common options

- `--out output-file`
this optional argument specifies the output file name. By default, the output file name will be the same as the PCAP file, but with the `.cfg` suffix.
- `--exporter host-address`
this optional argument specifies a host-address to filter the captured packets. Only packets from this exporter will be considered.
- `--collector host-address`
this optional argument specifies a host-address to filter the captured packets. Only packets from this collector will be considered.
- `--port port`
this optional argument specifies a port to filter the captured packets. Only packets to the specified collector port will be considered for simulation.
- `--start start`
- `--stop stop`
with these 2 options you can specify a range of packets to be recorded.
- `--exclude templatefile`
by default, certain flows will be simulated using a template in `template/sflow/`. This option allows you to prevent that.
- `--sequential`
by default, integer fields found to be in a range of values will be simulated as returning random numbers in the range (via the `range.mtc1` action script). By specifying this option, you will force sequential values in the range (via the `seq.mtc1` action).

Simulation Configuration

sFlow data to be generated by the simulated exporter is specified by the [configuration file](#) loaded into the agent instance. The [sFlow Wizard](#) gives you a user-friendly interface to edit a sFlow configuration file.

- Global parameters
These apply to all the sample sets defined in this configuration file.

Configurable	Description
Comments	User editable comment about the configuration. This value does not impact the simulation and is solely intended for self-documentation.

- Includes
The files containing definitions of the sFlow structures used in this configuration have to be included in this section.
- Sample sets
Each configuration generates one or more sample sets as defined here.

Configurable	Description
sub_agent_id	Used to distinguishing between datagram streams from separate agent sub entities within an device.
sys_uptime_offset	Offset from system uptime for this sample set for the <code>uptime</code> field in the packet header.
frame_sequence	Incremented with each sample datagram generated by a sub-agent within an agent for the <code>sequence_number</code> field in the packet header.
flow_sequence	Incremented with each flow sample generated by this <code>source_id</code> for the <code>sequence_number</code> field in the flow sample.
counter_sequence	Incremented with each counter sample generated by this <code>source_id</code> for the <code>sequence_number</code> field in the counter sample.

A sample set has counters samples and / or flow samples.

Configurable	Description
data_source	sFlowDataSource - the encoding is as in the sFlow specification. The following formats are supported: <ul style="list-style-type: none">i, a positive integer - a combined value of datasource type and datasource id;t:i, positive integers - datasource type t and datasource id i;i1 i2 i3, positive integers - multiple combined datasource type and id;t1:i1 t2:i2 t3:i3, positive integers - multiple datasource type t* and id i*;action-script.mtcl - an action script that dynamically generates the datasources;*Entry - generates instances of the tabular *Entry, eg. ifEntry;
sample	One or more sample blocks with a struct defined in the <code>includes</code> section.

Compatibility

Click [here](#) for the compatibility document. If you get an error, you need to download the optional update package with the [Update Wizard](#).



MIMIC WEB Services Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using WEB from MIMICView](#)
- [Using WEB from MIMICShell](#)
- [Recording WEB Sessions](#)
- [Simulation Configuration](#)
- [Compatibility](#)

1. Overview

The MIMIC WEB Services Module is an optional facility that simulates the standard web services via [SOAP](#) , [XML](#) , [OpenAPI](#) , [REST](#) , [WBEM](#) , [WS-MAN](#) , [Redfish](#) and many other API types.

On Windows, it additionally supports [SPNEGO-based Kerberos and NTLM HTTP Authentication](#).

2. Installation

Web services support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 13.00, you can use the [Protocol Wizard](#) to install the WEB module. If you prefer to enable WEB by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the WEB shared library (`web.dll` on Windows, `web.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the WEB module was properly loaded :

```
INFO - WEB : Loaded protocol from < path-to-DLL >
INFO - WEB v13.00
```

Once WEB is loaded, any agent instance configured to support the web services will be able to handle web services requests.

3. Using WEB from MIMICView

If the WEB module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display WEB as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new `WEB` pane will appear.

This WEB configuration pane lets the user configure the parameters for a WEB session:

- **Port**
This optional parameter specifies the port at which the server will be listening. The default is the standard port 80.
- **Use persistent connections**
This specifies whether connections are persistent. Default is `true`. The vast majority of connections are persistent according to [Wikipedia](#) but for the rare server that isn't, you need to set this to `false`.
- **Rule File**
This optional parameter specifies the rules that govern the web sessions to this agent. The rule file is in `scripts/web/`, and in effect implements a pattern match against web requests. Each rule consists of a regular expression that is matched against the URL in the request, and a response that is replied. Regular expressions are matched in the order they appear in the file, thus more generic regular expressions should be placed after more specific regular expressions, else they will take precedence. A sample rule file is shipped as `sample.ru1`. Details can be found [below](#).
- **WSDL File**
This optional parameter specifies the WSDL syntax of requests to this web service. The compiled WSDL file is in `scripts/web/`.
- **Username**
The username for access control. Blank will allow all access.
- **Password**
The password for access control. Blank will allow all access.
- **SSL Certificate File**
A WEB protocol enabled agent, can support HTTP-Secured WEB communication using a Server Certificate. Select a certificate file to be used as Server Certificate

here. If none is specified, the `server.pem` shipped with MIMIC is used by default.

- **Chunking Threshold**

The HTTP Transfer-Encoding: header is used to indicate how the contents will be transferred. The "chunked" encoding is used to transfer large contents by splitting it into manageable chunks. A positive integer value will enable chunking for responses larger than the specified threshold. Chunking is disabled by default (indicated by value 0).

i. Using WEB from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the WEB functionality. Here is a synopsis:

- **mimic protocol msg WEB get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg WEB get args
{{port} {Port} {integer} {} {optional} {80}}
{{is_persistent_connections} {Use persistent connections} {boolean} {} {optional} {true}}
{{rule} {Rule File} {file} {scripts/web {{*.rul {web rule files} {edit yes} {new yes}}} -
both} {optional} {}}
{{wsdl} {WSDL File} {file} {scripts/web {{*.cwd {compiled WSDL files} {edit no} {new no}}} -
both} {optional} {}}
{{username} {Username(blank for all-access)} {string} {} {optional} {}}
{{password} {Password(blank for all-access)} {string} {} {optional} {}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,WEB
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,WEB
mimicsh> mimic agent get protocol
snmpv1,WEB
```

- **mimic agent protocol msg WEB get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg WEB get config
{port=80} {is_persistent_connections=true} {rule=hello.rul} {wsdl=AXLAPI.cwd}
{username=} {password=}
```

- **mimic agent protocol msg WEB set config [config]**

This command lets the user change the current argument settings of all WEB sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg WEB get config
{port=80} {rule=hello.rul} {wsdl=AXLAPI.cwd}

mimicsh> mimic agent protocol msg WEB set config rule=sample_cuom.rul

mimicsh> mimic agent protocol msg WEB get config
{port=80} {is_persistent_connections=true} {rule=sample_cuom.rul} {wsdl=AXLAPI.cwd}
{username=} {password=}
```

- **mimic agent protocol msg WEB get trace**

- **mimic agent protocol msg WEB set trace [0 or 1]**

This command lets the user change the WEB tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 2

mimicsh> mimic agent protocol msg WEB get trace
0
mimicsh> mimic agent protocol msg WEB set trace 1

mimicsh> mimic agent protocol msg WEB get trace
1
```

and the log would show:

```
INFO 01/23.11:45:35 - agent 2 trace enabled for WEB
INFO 09/13.13:32:28 - WEB[AGT=2,CON=4,REM=10.0.0.2:46845]: connection request received
INFO 09/13.13:32:28 - WEB[AGT=2,CON=4,REM=10.0.0.2:46845]: received 403 byte(s) data in pkt #1
INFO 09/13.13:32:28 - 47 45 54 20 2F 20 48 54 54 50 2F 31 GET / HTTP/1
INFO 09/13.13:32:28 - 2E 31 0D 0A 48 6F 73 74 3A 20 31 30 .1..Host: 10
INFO 09/13.13:32:28 - 2E 30 2E 30 2E 32 0D 0A 55 73 65 72 .0.0.2..User
INFO 09/13.13:32:28 - 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 -Agent: Mozi
INFO 09/13.13:32:28 - 6C 6C 61 2F 35 2E 30 20 28 58 31 31 11a/5.0 (X11
```

```

INFO 09/13.13:32:28 - 3B 20 55 3B 20 4C 69 6E 75 78 20 69 ; U; Linux i
INFO 09/13.13:32:28 - 36 38 36 3B 20 65 6E 2D 55 53 3B 20 686; en-US;
INFO 09/13.13:32:28 - 72 76 3A 31 2E 39 2E 32 2E 32 34 29 rv:1.9.2.24)
INFO 09/13.13:32:28 - 20 47 65 63 6B 6F 2F 32 30 31 31 31 Gecko/20111
INFO 09/13.13:32:28 - 31 30 38 20 46 65 64 6F 72 61 2F 33 108 Fedora/3
INFO 09/13.13:32:28 - 2E 36 2E 32 34 2D 31 2E 66 63 31 34 .6.24-1.fc14
INFO 09/13.13:32:28 - 20 46 69 72 65 66 6F 78 2F 33 2E 36 Firefox/3.6
INFO 09/13.13:32:28 - 2E 32 34 0D 0A 41 63 63 65 70 74 3A .24..Accept:
INFO 09/13.13:32:28 - 20 74 65 78 74 2F 68 74 6D 6C 2C 61 text/html,a
INFO 09/13.13:32:28 - 70 70 6C 69 63 61 74 69 6F 6E 2F 78 pplication/x
INFO 09/13.13:32:28 - 68 74 6D 6C 2B 78 6D 6C 2C 61 70 70 html+xml,app
INFO 09/13.13:32:28 - 6C 69 63 61 74 69 6F 6E 2F 78 6D 6C lication/xml
INFO 09/13.13:32:28 - 3B 71 3D 30 2E 39 2C 2A 2F 2A 3B 71 ;q=0.9,*/*;q
INFO 09/13.13:32:28 - 3D 30 2E 38 0D 0A 41 63 63 65 70 74 =0.8..Accept
INFO 09/13.13:32:28 - 2D 4C 61 6E 67 75 61 67 65 3A 20 65 -Language: e
INFO 09/13.13:32:28 - 6E 2D 75 73 2C 65 6E 3B 71 3D 30 2E n-us,en;q=0.
INFO 09/13.13:32:28 - 35 0D 0A 41 63 63 65 70 74 2D 45 6E 5..Accept-En
INFO 09/13.13:32:28 - 63 6F 64 69 6E 67 3A 20 67 7A 69 70 coding: gzip
INFO 09/13.13:32:28 - 2C 64 65 66 6C 61 74 65 0D 0A 41 63 ,deflate..Zip
INFO 09/13.13:32:28 - 63 65 70 74 2D 43 68 61 72 73 65 74 cept-Charset
INFO 09/13.13:32:28 - 3A 20 49 53 4F 2D 38 38 35 39 2D 31 : ISO-8859-1
INFO 09/13.13:32:28 - 2C 75 74 66 2D 38 3B 71 3D 30 2E 37 ,utf-8;q=0.7
INFO 09/13.13:32:28 - 2C 2A 3B 71 3D 30 2E 37 0D 0A 4B 65 ,*/*;q=0.7..Ke
INFO 09/13.13:32:28 - 65 70 2D 41 6C 69 76 65 3A 20 31 31 ep-Alive: 11
INFO 09/13.13:32:28 - 35 0D 0A 43 6F 6E 6E 65 63 74 69 6F 5..Connectio
INFO 09/13.13:32:28 - 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 n: keep-aliv
INFO 09/13.13:32:28 - 65 0D 0A 43 61 63 68 65 2D 43 6F 6E e..Cache-Con
INFO 09/13.13:32:28 - 74 72 6F 6C 3A 20 6D 61 78 2D 61 67 trol: max-ag
INFO 09/13.13:32:28 - 65 3D 30 0D 0A 0D 0A e=0....
INFO 09/13.13:32:28 - WEB[AGT=2,CON=4,REM=10.0.0.2:46845]: sending 267 byte(s) data in pkt #1
INFO 09/13.13:32:28 - 48 54 54 50 2F 31 2E 31 20 32 30 30 HTTP/1.1 200
INFO 09/13.13:32:28 - 20 4F 4B 0D 0A 43 61 63 68 65 2D 43 OK..Cache-C
INFO 09/13.13:32:28 - 6F 6E 74 72 6F 6C 3A 20 70 72 69 76 ontrol: priv
INFO 09/13.13:32:28 - 61 74 65 0D 0A 45 78 70 69 72 65 73 ate..Expires
INFO 09/13.13:32:28 - 3A 20 57 65 64 2C 20 33 31 20 44 65 : Wed, 31 De
INFO 09/13.13:32:28 - 63 20 31 39 36 39 20 31 39 3A 30 30 c 1969 19:00
INFO 09/13.13:32:28 - 3A 30 30 20 45 53 54 0D 0A 43 6F 6E :00 EST..Con
INFO 09/13.13:32:28 - 74 65 6E 74 2D 54 79 70 65 3A 20 74 tent-Type: t
INFO 09/13.13:32:28 - 65 78 74 2F 68 74 6D 6C 3B 63 68 61 ext/html;cha
INFO 09/13.13:32:28 - 72 73 65 74 3D 49 53 4F 2D 38 38 35 rset=ISO-885
INFO 09/13.13:32:28 - 39 2D 31 0D 0A 43 6F 6E 74 65 6E 74 9-1..Content
INFO 09/13.13:32:28 - 2D 4C 65 6E 67 74 68 3A 20 35 35 0D -Length: 55.
INFO 09/13.13:32:28 - 0A 44 61 74 65 3A 20 57 65 64 2C 20 .Date: Wed,
INFO 09/13.13:32:28 - 32 32 20 4D 61 79 20 32 30 31 33 20 22 May 2013
INFO 09/13.13:32:28 - 32 30 3A 31 38 3A 35 30 20 47 4D 54 20:18:50 GMT
INFO 09/13.13:32:28 - 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E ..Connection
INFO 09/13.13:32:28 - 3A 20 63 6C 6F 73 65 0D 0A 53 65 72 : close..Ser
INFO 09/13.13:32:28 - 76 65 72 3A 0D 0A 0D 0A 3C 68 74 6D ver:....<htm
INFO 09/13.13:32:28 - 6C 3E 3C 62 6F 64 79 3E 3C 68 33 3E l><body><h3>
INFO 09/13.13:32:28 - 57 65 6C 63 6F 6D 65 20 74 6F 20 68 Welcome to h
INFO 09/13.13:32:28 - 65 6C 6C 6F 2E 72 75 6C 3C 2F 68 33 ello.rul</h3
INFO 09/13.13:32:28 - 3E 3C 2F 62 6F 64 79 3E 3C 2F 68 74 ></body></ht
INFO 09/13.13:32:28 - 6D 6C 3E ml>
INFO 09/13.13:32:28 - WEB[AGT=2,CON=4]: disconnecting connection
...

```

- **mimic protocol msg WEB get stats_hdr**
mimic agent protocol msg WEB get statistics
Returns WEB statistics information:

- a list of statistic headers, and
- current statistics values for the specified server.

In order, the statistic values are:

- Total number of connections established.
- Total number of disconnects.
- Total number of packets sent out.
- Total number of packets received.
- Total number of HTTP/HTTPS requests received.
- Total number of HTTP/HTTPS requests processed/responded.
- Total number of responses generated using only WSDL.
- Total number of responses generated using only rule based information.
- Total number of responses generated using combination of WSDL and rule based processing.

A sample exchange for these commands would be:

```

mimicsh> mimic protocol msg WEB get stats_hdr
{{connect} {Connect}} {{disconnect} {Disconnect}}
{{pktSnt} {PktsSent}} {{pktRcvd} {PktsRcvd}}
{{request} {Requests}} {{response} {Responses}}
{{wsdlonly} {WSDLOnly}} {{ruleonly} {RuleOnly}} {{rulewsdl} {Mixed}}

mimicsh> mimic agent protocol msg WEB get statistics
4 3 2 4 4 4 0 0 0

```

- **mimic agent protocol msg WEB server get state**

This command lets the user query the state of the server. This is particularly useful at agent startup time to wait for web server startup. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg WEB server get state
0
mimicsh> mimic agent start

mimicsh> while { ![mimic agent protocol msg WEB server get state] } {
    sleep 1
}
```

- **mimic agent protocol msg WEB port exists ...**

mimic agent protocol msg WEB port add ...

mimic agent protocol msg WEB port set ...

mimic agent protocol msg WEB port start ...

mimic agent protocol msg WEB port stop ...

mimic agent protocol msg WEB port remove ...

This set of commands manipulates extra ports for the WEB server. This allows to add extra service ports for the server, besides the default port in the server configuration. For example, to add a default SSL port would be:

```
mimicsh> if { ![mimic agent protocol msg WEB port exists 443] } {
    mimic agent protocol msg WEB port add 443
    mimic agent protocol msg WEB port set 443 ssl ssl23
    mimic agent protocol msg WEB port start 443
}
```

Recording WEB Sessions

The **WEB Wizard** provides a user-friendly GUI to create WEB simulations from a real-world WEB session / conversation. This section documents the utilities used by the wizard, in case you want to use them from the command line.

To create a default simulation, you have 2 options:

- **webrec**

You can record a web conversation between a WEB client and server with the **webrec** utility. This tool works as a proxy between client and server, and saves requests and responses in a temporary web session file.

This section documents the command-line options to this utility

- **--outfilename output-file**
this mandatory argument specifies the output file name.
- **--port port**
this optional argument specifies a port to listen for cleartext requests, by default 80.
- **--ssl-port port**
this optional argument specifies a port to listen for encrypted requests, by default 443.
- **--dest-ip**
this mandatory argument specifies a host-address of the web server.
- **--dest-port port**
this mandatory argument specifies a port on the web server to forward cleartext requests.
- **--dest-ssl-port port**
this mandatory argument specifies a port on the web server to forward encrypted requests.
- **--servercert**
Certificate PEM file to use as a server
- **--clientcert**
Certificate PEM file to use as a client
- **--cacert**
Certificate file issued by CA
- **--addr**
Bind to IP address (default - all)
- **--max-threads**
Maximum number of threads (default - 10)
- **--outfolder**
Absolute directory path for downloaded file
- **--overwrite**
Overwrite the existing files YES/NO (default - NO)
- **--debug**

Debug dump OFF/ON (default - OFF)

- **webpcap**

If you have a packet capture (PCAP) file of a web session from of the widely available network tools, such as WireShark, tcpdump, etc., you can run it through the `webpcap`, which saves requests and responses in a temporary web session file.

This section documents the command-line options to this utility

- `--file input-file`
this mandatory argument specifies the input PCAP file name. **Note:** only PCAP files are supported, not pcapng or other PCAP style files.
- `--out output-file`
this optional argument specifies the output file name. By default, this is the input file name appended with `.txt`.
- `--target host-address`
this optional argument specifies a host-address of the web server. By default, this is the first web server found in the PCAP.
- `--port port`
this optional argument specifies a port to listen for cleartext requests, by default the first port with a WEB request.
- `--start start`
- `--stop stop`
with these 2 optional parameters you can specify a range of packets to be recorded. By default, all packets will be recorded.
- `--directory directory-path`
this optional argument specifies the directory path where the output file(s) will be placed. By default, this is the folder that contains the input file.

The conversation file that was created by the above `webrec` or `webpcap` utility is then run through the WEB converter `webconv` to create a default WEB simulation.

Simulation Configuration

The definitions that govern the WEB module simulation are encoded in the `rule file`. This is a ASCII text file that is configured for a WEB server instance, and defines the behavior of the WEB simulation.

The rules database is located in the `scripts/web/` directory in the MIMIC directory hierarchy. The MIMIC distribution comes with an example file `sample_cuom.rul`.

NOTE: rules and script files are cached, and modifications to them will only be reloaded by the parser when the rule file modification time is newer than what is in the cache.

Rule File Format

The rules database defines the rules for the WEB services served by an agent instance. It is made up of a series of `service` blocks. A typical rules database will look something like:

```
version = 13.00

include = {
    rules = some-rules.rul
    tcl = some-include.mtcl
}

The include block does inline expansion of both rule files, eg. above the other file some-rules.rul is added to this rule file, or Tcl scripts, eg. above the file some-include.mtcl contains additional scripts.

service = {
    url_pattern = /pattern
    # multiple values can be specified by space separation
    http = HTTP ports to listen the service. e.g "8082 8083"
    https = HTTPS ports to listen the service. e.g "443 8443"
    server_cert = server certificate file with relative path to this file
    web_sim_dir_path = path relative to this file where simulation files are located
    dbpath = path relative to this file where xml database files are located

    runtime = {
        tcl = name of mtcl file
    }

    wsdlfile = name of compiled wsdl file
    porttype = name of portType in wsdl
}
```

The `service` block specifies a service to be served at the particular URL pattern. If no `url_pattern` is specified, it defaults to `/`.

Either `http` or `https` must be specified.

If `https` is specified, then `server_cert` may or may not be specified. If not specified then the default `server.pem` will be used.

Either `rule`, `rulefile`, `wsdlfile` or `redirect` should be specified. If none is specified then an error message will be displayed.

compiled wsdl to be used with this rule file

```

wsdl = {
  file = enterprise/cisco/AXLAPI.cwd
  port = AXLPort
}

```

The `wsdl` block specifies the compiled WSDL file (.`cwd` file) that the current rule file uses to handle commands. An optional port specifies the listening port for connections.

```

# runtime handling for initialization and cleanup
runtime = {
  tcl = runtime.mtcl
}

```

The runtime block defines runtime handlers in `runtime.mtcl`. Currently, there are handlers for agent initialization and cleanup. At agent startup, the `init` proc defined in `runtime.mtcl` is called, and at agent stop, the `cleanup` proc is called.

A request is matched in one of 3 ways:

1. **request =**

This matches a SOAP command. The request needs to be an exact string match, no pattern matching.

```

rule = {
  request = Body/getBillingServer
  response = <?xml version="1.0" encoding="UTF-8"?>
  response = <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.cisco.com/AXL/API/9.0">
  response =   <soap:Body>
  response =     <ns1:getBillingServerResponse/>
  response =   </soap:Body>
  response = </soap:Envelope>
}

```

This rule block specifies a single command and hardcodes the response.

These are sample `curl` commands and their outcome:

```

curl -X POST --data '<?xml version="1.0"?><soap:Envelope><soap:Body><getBillingServer>
</soap:Body></soap:Envelope>' http://10.9.201.134/abc
matches

```

```

curl -X POST --header 'Content-Length: 0' http://10.9.201.134/abc
does not match

```

```

curl -X POST --data '<?xml version="1.0"?><soap:Envelope></soap:Body></soap:Envelope>'
http://10.9.201.134/abc
does not match

```

```

rule = {
  request = Body/getSNMPCommunityString
  response_update = SNMPCommunityString/communityName public-cucm
  response_update = SNMPCommunityString/version 1
  response_update = SNMPCommunityString/accessPrivilege ReadOnly
}

```

When using a WSDL .`cwd` file in conjunction with the current rule file, the `response_update` command allows the inline patching of a XML branch in the command's response.

The `response_update` command also allows scripting of results.

- `response_update = script`
Use this directive to script a specific XML branch's content. Example:

```

response_update = script sample_cuom.mtcl::get_SNMPCommunityString_accessPrivilege

```

- `response_update = xpath script`
Use this directive to script multiple XML branches. Example:

```

response_update = return script sample_cuom.mtcl::getServiceParameter

```

Notice the `return` string is actually a XPath string that matches all XML tag `<return>`.

```

rule = {
  request = Body/listProcessNode
  response = <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.cisco.com/AXL/API/9.0">
  response =   <soap:Body>
  response =     <ns1:listProcessNodeResponse>
  response =       <return>
  response =         # empty element will be filled by script
  response =       </return>
  response =     </ns1:listProcessNodeResponse>
  response =   </soap:Body>
  response = </soap:Envelope>
  response_script = sample_cuom.mtcl::listProcessNode
}

```

```
}
```

Use the `response_script` command to create a general scripted response.

2. `request_header` =

Matches a HTTP command URI. The first line of the request's HTTP header lines needs to exactly match the field, case sensitive.

```
rule = {
  request_header = GET /ccmadmin/showHome.do;jsessionid=F959510A1266BD0CDAED2F5359402C9A
  HTTP/1.1
  response_file = ccmadmin/showHome.do
}
```

Use the `request_header` and `response_file` commands to handle non-SOAP HTTP commands, such as the GET command. This can be used in HTTP file transfer.

```
rule = {
  request_header = GET /ccmadmin HTTP/1.1
  response_header = HTTP/1.1 302 Moved Temporarily
  response_header = Location: http://192.9.200.3/ccmadmin/
  response_header = Date: Wed, 22 May 2013 20:17:27 GMT
  response_header = Server:
  response_header =
  response_header = 0
  response_header =
}
```

When used in conjunction with `request_header`, this rule block can create a HTTP redirect. Some NMS applications rely on HTTP redirect.

The `response_content_type` attribute sets the Content-Type of the response header, eg.

```
rule = {
  request_header = GET /redfish/v1/systems HTTP/1.1

  response_file = /redfish/v1/systems/mimic_1.html
  request_accept = application/json
  response_content_type = application/json
  is_persistent_connection = false
}
```

With the `request_accept` attribute you can control the response of different content types to a query, based on the client's `Accept:` attribute in the request header. If a rule that otherwise matches the request, also matches a specified content in the request header `Accept:` statement, then that rule will be used to return the response.

Use the `response_header_script` command to create a highly customized, scripted response. For example,

Notice that for this request, the connection is not persistent, as specified with the `is_persistent_connection` attribute.

```
rule = {
  request_header = GET /ast/AstIsapi.dll?GetAlertLog&IsInitialRequest=0 HTTP/1.1
  response_header_script = sample_cucm.mtcl::astisapiGetAlertLogIsInitialRequest0
}
```

You can customize HTTP redirects and HTTP authorizations in a `response_header_script` script. For example,

```
proc astisapiGetAlertLogIsInitialRequest0 { } {
  set response "<?xml version=\"1.0\"?>"
  set ret "HTTP/1.1 200 OK\r\n"
  append ret "Content-Type: text/xml;\r\n"
  append ret "Date: Wed, 22 May 2013 20:18:50 GMT\r\n"
  append ret "Server: \r\n"
  append ret "\r\n"
  return $ret
}
```

To allow HTTP chunking in a response, omit the "Transfer-Encoding:" and "Content-Length:" fields. Web.so will automatically fragment the response, as so,

```
proc astisapiGetAlertLogIsInitialRequest0 { } {
  set response "<?xml version=\"1.0\"?>"
  set ret "HTTP/1.1 200 OK\r\n"
  append ret "Content-Type: text/xml;\r\n"
  append ret "Date: Wed, 22 May 2013 20:18:50 GMT\r\n"
  append ret "Server: \r\n"
  append ret "\r\n"
  return $ret
}
```

3. `request_type` =

Matches a HTTP command (GET, POST, ...). This rule takes all the matching HTTP commands, and allows to call a script that handles pattern matching of URLs.

```
rule = {
  request_type = POST
  response_header_script = myrule.mtcl::common_handler
}
```

```
is_persistent_connection = false  
}
```

This rule has all POST commands being handled by the `common_handler` procedure in `myrule.mtc1`. That procedure can then do any pattern matching necessary.

4. Compatibility

Click [here](#) for the compatibility document. If you get an error, you need to download the optional update package with the [Update Wizard](#).

[<< Previous Section](#) [Next Section >>](#)



MIMIC MQTT Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using MQTT from MIMICView](#)
- [Using MQTT from MIMICShell](#)
- [Recording MQTT](#)
- [Simulation Configuration](#)
- [Compatibility](#)

1. Overview

The MIMIC MQTT Protocol Module is an optional facility that simulates the Internet of Things (IoT) Message Queue Telemetry Transport (MQTT) standard version 3.1.1 as detailed on [the MQTT portal site](#) and [ISO](#), as well as limited support for the new [MQTT 5.0](#). Each MIMIC agent instance configured with MQTT acts as a publisher client (eg. a sensor) that connects to the broker (system under test) and publishes the configured messages for third-party subscriber clients to consume. In addition, it can act as a subscriber client, eg. for a control channel, or to do end-to-end latency testing.

MIMIC supports both scenarios of a different IP address per simulated device instance (agent), as in a LAN at the edge, or same IP address for any number of them, as behind a NAT device.

Any type of payload is supported, whether ASCII, JSON or binary, whether stateless or stateful. For example, we have implemented the widely used [Sparkplug](#), [LWM2M-MQTT](#) and [TR50](#) standards, as well as a variety of ad-hoc schemes.

Compatibility with various on-site and cloud-based brokers is documented in the [section below](#).

2. Installation

MQTT support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 16.00, you can use the [Protocol Wizard](#) to install the MQTT module. If you prefer to enable MQTT by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the MQTT shared library (`mqtt.dll` on Windows, `mqtt.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the MQTT module was properly loaded :
INFO - MQTT : Loaded protocol from < path-to-DLL >
INFO - MQTT v16.00

Once MQTT is loaded, any agent instance configured to support the MQTT protocol will be able to publish MQTT messages to a MQTT broker.

3. Using MQTT from MIMICView

If the MQTT module is enabled, then `Agent->Add`, `Agent->Configure` and `Agent->Paste` dialogs will display MQTT as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new `MQTT` pane will appear.

This MQTT configuration pane lets the user configure the parameters for a MQTT session:

- **Config file**
This mandatory parameter specifies the MQTT configuration file, to determine what MQTT packet is generated by the corresponding sensor. You will not be able to start the agent unless this parameter is set.

The configuration file is detailed [below](#). You can edit configuration files directly.
- **Broker Address**
This mandatory parameter specifies the address of the broker server.
- **Port**
This optional parameter specifies the broker port to use. The default port is 1883.
- **Client ID**
By default, the client identifier field in the `CONNECT` message will be a string like `mimic/mqtt/00001`, the last number indicating the agent instance number. You can override this string with this configurable. If the string ends with `#`, then this is meant to be a prefix, and the client identifier sent to the server will replace the `#` with the agent instance number.

- Username
This optional parameter specifies the username to use.
- Password
This optional parameter specifies the username to use.
- Version
This optional parameter specifies the MQTT version to use. You can set this to 3.1.1 or 5.0. By default, version 3.1.1 is used.
- Use TLS
This optional boolean parameter determines whether [TLS security](#) is enabled for this simulated sensor. The default is `false`.
- TLS Config File
If the sensor is configured to Use TLS security as above, then this parameter specifies the [TLS configuration](#) to use. If no configuration file specified, the agent's TLS client will not have a default certification. To use the sample certification included in MIMIC installation, you need to load the `tls.cfg` file.

NOTE: if configuring a large number of agents at different IP addresses, and if SNMP access to the device is not required, there is a performance benefit in configuring non-privileged ports for the SNMP agent (reference 5377).

Using MQTT from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the MQTT functionality. Here is a synopsis:

- **mimic protocol msg MQTT get args**

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg MQTT get args
{{filename} {Config File} {file} {scripts/mqtt {*.cfg {MQTT config files}
{edit yes} {new yes}}} - both {mandatory} {}
{{broker} {Broker Address} {string} {} {mandatory} {}
{{port} {Port} {integer} {} {optional} {1883}
{{clientid} {Client ID} {string} {} {optional} {}
{{username} {Username(blank for all-access)} {string} {} {optional} {}
{{password} {Password(blank for all-access)} {string} {} {optional} {}
{{version} {Version} {string} {} {optional} {}
{{is_tls} {Use TLS} {boolean} {} {optional} {false}}
{{tls_conf_filename} {TLS Config (Optional)} {file} {config/mqtt
{*.cfg {TLS config files} {edit yes} {new yes}}} - both {optional} {}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,MQTT
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,MQTT
mimicsh> mimic agent get protocol
snmpv1,MQTT
```

- **mimic agent protocol msg MQTT get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg MQTT get config
{filename=mqtt1.cfg} {broker=2001:a18:1:10:d6be:d9ff:feeb:589f} {port=1883} {clientid=}
{username=} {password=} {version=} {is_tls=false} {tls_conf_filename=}
```

- **mimic agent protocol msg MQTT set config [config]**

This command lets the user change the current argument settings of all MQTT sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg MQTT get config
{filename=mqtt1.cfg} {broker=2001:a18:1:10:d6be:d9ff:feeb:589f} {port=1883} {clientid=}
{username=} {password=} {version=} {is_tls=false} {tls_conf_filename=}

mimicsh> mimic agent protocol msg MQTT set config \{broker=192.9.200.71\}

mimicsh> mimic agent protocol msg MQTT get config
{filename=mqtt1.cfg} {broker=192.9.200.71} {port=1883} {clientid=}
{username=} {password=} {version=} {is_tls=false} {tls_conf_filename=}

mimicsh> mimic agent protocol msg MQTT set config \{filename=mqtt2.cfg\}
```

```
mimicsh> mimic agent protocol msg MQTT get config
{filename=mqtt2.cfg} {broker=192.9.200.71} {port=1883} {clientid=}
{username=} {password=} {version=} {is_tls=false} {tls_conf_filename=}
```

- **mimic agent protocol msg MQTT get trace**
mimic agent protocol msg MQTT set trace [0 or 1]

This command lets the user change the MQTT tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 1

mimicsh> mimic agent protocol msg MQTT get trace
0
mimicsh> mimic agent protocol msg MQTT set trace 1

mimicsh> mimic agent protocol msg MQTT get trace
1
```

and the log would show:

```
INFO 01/05.09:42:35 - agent 1 trace enabled for MQTT
INFO 01/05.09:42:51 - agent 1 loaded device
INFO 01/05.09:42:51 - MQTT[AGT=1] - sent CONNECT (30 bytes)
INFO 01/05.09:42:51 - MQTT[AGT=1]: client started
INFO 01/05.09:42:51 - MQTT[AGT=1] - rcvd CONNACK (4 bytes)
INFO 01/05.09:43:00 - MQTT[AGT=1] - sent PUBLISH (119 bytes)
INFO 01/05.09:43:00 - MQTT[AGT=1] - rcvd PUBACK (4 bytes)
...
```

- **mimic protocol msg MQTT get stats_hdr**
mimic agent protocol msg MQTT get statistics
Returns MQTT statistics information:

- a list of statistic headers, and
- current statistics values for the specified client.

In order, the statistic values are:

- Total number of client TCP connections
- Total number of client TCP disconnects
- Total number of PUBLISH sent.
- Total number of packets sent.
- Total number of packets received.
- Total number of packets discarded.
- Total number of connections aborted.
- Total number of CONNECT sent.
- Total number of CONNACK received.
- Total number of CONNACK errors.
- Total number of PINGREQ sent.
- Total number of PINGRESP received.
- Total number of PUBLISH sent.
- Total number of PUBLISH payload bytes sent.
- Total number of PUBACK received.
- Total number of PUBREC received.
- Total number of PUBREL sent.
- Total number of PUBCOMP received.
- Total number of PUBLISH received.
- Total number of PUBLISH payload bytes received.
- Total number of PUBACK sent.
- Total number of PUBREC sent.
- Total number of PUBREL received.
- Total number of PUBCOMP sent.
- Total number of SUBSCRIBE sent.
- Total number of SUBACK received.
- Total number of UNSUBSCRIBE sent.
- Total number of UNSUBACK received.
- Total number of DISCONNECT sent.
- Total number of DISCONNECT received.
- Total number of CONNACK discarded.
- Total number of PUBLISH discarded.
- Total number of PUBACK discarded.
- Total number of PUBREC discarded.
- Total number of PUBCOMP discarded.
- Total number of SUBACK discarded.
- Total number of PINGRESP discarded.

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg MQTT get stats_hdr
{{connect} {Connect}} {{disconnect} {Disconnect}} {{published} {Published}}
{{pktSnt} {PktSent}} {{pktRcv} {PktRcvd}} {{pktDisc} {PktDisc}}
{{abort} {Abort}} {{CONNECTsent} {CONNECTsent}} {{CONNACKrcvd} {CONNACKrcvd}}
{{CONNACKerr} {CONNACKerr}} {{PINGREQsent} {PINGREQsent}}
{{PINGRESPrcvd} {PINGRESPrcvd}} {{PUBLISHsent} {PUBLISHsent}}
{{PUBLISHsentbytes} {PUBLISHsentbytes}} {{PUBACKrcvd} {PUBACKrcvd}}
{{PUBRECrvd} {PUBRECrvd}} {{PUBRELSent} {PUBRELSent}}
{{PUBCOMPrcvd} {PUBCOMPrcvd}} {{PUBLISHrcvd} {PUBLISHrcvd}}
```

```

{{PUBLISHrcvdbytes} {PUBLISHrcvdbytes}} {{PUBACKsent} {PUBACKsent}}
{{PUBRECsent} {PUBRECsent}} {{PUBRELrcvd} {PUBRELrcvd}}
{{PUBCOMPsent} {PUBCOMPsent}}
{{SUBSCRIBEsent} {SUBSCRIBEsent}} {{SUBACKrcvd} {SUBACKrcvd}}
{{UNSUBSCRIBEsent} {UNSUBSCRIBEsent}} {{UNSUBACKrcvd} {UNSUBACKrcvd}}
{{DISCONNECTsent} {DISCONNECTsent}} {{DISCONNECTrcvd} {DISCONNECTrcvd}}
{{CONNACKdisc} {CONNACKdisc}}
{{PUBLISHdisc} {PUBLISHdisc}} {{PUBACKdisc} {PUBACKdisc}}
{{PUBRECdisc} {PUBRECdisc}} {{PUBRELDisc} {PUBRELDisc}}
{{PUBCOMPdisc} {PUBCOMPdisc}} {{SUBACKdisc} {SUBACKdisc}}
{{UNSUBACKdisc} {UNSUBACKdisc}} {{PINGRESPdisc} {PINGRESPdisc}}

```

```

mimicsh> mimic agent protocol msg MQTT get statistics
1 0 2 11 11 0 0 1 1 0 1 1 2 48 0 2 2 2 2 48 0 2 2 2 1 1 0 0 0 0 0 0 0 0 0

```

- **mimic agent protocol msg MQTT client get state**

This command lets the user query the state of the client. This is particularly useful at agent startup time to wait for MQTT client startup. A sample exchange for this command would be:

```

mimicsh> mimic agent protocol msg MQTT client get state
0
mimicsh> mimic agent start

mimicsh> while { ![mimic agent protocol msg MQTT client get state] } {
    sleep 1
}

```

- **mimic agent protocol msg MQTT client get protstate**

This command returns the protocol state of the client. The following are the values:

- 0 - stopped
- 2 - disconnected
- 3 - connecting
- 4 - connected
- 5 - waiting for CONNACK
- 6 - waiting for SUBACK
- 7 - CONNACK received, in steady state

- **mimic agent protocol msg MQTT client set broker [BROKER-ADDR]**

```

mimic agent protocol msg MQTT client set port [PORT]
mimic agent protocol msg MQTT client set clientid [CLIENTID]
mimic agent protocol msg MQTT client set username [USERNAME]
mimic agent protocol msg MQTT client set password [PASSWORD]
mimic agent protocol msg MQTT client set willtopic [TOPIC]
mimic agent protocol msg MQTT client set willmsg [MSG]
mimic agent protocol msg MQTT client set willretain [RETAIN]
mimic agent protocol msg MQTT client set willqos [QOS]
mimic agent protocol msg MQTT client set cleansession [0 or 1]
mimic agent protocol msg MQTT client set keepalive [KEEPALIVE]
mimic agent protocol msg MQTT client set on_disconnect [ACTION]

```

You can change these configurable parameters for the MQTT client after agent startup provided that the agent is HALTED and not connected to a broker (ie. mimic agent protocol msg MQTT client get protstate returns 2). For example:

```

mimicsh> mimic agent protocol msg MQTT get config
{filename=mqtt2.cfg} {broker=192.9.200.71} {port=1883} {clientid=}
{username=} {password=} {version=} {is_tls=false} {tls_conf_filename=}

```

```

mimicsh> mimic agent start

```

```

# if the agent has a start action script that sets it's state to HALTED,
# then the MQTT client does not establish a connection

```

```

mimicsh> mimic agent get state
3
mimicsh> mimic agent protocol msg MQTT client get protstate
2
mimicsh> mimic agent protocol msg MQTT client set broker iot.eclipse.org

```

- **mimic agent protocol msg MQTT client runtime abort**

```

mimic agent protocol msg MQTT client runtime disconnect
mimic agent protocol msg MQTT client runtime connect

```

To control the client session at runtime, these commands allow you to abort the session (without sending a DISCONNECT command), disconnect gracefully from the server, and re-connect, respectively. For example:

```

mimicsh> mimic agent start

```

```

mimicsh> mimic agent protocol msg MQTT client get state
1
mimicsh> mimic agent protocol msg MQTT client get protstate
6
mimicsh> mimic agent protocol msg MQTT client runtime disconnect

mimicsh> mimic agent protocol msg MQTT client get state
1
mimicsh> mimic agent protocol msg MQTT client get protstate
2
mimicsh> mimic agent protocol msg MQTT client runtime connect

mimicsh> mimic agent protocol msg MQTT client get state
1
mimicsh> mimic agent protocol msg MQTT client get protstate
6
mimicsh> mimic agent protocol msg MQTT client runtime abort

mimicsh> mimic agent protocol msg MQTT client get protstate
2

```

- **mimic agent protocol msg MQTT client message card**
mimic agent protocol msg MQTT client message get [MSGNUM] [ATTR]
mimic agent protocol msg MQTT client message set [MSGNUM] [ATTR] [VALUE]

To control message generation by the client at runtime, these commands allow you to get and set message simulation parameters. The first parameter returns the cardinality of the messages, ie. 0 or more. You can then control each message (starting at 0) with the get and set commands.

ATTR (defined in [message configuration below](#) can be one of the following:

- topic
- interval
- count
- sent (cannot set)
- pre
- post
- properties (cannot set) - list of PUBLISH properties
- properties.i - i-th PUBLISH property
- properties.PROP-NAME - PUBLISH property with name PROP-NAME

Eg. to PUBLISH exactly one of the first configured message on demand (where count was configured to 0):

```

mimicsh> mimic agent protocol msg MQTT client message card
1
mimicsh> mimic agent protocol msg MQTT client message get 0 count
0
mimicsh> set sent [mimic agent protocol msg MQTT client message get 0 sent]
0
mimicsh> incr sent
1
mimicsh> mimic agent protocol msg MQTT client message set 0 count $sent
# message is PUBLISHED at this moment
mimicsh> mimic agent protocol msg MQTT client message get 0 count
1
mimicsh> set sent [mimic agent protocol msg MQTT client message get 0 sent]
1

```

Or, to set and clear the Message Expiry Interval for a message:

```

mimicsh> mimic agent protocol msg MQTT client message card
1
mimicsh> mimic agent protocol msg MQTT client message get 0 properties
User-Property User-Property
mimicsh> mimic agent protocol msg MQTT client message get 0 properties.0
prop1 = some value 1
mimicsh> mimic agent protocol msg MQTT client message get 0 properties.1
prop2 = some value 2
mimicsh> mimic agent protocol msg MQTT client message get 0 properties.2
{cannot get property 2}
mimicsh> mimic agent protocol msg MQTT client message set 0 properties.Message-Expiry-Interval 100

mimicsh> mimic agent protocol msg MQTT client message get 0 properties.2
100
mimicsh> mimic agent protocol msg MQTT client message set 0 properties.Message-Expiry-Interval {""}

mimicsh> mimic agent protocol msg MQTT client message get 0 properties
User-Property User-Property
mimicsh> mimic agent protocol msg MQTT client message get 0 properties.2
{cannot get property 2}

```

Or, to change one of the user properties:

```

mimicsh> mimic agent protocol msg MQTT client message get 0 properties
User-Property User-Property

```

```
mimicsh> mimic agent protocol msg MQTT client message get 0 properties.0
prop3 = some value 3
mimicsh> mimic agent protocol msg MQTT client message get 0 properties.1
prop2 = some value 2
mimicsh> mimic agent protocol msg MQTT client message set 0 properties.1 {{prop3=some value 3}}

mimicsh> mimic agent protocol msg MQTT client message get 0 properties.0
prop3 = some value 3
mimicsh> mimic agent protocol msg MQTT client message get 0 properties.1
prop3 = some value 3
```

- **mimic agent protocol msg MQTT client subscribe card**
mimic agent protocol msg MQTT client subscribe get [SUBNUM] [ATTR]
mimic agent protocol msg MQTT client subscribe unsubscribe [SUBNUM]
mimic agent protocol msg MQTT client subscribe set [SUBNUM] [ATTR] [VALUE]
mimic agent protocol msg MQTT client subscribe resubscribe [SUBNUM]

To control subscriptions by the client at runtime, these commands allow you to get and set subscription simulation parameters. The first parameter returns the cardinality of the subscriptions, ie. 0 or more. You can then control each subscriptions (starting at 0) with the `get` and `set` commands. The `unsubscribe` command unsubscribes with an UNSUBSCRIBE message, and the `resubscribe` command resubscribes.

ATTR (defined in [subscription configuration below](#) can be one of the following:

- `topic`
- `properties` (cannot set) - list of SUBSCRIBE properties
- `properties.i` - i-th SUBSCRIBE property
- `properties.PROP-NAME` - SUBSCRIBE property with name PROP-NAME

For example, to unsubscribe to the current topic, and re-subscribe with a different one

```
mimicsh> mimic agent protocol msg MQTT client subscribe unsubscribe 0

mimicsh> mimic agent protocol msg MQTT client subscribe get 0 topic
some-topic/#
mimicsh> mimic agent protocol msg MQTT client subscribe get 0 properties
Subscription-Identifier User-Property
mimicsh> mimic agent protocol msg MQTT client subscribe get 0 properties.Subscription-Identifier
123456
mimicsh> mimic agent protocol msg MQTT client subscribe set 0 properties.Subscription-Identifier 123457

mimicsh> mimic agent protocol msg MQTT client subscribe get 0 properties.Subscription-Identifier
123457
mimicsh> mimic agent protocol msg MQTT client subscribe set 0 topic some-topic/2

mimicsh> mimic agent protocol msg MQTT client subscribe resubscribe 0
```

Recording MQTT

The MIMIC `mqttrec` and `mqttconv` utilities work together to transform real-world MQTT client and broker network communications into customizable MIMIC MQTT simulations.

To create a default MQTT simulation, you record MQTT sessions from real-world publisher or subscriber clients (eg. sensors) with the `mqttrec` utility. This tool works as a proxy between MQTT client and broker, and saves commands and responses in a temporary MQTT session file. Alternatively, you can record from a previously captured packet capture (PCAP) file..

The usage is as below:

```
mqttrec --help
Usage: obj.Linux/mqttrec
  [--help, -h]          print this usage message
  or
  --out, -o output file  Recording output file
  --target, -t target    Target ip address or host name
  [--port, -p target port] Target port
  [--start, -S start]   Packet number to start recording from
  [--stop, -E stop]     Ending packet number
  [--file, -f input file] pcap input file
  [--directory, -D path] default path combined with filenames
  [--localport, -l port] Local port
  [--ipv4, -4]          ipv4 (default)
  [--ipv6, -6]          ipv6
```

The MQTT client to be recorded (ie. the real sensor) should be configured to connect to the MIMIC host system, where `mqttrec` is running, as its MQTT broker. The `mqttrec` application will forward mqtt traffic to the broker and broker traffic back to the client.

The MQTT session will be recorded and written to the output file specified. Recorded sessions can be converted by `mqttconv` into MIMIC MQTT simulations.

```
mqttconv --help
INFO 11/16.15:25:21 - MQTTCNV starting
Usage: mqttconv
  [--help, -h]          print this usage message
```

```

or
[--out, -o output file name]    mqttconv base output name (no extension)
[--in, -i input file]          mqttrec input
[--path, -p directory path]    override default path
[--single, -s single output stream] combine MQTT CONNECTs
[--multiple, -m multiple output streams] MIMIC cfg file for each CONNECT

```

mqttconv will generate two types of files, a MIMIC MQTT agent configuration file (.cfg) and script files as detailed below.

mqttconv will process MQTT CONNECT, PUBLISH and SUBSCRIBE commands into a format that can be read by the MIMIC MQTT agent. The MQTT CONNECT information can be further customized using the MIMIC `Configure Agents` dialog tab.

The stream of PUBLISH commands is evaluated for periodic repetitions that can be consolidated into a single MIMIC MQTT simulation configuration file `message statement`.

For non-JSON mqtt payloads only identical payloads are consolidated. When a JSON payload is detected the schema is extracted and compared against the other JSON payload schema. If the schema match the timing is used to determine the period of messages.

Currently the MQTT command times are averaged to create the `interval` value in the MIMIC MQTT configuration `message statement`.

When a JSON payload is identified several support files are created to facilitate customizing the MIMIC MQTT simulation:

- A unique JSON file, `[output name]_[simulation number]_[mqtt command number].json`, that contains the JSON schema and MIMIC actions to retrieve MIMIC agent store variable values for each element.
- A unique MIMIC agent startup TCL script, `start+[output name]_[simulation number]_[mqtt command number].tcl`, that initializes the values of the MIMIC agent store variables read by the JSON file specified in the MIMIC agent `cfg` message statement.

Several general TCL scripts are generated to provide examples of how a MIMIC mqtt simulation can be customized using TCL scripting:

- `action-on-disconnect.tcl` prints a message when an agent is disconnected.
- `mqtt_subscribe_action.tcl` prints a message displaying the payload when an agent receives an mqtt publish message as a result of an mqtt subscribe command.

Simulation Configuration

The behavior of the MQTT simulation (data to be published by the simulated sensor, as well as subscriptions) is specified by the [MQTT configuration file\(s\)](#) loaded into the agent instance. This configuration allows to control a simulated MQTT session from establishment until disconnect.

Globals

This section documents the global parameters for this client. All of these are optional with specified defaults.

Parameter	Description
init	<p>There can be zero or more initialization scripts for this simulation. Currently it supports</p> <ul style="list-style-type: none"> ◦ <code>action</code> The action script to run. These global variables are passed into the action script: <ul style="list-style-type: none"> ▪ <code>gCurrentAgent IN</code> <p>These can be used to initialize agent store variables, start timer scripts, etc.</p>

Connect

This section documents the connect-time parameters for this client. All of these are optional with specified defaults.

Parameter	Description
will-topic	The <code>Will Topic</code> as detailed in section 3.1.2.5 and 3.1.3.2 of the MQTT spec . If this option is not specified, no <code>Will Message</code> will be published.
will-message	The <code>Will Message</code> payload as detailed in section 3.1.2.5 and 3.1.3.3 of the MQTT spec . If this value is the name of an action script, then the payload will be dynamically generated.
will-qos	The <code>Will QoS</code> as detailed in section 3.1.2.5 and 3.1.2.6 of the MQTT spec . The default is 0.
will-retain	The <code>Will Retain</code> as detailed in section 3.1.2.5 and 3.1.2.7 of the MQTT spec . The default is 0.
	<p>The <code>will-properties</code> block specifies MQTT 5 Will Properties to be sent with the <code>CONNECT</code> message (optional). This is an example of the properties block:</p>

will-properties	<pre> will-properties = { Will-Delay-Interval = 20 Payload-Format-Indicator = 1 Message-Expiry-Interval = 123 Response-Topic = some-will-topic User-Property = { name = willprop7 value = some value 7 } User-Property = { name = willprop8 value = some value 8 } } </pre>
clean-session	The Clean Session flag as detailed in section 3.1.2.4 of the MQTT spec . The default is 0.
keepalive	The Keep Alive time as detailed in section 3.1.2.10 of the MQTT spec . The default is 60.
on-disconnect	This parameter specifies the action script to invoke on receiving a DISCONNECT event on the connection. If absent, then no action is taken on disconnect.
on-error	<p>This parameter specifies the action script to invoke on receiving an error event on the connection. If absent, then a default error message is printed in the log. These global variables are passed into the action script:</p> <ul style="list-style-type: none"> gCurrentAgent IN the MQTT control message that generated the error gPacketId IN the packet identifier for the erroneous message gCode IN the reason code in the MQTT control message that generated the error gMsg IN message number (as defined for the mimic agent protocol msg MQTT client message above) gData IN Payload with binary data represented as a string of hexadecimal bytes starting with \x gAscii IN The same payload represented as a ASCII string. This handles the majority of payloads.
properties	<p>The properties block specifies MQTT 5 CONNECT Properties to be sent with the CONNECT message (optional). This is an example of the properties block:</p> <pre> properties = { Session-Expiry-Interval = 0xFFFFFFFF Receive-Maximum = 10 Maximum-Packet-Size = 100 Topic-Alias-Maximum = 5 } </pre>

Message

This section documents the **PUBLISH** message to be sent to the broker as detailed in section 3.3 of the [MQTT spec](#) . There can be multiple different messages published from this client. All parameters are mandatory unless specified. Most parameters can be changed at run-time as detailed below.

Parameter	Description
topic	The topic that all messages of this type are published to. If this value is the name of an action script, then the action script defines the topic, allowing to make the topic unique for the particular agent instance.
interval	The interval in msec that this message is generated.
count	The count of messages to be generated (optional). If this parameter is absent, then the message is generated indefinitely.
qos	The qos that this message is sent at, as detailed in this section (optional). By default it is 0.
	<p>The data section specifies the payload of the message. These are the alternatives:</p> <ul style="list-style-type: none"> const The constant string that is sent in the payload. action

The action script that generates this payload. It is invoked each time a message needs to be published, and returns the content of the payload. These global variables are passed into the action script:

- gCurrentAgent IN
- gMsg IN
message number (as defined for the mimic agent protocol msg MQTT client message above)
- gSeq IN
- gQos IN/OUT
- gRetain IN/OUT
- gTopic IN/OUT

The IN/OUT variables override the message parameters on return, so you can change topic/QOS/retain flag for individual messages in addition to the payload.

- o json_file
The native JSON that specifies the payload of this message. The JSON file defines the payload syntax PAYLOAD as follows:

```
PAYLOAD ::= BLOCK
BLOCK ::= { JSON-STMTS }
JSON-STMTS ::= JSON-STMT | JSON-STMT, JSON-STMTS
JSON-STMT ::= LABEL:BLOCK | LABEL:VALUE-STMT
LABEL ::= "STRING"
VALUE-STMT ::= STRING | VALUE-BLOCK
VALUE-BLOCK ::= { "MIMIC_action": MIMIC-ACTION VALUE-TYPE }
VALUE-TYPE ::= NULL | , "MIMIC_json_type": MIMIC-JSON-TYPE
MIMIC-ACTION ::= "store_get" | "agent_get_host" | "agent_get_ipv6" |
"range" | "seq" | "timeformat(now,format)"
MIMIC-JSON-TYPE ::= "string" | "number" | "int" | "uint" | "int64" | "uint64" |
"boolean" | "object" | "array" | "null"
STRING is an arbitrary constant string
NULL is the empty string
```

For example this JSON PAYLOAD

```
{
  "d": {
    "host": {
      "MIMIC_action" : "agent_get_host()"
    },
    "mem": {
      "MIMIC_action" : "seq(20,60)",
      "MIMIC_json_type" : "number"
    }
  },
  "ts": {
    "MIMIC_action" : "timeformat(now)"
  }
}
```

generates this payload for a sensor at address 10.0.0.2 if generated every 10 seconds:

```
{"d":{"host":"10.0.0.2","mem":20},"ts":"Tue, 02 Aug 2016 10:25:27"}
{"d":{"host":"10.0.0.2","mem":21},"ts":"Tue, 02 Aug 2016 10:25:37"}
{"d":{"host":"10.0.0.2","mem":22},"ts":"Tue, 02 Aug 2016 10:25:47"}
```

The properties block specifies MQTT 5 [PUBLISH Properties](#) to be sent with each message (optional).

This block must be defined after the data block.

This is an example of the properties block:

```
properties = {
  Payload-Format-Indicator = 0
  Message-Expiry-Interval = 10
  Response-Topic = some-topic
  Topic-Alias = 1
  User-Property = {
```

data

properties


```

        name = userprop1
        val = some value 1
    }
    User-Property = {
        name = userprop2
        value = some value 2
    }
}

```

Notice that all properties can only appear once, except for `User-Property` blocks. Properties can be changed at run-time with the `mimic agent protocol msg MQTT client` message [command above](#) .

pre

The `pre` block specifies a pre-processing action script that is executed before sending every payload and before any action script in the `data` block (optional). This block must be defined after the `data` block.

These global variables are passed into the action script:

- `gCurrentAgent` IN
- `gMsg` IN
message number (as defined for the `mimic agent protocol msg MQTT client` message [command above](#))
- `gSeq` IN

For example, the pre-processing script can do calculations and set agent store variables to be retrieved in the `json` data action, or a C++ action script such as that for the [Sparkplug](#) standard.

```
pre = {
    action = action-some-preprocessing.mtcl
}
```

post

The `post` block specifies a post-processing action script that is executed after sending every payload (optional). It is executed only if the `PUBLISH` is sent successfully. This block must be defined after the `data` block.

Same global variables are passed into the action script as for `pre`.

For example, the post-processing script can be used to implement the [MQTT 5 Topic Alias](#) feature, by setting the topic name to empty after the first `PUBLISH`.

```
post = {
    action = action-some-postprocessing.mtcl
}
```

Most of these configurables can be changed at runtime with the `mimic agent protocol msg MQTT client` message `set` [command above](#) .

Subscribe

This section documents subscriber features of the MQTT client as detailed in [section 3.8 of the MQTT spec](#) .

Parameter	Description
topic	The topic that the subscriber wants to receive. If it is the name of an action script, then that topic is dynamically generated at agent startup time and can be unique for the agent.
action	<p>The name of an action script to be invoked upon receipt of <code>PUBLISH</code> messages from the broker on the subscribed topic.</p> <p>These global variables are passed into the action script with information from the <code>PUBLISH</code> message:</p> <ul style="list-style-type: none"> ◦ <code>gCurrentAgent</code> IN ◦ <code>gPacketId</code> IN the packet identifier for the <code>PUBLISH</code> message ◦ <code>gDup</code> IN ◦ <code>gQos</code> IN ◦ <code>gRetain</code> IN ◦ <code>gData</code> IN

- Payload with binary data represented as a string of hexadecimal bytes starting with `\x`
- `gAscii` IN
The same payload represented as a ASCII string. This handles the majority of payloads.
- `gTopic` IN
- `gProps` IN
For MQTT 5, the Properties in the PUBLISH message.

An error return, eg. with a "return -code error" statement from this script results in the PUBLISH to be discarded (and not acknowledged if QOS > 1).

Most of these configurables can be changed at runtime with the `mimic agent protocol msg MQTT client subscribe set` [command above](#) .

TLS configuration

This section documents the TLS configuration file.

Parameter	Description
certificate	<p>The client certificate file. The sample command</p> <pre>openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365 -nodes</pre> <p>created the certificate and key file that is shipped in <code>config/mqtt/client.crt</code> . For details see this primer.</p> <p>This command diagnoses the SSL connection to a server at IP address IPADDR and port PORT (usually 8883) for a Certificate Authority specified in CAFILE:</p> <pre>openssl s_client -connect IPADDR:PORT -CAfile CAFILE</pre>
key	The client private key file.

Compatibility

Click [here](#) for the compatibility document. If you get an error, you need to download the optional update package with the [Update Wizard](#).

[<< Previous Section](#) [Next Section >>](#)



MIMIC CoAP Protocol Module Guide

Table of Contents

- [Overview](#)
- [Installation](#)
- [Using CoAP from MIMICView](#)
- [Using CoAP from MIMICShell](#)
- [Recording CoAP](#)
- [Simulation Configuration](#)
- [Compatibility](#)

1. Overview

The MIMIC CoAP Protocol Module is an optional facility that simulates the Internet of Things (IoT) Constrained Application Protocol (CoAP) standard as detailed on [IETF RFC 7252](#) . Each MIMIC agent instance configured with CoAP acts as a server

Compatibility with various clients and applications is documented in the [section below](#) .

2. Installation

CoAP support is made available in MIMIC as an optional dynamically loadable module. Starting with MIMIC 18.00, you can use the [Protocol Wizard](#) to install the CoAP module. If you prefer to enable CoAP by hand, you need to do the following:

- Use `File->Terminate` to stop the any running MIMIC daemon.
- Copy the CoAP shared library (`coap.dll` on Windows, `coap.so` on Unix) from "bin/dynamic/optional" to "bin/dynamic" in the install directory.
- Install the license keys as detailed in the instructions e-mailed to you.
- Restart MIMIC. You should see the following type of message in the MIMICLog that confirms that the CoAP module was properly loaded :
INFO - COAP : Loaded protocol from < path-to-DLL >
INFO - COAP v18.00

Once CoAP is loaded, any agent instance configured to support the CoAP protocol will be able to respond to requests from clients.

3. Using CoAP from MIMICView

If the CoAP module is enabled, then `Agent ->Add`, `Agent ->Configure` and `Agent ->Paste` dialogs will display COAP as an additional checkbox in the `Advanced` pane along with the SNMP protocols. On selecting the checkbox a new `COAP` pane will appear.

This CoAP configuration pane lets the user configure the parameters for a CoAP session:

- **Primary Port**
This optional parameter specifies the non-encrypted, server UDP port to use. If set to 0, this port is disabled. The default port is 5683.
- **Rule file**
This mandatory parameter specifies the CoAP rule file, to determine the response for each REST command with URI. You will not be able to start the agent unless this parameter is set.

The rule file is detailed [below](#) . You can edit rule files directly.
- **Secure Port**
This optional parameter specifies the DTLS-encrypted, server UDP port to use. If set to 0, this port is disabled. The default port is 5684.
- **Key Store**
This optional parameter specifies the keystore file to use. If the keystore file is not set, the DTLS CoAP agent can not receive encrypted commands because the agent needs the correct key to decrypt encrypted traffic.

4. Using CoAP from MIMICShell

A few new commands and some enhanced old commands can be used from the MIMICShell to control the CoAP functionality. Here is a synopsis:

- `mimic protocol msg COAP get args`

This command lets the user gather the self-defining list of arguments required and their particulars. The parameters are detailed above. A sample exchange for this command would be:

```
mimicsh> mimic protocol msg COAP get args
{{primary_port} {Primary Port} {integer} {} {optional} {5683}}
{{rule} {Rule File} {file} {scripts/coap {*.rul {coap rule files} {edit yes} {new yes}}}}
- both {mandatory} {} {{secure_port} {Secure Port} {integer} {} {optional} {5684}}
{{keystore} {Key Store} {file} {config/coap {*.db {Key store files} {edit yes} {new yes}}}}
- both {optional} {samplekeystore.db}}
```

- **mimic agent get protocol**

This command lets the user look at the protocols currently configured on the agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1,snmpv2c,COAP
```

- **mimic agent set protocol**

This command lets the user change the protocol setting for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent get protocol
snmpv1
mimicsh> mimic agent set protocol snmpv1,COAP
mimicsh> mimic agent get protocol
snmpv1,COAP
```

- **mimic agent protocol msg COAP get config**

This command lets the user get the current argument settings. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg COAP get config
{primary_port=5683} {rule=iot-eclipse-org.rul} {secure_port=0}
{keystore=samplekeystore.db}
```

- **mimic agent protocol msg COAP set config [config]**

This command lets the user change the current argument settings of all CoAP sessions for an agent. A sample exchange for this command would be:

```
mimicsh> mimic agent protocol msg COAP get config
{primary_port=5683} {rule=iot-eclipse-org.rul} {secure_port=0}
{keystore=samplekeystore.db}

mimicsh> mimic agent protocol msg COAP set config \{secure_port=5684\}

mimicsh> mimic agent protocol msg COAP get config
{primary_port=5683} {rule=iot-eclipse-org.rul} {secure_port=5684}
{keystore=samplekeystore.db}

mimicsh> mimic agent protocol msg COAP set config \{rule=ikea-e1526.cfg\}

mimicsh> mimic agent protocol msg COAP get config
{primary_port=5683} {rule=ikea-e1526.cfg} {secure_port=5684} {keystore=samplekeystore.db}
```

- **mimic agent protocol msg COAP get trace**

- **mimic agent protocol msg COAP set trace [0 or 1]**

This command lets the user change the CoAP tracing configuration for an agent. A sample exchange would be:

```
mimicsh> mimic agent assign 1

mimicsh> mimic agent protocol msg COAP get trace
0
mimicsh> mimic agent protocol msg COAP set trace 1

mimicsh> mimic agent protocol msg COAP get trace
1
```

and the log would show:

```
INFO 01/05.09:42:35 - agent 1 trace enabled for COAP
INFO 01/05.09:42:51 - agent 1 loaded device
INFO 01/05.09:42:51 - COAP[AGT=1]: rules database initialized
INFO 01/05.09:42:51 - COAP [AGT=1]: server started
...
```

- **mimic protocol msg COAP get stats_hdr**

- **mimic agent protocol msg COAP get statistics**

Returns CoAP statistics information:

- a list of statistic headers, and
- current statistics values for the specified server.

In order, the statistic values are:

- Total number of CoAP packets sent.
- Total number of CoAP packets received.

A sample exchange for these commands would be:

```
mimicsh> mimic protocol msg COAP get stats_hdr
{{connect} {Connect}} {{disconnect} {Disconnect}} {{published} {Published}}
{{pktSnt} {PktSent}} {{pktRcv} {PktRcv}} {{pktDisc} {PktDisc}}

mimicsh> mimic agent protocol msg COAP get statistics
1 0 18 18 18 0
```

Recording CoAP

TBD

Simulation Configuration

CoAP responses to be returned by the simulated sensor are specified by the [rule file\(s\)](#) loaded into the agent instance.

Overview

The rules database is a ASCII text file and the loader reads and sources any Tcl scripts as associated. The contents of the file are used to initialize data structures that will then be used to return a response string (output) for a given request REST command. In the absence of a match, a CoAP error is returned.

The rules database is located in the `scripts/coap/` directory in the MIMIC directory hierarchy. Any custom Tcl scripts also need to be in this same directory.

NOTE: rules and script files are cached, and modifications to them will only be reloaded by the parser when the rule file modification time is newer than what is in the cache.

Format

The rules database is made of a series of `rule` blocks. A rule block must have 1 `request_header` field and either one `response` or one `response_script` field.

```
version = 18.00
rule = {
  request_header = GET /15011/9063/
  response_script = ikea-e1526.mtcl::generate_key
}
rule = {
  request_header = GET /15001/
  response = [65536,65537,65538,65539,65540,65541,65542]
}
```

Global Tcl Variables

Any Tcl procedure invoked will have a predefined set of global variables passed to it which identify the current environment.

- `gCurrentRequest`
This variable holds the POST command data received from the CoAP client.

Compatibility

Click [here](#) for the compatibility document. If you get an error, you need to download the optional update package with the [Update Wizard](#).

[<< Previous Section](#) [Next Section >>](#)



Appendix A: MIMIC Directory Hierarchy

Overview

The MIMIC distribution directory hierarchy has the following layout:

```
bin/ ..... OS-specific executables (symbolic link on Unix)
dynamic/ ..... dynamically loadable protocol modules
mimicd ..... MIMIC SNMP Agent Simulator daemon
mimiccom ..... MIMIC Compiler
mimicrec ..... MIMIC Recorder
mimicview ..... MIMICView GUI front-end
execlog ..... MIMIC log window
mimicsh ..... MIMICShell
mib2walk ..... Walkfile template generator
oidinfo ..... MIB object information tool
tcl/ ..... Tcl/Tk files for Tcl-based MIMIC clients
trapper ..... Trap Recorder
etc.

bitmaps/ ..... Bitmaps for MIMIC clients (MIMICView, etc)
3Com ..... (one per type of device)
FORE
etc. .... (You can add your own)
common/ ..... Common OS-independent scripts
hpov.csh ..... HP OpenView walkfile conversion
etc.

config/ ..... MIMIC configuration files
*.lic ..... License file per tool
acl.conf ..... SNMPv2 access control configuration file
admin/ ..... MIMIC administrative configuration files
agents/ ..... MIMIC lab configuration files
  OLD/ ..... old versioned lab configuration files
context.conf ..... SNMPv2 context configuration file
[debug.cfg] ..... Optional debugging configuration
dev.cfg ..... MIMIC device simulation configuration (obsoleted)
devices ..... MIMIC device category database
execlog.cfg ..... Log viewer configuration
filters.cfg ..... filters for walkfile converter
gui.cfg ..... MIMICView GUI configuration
iosdisc.cfg ..... IOS Discovery program configuration
iosrec.cfg ..... IOS Recorder program configuration
java.cfg ..... Java configuration
mibequiv.db ..... MIB equivalences database
mimiccom.types ..... MIMIC Compiler internal types
[mimicd.cfg] ..... Optional MIMIC Simulator configuration
mimicview.cfg ..... MIMICView configuration
party.conf ..... SNMPv2 party configuration file
persist.cfg ..... MIMICView persistency configuration
timerscripts.cfg ..... timerscript configuration
traps.cfg ..... MIMIC global trap destinations
version ..... MIMIC version
view.conf ..... SNMPv2 view configuration file
v3*.conf ..... SNMPv3 security configuration
vlab/ ..... Virtual Lab configuration files
cpp/ ..... C++ API
inc/ ..... header files
lib/ ..... library files
src/ ..... source files
data/ ..... MIMIC data files
discovery/ ..... Discovery Wizard data files
mibs/
  mimic.dir ..... MIB directory
  *.db, *.typ ..... Compiled MIB files
  3com/ ..... Enterprise-specific MIB files
  ascend/ ..... (Each in its own subdirectory)
  etc.
sim/
  <SIMULATION>/ ..... Per-simulation directories
  dev.cfg ..... device configuration file
  *.mtcl ..... action scripts
  <MIB>*/ ..... Compiled simulation files per MIB
    *.sdb ..... Simulation data base files
    *.idb ..... Index data base files
    etc.
  <SCENARIO>/ .. Per-scenario directories
```

```

*.tab .... Table files
*.var .... Variable files
etc.
snapshots/ ..... Snapshot Wizard data files
traps/ ..... Trap Wizard data files
help/ ..... MIMIC online documentation
mimic.htm ..... main page
etc.
java/ ..... Java-based tools
Apache-Oro/ ..... Topology Editor related classes
JGo/ ..... same
TopEd/ ..... same
Utils/ ..... same
Mimic/ ..... MIMIC API classes
applet/ ..... applet source code
test/ ..... test client /w source code
*.class, *.html ..... applet /w web page
vlab/ ..... Virtual Lab client
webui/ ..... WEBUI
etc.
lib/ ..... libraries for Tcl-based clients
linux/ ..... Linux-specific executables
etc.
mibs/ ..... MIB source files
rfc1213.mib ..... RFC MIBs in top directory
etc. .... Other standard MIB source files
3com/ ..... Enterprise-specific MIB files
ascend/ ..... (Each in its own subdirectory)
bay/
cabletron/
cisco/
dec/
hp/
shiva/
etc.
packages/ ..... MIMIC package for Tcl-based shells
perl/ ..... Perl API
php/ ..... PHP API
python/ ..... Python API
scripts/ ..... MIMICshell script files, action scripts, IOS rule scripts
bin/ ..... dynamically loadable action scripts
exercise/ ..... Virtual Lab exercises
mqtt/ ..... MQTT protocol module scripts
netflow/ ..... NETFLOW protocol module scripts
telnet/ ..... Telnet protocol module scripts
vlab/ ..... Virtual Lab scripts
web/ ..... WEB protocol module scripts
sim/ ..... Per-simulation directories
<SIMULATION>/ ..... Source files for <SIMULATION>
  <MIB>.sim ..... Simulation source file per MIB
  etc.
  etc. .... Other simulations
snapshots/ ..... snapshots created by Snapshot Wizard
solaris/ ..... Solaris-specific executables
etc.
traps/ ..... trap series created by Trap Wizard

```



Appendix B: Pre-compiled MIBs

Overview

The MIMIC distribution supplies the following pre-compiled MIBs in compiled form in the base product. The MIB source files are installable as an optional update. If you have installed the MIB sources, click on the MIB name to browse the source file.

MIB Name	Compiled File Name
1. RFC1155-SMI	data/mibs/RFC1155-SMI.db
2. SNMPv2-CONF	data/mibs/SNMPv2-CONF.db
3. RFC1213-MIB	data/mibs/RFC1213-MIB.db
4. SNMPv2-SMI-v1	data/mibs/SNMPv2-SMI-v1.db
5. SNMPv2-TC-v1	data/mibs/SNMPv2-TC-v1.db
6. SNMPv2-SMI	data/mibs/SNMPv2-SMI.db
7. SNMPv2-TC	data/mibs/SNMPv2-TC.db
8. SNMPv2-TM	data/mibs/SNMPv2-TM.db
9. SNMPv2-MIB	data/mibs/SNMPv2-MIB.db
10. IANAifType-MIB	data/mibs/IANAifType-MIB.db
11. IF-MIB	data/mibs/IF-MIB.db
12. SNMPv1-TRAPS	data/mibs/SNMPv1-TRAPS.db
13. SMUX-MIB	data/mibs/SMUX-MIB.db
14. RFC1229-MIB	data/mibs/RFC1229-MIB.db
15. RFC1230-MIB	data/mibs/RFC1230-MIB.db
16. RFC1231-MIB	data/mibs/RFC1231-MIB.db
17. RFC1232-MIB	data/mibs/RFC1232-MIB.db
18. RFC1233-MIB	data/mibs/RFC1233-MIB.db
19. CLNS-MIB	data/mibs/CLNS-MIB.db
20. RFC1285-MIB	data/mibs/RFC1285-MIB.db
21. SIP-MIB	data/mibs/SIP-MIB.db
22. CHARACTER-MIB	data/mibs/CHARACTER-MIB.db
23. RFC1318-MIB	data/mibs/RFC1318-MIB.db
24. RFC1353-MIB	data/mibs/RFC1353-MIB.db
25. RFC1381-MIB	data/mibs/RFC1381-MIB.db
26. RFC1414-MIB	data/mibs/RFC1414-MIB.db
27. SNMPv2-PARTY-MIB	data/mibs/SNMPv2-PARTY-MIB.db
28. PPP-LCP-MIB	data/mibs/PPP-LCP-MIB.db
29. PPP-SEC-MIB	data/mibs/PPP-SEC-MIB.db
30. PPP-IP-NCP-MIB	data/mibs/PPP-IP-NCP-MIB.db
31. PPP-BRIDGE-NCP-MIB	data/mibs/PPP-BRIDGE-NCP-MIB.db
32. BRIDGE-MIB	data/mibs/BRIDGE-MIB.db
33. FDDI-SMT73-MIB	data/mibs/FDDI-SMT73-MIB.db
34. HOST-RESOURCES-MIB HOST-RESOURCES-TYPES	data/mibs/HOST-RESOURCES-MIB HOST-RESOURCES-TYPES.db
35. SNMP-REPEATER-MIB	data/mibs/SNMP-REPEATER-MIB.db
36. SOURCE-ROUTING-MIB	data/mibs/SOURCE-ROUTING-MIB.db
37. DECNET-PHIV-MIB	data/mibs/DECNET-PHIV-MIB.db
38. DPI20-MIB	data/mibs/DPI20-MIB.db
39. FRNETSERV-MIB	data/mibs/FRNETSERV-MIB.db
40. UPS-MIB	data/mibs/UPS-MIB.db
41. BGP4-MIB	data/mibs/BGP4-MIB.db
42. RS-232-MIB	data/mibs/RS-232-MIB.db
43. SNA-NAU-MIB	data/mibs/SNA-NAU-MIB.db
44. Modem-MIB	data/mibs/Modem-MIB.db
45. RIPv2-MIB	data/mibs/RIPv2-MIB.db
46. APPLETTALK-MIB	data/mibs/APPLETTALK-MIB.db
47. SNA-SDLC-MIB --	data/mibs/SNA-SDLC-MIB --.db
48. RMON-MIB	data/mibs/RMON-MIB.db
49. RFC1382-MIB	data/mibs/RFC1382-MIB.db
50. TCPIP-MIB	data/mibs/TCPIP-MIB.db
51. TOKEN-RING-RMON-MIB	data/mibs/TOKEN-RING-RMON-MIB.db
52. RMON2-MIB	data/mibs/RMON2-MIB.db
53. IANA-PRINTER-MIB	data/mibs/IANA-PRINTER-MIB.db
54. IANA-CHARSET-MIB	data/mibs/IANA-CHARSET-MIB.db
55. Printer-MIB	data/mibs/Printer-MIB.db
56. OSPF-MIB OSPF-TRAP-MIB	data/mibs/OSPF-MIB OSPF-TRAP-MIB.db
57. DOT12-IF-MIB	data/mibs/DOT12-IF-MIB.db
58. MIP-MIB	data/mibs/MIP-MIB.db
59. DLSW-MIB --	data/mibs/DLSW-MIB --.db
60. SNMP-FRAMEWORK-MIB	data/mibs/SNMP-FRAMEWORK-MIB.db

61. IANA-ENTITY-MIB UUID-TC-MIB ENTITY-MIB data/mibs/IANA-ENTITY-MIB UUID-TC-MIB ENTITY-MIB.db

62. AGENTX-MIB data/mibs/AGENTX-MIB.db

63. APPC-MIB data/mibs/APPC-MIB.db

64. FLOW-METER-MIB data/mibs/FLOW-METER-MIB.db

65. INET-ADDRESS-MIB data/mibs/INET-ADDRESS-MIB.db

66. IANA-RTPROTO-MIB data/mibs/IANA-RTPROTO-MIB.db

67. IP-FORWARD-MIB data/mibs/IP-FORWARD-MIB.db

68. FRAME-RELAY-DTE-MIB data/mibs/FRAME-RELAY-DTE-MIB.db

69. ISDN-MIB data/mibs/ISDN-MIB.db

70. DIAL-CONTROL-MIB data/mibs/DIAL-CONTROL-MIB.db

71. APPN-MIB data/mibs/APPN-MIB.db

72. INTEGRATED-SERVICES-MIB data/mibs/INTEGRATED-SERVICES-MIB.db

73. INTEGRATED-SERVICES-GUARANTEED-MIB data/mibs/INTEGRATED-SERVICES-GUARANTEED-MIB.db

74. RSVP-MIB data/mibs/RSVP-MIB.db

75. APPN-DLUR-MIB data/mibs/APPN-DLUR-MIB.db

76. HPR-MIB data/mibs/HPR-MIB.db

77. NETWORK-SERVICES-MIB data/mibs/NETWORK-SERVICES-MIB.db

78. RDBMS-MIB data/mibs/RDBMS-MIB.db

79. MTA-MIB data/mibs/MTA-MIB.db

80. DOT12-RPTR-MIB data/mibs/DOT12-RPTR-MIB.db

81. SYSAPPL-MIB data/mibs/SYSAPPL-MIB.db

82. IPOA-MIB data/mibs/IPOA-MIB.db

83. APPN-TRAP-MIB data/mibs/APPN-TRAP-MIB.db

84. EBN-MIB data/mibs/EBN-MIB.db

85. IPV6-TC data/mibs/IPV6-TC.db

86. IPV6-MIB data/mibs/IPV6-MIB.db

87. IPV6-TCP-MIB data/mibs/IPV6-TCP-MIB.db

88. IPV6-UDP-MIB data/mibs/IPV6-UDP-MIB.db

89. IPV6-ICMP-MIB data/mibs/IPV6-ICMP-MIB.db

90. PerfHist-TC-MIB data/mibs/PerfHist-TC-MIB.db

91. DS0-MIB data/mibs/DS0-MIB.db

92. DS0BUNDLE-MIB data/mibs/DS0BUNDLE-MIB.db

93. DS1-MIB data/mibs/DS1-MIB.db

94. DS3-MIB data/mibs/DS3-MIB.db

95. ACCOUNTING-CONTROL-MIB data/mibs/ACCOUNTING-CONTROL-MIB.db

96. ATM-TC-MIB data/mibs/ATM-TC-MIB.db

97. ATM-MIB data/mibs/ATM-MIB.db

98. ATM-FORUM-SNMP-M4-MIB data/mibs/ATM-FORUM-SNMP-M4-MIB.db

99. NCCI-MIB data/mibs/NCCI-MIB.db

100. PNNI-MIB data/mibs/PNNI-MIB.db

101. ATM-TRACE-MIB data/mibs/ATM-TRACE-MIB.db

102. ATM-REROUTING-MIB data/mibs/ATM-REROUTING-MIB.db

103. HOPCOUNT-MIB data/mibs/HOPCOUNT-MIB.db

104. ATM-DXI-MIB data/mibs/ATM-DXI-MIB.db

105. ATM-FORUM-TC-MIB ATM-FORUM-MIB ATM-FORUM-ADDR-REG ATM-FORUM-SRVC-REG data/mibs/ATM-FORUM-TC-MIB ATM-FORUM-MIB ATM-FORUM-ADDR-REG ATM-FORUM-SRVC-REG.db

106. LAN-EMULATION-CLIENT-MIB data/mibs/LAN-EMULATION-CLIENT-MIB.db

107. LAN-EMULATION-ELAN-MIB LAN-EMULATION-LES-MIB LAN-EMULATION-BUS-MIB data/mibs/LAN-EMULATION-ELAN-MIB LAN-EMULATION-LES-MIB LAN-EMULATION-BUS-MIB.db

108. ATM-SOFT-PVC-MIB data/mibs/ATM-SOFT-PVC-MIB.db

109. ATMF-CES data/mibs/ATMF-CES.db

110. ATM-FORUM-AUTO-CONFIG data/mibs/ATM-FORUM-AUTO-CONFIG.db

111. IMA-MIB-OLD data/mibs/IMA-MIB-OLD.db

112. IMA-MIB data/mibs/IMA-MIB.db

113. IPV6-MLD-MIB data/mibs/IPV6-MLD-MIB.db

114. SONET-MIB data/mibs/SONET-MIB.db

115. APPLICATION-MIB data/mibs/APPLICATION-MIB.db

116. SNMP-MPD-MIB data/mibs/SNMP-MPD-MIB.db

117. SNMP-USER-BASED-SM-MIB data/mibs/SNMP-USER-BASED-SM-MIB.db

118. SNMP-VIEW-BASED-ACM-MIB data/mibs/SNMP-VIEW-BASED-ACM-MIB.db

119. HPR-IP-MIB data/mibs/HPR-IP-MIB.db

120. DISMAN-SCHEDULE-MIB data/mibs/DISMAN-SCHEDULE-MIB.db

121. DISMAN-SCRIPT-MIB data/mibs/DISMAN-SCRIPT-MIB.db

122. WWW-MIB data/mibs/WWW-MIB.db

123. SMON-MIB data/mibs/SMON-MIB.db

124. RADIUS-AUTH-CLIENT-MIB data/mibs/RADIUS-AUTH-CLIENT-MIB.db

125. RADIUS-AUTH-SERVER-MIB data/mibs/RADIUS-AUTH-SERVER-MIB.db

126. ADSL-TC-MIB data/mibs/ADSL-TC-MIB.db

127. ADSL-LINE-MIB data/mibs/ADSL-LINE-MIB.db

128. EtherLike-MIB data/mibs/EtherLike-MIB.db

129. IPV6-FLOW-LABEL-MIB data/mibs/IPV6-FLOW-LABEL-MIB.db

130. TUNNEL-MIB data/mibs/TUNNEL-MIB.db

131. IANA-MAU-MIB data/mibs/IANA-MAU-MIB.db

132. MAU-MIB data/mibs/MAU-MIB.db

133. DOCS-CABLE-DEVICE-MIB data/mibs/DOCS-CABLE-DEVICE-MIB.db

134. P-BRIDGE-MIB Q-BRIDGE-MIB data/mibs/P-BRIDGE-MIB Q-BRIDGE-MIB.db

135. SNMP-USM-DH-OBJECTS-MIB data/mibs/SNMP-USM-DH-OBJECTS-MIB.db

136. VRRP-MIB data/mibs/VRRP-MIB.db

137. IANA-ADDRESS-FAMILY-NUMBERS-MIB data/mibs/IANA-ADDRESS-FAMILY-NUMBERS-MIB.db

138. DOCS-IF-MIB data/mibs/DOCS-IF-MIB.db

139. DOCS-QOS-MIB data/mibs/DOCS-QOS-MIB.db

140. DOCS-SUBMGT-MIB data/mibs/DOCS-SUBMGT-MIB.db

141. DOCS-BPI2-MIB data/mibs/DOCS-BPI2-MIB.db
142. DOCS-IF-EXT-MIB DOCS-CABLE-DEVICE-TRAP-MIB data/mibs/DOCS-IF-EXT-MIB DOCS-CABLE-DEVICE-TRAP-MIB.db
143. DOCS-BPI-MIB data/mibs/DOCS-BPI-MIB.db
144. FIBRE-CHANNEL-FE-MIB data/mibs/FIBRE-CHANNEL-FE-MIB.db
145. HCNUM-TC data/mibs/HCNUM-TC.db
146. IF-INVERTED-STACK-MIB data/mibs/IF-INVERTED-STACK-MIB.db
147. PTOPO-MIB data/mibs/PTOPO-MIB.db
148. DISMAN-PING-MIB DISMAN-TRACEROUTE-MIB DISMAN-NSLOOKUP-MIB data/mibs/DISMAN-PING-MIB DISMAN-TRACEROUTE-MIB DISMAN-NSLOOKUP-MIB.db
149. SNMP-TARGET-MIB data/mibs/SNMP-TARGET-MIB.db
150. SNMP-NOTIFICATION-MIB data/mibs/SNMP-NOTIFICATION-MIB.db
151. SNMP-PROXY-MIB data/mibs/SNMP-PROXY-MIB.db
152. IPMROUTE-STD-MIB data/mibs/IPMROUTE-STD-MIB.db
153. IGMP-STD-MIB data/mibs/IGMP-STD-MIB.db
154. PIM-MIB data/mibs/PIM-MIB.db
155. RTP-MIB data/mibs/RTP-MIB.db
156. NOTIFICATION-LOG-MIB data/mibs/NOTIFICATION-LOG-MIB.db
157. MGMD-STD-MIB data/mibs/MGMD-STD-MIB.db
158. FR-MFR-MIB data/mibs/FR-MFR-MIB.db
159. PINT-MIB data/mibs/PINT-MIB.db
160. DMTF-DMI-MIB data/mibs/DMTF-DMI-MIB.db
161. SFLOW-MIB data/mibs/SFLOW-MIB.db
162. CIRCUIT-IF-MIB data/mibs/CIRCUIT-IF-MIB.db
163. FRSLD-MIB data/mibs/FRSLD-MIB.db
164. HC-RMON-MIB data/mibs/HC-RMON-MIB.db
165. HDLSL2-SHDSL-LINE-MIB data/mibs/HDLSL2-SHDSL-LINE-MIB.db
166. DIFFSERV-DSCP-TC DIFFSERV-MIB data/mibs/DIFFSERV-DSCP-TC DIFFSERV-MIB.db
167. GSMP-MIB data/mibs/GSMP-MIB.db
168. L2TP-MIB data/mibs/L2TP-MIB.db
169. ENTITY-SENSOR-MIB data/mibs/ENTITY-SENSOR-MIB.db
170. HC-ALARM-MIB data/mibs/HC-ALARM-MIB.db
171. ADSL-LINE-EXT-MIB data/mibs/ADSL-LINE-EXT-MIB.db
172. APS-MIB data/mibs/APS-MIB.db
173. SNMP-COMMUNITY-MIB data/mibs/SNMP-COMMUNITY-MIB.db
174. OPT-IF-MIB data/mibs/OPT-IF-MIB.db
175. ATM2-MIB data/mibs/ATM2-MIB.db
176. POWER-ETHERNET-MIB data/mibs/POWER-ETHERNET-MIB.db
177. ETHER-WIS data/mibs/ETHER-WIS.db
178. HC-PerfHist-TC-MIB data/mibs/HC-PerfHist-TC-MIB.db
179. VDSL-LINE-MIB data/mibs/VDSL-LINE-MIB.db
180. APM-MIB data/mibs/APM-MIB.db
181. DIFFSERV-CONFIG-MIB data/mibs/DIFFSERV-CONFIG-MIB.db
182. IANA-FINISHER-MIB data/mibs/IANA-FINISHER-MIB.db
183. Finisher-MIB data/mibs/Finisher-MIB.db
184. MPLS-TC-STD-MIB data/mibs/MPLS-TC-STD-MIB.db
185. MPLS-TE-STD-MIB data/mibs/MPLS-TE-STD-MIB.db
186. MPLS-LSR-STD-MIB data/mibs/MPLS-LSR-STD-MIB.db
187. MPLS-FTN-STD-MIB data/mibs/MPLS-FTN-STD-MIB.db
188. MPLS-LDP-STD-MIB MPLS-LDP-ATM-STD-MIB MPLS-LDP-FRAME-RELAY-STD-MIB MPLS-LDP-GENERIC-STD-MIB data/mibs/MPLS-LDP-STD-MIB MPLS-LDP-ATM-STD-MIB MPLS-LDP-FRAME-RELAY-STD-MIB MPLS-LDP-GENERIC-STD-MIB.db
189. ROHC-MIB ROHC-UNCOMPRESSED-MIB ROHC-RTP-MIB data/mibs/ROHC-MIB ROHC-UNCOMPRESSED-MIB ROHC-RTP-MIB.db
190. SNMP-USM-AES-MIB data/mibs/SNMP-USM-AES-MIB.db
191. SCTP-MIB data/mibs/SCTP-MIB.db
192. ALARM-MIB SNMP-NOTIFICATION-MIB IANA-ITU-ALARM-TC-MIB ITU-ALARM-TC-MIB ITU-ALARM-MIB data/mibs/ALARM-MIB SNMP-NOTIFICATION-MIB IANA-ITU-ALARM-TC-MIB ITU-ALARM-TC-MIB ITU-ALARM-MIB.db
193. ARC-MIB data/mibs/ARC-MIB.db
194. TE-MIB data/mibs/TE-MIB.db
195. TCP-MIB data/mibs/TCP-MIB.db
196. DOCS-IETF-SUBMGT-MIB data/mibs/DOCS-IETF-SUBMGT-MIB.db
197. FC-MGMT-MIB data/mibs/FC-MGMT-MIB.db
198. VDSL-LINE-EXT-SCM-MIB data/mibs/VDSL-LINE-EXT-SCM-MIB.db
199. VDSL-LINE-EXT-MCM-MIB data/mibs/VDSL-LINE-EXT-MCM-MIB.db
200. UDP-MIB data/mibs/UDP-MIB.db
201. DOCS-IETF-BPI2-MIB data/mibs/DOCS-IETF-BPI2-MIB.db
202. SSPM-MIB data/mibs/SSPM-MIB.db
203. TPM-MIB data/mibs/TPM-MIB.db
204. DISMAN-EVENT-MIB data/mibs/DISMAN-EVENT-MIB.db
205. DISMAN-EXPRESSION-MIB data/mibs/DISMAN-EXPRESSION-MIB.db
206. TE-LINK-STD-MIB data/mibs/TE-LINK-STD-MIB.db
207. VPN-TC-STD-MIB data/mibs/VPN-TC-STD-MIB.db
208. ENTITY-STATE-TC-MIB ENTITY-STATE-MIB data/mibs/ENTITY-STATE-TC-MIB ENTITY-STATE-MIB.db
209. IP-MIB data/mibs/IP-MIB.db
210. MOBILEIPV6-MIB data/mibs/MOBILEIPV6-MIB.db
211. RSTP-MIB data/mibs/RSTP-MIB.db
212. DOCS-IETF-QOS-MIB data/mibs/DOCS-IETF-QOS-MIB.db
213. LMP-MIB data/mibs/LMP-MIB.db
214. MPLS-LC-ATM-STD-MIB MPLS-LC-FR-STD-MIB data/mibs/MPLS-LC-ATM-STD-MIB MPLS-LC-FR-STD-MIB.db
215. IFCP-MGMT-MIB data/mibs/IFCP-MGMT-MIB.db
216. MPLS-L3VPN-STD-MIB data/mibs/MPLS-L3VPN-STD-MIB.db
217. FCIP-MGMT-MIB data/mibs/FCIP-MGMT-MIB.db
218. T11-TC-MIB T11-FC-FABRIC-ADDR-MGR-MIB data/mibs/T11-TC-MIB T11-FC-FABRIC-ADDR-MGR-MIB.db
219. T11-FC-NAME-SERVER-MIB data/mibs/T11-FC-NAME-SERVER-MIB.db

220. ISIS-MIB data/mibs/ISIS-MIB.db
221. SCSI-MIB data/mibs/SCSI-MIB.db
222. AGGREGATE-MIB TIME-AGGREGATE-MIB data/mibs/AGGREGATE-MIB TIME-AGGREGATE-MIB.db
223. ISCSI-MIB data/mibs/ISCSI-MIB.db
224. IPS-AUTH-MIB data/mibs/IPS-AUTH-MIB.db
225. DOCS-IETF-CABLE-DEVICE-NOTIFICATION-MIB data/mibs/DOCS-IETF-CABLE-DEVICE-NOTIFICATION-MIB.db
226. T11-FC-ROUTE-MIB data/mibs/T11-FC-ROUTE-MIB.db
227. PKTC-IETF-MTA-MIB data/mibs/PKTC-IETF-MTA-MIB.db
228. ADSL2-LINE-TC-MIB ADSL2-LINE-MIB data/mibs/ADSL2-LINE-TC-MIB ADSL2-LINE-MIB.db
229. RAQMON-MIB data/mibs/RAQMON-MIB.db
230. RAQMON-RDS-MIB data/mibs/RAQMON-RDS-MIB.db
231. T11-FC-VIRTUAL-FABRIC-MIB data/mibs/T11-FC-VIRTUAL-FABRIC-MIB.db
232. SNMP-IEEE802-TM-MIB data/mibs/SNMP-IEEE802-TM-MIB.db
233. T11-FC-FSPF-MIB data/mibs/T11-FC-FSPF-MIB.db
234. RADIUS-ACC-CLIENT-MIB data/mibs/RADIUS-ACC-CLIENT-MIB.db
235. RADIUS-DYNAUTH-CLIENT-MIB data/mibs/RADIUS-DYNAUTH-CLIENT-MIB.db
236. RADIUS-DYNAUTH-SERVER-MIB data/mibs/RADIUS-DYNAUTH-SERVER-MIB.db
237. GMPLS-TC-STD-MIB data/mibs/GMPLS-TC-STD-MIB.db
238. IANA-GMPLS-TC-MIB data/mibs/IANA-GMPLS-TC-MIB.db
239. GMPLS-TE-STD-MIB data/mibs/GMPLS-TE-STD-MIB.db
240. GMPLS-LSR-STD-MIB data/mibs/GMPLS-LSR-STD-MIB.db
241. IPSEC-SPD-MIB data/mibs/IPSEC-SPD-MIB.db
242. DNS-SERVER-MIB data/mibs/DNS-SERVER-MIB.db
243. DNS-RESOLVER-MIB data/mibs/DNS-RESOLVER-MIB.db
244. DOT3-EPON-MIB data/mibs/DOT3-EPON-MIB.db
245. DOT3-OAM-MIB data/mibs/DOT3-OAM-MIB.db
246. TCP-ESTATS-MIB data/mibs/TCP-ESTATS-MIB.db
247. T11-FC-FABRIC-CONFIG-SERVER-MIB data/mibs/T11-FC-FABRIC-CONFIG-SERVER-MIB.db
248. T11-FC-FABRIC-LOCK-MIB T11-FC-ZONE-SERVER-MIB data/mibs/T11-FC-FABRIC-LOCK-MIB T11-FC-ZONE-SERVER-MIB.db
249. ISNS-MIB data/mibs/ISNS-MIB.db
250. T11-FC-RSCN-MIB data/mibs/T11-FC-RSCN-MIB.db
251. URI-TC-MIB data/mibs/URI-TC-MIB.db
252. IF-CAP-STACK-MIB EFM-CU-MIB .. data/mibs/IF-CAP-STACK-MIB EFM-CU-MIB.db
253. LANGTAG-TC-MIB data/mibs/LANGTAG-TC-MIB.db
254. IPMCAST-MIB data/mibs/IPMCAST-MIB.db
255. DSMON-MIB data/mibs/DSMON-MIB.db
256. NAT-MIB data/mibs/NAT-MIB.db
257. MIDCOM-MIB data/mibs/MIDCOM-MIB.db
258. PIM-BSR-MIB data/mibs/PIM-BSR-MIB.db
259. T11-FC-SP-TC-MIB T11-FC-SP-AUTHENTICATION-MIB T11-FC-SP-ZONING-MIB T11-FC-SP-POLICY-MIB T11-FC-SP-SA-MIB data/mibs/T11-FC-SP-TC-MIB T11-FC-SP-AUTHENTICATION-MIB T11-FC-SP-ZONING-MIB T11-FC-SP-POLICY-MIB T11-FC-SP-SA-MIB.db
260. SYSLOG-TC-MIB data/mibs/SYSLOG-TC-MIB.db
261. PKTC-IETF-EVENT-MIB data/mibs/PKTC-IETF-EVENT-MIB.db
262. NEMO-MIB data/mibs/NEMO-MIB.db
263. RSERPOOL-MIB data/mibs/RSERPOOL-MIB.db
264. PW-TC-STD-MIB data/mibs/PW-TC-STD-MIB.db
265. IANA-PWE3-MIB PW-STD-MIB data/mibs/IANA-PWE3-MIB PW-STD-MIB.db
266. PW-MPLS-STD-MIB data/mibs/PW-MPLS-STD-MIB.db
267. PW-ENET-STD-MIB data/mibs/PW-ENET-STD-MIB.db
268. PW-TDM-MIB data/mibs/PW-TDM-MIB.db
269. PW-ATM-MIB data/mibs/PW-ATM-MIB.db
270. OSPFV3-MIB data/mibs/OSPFV3-MIB.db
271. NHRP-MIB data/mibs/NHRP-MIB.db
272. VDSL2-LINE-TC-MIB VDSL2-LINE-MIB data/mibs/VDSL2-LINE-TC-MIB VDSL2-LINE-MIB.db
273. SYSLOG-MSG-MIB data/mibs/SYSLOG-MSG-MIB.db
274. DVB-RCS-MIB data/mibs/DVB-RCS-MIB.db
275. FORCES-MIB data/mibs/FORCES-MIB.db
276. IPFIX-MIB IPFIX-SELECTOR-MIB . data/mibs/IPFIX-MIB IPFIX-SELECTOR-MIB.db
277. CAPWAP-BASE-MIB data/mibs/CAPWAP-BASE-MIB.db
278. CAPWAP-DOT11-MIB data/mibs/CAPWAP-DOT11-MIB.db
279. NTPv4-MIB data/mibs/NTPv4-MIB.db
280. PW-CEP-STD-MIB data/mibs/PW-CEP-STD-MIB.db
281. MPLS-FRR-GENERAL-STD-MIB MPLS-FRR-ONEZONE-STD-MIB MPLS-FRR-FACILITY-STD-MIB data/mibs/MPLS-FRR-GENERAL-STD-MIB MPLS-FRR-ONEZONE-STD-MIB MPLS-FRR-FACILITY-STD-MIB.db
282. IEEE8021-TC-MIB data/mibs/IEEE8021-TC-MIB.db
283. IEEE8021-BRIDGE-MIB data/mibs/IEEE8021-BRIDGE-MIB.db
284. IEEE8021-MSTP-MIB data/mibs/IEEE8021-MSTP-MIB.db
285. IEEE8021-PBB-MIB data/mibs/IEEE8021-PBB-MIB.db
286. IEEE8021-PB-MIB data/mibs/IEEE8021-PB-MIB.db
287. IEEE8021-Q-BRIDGE-MIB data/mibs/IEEE8021-Q-BRIDGE-MIB.db
288. IEEE8021-SPANNING-TREE-MIB ... data/mibs/IEEE8021-SPANNING-TREE-MIB.db
289. IEEE8021-SECY-MIB data/mibs/IEEE8021-SECY-MIB.db
290. LLDP-V2-TC-MIB data/mibs/LLDP-V2-TC-MIB.db
291. LLDP-V2-MIB data/mibs/LLDP-V2-MIB.db
292. DEFINITIONS data/mibs/DEFINITIONS.db
293. DEFINITIONS data/mibs/DEFINITIONS.db
294. IEEE8021-AS-MIB data/mibs/IEEE8021-AS-MIB.db
295. IEEE8021-EVB-MIB data/mibs/IEEE8021-EVB-MIB.db
296. IEEE8021-FQTS-MIB data/mibs/IEEE8021-FQTS-MIB.db
297. LLDP-EXT-DOT1-V2-MIB data/mibs/LLDP-EXT-DOT1-V2-MIB.db
298. LLDP-EXT-DOT1-EVB-EXTENSIONS-MIB data/mibs/LLDP-EXT-DOT1-EVB-EXTENSIONS-MIB.db

299. DEFINITIONS data/mibs/DEFINITIONS.db
300. DEFINITIONS data/mibs/DEFINITIONS.db
301. IEEE8021-PAE-MIB data/mibs/IEEE8021-PAE-MIB.db
302. IEEE8021-PBBTE-MIB data/mibs/IEEE8021-PBBTE-MIB.db
303. IEEE8021-PE-MIB data/mibs/IEEE8021-PE-MIB.db
304. IEEE8021-PFC-MIB data/mibs/IEEE8021-PFC-MIB.db
305. IEEE8021-SRP-MIB data/mibs/IEEE8021-SRP-MIB.db
306. IEEE8021-TEIPS-V2-MIB data/mibs/IEEE8021-TEIPS-V2-MIB.db
307. IEEE8021-TPMR-MIB data/mibs/IEEE8021-TPMR-MIB.db
308. IEEE8021X-PAE-MIB data/mibs/IEEE8021X-PAE-MIB.db
309. IEEE8023-LAG-MIB data/mibs/IEEE8023-LAG-MIB.db
310. LLDP-EXT-DOT1-PE-MIB data/mibs/LLDP-EXT-DOT1-PE-MIB.db
311. LLDP-MIB data/mibs/LLDP-MIB.db
312. IEEE8021-CFM-MIB data/mibs/IEEE8021-CFM-MIB.db
313. IEEE8021-CFM-V2-MIB data/mibs/IEEE8021-CFM-V2-MIB.db
314. IEEE8021-DDCFM-MIB data/mibs/IEEE8021-DDCFM-MIB.db
315. IEEE8021-SPB-MIB -- -- data/mibs/IEEE8021-SPB-MIB -- --.db
316. IEEE8021-ECMP-MIB -- data/mibs/IEEE8021-ECMP-MIB --.db
317. ESO-CONSORTIUM-MIB data/mibs/ESO-CONSORTIUM-MIB.db
318. PMIPV6-TC-MIB PMIPV6-MIB data/mibs/PMIPV6-TC-MIB PMIPV6-MIB.db
319. VRRPV3-MIB data/mibs/VRRPV3-MIB.db
320. FLOAT-TC-MIB data/mibs/FLOAT-TC-MIB.db
321. PSAMP-MIB data/mibs/PSAMP-MIB.db
322. IANA-GBOND-TC-MIB GBOND-MIB .. data/mibs/IANA-GBOND-TC-MIB GBOND-MIB.db
323. G9983-MIB data/mibs/G9983-MIB.db
324. G9982-MIB data/mibs/G9982-MIB.db
325. G9981-MIB data/mibs/G9981-MIB.db
326. NHDP-MIB data/mibs/NHDP-MIB.db
327. TED-MIB data/mibs/TED-MIB.db
328. RBRIDGE-MIB data/mibs/RBRIDGE-MIB.db
329. RPKI-ROUTER-MIB data/mibs/RPKI-ROUTER-MIB.db
330. LISP-MIB IANA-ADDRESS-FAMILY-NUMBERS-MIB data/mibs/LISP-MIB IANA-ADDRESS-FAMILY-NUMBERS-MIB.db
331. IANA-OLSRv2-LINK-METRIC-TYPE-MIB OLSRv2-MIB data/mibs/IANA-OLSRv2-LINK-METRIC-TYPE-MIB OLSRv2-MIB.db
332. BFD-TC-STD-MIB IANA-BFD-TC-STD-MIB data/mibs/BFD-TC-STD-MIB IANA-BFD-TC-STD-MIB.db
333. BFD-STD-MIB data/mibs/BFD-STD-MIB.db
334. IANA-SMF-MIB SMF-MIB data/mibs/IANA-SMF-MIB SMF-MIB.db
335. LOWPAN-MIB data/mibs/LOWPAN-MIB.db
336. PCE-PCEP-MIB data/mibs/PCE-PCEP-MIB.db
337. MPLS-TC-EXT-STD-MIB MPLS-ID-STD-MIB MPLS-LSR-EXT-STD-MIB MPLS-TE-EXT-STD-MIB data/mibs/MPLS-TC-EXT-STD-MIB MPLS-ID-STD-MIB MPLS-LSR-EXT-STD-MIB MPLS-TE-EXT-STD-MIB.db
338. IANAPowerStateSet-MIB ENERGY-OBJECT-MIB POWER-ATTRIBUTES-MIB data/mibs/IANAPowerStateSet-MIB ENERGY-OBJECT-MIB POWER-ATTRIBUTES-MIB.db
339. IANA-ENERGY-RELATION-MIB ENERGY-OBJECT-CONTEXT-MIB data/mibs/IANA-ENERGY-RELATION-MIB ENERGY-OBJECT-CONTEXT-MIB.db
340. BATTERY-MIB data/mibs/BATTERY-MIB.db
341. NATV2-MIB data/mibs/NATV2-MIB.db
342. IANA-STORAGE-MEDIA-TYPE-MIB VM-MIB data/mibs/IANA-STORAGE-MEDIA-TYPE-MIB VM-MIB.db
343. MPLS-OAM-ID-STD-MIB data/mibs/MPLS-OAM-ID-STD-MIB.db
344. TRILL-OAM-MIB data/mibs/TRILL-OAM-MIB.db
345. SOFTWARE-MESH-MIB data/mibs/SOFTWARE-MESH-MIB.db
346. DSLite-MIB data/mibs/DSLite-MIB.db
347. MPLS-LPS-MIB data/mibs/MPLS-LPS-MIB.db
348. PTPBASE-MIB data/mibs/PTPBASE-MIB.db
349. IANA-LANGUAGE-MIB data/mibs/IANA-LANGUAGE-MIB.db
350. IANA-MALLOC-MIB data/mibs/IANA-MALLOC-MIB.db
351. L2L3-VPN-MULTICAST-TC-MIB L2L3-VPN-MULTICAST-MIB data/mibs/L2L3-VPN-MULTICAST-TC-MIB L2L3-VPN-MULTICAST-MIB.db
352. BGP-MPLS-LAYER3-VPN-MULTICAST-MIB data/mibs/BGP-MPLS-LAYER3-VPN-MULTICAST-MIB.db
353. Job-Monitoring-MIB data/mibs/Job-Monitoring-MIB.db
354. IBM6611-DLS-MIB data/mibs/3com/IBM6611-DLS-MIB.db
355. -- A3com-SNMP-REPEATER-MIB ... data/mibs/3com/-- A3com-SNMP-REPEATER-MIB.db
356. ARTEL-MIB data/mibs/3com/ARTEL-MIB.db
357. LANPLEX-MIB data/mibs/3com/LANPLEX-MIB.db
358. LANPLEX-OPT-PDDI-MIB data/mibs/3com/LANPLEX-OPT-PDDI-MIB.db
359. Spider-MIB data/mibs/3com/Spider-MIB.db
360. SWITCH-MIB data/mibs/3com/SWITCH-MIB.db
361. A3Com-System-r6-MIB data/mibs/3com/A3Com-System-r6-MIB.db
362. CHIP-Wgrp-hub data/mibs/3com/CHIP-Wgrp-hub.db
363. CHIPCOMMIB data/mibs/3com/CHIPCOMMIB.db
364. A3Com-Bridge-MIB data/mibs/3com/A3Com-Bridge-MIB.db
365. A3Com-Filter-MIB data/mibs/3com/A3Com-Filter-MIB.db
366. A3Com-IPSO-MIB data/mibs/3com/A3Com-IPSO-MIB.db
367. A3Com-IPX-MIB data/mibs/3com/A3Com-IPX-MIB.db
368. A3Com-IPXpolicy-MIB data/mibs/3com/A3Com-IPXpolicy-MIB.db
369. A3Com-IPextns-MIB data/mibs/3com/A3Com-IPextns-MIB.db
370. A3Com-PortPath-MIB data/mibs/3com/A3Com-PortPath-MIB.db
371. A3Com-RIP-IPextns-MIB data/mibs/3com/A3Com-RIP-IPextns-MIB.db
372. A3Com-System-MIB data/mibs/3com/A3Com-System-MIB.db
373. BLC-MIB data/mibs/3com/BLC-MIB.db
374. softhub data/mibs/3com/softhub.db
375. LB3GH data/mibs/3com/LB3GH.db
376. -- LBFMS-MIB data/mibs/3com/-- LBFMS-MIB.db
377. -- LBMSH-MIB data/mibs/3com/-- LBMSH-MIB.db
378. LINKB-OPT-PDDI-MIB data/mibs/3com/LINKB-OPT-PDDI-MIB.db

379. -- MSH-MIB data/mibs/3com/-- MSH-MIB.db
380. TermServLineCardMIB data/mibs/3com/TermServLineCardMIB.db
381. GENERIC-3COM-VLAN-MIB-1-0-7 .. data/mibs/3com/GENERIC-3COM-VLAN-MIB-1-0-7.db
382. ANI-MIB data/mibs/accelerated/ANI-MIB.db
383. ANI-TM-PROFILE-MIB data/mibs/accelerated/ANI-TM-PROFILE-MIB.db
384. ANI-VC-MAPPING-MIB data/mibs/accelerated/ANI-VC-MAPPING-MIB.db
385. ANI-VC-STATISTICS-MIB data/mibs/accelerated/ANI-VC-STATISTICS-MIB.db
386. ANI-CPE-PP-MIB data/mibs/accelerated/ANI-CPE-PP-MIB.db
387. ANI-FR-EXTENSIONS-MIB data/mibs/accelerated/ANI-FR-EXTENSIONS-MIB.db
388. ANI-FRATM-NI-MIB data/mibs/accelerated/ANI-FRATM-NI-MIB.db
389. ANI-FRATM-SI-MIB data/mibs/accelerated/ANI-FRATM-SI-MIB.db
390. ANI-FRGENERAL-MIB data/mibs/accelerated/ANI-FRGENERAL-MIB.db
391. ANI-FR-PM-MIB data/mibs/accelerated/ANI-FR-PM-MIB.db
392. ANI-CPE-FR-PP-MIB data/mibs/accelerated/ANI-CPE-FR-PP-MIB.db
393. ANI-FRSW-MIB data/mibs/accelerated/ANI-FRSW-MIB.db
394. ANI-HDLCATM-MIB data/mibs/accelerated/ANI-HDLCATM-MIB.db
395. ANI-CPE-PROXY-PORT-MIB data/mibs/accelerated/ANI-CPE-PROXY-PORT-MIB.db
396. ANI-CPE-CONN-MIB data/mibs/accelerated/ANI-CPE-CONN-MIB.db
397. ANI-CPE-TLS-MIB data/mibs/accelerated/ANI-CPE-TLS-MIB.db
398. ANI-CPE-DHCP-MIB data/mibs/accelerated/ANI-CPE-DHCP-MIB.db
399. ANI-CPE-PROXY-SYSTEM-MIB data/mibs/accelerated/ANI-CPE-PROXY-SYSTEM-MIB.db
400. ANI-CPE-VOICE-PP-MIB data/mibs/accelerated/ANI-CPE-VOICE-PP-MIB.db
401. ANI-CLOCK-SYNC-MIB data/mibs/accelerated/ANI-CLOCK-SYNC-MIB.db
402. ANI-CDB-MIB data/mibs/accelerated/ANI-CDB-MIB.db
403. ANI-CPE-LP-MIB data/mibs/accelerated/ANI-CPE-LP-MIB.db
404. ANI-DS3-MIB data/mibs/accelerated/ANI-DS3-MIB.db
405. ANI-ENET-MIB data/mibs/accelerated/ANI-ENET-MIB.db
406. ANI-OC3-MIB data/mibs/accelerated/ANI-OC3-MIB.db
407. ANI-PORTMANAGER-MIB data/mibs/accelerated/ANI-PORTMANAGER-MIB.db
408. ANI-QOSCONFIG-MIB data/mibs/accelerated/ANI-QOSCONFIG-MIB.db
409. ATM-MIB data/mibs/accelerated/ATM-MIB.db
410. ANI-RIP-GLOBAL-MIB data/mibs/accelerated/ANI-RIP-GLOBAL-MIB.db
411. ANI-SDSL-MIB data/mibs/accelerated/ANI-SDSL-MIB.db
412. ANI-SHM-MIB data/mibs/accelerated/ANI-SHM-MIB.db
413. ANI-SSCBRSRVCS-MIB data/mibs/accelerated/ANI-SSCBRSRVCS-MIB.db
414. ANI-GEN-SYS-INFO-MIB data/mibs/accelerated/ANI-GEN-SYS-INFO-MIB.db
415. ANI-SYSMOD-MIB data/mibs/accelerated/ANI-SYSMOD-MIB.db
416. ANI-T1-MIB data/mibs/accelerated/ANI-T1-MIB.db
417. ANI-CPE-VOICE-CHAN-MIB data/mibs/accelerated/ANI-CPE-VOICE-CHAN-MIB.db
418. ANI-VC-RESOURCE-MIB data/mibs/accelerated/ANI-VC-RESOURCE-MIB.db
419. ANI-VOICECONN-MIB data/mibs/accelerated/ANI-VOICECONN-MIB.db
420. PowerNet-MIB data/mibs/apc/PowerNet-MIB.db
421. ASCEND-MIB data/mibs/ascend/ASCEND-MIB.db
422. ASCEND-WAN-MIB data/mibs/ascend/ASCEND-WAN-MIB.db
423. ASCEND-ADSL-CAP-MIB data/mibs/ascend/ASCEND-ADSL-CAP-MIB.db
424. ASCEND-ADVANCED-AGENT-MIB data/mibs/ascend/ASCEND-ADVANCED-AGENT-MIB.db
425. ASCEND-CALL-MIB data/mibs/ascend/ASCEND-CALL-MIB.db
426. ASCEND-EVENT-MIB data/mibs/ascend/ASCEND-EVENT-MIB.db
427. ASCEND-FIREWALL-MIB data/mibs/ascend/ASCEND-FIREWALL-MIB.db
428. ASCEND-LANMODEM-MIB data/mibs/ascend/ASCEND-LANMODEM-MIB.db
429. ASCEND-MCAST-MIB data/mibs/ascend/ASCEND-MCAST-MIB.db
430. ASCEND-MULTI-SHELF-MIB data/mibs/ascend/ASCEND-MULTI-SHELF-MIB.db
431. ASCEND-POWER-SUPPLY-MIB data/mibs/ascend/ASCEND-POWER-SUPPLY-MIB.db
432. ASCEND-RADIUS-MIB data/mibs/ascend/ASCEND-RADIUS-MIB.db
433. ASCEND-SDSL-MIB data/mibs/ascend/ASCEND-SDSL-MIB.db
434. ASCEND-SESSION-MIB data/mibs/ascend/ASCEND-SESSION-MIB.db
435. ASCEND-WAN-DIALOUT-PKT-MIB ... data/mibs/ascend/ASCEND-WAN-DIALOUT-PKT-MIB.db
436. SH-ATT-MIB data/mibs/att/SH-ATT-MIB.db
437. ATT-CNM-DS1-MIB data/mibs/att/ATT-CNM-DS1-MIB.db
438. ATT-CNM-DS3-MIB data/mibs/att/ATT-CNM-DS3-MIB.db
439. ATT-CNM-ENHANCED-FRAME-RELAY-MIB data/mibs/att/ATT-CNM-ENHANCED-FRAME-RELAY-MIB.db
440. ATT-CNM-FRAME-RELAY-MIB data/mibs/att/ATT-CNM-FRAME-RELAY-MIB.db
441. ATT-CNM-SIP-MIB data/mibs/att/ATT-CNM-SIP-MIB.db
442. ATT-CNM-SMDS-MIB data/mibs/att/ATT-CNM-SMDS-MIB.db
443. ATT-CNM-SYSTEM-MIB data/mibs/att/ATT-CNM-SYSTEM-MIB.db
444. RH-ATT-MIB data/mibs/att/RH-ATT-MIB.db
445. SYNOPTICS-ROOT-MIB data/mibs/bay/SYNOPTICS-ROOT-MIB.db
446. KAL-BASE-MIB data/mibs/bay/KAL-BASE-MIB.db
447. SYNOPTICS-LS-ROOT-MIB data/mibs/bay/SYNOPTICS-LS-ROOT-MIB.db
448. S5-ROOT-MIB data/mibs/bay/S5-ROOT-MIB.db
449. KAL-ES-MIB data/mibs/bay/KAL-ES-MIB.db
450. SYNOPTICS-COMMON-MIB data/mibs/bay/SYNOPTICS-COMMON-MIB.db
451. SYNOPTICS-FDDI-MIB data/mibs/bay/SYNOPTICS-FDDI-MIB.db
452. SYNOPTICS-FDDI-SNMP-MIB data/mibs/bay/SYNOPTICS-FDDI-SNMP-MIB.db
453. SYNOPTICS-FDDI-OPT-MIB data/mibs/bay/SYNOPTICS-FDDI-OPT-MIB.db
454. SYNOPTICS-IEEB8023-MIB data/mibs/bay/SYNOPTICS-IEEB8023-MIB.db
455. SYNOPTICS-IPX-MIB data/mibs/bay/SYNOPTICS-IPX-MIB.db
456. SYNOPTICS-LS-TCS-MIB data/mibs/bay/SYNOPTICS-LS-TCS-MIB.db
457. SYNOPTICS-LS-AGENT-MIB data/mibs/bay/SYNOPTICS-LS-AGENT-MIB.db
458. SYNOPTICS-LS-CHASSIS-MIB data/mibs/bay/SYNOPTICS-LS-CHASSIS-MIB.db
459. SYNOPTICS-LS-DCM-MIB data/mibs/bay/SYNOPTICS-LS-DCM-MIB.db

460. SYNOPTICS-LS-ETHERNET-MIB data/mibs/bay/SYNOPTICS-LS-ETHERNET-MIB.db
461. SYNOPTICS-LS-CONFIG-MIB data/mibs/bay/SYNOPTICS-LS-CONFIG-MIB.db
462. SYNOPTICS-LS-REG-MIB data/mibs/bay/SYNOPTICS-LS-REG-MIB.db
463. SYNOPTICS-ETHERNET-MIB data/mibs/bay/SYNOPTICS-ETHERNET-MIB.db
464. SYNOPTICS-TOKENRING-MIB data/mibs/bay/SYNOPTICS-TOKENRING-MIB.db
465. S5-TCS-MIB data/mibs/bay/S5-TCS-MIB.db
466. S5-AGENT-MIB data/mibs/bay/S5-AGENT-MIB.db
467. S5-COMMON-BAYSECURE-MIB data/mibs/bay/S5-COMMON-BAYSECURE-MIB.db
468. S5-CHASSIS-MIB data/mibs/bay/S5-CHASSIS-MIB.db
469. S5-ETHERNET-MIB data/mibs/bay/S5-ETHERNET-MIB.db
470. S5-COMMON-STATS-MIB data/mibs/bay/S5-COMMON-STATS-MIB.db
471. S5-ETHERNET-COMMON-MIB data/mibs/bay/S5-ETHERNET-COMMON-MIB.db
472. S5-ETH-MULTISEG-TOPOLOGY-MIB . data/mibs/bay/S5-ETH-MULTISEG-TOPOLOGY-MIB.db
473. S5-ETH-REDUNDANT-LINKS-MIB ... data/mibs/bay/S5-ETH-REDUNDANT-LINKS-MIB.db
474. S5-ETH-TOPOLOGY-MIB data/mibs/bay/S5-ETH-TOPOLOGY-MIB.db
475. S5-REG-MIB data/mibs/bay/S5-REG-MIB.db
476. S5-TOKENRING-MIB data/mibs/bay/S5-TOKENRING-MIB.db
477. S5-TOKENRING-COMMON-MIB data/mibs/bay/S5-TOKENRING-COMMON-MIB.db
478. S5-TOK-TOPOLOGY-MIB data/mibs/bay/S5-TOK-TOPOLOGY-MIB.db
479. SYNOPTICS-SUPERAGENT-MIB data/mibs/bay/SYNOPTICS-SUPERAGENT-MIB.db
480. SYNOPTICS-FDDI-SMT73-MIB data/mibs/bay/SYNOPTICS-FDDI-SMT73-MIB.db
481. SYNOPTICS-TRB-MIB data/mibs/bay/SYNOPTICS-TRB-MIB.db
482. ANNEX-MIB-R8 data/mibs/bay/ANNEX-MIB-R8.db
483. LATTISCELL-MIB data/mibs/bay/LATTISCELL-MIB.db
484. RETIX-BASE-MIB data/mibs/bay/RETIX-BASE-MIB.db
485. SYNOPTICS-ATM-ETH-MIB data/mibs/bay/SYNOPTICS-ATM-ETH-MIB.db
486. S5-COMMON-TRAP-MIB data/mibs/bay/S5-COMMON-TRAP-MIB.db
487. SYNOPTICS-COMMON-TRAP-MIB data/mibs/bay/SYNOPTICS-COMMON-TRAP-MIB.db
488. SYNOPTICS-LATTISCELL-MIB data/mibs/bay/SYNOPTICS-LATTISCELL-MIB.db
489. SYNOPTICS-ETHERCELL-TRAP-MIB . data/mibs/bay/SYNOPTICS-ETHERCELL-TRAP-MIB.db
490. SYNOPTICS-LS-TRAP-MIB data/mibs/bay/SYNOPTICS-LS-TRAP-MIB.db
491. SYNOPTICS-ETHERNET-TRAP-MIB .. data/mibs/bay/SYNOPTICS-ETHERNET-TRAP-MIB.db
492. SYNOPTICS-ORION-MIB data/mibs/bay/SYNOPTICS-ORION-MIB.db
493. SYNOPTICS-FDDI-COM-TRAP-MIB .. data/mibs/bay/SYNOPTICS-FDDI-COM-TRAP-MIB.db
494. SYNOPTICS-FDDI-TRAP-MIB data/mibs/bay/SYNOPTICS-FDDI-TRAP-MIB.db
495. SYNOPTICS-TOKENRING-TRAP-MIB . data/mibs/bay/SYNOPTICS-TOKENRING-TRAP-MIB.db
496. LAN-EMULATION-CLIENT-MIB data/mibs/bay/LAN-EMULATION-CLIENT-MIB.db
497. LAN-EMULATION-ELAN-MIB data/mibs/bay/LAN-EMULATION-ELAN-MIB.db
498. LAN-EMULATION-LES-MIB data/mibs/bay/LAN-EMULATION-LES-MIB.db
499. LAN-EMULATION-BUS-MIB data/mibs/bay/LAN-EMULATION-BUS-MIB.db
500. ATM-FORUM-TC-MIB data/mibs/bay/ATM-FORUM-TC-MIB.db
501. ATM-FORUM-SRVC-REG data/mibs/bay/ATM-FORUM-SRVC-REG.db
502. ATM-FORUM-ADDR-REG data/mibs/bay/ATM-FORUM-ADDR-REG.db
503. -- ATM-TC-MIB data/mibs/bay/-- ATM-TC-MIB.db
504. BNET-ATM-ATOM-AUG-MIB data/mibs/bay/BNET-ATM-ATOM-AUG-MIB.db
505. ATM-FORUM-MIB data/mibs/bay/ATM-FORUM-MIB.db
506. BNET-ATM-TOPOLOGY-MIB data/mibs/bay/BNET-ATM-TOPOLOGY-MIB.db
507. ATM-MIB data/mibs/bay/ATM-MIB.db
508. ATM2-MIB data/mibs/bay/ATM2-MIB.db
509. CENTILLION-ROOT-MIB data/mibs/bay/CENTILLION-ROOT-MIB.db
510. CENTILLION-BRIDGE-MIB data/mibs/bay/CENTILLION-BRIDGE-MIB.db
511. CENTILLION-BRIDGEGROUP-MIB ... data/mibs/bay/CENTILLION-BRIDGEGROUP-MIB.db
512. CENTILLION-PORT-RING-MASTER-MONITOR-MIB data/mibs/bay/CENTILLION-PORT-RING-MASTER-MONITOR-MIB.db
513. CENTILLION-SYSTEM-MONITOR-MIB data/mibs/bay/CENTILLION-SYSTEM-MONITOR-MIB.db
514. CENTILLION-ATMMON-MIB data/mibs/bay/CENTILLION-ATMMON-MIB.db
515. CENTILLION-CONFIG-MIB data/mibs/bay/CENTILLION-CONFIG-MIB.db
516. CENTILLION-DOT3-EXTENSIONS-MIB data/mibs/bay/CENTILLION-DOT3-EXTENSIONS-MIB.db
517. CENTILLION-DOT5-EXTENSIONS-MIB data/mibs/bay/CENTILLION-DOT5-EXTENSIONS-MIB.db
518. RFC1407-EXT-MIB data/mibs/bay/RFC1407-EXT-MIB.db
519. CENTILLION-FDB-MIB data/mibs/bay/CENTILLION-FDB-MIB.db
520. CENTILLION-FILTERS-MIB data/mibs/bay/CENTILLION-FILTERS-MIB.db
521. CENTILLION-IF-EXTENSIONS-MIB . data/mibs/bay/CENTILLION-IF-EXTENSIONS-MIB.db
522. CENTILLION-LANE-V2-EXT-MIB ... data/mibs/bay/CENTILLION-LANE-V2-EXT-MIB.db
523. CENTILLION-LESBUS-MIB data/mibs/bay/CENTILLION-LESBUS-MIB.db
524. PNNI-EXT-MIB data/mibs/bay/PNNI-EXT-MIB.db
525. CENTILLION-SONET-MIB data/mibs/bay/CENTILLION-SONET-MIB.db
526. CENTILLION-SOURCE-ROUTING-MIB data/mibs/bay/CENTILLION-SOURCE-ROUTING-MIB.db
527. CENTILLION-VIRTUALSEGMENT-MIB data/mibs/bay/CENTILLION-VIRTUALSEGMENT-MIB.db
528. CENTILLION-ATMCFG-MIB data/mibs/bay/CENTILLION-ATMCFG-MIB.db
529. LANPLEX-MIB data/mibs/bay/LANPLEX-MIB.db
530. CENTILLION-MCAST-MIB data/mibs/bay/CENTILLION-MCAST-MIB.db
531. SYNOPTICS-MULTICAST-SERVER-MIB data/mibs/bay/SYNOPTICS-MULTICAST-SERVER-MIB.db
532. PNNI-MIB data/mibs/bay/PNNI-MIB.db
533. SYNOPTICS-RMON-EXT-MIB data/mibs/bay/SYNOPTICS-RMON-EXT-MIB.db
534. S5-CHASSIS-TRAP-MIB data/mibs/bay/S5-CHASSIS-TRAP-MIB.db
535. BN-IF-EXTENSIONS-MIB data/mibs/bay/BN-IF-EXTENSIONS-MIB.db
536. REMOTE-LOGIN-TRAP-MIB data/mibs/bay/REMOTE-LOGIN-TRAP-MIB.db
537. S5-SWITCH-BAYSECURE-MIB data/mibs/bay/S5-SWITCH-BAYSECURE-MIB.db
538. S5-ETHERNET-TRAP-MIB data/mibs/bay/S5-ETHERNET-TRAP-MIB.db
539. S5-TOKENRING-TRAP-MIB data/mibs/bay/S5-TOKENRING-TRAP-MIB.db
540. Netwave-MIB data/mibs/bay/Netwave-MIB.db

541.	BN-LOG-MESSAGE-MIB	data/mibs/bay/BN-LOG-MESSAGE-MIB.db
542.	POLICY-FRAMEWORK-PIB	data/mibs/bay/POLICY-FRAMEWORK-PIB.db
543.	NTN-QOS-POLICY-EVOL-PIB	data/mibs/bay/NTN-QOS-POLICY-EVOL-PIB.db
544.	IEEE8021-PAE-MIB	data/mibs/bay/IEEE8021-PAE-MIB.db
545.	BAY-STACK-MIB	data/mibs/bay/BAY-STACK-MIB.db
546.	BAY-STACK-NOTIFICATIONS-MIB	data/mibs/bay/BAY-STACK-NOTIFICATIONS-MIB.db
547.	BAY-STACK-EAPOL-EXTENSION-MIB	data/mibs/bay/BAY-STACK-EAPOL-EXTENSION-MIB.db
548.	BAY-STACK-MULTICAST-FLOODING-MIB	data/mibs/bay/BAY-STACK-MULTICAST-FLOODING-MIB.db
549.	BAY-STACK-LACP-EXT-MIB	data/mibs/bay/BAY-STACK-LACP-EXT-MIB.db
550.	BAY-STACK-PETH-EXT-MIB	data/mibs/bay/BAY-STACK-PETH-EXT-MIB.db
551.	BAY-STACK-ADAC-MIB	data/mibs/bay/BAY-STACK-ADAC-MIB.db
552.	BAY-STACK-VRRP-EXT-MIB	data/mibs/bay/BAY-STACK-VRRP-EXT-MIB.db
553.	BAY-STACK-STATS-MIB	data/mibs/bay/BAY-STACK-STATS-MIB.db
554.	BAY-STACK-OSPF-EXT-MIB	data/mibs/bay/BAY-STACK-OSPF-EXT-MIB.db
555.	BAY-STACK-ECMP-MIB	data/mibs/bay/BAY-STACK-ECMP-MIB.db
556.	BAY-STACK-DHCP-SNOOPING-MIB	data/mibs/bay/BAY-STACK-DHCP-SNOOPING-MIB.db
557.	BAY-STACK-ARP-INSPECTION-MIB	data/mibs/bay/BAY-STACK-ARP-INSPECTION-MIB.db
558.	BAY-STACK-ERROR-MESSAGE-MIB	data/mibs/bay/BAY-STACK-ERROR-MESSAGE-MIB.db
559.	BAY-STACK-SOURCE-GUARD-MIB	data/mibs/bay/BAY-STACK-SOURCE-GUARD-MIB.db
560.	BAY-STACK-RADIUS-MIB	data/mibs/bay/BAY-STACK-RADIUS-MIB.db
561.	BAY-STACK-PIM-EXT-MIB	data/mibs/bay/BAY-STACK-PIM-EXT-MIB.db
562.	BAY-STACK-PORT-MIRRORING-MIB	data/mibs/bay/BAY-STACK-PORT-MIRRORING-MIB.db
563.	BAY-STACK-NES-MIB	data/mibs/bay/BAY-STACK-NES-MIB.db
564.	BAY-STACK-EDM-MIB	data/mibs/bay/BAY-STACK-EDM-MIB.db
565.	BAY-STACK-IGMP-EXT-MIB	data/mibs/bay/BAY-STACK-IGMP-EXT-MIB.db
566.	BAY-STACK-IP-FWD-NH-MIB	data/mibs/bay/BAY-STACK-IP-FWD-NH-MIB.db
567.	LLDP-MIB	data/mibs/bay/LLDP-MIB.db
568.	BAY-STACK-LLDP-EXT-MED-MIB	data/mibs/bay/BAY-STACK-LLDP-EXT-MED-MIB.db
569.	BAY-STACK-NOTIFY-CONTROL-MIB	data/mibs/bay/BAY-STACK-NOTIFY-CONTROL-MIB.db
570.	BAY-STACK-UNICAST-STORM-CONTROL-MIB	data/mibs/bay/BAY-STACK-UNICAST-STORM-CONTROL-MIB.db
571.	CTRON-OIDS	data/mibs/cabletron/CTRON-OIDS.db
572.	IRM-OIDS	data/mibs/cabletron/IRM-OIDS.db
573.	CTRON-MIB-NAMES	data/mibs/cabletron/CTRON-MIB-NAMES.db
574.	ROUTER-OIDS	data/mibs/cabletron/ROUTER-OIDS.db
575.	TR-STNASSIGN-MIB	data/mibs/cabletron/TR-STNASSIGN-MIB.db
576.	EVENT-ACTIONS-MIB	data/mibs/cabletron/EVENT-ACTIONS-MIB.db
577.	IRM3-MIB	data/mibs/cabletron/IRM3-MIB.db
578.	REPEATER-MIB-2	data/mibs/cabletron/REPEATER-MIB-2.db
579.	CABLETRON-TRAPS-IRM	data/mibs/cabletron/CABLETRON-TRAPS-IRM.db
580.	CHASSIS-MIB	data/mibs/cabletron/CHASSIS-MIB.db
581.	CTRON-COMMON-MIB	data/mibs/cabletron/CTRON-COMMON-MIB.db
582.	COMMUNITY-MIB	data/mibs/cabletron/COMMUNITY-MIB.db
583.	CT-CONTAINER-MIB	data/mibs/cabletron/CT-CONTAINER-MIB.db
584.	CTRON-DLSW-MIB	data/mibs/cabletron/CTRON-DLSW-MIB.db
585.	CTATX-MIB	data/mibs/cabletron/CTATX-MIB.db
586.	FN10-MIB	data/mibs/cabletron/FN10-MIB.db
587.	FN100-MIB	data/mibs/cabletron/FN100-MIB.db
588.	CTRON-APPN-MIB	data/mibs/cabletron/CTRON-APPN-MIB.db
589.	CTRON-AppleTalk-ROUTER-MIB	data/mibs/cabletron/CTRON-AppleTalk-ROUTER-MIB.db
590.	CTATM-CONFIG-MIB	data/mibs/cabletron/CTATM-CONFIG-MIB.db
591.	CTRON-BRIDGE-MIB	data/mibs/cabletron/CTRON-BRIDGE-MIB.db
592.	CT-BROADCAST-MIB	data/mibs/cabletron/CT-BROADCAST-MIB.db
593.	CTRON-CSMACD-MIB	data/mibs/cabletron/CTRON-CSMACD-MIB.db
594.	CTRON-DECIV-ROUTER-MIB	data/mibs/cabletron/CTRON-DECIV-ROUTER-MIB.db
595.	CTRON-DEVICE-MIB	data/mibs/cabletron/CTRON-DEVICE-MIB.db
596.	CTRON-DOWNLOAD-MIB	data/mibs/cabletron/CTRON-DOWNLOAD-MIB.db
597.	CTRON-CHASSIS-MIB	data/mibs/cabletron/CTRON-CHASSIS-MIB.db
598.	CTRON-ENVIRONMENT-MIB	data/mibs/cabletron/CTRON-ENVIRONMENT-MIB.db
599.	CTRON-FDDI-FNB-MIB	data/mibs/cabletron/CTRON-FDDI-FNB-MIB.db
600.	CTRON-FDDI-STAT-MIB	data/mibs/cabletron/CTRON-FDDI-STAT-MIB.db
601.	CT-FLASH-MIB	data/mibs/cabletron/CT-FLASH-MIB.db
602.	CTRON-FRONTANEL-MIB	data/mibs/cabletron/CTRON-FRONTANEL-MIB.db
603.	CTRON-IF-REMAP-2-MIB	data/mibs/cabletron/CTRON-IF-REMAP-2-MIB.db
604.	CTRON-IF-REMAP-MIB	data/mibs/cabletron/CTRON-IF-REMAP-MIB.db
605.	CTRON-IMIM-ADDRESS-MIB	data/mibs/cabletron/CTRON-IMIM-ADDRESS-MIB.db
606.	CTRON-IP-ROUTER-MIB	data/mibs/cabletron/CTRON-IP-ROUTER-MIB.db
607.	CTRON-IPX-ROUTER-MIB	data/mibs/cabletron/CTRON-IPX-ROUTER-MIB.db
608.	NETWORK-DIAGS	data/mibs/cabletron/NETWORK-DIAGS.db
609.	CT-PIC-MIB	data/mibs/cabletron/CT-PIC-MIB.db
610.	CTRON-POWER-SUPPLY-MIB	data/mibs/cabletron/CTRON-POWER-SUPPLY-MIB.db
611.	CTRON-PRIORITY-EXTENSIONS-MIB	data/mibs/cabletron/CTRON-PRIORITY-EXTENSIONS-MIB.db
612.	CTRMONT-MIB	data/mibs/cabletron/CTRMONT-MIB.db
613.	CTRON-FNBTR-MIB	data/mibs/cabletron/CTRON-FNBTR-MIB.db
614.	CTRON-BDG-MIB	data/mibs/cabletron/CTRON-BDG-MIB.db
615.	CTRON-ROUTERS-MIB	data/mibs/cabletron/CTRON-ROUTERS-MIB.db
616.	CTSMTMIB-MIB	data/mibs/cabletron/CTSMTMIB-MIB.db
617.	CTRON-TRANSLATION-MIB	data/mibs/cabletron/CTRON-TRANSLATION-MIB.db
618.	CTRON-UPS-MIB	data/mibs/cabletron/CTRON-UPS-MIB.db
619.	UPS2-MIB	data/mibs/cabletron/UPS2-MIB.db
620.	CTRON-WAN-MIB	data/mibs/cabletron/CTRON-WAN-MIB.db
621.	DLM-MIB	data/mibs/cabletron/DLM-MIB.db

622. DOT5-LOG-MIB data/mibs/cabletron/DOT5-LOG-MIB.db
623. DOT5-PHYS-MIB data/mibs/cabletron/DOT5-PHYS-MIB.db
624. CTRON-ETWMIM-MIB data/mibs/cabletron/CTRON-ETWMIM-MIB.db
625. FAST-ETHERNET-MIB data/mibs/cabletron/FAST-ETHERNET-MIB.db
626. NB30MIB data/mibs/cabletron/NB30MIB.db
627. REPEATER-REV4-MIB data/mibs/cabletron/REPEATER-REV4-MIB.db
628. SYSTEM-RESOURCE-MIB data/mibs/cabletron/SYSTEM-RESOURCE-MIB.db
629. TRAP-MIB data/mibs/cabletron/TRAP-MIB.db
630. CT-FASTPATH-DHCPSERVER-MIB ... data/mibs/cabletron/CT-FASTPATH-DHCPSERVER-MIB.db
631. CT-FASTPATH-PROTECTED-PORT-MIB data/mibs/cabletron/CT-FASTPATH-PROTECTED-PORT-MIB.db
632. CTRON-ALIAS-MIB data/mibs/cabletron/CTRON-ALIAS-MIB.db
633. CTRON-AP3000-MIB data/mibs/cabletron/CTRON-AP3000-MIB.db
634. CTRON-CDP-MIB data/mibs/cabletron/CTRON-CDP-MIB.db
635. CTIF-EXT-MIB data/mibs/cabletron/CTIF-EXT-MIB.db
636. CTRON-ETHERNET-PARAMETERS-MIB data/mibs/cabletron/CTRON-ETHERNET-PARAMETERS-MIB.db
637. CTRON-IGMP-MIB data/mibs/cabletron/CTRON-IGMP-MIB.db
638. CTRON-PPC-BAD-PACKETS-MIB data/mibs/cabletron/CTRON-PPC-BAD-PACKETS-MIB.db
639. CTRON-PRIORITY-CLASSIFY-MIB .. data/mibs/cabletron/CTRON-PRIORITY-CLASSIFY-MIB.db
640. CTRON-Q-BRIDGE-MIB-EXT data/mibs/cabletron/CTRON-Q-BRIDGE-MIB-EXT.db
641. CTRON-RATE-POLICING-MIB data/mibs/cabletron/CTRON-RATE-POLICING-MIB.db
642. CTRON-SMARTTRUNK-MIB data/mibs/cabletron/CTRON-SMARTTRUNK-MIB.db
643. CTRON-TIMED-RESET-MIB data/mibs/cabletron/CTRON-TIMED-RESET-MIB.db
644. CTRON-TX-QUEUE-ARBITRATION-MIB data/mibs/cabletron/CTRON-TX-QUEUE-ARBITRATION-MIB.db
645. CTRON-VLAN-CLASSIFY-MIB data/mibs/cabletron/CTRON-VLAN-CLASSIFY-MIB.db
646. CTRON-WEBVIEW-MIB data/mibs/cabletron/CTRON-WEBVIEW-MIB.db
647. ELS100-S24TX2M-MIB data/mibs/cabletron/ELS100-S24TX2M-MIB.db
648. V2H124-24-MIB data/mibs/cabletron/V2H124-24-MIB.db
649. CT-FASTPATH-ARPAACL-MIB data/mibs/cabletron/CT-FASTPATH-ARPAACL-MIB.db
650. CT-FASTPATH-DHCPSNOOPING-MIB . data/mibs/cabletron/CT-FASTPATH-DHCPSNOOPING-MIB.db
651. CT-FASTPATH-DYNAMIC-ARP-INSPECTION-MIB data/mibs/cabletron/CT-FASTPATH-DYNAMIC-ARP-INSPECTION-MIB.db
652. CTFPS-MIB data/mibs/cabletron/CTFPS-MIB.db
653. CT-FPS-SERVICES-MIB data/mibs/cabletron/CT-FPS-SERVICES-MIB.db
654. CTFRAMER-CONFIG-MIB data/mibs/cabletron/CTFRAMER-CONFIG-MIB.db
655. CT-HSIMPHYS-MIB data/mibs/cabletron/CT-HSIMPHYS-MIB.db
656. CT-CMMPHYS-MIB data/mibs/cabletron/CT-CMMPHYS-MIB.db
657. CTINB2-MIB data/mibs/cabletron/CTINB2-MIB.db
658. CTINB-MIB data/mibs/cabletron/CTINB-MIB.db
659. CT-PRIORITY-CLASSIFY-MIB data/mibs/cabletron/CT-PRIORITY-CLASSIFY-MIB.db
660. CT-PRIORITY-QUEUEING data/mibs/cabletron/CT-PRIORITY-QUEUEING.db
661. CTRON-BUS-MIB data/mibs/cabletron/CTRON-BUS-MIB.db
662. CTRON-DHCP-MIB data/mibs/cabletron/CTRON-DHCP-MIB.db
663. CTRON-ELAN-MIB data/mibs/cabletron/CTRON-ELAN-MIB.db
664. CTRON-ENTITY-STATE-TC-MIB data/mibs/cabletron/CTRON-ENTITY-STATE-TC-MIB.db
665. CTRON-ENTITY-STATE-MIB data/mibs/cabletron/CTRON-ENTITY-STATE-MIB.db
666. CTRON-NAT-MIB data/mibs/cabletron/CTRON-NAT-MIB.db
667. CTRON-ORP-HSIM-MIB data/mibs/cabletron/CTRON-ORP-HSIM-MIB.db
668. CTRON-PORTMAP-MIB data/mibs/cabletron/CTRON-PORTMAP-MIB.db
669. CTRON-REMOTE-ACCESS-MIB data/mibs/cabletron/CTRON-REMOTE-ACCESS-MIB.db
670. CTRON-ROUTERS-INTERNAL-MIB ... data/mibs/cabletron/CTRON-ROUTERS-INTERNAL-MIB.db
671. CTRON-SFCS-MIB data/mibs/cabletron/CTRON-SFCS-MIB.db
672. CTRON-SFPPS-INCLUDE-MIB data/mibs/cabletron/CTRON-SFPPS-INCLUDE-MIB.db
673. CTRON-SFPPS-BASE-MIB data/mibs/cabletron/CTRON-SFPPS-BASE-MIB.db
674. CTRON-SFPPS-BINDERY-MIB data/mibs/cabletron/CTRON-SFPPS-BINDERY-MIB.db
675. CTRON-SFPPS-CALL-MIB data/mibs/cabletron/CTRON-SFPPS-CALL-MIB.db
676. CTRON-SFPPS-CHASSIS-MIB data/mibs/cabletron/CTRON-SFPPS-CHASSIS-MIB.db
677. CTRON-SFPPS-COMMON-MIB data/mibs/cabletron/CTRON-SFPPS-COMMON-MIB.db
678. CTRON-SFPPS-CONNECTION-MIB data/mibs/cabletron/CTRON-SFPPS-CONNECTION-MIB.db
679. CTRON-SFPPS-CONN-MIB data/mibs/cabletron/CTRON-SFPPS-CONN-MIB.db
680. CTRON-SFPPS-DIAGSTATS-MIB data/mibs/cabletron/CTRON-SFPPS-DIAGSTATS-MIB.db
681. CTRON-SFPPS-DIRECTORY-MIB data/mibs/cabletron/CTRON-SFPPS-DIRECTORY-MIB.db
682. CTRON-SFPPS-ESYS-MIB data/mibs/cabletron/CTRON-SFPPS-ESYS-MIB.db
683. CTRON-SFPPS-EVENTLOG-MIB data/mibs/cabletron/CTRON-SFPPS-EVENTLOG-MIB.db
684. CTRON-SFPPS-FLOOD-MIB data/mibs/cabletron/CTRON-SFPPS-FLOOD-MIB.db
685. CTRON-SFPPS-L4SS-MIB data/mibs/cabletron/CTRON-SFPPS-L4SS-MIB.db
686. CTRON-SFPPS-MCAST-MIB data/mibs/cabletron/CTRON-SFPPS-MCAST-MIB.db
687. CTRON-SFPPS-PATH-MIB data/mibs/cabletron/CTRON-SFPPS-PATH-MIB.db
688. CTRON-SFPPS-PKTGR-MIB data/mibs/cabletron/CTRON-SFPPS-PKTGR-MIB.db
689. CTRON-SFPPS-POLICY-MIB data/mibs/cabletron/CTRON-SFPPS-POLICY-MIB.db
690. CTRON-SFPPS-PORT-MIB data/mibs/cabletron/CTRON-SFPPS-PORT-MIB.db
691. CTRON-SFPPS-RESOLVE-MIB data/mibs/cabletron/CTRON-SFPPS-RESOLVE-MIB.db
692. CTRON-SFPPS-SFLSP-MIB data/mibs/cabletron/CTRON-SFPPS-SFLSP-MIB.db
693. CTRON-SFPPS-SIZE-MIB data/mibs/cabletron/CTRON-SFPPS-SIZE-MIB.db
694. CTRON-SFPPS-SOFTLINK-MIB data/mibs/cabletron/CTRON-SFPPS-SOFTLINK-MIB.db
695. CTRON-SFPPS-TAP-MIB data/mibs/cabletron/CTRON-SFPPS-TAP-MIB.db
696. CTRON-SFPPS-TOPOLOGY-MIB data/mibs/cabletron/CTRON-SFPPS-TOPOLOGY-MIB.db
697. CTRON-SFPPS-VLAN-MIB data/mibs/cabletron/CTRON-SFPPS-VLAN-MIB.db
698. CTRON-SFPPS-VSTP-MIB data/mibs/cabletron/CTRON-SFPPS-VSTP-MIB.db
699. CTRON-SSR-SMI-MIB data/mibs/cabletron/CTRON-SSR-SMI-MIB.db
700. CTRON-SSR-CAPACITY-MIB data/mibs/cabletron/CTRON-SSR-CAPACITY-MIB.db
701. CTRON-SSR-CONFIG-MIB data/mibs/cabletron/CTRON-SSR-CONFIG-MIB.db
702. CTRON-SSR-HARDWARE-MIB data/mibs/cabletron/CTRON-SSR-HARDWARE-MIB.db

703.	CTRON-SSR-L2-MIB	data/mibs/cabletron/CTRON-SSR-L2-MIB.db
704.	CTRON-SSR-L3-MIB	data/mibs/cabletron/CTRON-SSR-L3-MIB.db
705.	CTRON-SSR-POLICY-MIB	data/mibs/cabletron/CTRON-SSR-POLICY-MIB.db
706.	CTRON-SSR-SERVICE-STATUS-MIB	data/mibs/cabletron/CTRON-SSR-SERVICE-STATUS-MIB.db
707.	CTRON-SSR-TRAP-MIB	data/mibs/cabletron/CTRON-SSR-TRAP-MIB.db
708.	CTRON-VLAN-EXTENSIONS-MIB	data/mibs/cabletron/CTRON-VLAN-EXTENSIONS-MIB.db
709.	CTRON-WAN-IMUX-MIB	data/mibs/cabletron/CTRON-WAN-IMUX-MIB.db
710.	CTRON-WAN-MULTI-IMUX-MIB	data/mibs/cabletron/CTRON-WAN-MULTI-IMUX-MIB.db
711.	SADMIN-MIB	data/mibs/cai/SADMIN-MIB.db
712.	CA-NTOS-MIB --	data/mibs/cai/CA-NTOS-MIB --.db
713.	CA-WIN95-MIB --	data/mibs/cai/CA-WIN95-MIB --.db
714.	EXAGENT-MIB	data/mibs/cai/EXAGENT-MIB.db
715.	LOG220-MIB	data/mibs/cai/LOG220-MIB.db
716.	LOGAGENT-MIB	data/mibs/cai/LOGAGENT-MIB.db
717.	ORAAGENT-MIB	data/mibs/cai/ORAAGENT-MIB.db
718.	OS220-MIB	data/mibs/cai/OS220-MIB.db
719.	OSAGENT-MIB	data/mibs/cai/OSAGENT-MIB.db
720.	PRFAGENT-MIB	data/mibs/cai/PRFAGENT-MIB.db
721.	PRO220-MIB	data/mibs/cai/PRO220-MIB.db
722.	PRO-MIB	data/mibs/cai/PRO-MIB.db
723.	SAP-MIB	data/mibs/cai/SAP-MIB.db
724.	SQLAGENT-MIB	data/mibs/cai/SQLAGENT-MIB.db
725.	SYBAGENT-MIB	data/mibs/cai/SYBAGENT-MIB.db
726.	ORAAGTVMS-MIB	data/mibs/cai/ORAAGTVMS-MIB.db
727.	CA-AS400OS-MIB --	data/mibs/cai/CA-AS400OS-MIB --.db
728.	CAIDB2MVS-MIB	data/mibs/cai/CAIDB2MVS-MIB.db
729.	INGRES-MIB	data/mibs/cai/INGRES-MIB.db
730.	CA-LOGA2-MIB	data/mibs/cai/CA-LOGA2-MIB.db
731.	CA-CICS_SYSTEM_AGENT --	data/mibs/cai/CA-CICS_SYSTEM_AGENT --.db
732.	CA-HDS_SYSTEM_AGENT	data/mibs/cai/CA-HDS_SYSTEM_AGENT.db
733.	CA-MQS_SYSTEM_AGENT --	data/mibs/cai/CA-MQS_SYSTEM_AGENT --.db
734.	CA-MVS_SYSTEM_AGENT --	data/mibs/cai/CA-MVS_SYSTEM_AGENT --.db
735.	CAIUXOS	data/mibs/cai/CAIUXOS.db
736.	CA-VMSOS-MIB --	data/mibs/cai/CA-VMSOS-MIB --.db
737.	DB2AGENT-MIB --	data/mibs/cai/DB2AGENT-MIB --.db
738.	HPAAGENT-MIB	data/mibs/cai/HPAAGENT-MIB.db
739.	IMXAGENT-MIB	data/mibs/cai/IMXAGENT-MIB.db
740.	MKINVEXP-MIB	data/mibs/cai/MKINVEXP-MIB.db
741.	MKLATECUSTPAY-MIB	data/mibs/cai/MKLATECUSTPAY-MIB.db
742.	MKLATEPRODORD-MIB	data/mibs/cai/MKLATEPRODORD-MIB.db
743.	MKLATEVENDSHIP-MIB	data/mibs/cai/MKLATEVENDSHIP-MIB.db
744.	PPLAGENT-MIB	data/mibs/cai/PPLAGENT-MIB.db
745.	rpagent-MIB	data/mibs/cai/rpagent-MIB.db
746.	CA-NTLOG-MIB	data/mibs/cai/CA-NTLOG-MIB.db
747.	CAIDATACOM-MIB	data/mibs/cai/CAIDATACOM-MIB.db
748.	IDMSAGENT-MIB	data/mibs/cai/IDMSAGENT-MIB.db
749.	CAIORAA2-MIB	data/mibs/cai/CAIORAA2-MIB.db
750.	CAISYBA2	data/mibs/cai/CAISYBA2.db
751.	CA-W2KOS-MIB	data/mibs/cai/CA-W2KOS-MIB.db
752.	CELLAGENT-MIB	data/mibs/cai/CELLAGENT-MIB.db
753.	CASCADE-MIB	data/mibs/cascade/CASCADE-MIB.db
754.	ATM-FORUM-MIB	data/mibs/cascade/ATM-FORUM-MIB.db
755.	PNNI-MIB	data/mibs/cascade/PNNI-MIB.db
756.	PS-MIB-REV2	data/mibs/cascade/PS-MIB-REV2.db
757.	CASCADE-ACCTSERVER-MIB	data/mibs/cascade/CASCADE-ACCTSERVER-MIB.db
758.	CASCADE-MPT-MIB	data/mibs/cascade/CASCADE-MPT-MIB.db
759.	CASCADEVIEW-MIB	data/mibs/cascade/CASCADEVIEW-MIB.db
760.	CASCADE-PROTCONNECT-MIB	data/mibs/cascade/CASCADE-PROTCONNECT-MIB.db
761.	DS3SUPPMIB	data/mibs/cascade/DS3SUPPMIB.db
762.	CASCADE-TCPIP-MIB	data/mibs/cascade/CASCADE-TCPIP-MIB.db
763.	SNMPv2-SMI-v1	data/mibs/cisco/SNMPv2-SMI-v1.db
764.	SNMPv2-TC-v1	data/mibs/cisco/SNMPv2-TC-v1.db
765.	A100-R1-MIB	data/mibs/cisco/A100-R1-MIB.db
766.	CISCO-SMI	data/mibs/cisco/CISCO-SMI.db
767.	ACCOUNTING-CONTROL-MIB	data/mibs/cisco/ACCOUNTING-CONTROL-MIB.db
768.	PerfHist-TC-MIB	data/mibs/cisco/PerfHist-TC-MIB.db
769.	ADSL-LINE-MIB	data/mibs/cisco/ADSL-LINE-MIB.db
770.	ADSL-CAP-LINE-MIB	data/mibs/cisco/ADSL-CAP-LINE-MIB.db
771.	ADSL-DMT-LINE-MIB	data/mibs/cisco/ADSL-DMT-LINE-MIB.db
772.	APPN-DLUR-MIB	data/mibs/cisco/APPN-DLUR-MIB.db
773.	ATM-ACCOUNTING-INFORMATION-MIB	data/mibs/cisco/ATM-ACCOUNTING-INFORMATION-MIB.db
774.	ATM-RMON-MIB	data/mibs/cisco/ATM-RMON-MIB.db
775.	CAT2600-MIB	data/mibs/cisco/CAT2600-MIB.db
776.	CISCO-5800-HEALTH-MON-MIB	data/mibs/cisco/CISCO-5800-HEALTH-MON-MIB.db
777.	CISCO-6400-CHASSIS-MIB	data/mibs/cisco/CISCO-6400-CHASSIS-MIB.db
778.	Cisco90Series-MIB	data/mibs/cisco/Cisco90Series-MIB.db
779.	CISCO-AAA-SERVER-CAPABILITY	data/mibs/cisco/CISCO-AAA-SERVER-CAPABILITY.db
780.	CISCO-TC	data/mibs/cisco/CISCO-TC.db
781.	CISCO-AAA-SERVER-MIB	data/mibs/cisco/CISCO-AAA-SERVER-MIB.db
782.	CISCO-AAA-SESSION-MIB	data/mibs/cisco/CISCO-AAA-SESSION-MIB.db
783.	CISCO-AAA-SERVER-EXT-MIB	data/mibs/cisco/CISCO-AAA-SERVER-EXT-MIB.db

784. CISCO-AAL5-MIB data/mibs/cisco/CISCO-AAL5-MIB.db
785. CISCO-ENVMON-MIB data/mibs/cisco/CISCO-ENVMON-MIB.db
786. CISCO-ACCESS-ENVMON-MIB data/mibs/cisco/CISCO-ACCESS-ENVMON-MIB.db
787. CISCO-ADAPTER-MIB data/mibs/cisco/CISCO-ADAPTER-MIB.db
788. CISCO-ADSL-CAP-LINE-MIB *** data/mibs/cisco/CISCO-ADSL-CAP-LINE-MIB ***.db
789. CISCO-ADSL-DMT-LINE-MIB *** data/mibs/cisco/CISCO-ADSL-DMT-LINE-MIB ***.db
790. CISCO-ALPS-MIB data/mibs/cisco/CISCO-ALPS-MIB.db
791. CISCO-ATM-ACCESS-LIST-MIB data/mibs/cisco/CISCO-ATM-ACCESS-LIST-MIB.db
792. CISCO-ATM-ADDR-MIB data/mibs/cisco/CISCO-ATM-ADDR-MIB.db
793. CISCO-ATM-IF-MIB data/mibs/cisco/CISCO-ATM-IF-MIB.db
794. CISCO-ATM-RM-MIB data/mibs/cisco/CISCO-ATM-RM-MIB.db
795. CISCO-ATM-CONN-MIB data/mibs/cisco/CISCO-ATM-CONN-MIB.db
796. CISCO-ATM-DUAL-PHY-MIB data/mibs/cisco/CISCO-ATM-DUAL-PHY-MIB.db
797. CISCO-ATM-EXT-MIB data/mibs/cisco/CISCO-ATM-EXT-MIB.db
798. CISCO-ATM-IF-PHYS-MIB data/mibs/cisco/CISCO-ATM-IF-PHYS-MIB.db
799. CISCO-ATM-PVC-MIB data/mibs/cisco/CISCO-ATM-PVC-MIB.db
800. CISCO-ATM-SERVICE-REGISTRY-MIB data/mibs/cisco/CISCO-ATM-SERVICE-REGISTRY-MIB.db
801. CISCO-ATM-SIG-DIAG-MIB data/mibs/cisco/CISCO-ATM-SIG-DIAG-MIB.db
802. CISCO-ATM-SWITCH-ADDR-MIB data/mibs/cisco/CISCO-ATM-SWITCH-ADDR-MIB.db
803. CISCO-ATM-SWITCH-CUG-MIB data/mibs/cisco/CISCO-ATM-SWITCH-CUG-MIB.db
804. CISCO-ATM-TRAFFIC-MIB data/mibs/cisco/CISCO-ATM-TRAFFIC-MIB.db
805. CISCO-ATM2-MIB data/mibs/cisco/CISCO-ATM2-MIB.db
806. CISCO-BGP-POLICY-ACCOUNTING-MIB data/mibs/cisco/CISCO-BGP-POLICY-ACCOUNTING-MIB.db
807. CISCO-BSC-MIB data/mibs/cisco/CISCO-BSC-MIB.db
808. CISCO-BSTUN-MIB data/mibs/cisco/CISCO-BSTUN-MIB.db
809. CISCO-BULK-FILE-MIB data/mibs/cisco/CISCO-BULK-FILE-MIB.db
810. CISCO-BUS-MIB data/mibs/cisco/CISCO-BUS-MIB.db
811. CISCO-C2900-MIB data/mibs/cisco/CISCO-C2900-MIB.db
812. CISCO-6200-MIB data/mibs/cisco/CISCO-6200-MIB.db
813. CISCO-CALL-APPLICATION-MIB ... data/mibs/cisco/CISCO-CALL-APPLICATION-MIB.db
814. CISCO-CALL-HISTORY-MIB data/mibs/cisco/CISCO-CALL-HISTORY-MIB.db
815. CISCO-CALL-RESOURCE-POOL-MIB . data/mibs/cisco/CISCO-CALL-RESOURCE-POOL-MIB.db
816. CISCO-CALL-TRACKER-MIB data/mibs/cisco/CISCO-CALL-TRACKER-MIB.db
817. CISCO-CALL-TRACKER-MODEM-MIB . data/mibs/cisco/CISCO-CALL-TRACKER-MODEM-MIB.db
818. CISCO-CALL-TRACKER-TCP-MIB ... data/mibs/cisco/CISCO-CALL-TRACKER-TCP-MIB.db
819. CISCO-CAR-MIB data/mibs/cisco/CISCO-CAR-MIB.db
820. CISCO-CAS-IF-MIB data/mibs/cisco/CISCO-CAS-IF-MIB.db
821. CISCO-VTP-MIB data/mibs/cisco/CISCO-VTP-MIB.db
822. CISCO-CDP-MIB data/mibs/cisco/CISCO-CDP-MIB.db
823. CISCO-CHANNEL-MIB data/mibs/cisco/CISCO-CHANNEL-MIB.db
824. CISCO-SNA-LLC-MIB data/mibs/cisco/CISCO-SNA-LLC-MIB.db
825. CISCO-CIPCSNA-MIB data/mibs/cisco/CISCO-CIPCSNA-MIB.db
826. CISCO-CIPTG-MIB data/mibs/cisco/CISCO-CIPTG-MIB.db
827. CISCO-CIPCMP-MIB data/mibs/cisco/CISCO-CIPCMP-MIB.db
828. CISCO-CIPLAN-MIB data/mibs/cisco/CISCO-CIPLAN-MIB.db
829. CISCO-CIPTCPIP-MIB data/mibs/cisco/CISCO-CIPTCPIP-MIB.db
830. CISCO-CIRCUIT-INTERFACE-MIB .. data/mibs/cisco/CISCO-CIRCUIT-INTERFACE-MIB.db
831. OLD-CISCO-CHASSIS-MIB data/mibs/cisco/OLD-CISCO-CHASSIS-MIB.db
832. CISCO-COMPRESSION-SERVICE-ADAPTER-MIB data/mibs/cisco/CISCO-COMPRESSION-SERVICE-ADAPTER-MIB.db
833. CISCO-ST-TC data/mibs/cisco/CISCO-ST-TC.db
834. CISCO-CONFIG-COPY-MIB data/mibs/cisco/CISCO-CONFIG-COPY-MIB.db
835. CISCO-CONFIG-MAN-MIB data/mibs/cisco/CISCO-CONFIG-MAN-MIB.db
836. CISCO-COPS-CLIENT-MIB data/mibs/cisco/CISCO-COPS-CLIENT-MIB.db
837. CISCO-DIAL-CONTROL-MIB data/mibs/cisco/CISCO-DIAL-CONTROL-MIB.db
838. CISCO-DLC-SWITCH-MIB data/mibs/cisco/CISCO-DLC-SWITCH-MIB.db
839. CISCO-DLSW-MIB -- data/mibs/cisco/CISCO-DLSW-MIB --.db
840. CISCO-QOS-PIB-MIB data/mibs/cisco/CISCO-QOS-PIB-MIB.db
841. CISCO-DSP-MGMT-MIB data/mibs/cisco/CISCO-DSP-MGMT-MIB.db
842. CISCO-DSPU-MIB data/mibs/cisco/CISCO-DSPU-MIB.db
843. CISCO-ENTITY-ALARM-MIB data/mibs/cisco/CISCO-ENTITY-ALARM-MIB.db
844. CISCO-ENTITY-ASSET-MIB data/mibs/cisco/CISCO-ENTITY-ASSET-MIB.db
845. CISCO-ENTITY-FRU-CONTROL-MIB . data/mibs/cisco/CISCO-ENTITY-FRU-CONTROL-MIB.db
846. CISCO-ENTITY-PERFORMANCE-MIB . data/mibs/cisco/CISCO-ENTITY-PERFORMANCE-MIB.db
847. CISCO-ENTITY-PROVISIONING-MIB data/mibs/cisco/CISCO-ENTITY-PROVISIONING-MIB.db
848. CISCO-ENTITY-SENSOR-MIB data/mibs/cisco/CISCO-ENTITY-SENSOR-MIB.db
849. CISCO-ENTITY-SENSOR-EXT-MIB .. data/mibs/cisco/CISCO-ENTITY-SENSOR-EXT-MIB.db
850. CISCO-ENTITY-VENDORTYPE-OID-MIB data/mibs/cisco/CISCO-ENTITY-VENDORTYPE-OID-MIB.db
851. CISCO-EPC-GATEWAY-MIB data/mibs/cisco/CISCO-EPC-GATEWAY-MIB.db
852. CISCO-FASTHUB-MIB data/mibs/cisco/CISCO-FASTHUB-MIB.db
853. CISCO-FC-PM-MIB data/mibs/cisco/CISCO-FC-PM-MIB.db
854. CISCO-FLASH-MIB data/mibs/cisco/CISCO-FLASH-MIB.db
855. CISCO-FRAME-RELAY-MIB data/mibs/cisco/CISCO-FRAME-RELAY-MIB.db
856. CISCO-FTP-CLIENT-MIB data/mibs/cisco/CISCO-FTP-CLIENT-MIB.db
857. CISCO-H323-TC-MIB data/mibs/cisco/CISCO-H323-TC-MIB.db
858. CISCO-GATEKEEPER-MIB data/mibs/cisco/CISCO-GATEKEEPER-MIB.db
859. CISCO-HSRP-MIB data/mibs/cisco/CISCO-HSRP-MIB.db
860. CISCO-HSRP-EXT-MIB data/mibs/cisco/CISCO-HSRP-EXT-MIB.db
861. CISCO-ICSUDSU-MIB data/mibs/cisco/CISCO-ICSUDSU-MIB.db
862. CISCO-IDSL-LINE-MIB -- data/mibs/cisco/CISCO-IDSL-LINE-MIB --.db
863. CISCO-IETF-ATM2-PVCTRAP-MIB .. data/mibs/cisco/CISCO-IETF-ATM2-PVCTRAP-MIB.db
864. CISCO-IMAGE-MIB data/mibs/cisco/CISCO-IMAGE-MIB.db

865. CISCO-IP-ENCRYPTION-MIB data/mibs/cisco/CISCO-IP-ENCRYPTION-MIB.db
866. CISCO-IP-STAT-MIB data/mibs/cisco/CISCO-IP-STAT-MIB.db
867. IPROUTE-MIB data/mibs/cisco/IPROUTE-MIB.db
868. CISCO-IPROUTE-MIB data/mibs/cisco/CISCO-IPROUTE-MIB.db
869. CISCO-ISDN-MIB data/mibs/cisco/CISCO-ISDN-MIB.db
870. CISCO-ISDNU-IF-MIB data/mibs/cisco/CISCO-ISDNU-IF-MIB.db
871. CISCO-L2L3-INTERFACE-CONFIG-MIB data/mibs/cisco/CISCO-L2L3-INTERFACE-CONFIG-MIB.db
872. CISCO-LECS-MIB data/mibs/cisco/CISCO-LECS-MIB.db
873. CISCO-LES-MIB data/mibs/cisco/CISCO-LES-MIB.db
874. CISCO-MEMORY-POOL-MIB data/mibs/cisco/CISCO-MEMORY-POOL-MIB.db
875. INT-SERV-GUARANTEED-MIB data/mibs/cisco/INT-SERV-GUARANTEED-MIB.db
876. RSVP-MIB data/mibs/cisco/RSVP-MIB.db
877. CISCO-VOICE-COMMON-DIAL-CONTROL-MIB data/mibs/cisco/CISCO-VOICE-COMMON-DIAL-CONTROL-MIB.db
878. CISCO-MLD-SNOOPING-MIB data/mibs/cisco/CISCO-MLD-SNOOPING-MIB.db
879. CISCO-MMAIL-DIAL-CONTROL-MIB . data/mibs/cisco/CISCO-MMAIL-DIAL-CONTROL-MIB.db
880. CISCO-MODEM-MGMT-MIB data/mibs/cisco/CISCO-MODEM-MGMT-MIB.db
881. CISCO-NETSYNC-MIB data/mibs/cisco/CISCO-NETSYNC-MIB.db
882. CISCO-OAM-MIB data/mibs/cisco/CISCO-OAM-MIB.db
883. CISCO-PAGP-MIB data/mibs/cisco/CISCO-PAGP-MIB.db
884. CISCO-PING-MIB data/mibs/cisco/CISCO-PING-MIB.db
885. CISCO-PNNI-MIB data/mibs/cisco/CISCO-PNNI-MIB.db
886. CISCO-POP-MGMT-MIB data/mibs/cisco/CISCO-POP-MGMT-MIB.db
887. CISCO-PROCESS-MIB data/mibs/cisco/CISCO-PROCESS-MIB.db
888. CISCO-PRODUCTS-MIB data/mibs/cisco/CISCO-PRODUCTS-MIB.db
889. CISCO-PTP-MIB data/mibs/cisco/CISCO-PTP-MIB.db
890. CISCO-QLLC01-MIB data/mibs/cisco/CISCO-QLLC01-MIB.db
891. CISCO-QUEUE-MIB data/mibs/cisco/CISCO-QUEUE-MIB.db
892. CISCO-REPEATER-MIB data/mibs/cisco/CISCO-REPEATER-MIB.db
893. CISCO-RHINO-MIB data/mibs/cisco/CISCO-RHINO-MIB.db
894. CISCO-RSRB-MIB data/mibs/cisco/CISCO-RSRB-MIB.db
895. CISCO-RTTMON-TC-MIB data/mibs/cisco/CISCO-RTTMON-TC-MIB.db
896. CISCO-ETHER-CFM-MIB data/mibs/cisco/CISCO-ETHER-CFM-MIB.db
897. CISCO-RTTMON-MIB DEFINITIONS: -- data/mibs/cisco/CISCO-RTTMON-MIB DEFINITIONS: --.db
898. CISCO-SDLLC-MIB data/mibs/cisco/CISCO-SDLLC-MIB.db
899. CISCO-SDSL-LINE-MIB *** data/mibs/cisco/CISCO-SDSL-LINE-MIB ***.db
900. CISCO-SNAPSHOT-MIB data/mibs/cisco/CISCO-SNAPSHOT-MIB.db
901. CISCO-SRP-MIB data/mibs/cisco/CISCO-SRP-MIB.db
902. CISCO-STACK-MIB data/mibs/cisco/CISCO-STACK-MIB.db
903. CISCO-STP-EXTENSIONS-MIB data/mibs/cisco/CISCO-STP-EXTENSIONS-MIB.db
904. CISCO-STUN-MIB data/mibs/cisco/CISCO-STUN-MIB.db
905. MPLS-VPN-MIB data/mibs/cisco/MPLS-VPN-MIB.db
906. CISCO-SWITCH-ENGINE-MIB data/mibs/cisco/CISCO-SWITCH-ENGINE-MIB.db
907. CISCO-SYSLOG-MIB data/mibs/cisco/CISCO-SYSLOG-MIB.db
908. CISCO-SYSTEM-MIB data/mibs/cisco/CISCO-SYSTEM-MIB.db
909. CISCO-TCP-MIB data/mibs/cisco/CISCO-TCP-MIB.db
910. CISCO-TCPOFFLOAD-MIB data/mibs/cisco/CISCO-TCPOFFLOAD-MIB.db
911. CISCO-TELEPRESENCE-EXCHANGE-SYSTEM-MIB data/mibs/cisco/CISCO-TELEPRESENCE-EXCHANGE-SYSTEM-MIB.db
912. CISCO-TN3270SERVER-MIB data/mibs/cisco/CISCO-TN3270SERVER-MIB.db
913. CISCO-TRANSACTION-CONNECTION-MIB data/mibs/cisco/CISCO-TRANSACTION-CONNECTION-MIB.db
914. CISCO-VINES-MIB data/mibs/cisco/CISCO-VINES-MIB.db
915. CISCO-VLAN-BRIDGING-MIB data/mibs/cisco/CISCO-VLAN-BRIDGING-MIB.db
916. CISCO-VLAN-MEMBERSHIP-MIB data/mibs/cisco/CISCO-VLAN-MEMBERSHIP-MIB.db
917. CISCO-VMPS-MIB data/mibs/cisco/CISCO-VMPS-MIB.db
918. CISCO-VOICE-ANALOG-IF-MIB data/mibs/cisco/CISCO-VOICE-ANALOG-IF-MIB.db
919. CISCO-VOICE-IF-MIB data/mibs/cisco/CISCO-VOICE-IF-MIB.db
920. CISCO-VPDN-MGMT-MIB data/mibs/cisco/CISCO-VPDN-MGMT-MIB.db
921. CISCO-VSIMASTER-MIB data/mibs/cisco/CISCO-VSIMASTER-MIB.db
922. CISCO-WIRELESS-EXP-MIB data/mibs/cisco/CISCO-WIRELESS-EXP-MIB.db
923. CISCO-WIRELESS-TC-MIB data/mibs/cisco/CISCO-WIRELESS-TC-MIB.db
924. CISCO-WIRELESS-IF-MIB data/mibs/cisco/CISCO-WIRELESS-IF-MIB.db
925. CISCO-WIRELESS-P2P-BPI-MIB ... data/mibs/cisco/CISCO-WIRELESS-P2P-BPI-MIB.db
926. CISCO-WRED-MIB data/mibs/cisco/CISCO-WRED-MIB.db
927. DTRConcentratorMIB data/mibs/cisco/DTRConcentratorMIB.db
928. ES-MODULE-MIB data/mibs/cisco/ES-MODULE-MIB.db
929. ESSWITCH-MIB data/mibs/cisco/ESSWITCH-MIB.db
930. IANATn3270eTC-MIB data/mibs/cisco/IANATn3270eTC-MIB.db
931. IBM-6611-APPN-MIB data/mibs/cisco/IBM-6611-APPN-MIB.db
932. IGMP-MIB data/mibs/cisco/IGMP-MIB.db
933. LANOPTICS-ALERTS-MIB data/mibs/cisco/LANOPTICS-ALERTS-MIB.db
934. LANOPTICS-BRIDGE-OPTION-MIB .. data/mibs/cisco/LANOPTICS-BRIDGE-OPTION-MIB.db
935. LANOPTICS-ETHERNET-OPTION-MIB data/mibs/cisco/LANOPTICS-ETHERNET-OPTION-MIB.db
936. LANOPTICS-HUB-MIB data/mibs/cisco/LANOPTICS-HUB-MIB.db
937. LANOPTICS-RING-MANAGER-MIB ... data/mibs/cisco/LANOPTICS-RING-MANAGER-MIB.db
938. LANOPTICS-SYSTEM-MIB data/mibs/cisco/LANOPTICS-SYSTEM-MIB.db
939. LS100-MIB data/mibs/cisco/LS100-MIB.db
940. LIGHTSTREAM-MIB data/mibs/cisco/LIGHTSTREAM-MIB.db
941. MADGEBOS-MIB data/mibs/cisco/MADGEBOS-MIB.db
942. MADGERSW-MIB data/mibs/cisco/MADGERSW-MIB.db
943. NOVELL-IPX-MIB data/mibs/cisco/NOVELL-IPX-MIB.db
944. NOVELL-NLSP-MIB data/mibs/cisco/NOVELL-NLSP-MIB.db
945. NOVELL-RIPSAP-MIB data/mibs/cisco/NOVELL-RIPSAP-MIB.db

946. OLD-CISCO-SYS-MIB data/mibs/cisco/OLD-CISCO-SYS-MIB.db
947. OLD-CISCO-APPLETALK-MIB data/mibs/cisco/OLD-CISCO-APPLETALK-MIB.db
948. OLD-CISCO-CPU-MIB data/mibs/cisco/OLD-CISCO-CPU-MIB.db
949. OLD-CISCO-DECNET-MIB data/mibs/cisco/OLD-CISCO-DECNET-MIB.db
950. OLD-CISCO-ENV-MIB data/mibs/cisco/OLD-CISCO-ENV-MIB.db
951. OLD-CISCO-FLASH-MIB data/mibs/cisco/OLD-CISCO-FLASH-MIB.db
952. OLD-CISCO-INTERFACES-MIB data/mibs/cisco/OLD-CISCO-INTERFACES-MIB.db
953. OLD-CISCO-IP-MIB data/mibs/cisco/OLD-CISCO-IP-MIB.db
954. OLD-CISCO-MEMORY-MIB data/mibs/cisco/OLD-CISCO-MEMORY-MIB.db
955. OLD-CISCO-NOVELL-MIB data/mibs/cisco/OLD-CISCO-NOVELL-MIB.db
956. OLD-CISCO-SYSTEM-MIB data/mibs/cisco/OLD-CISCO-SYSTEM-MIB.db
957. OLD-CISCO-TCP-MIB data/mibs/cisco/OLD-CISCO-TCP-MIB.db
958. OLD-CISCO-TS-MIB data/mibs/cisco/OLD-CISCO-TS-MIB.db
959. OLD-CISCO-VINES-MIB data/mibs/cisco/OLD-CISCO-VINES-MIB.db
960. OLD-CISCO-XNS-MIB data/mibs/cisco/OLD-CISCO-XNS-MIB.db
961. TN3270E-MIB data/mibs/cisco/TN3270E-MIB.db
962. TN3270E-RT-MIB data/mibs/cisco/TN3270E-RT-MIB.db
963. UDP-MIB data/mibs/cisco/UDP-MIB.db
964. VIPER-MIB data/mibs/cisco/VIPER-MIB.db
965. XGCP-MIB data/mibs/cisco/XGCP-MIB.db
966. CISCO-DLSW-EXT-MIB data/mibs/cisco/CISCO-DLSW-EXT-MIB.db
967. CISCOTRAP-MIB data/mibs/cisco/CISCOTRAP-MIB.db
968. CISCO-AAA-CLIENT-MIB data/mibs/cisco/CISCO-AAA-CLIENT-MIB.db
969. CISCO-APS-MIB data/mibs/cisco/CISCO-APS-MIB.db
970. CISCO-APS-EXT-MIB data/mibs/cisco/CISCO-APS-EXT-MIB.db
971. CISCO-ATM-CELL-LAYER-MIB data/mibs/cisco/CISCO-ATM-CELL-LAYER-MIB.db
972. CISCO-ATM-VIRTUAL-IF-MIB data/mibs/cisco/CISCO-ATM-VIRTUAL-IF-MIB.db
973. CISCO-CL2000-IF-HC-COUNTERS-MIB data/mibs/cisco/CISCO-CL2000-IF-HC-COUNTERS-MIB.db
974. CISCO-CABLE-SPECTRUM-MIB data/mibs/cisco/CISCO-CABLE-SPECTRUM-MIB.db
975. CISCO-CAT6K-CROSSBAR-MIB data/mibs/cisco/CISCO-CAT6K-CROSSBAR-MIB.db
976. CISCO-CATOS-ACL-QOS-MIB data/mibs/cisco/CISCO-CATOS-ACL-QOS-MIB.db
977. CISCO-CCM-MIB data/mibs/cisco/CISCO-CCM-MIB.db
978. CISCO-CLASS-BASED-QOS-MIB data/mibs/cisco/CISCO-CLASS-BASED-QOS-MIB.db
979. CISCO-CLUSTER-MIB data/mibs/cisco/CISCO-CLUSTER-MIB.db
980. CISCO-DS3-MIB data/mibs/cisco/CISCO-DS3-MIB.db
981. CISCO-DSL-PROVISION-MIB data/mibs/cisco/CISCO-DSL-PROVISION-MIB.db
982. CISCO-ENHANCED-MEMPOOL-MIB data/mibs/cisco/CISCO-ENHANCED-MEMPOOL-MIB.db
983. CISCO-ENTITY-EXT-MIB data/mibs/cisco/CISCO-ENTITY-EXT-MIB.db
984. CISCO-FIREWALL-MIB data/mibs/cisco/CISCO-FIREWALL-MIB.db
985. CISCO-IP-THRESHOLD-MIB data/mibs/cisco/CISCO-IP-THRESHOLD-MIB.db
986. CISCO-FILTER-GROUP-MIB data/mibs/cisco/CISCO-FILTER-GROUP-MIB.db
987. CISCO-FLOW-MONITOR-TC-MIB data/mibs/cisco/CISCO-FLOW-MONITOR-TC-MIB.db
988. CISCO-REPORT-INTERVAL-TC-MIB data/mibs/cisco/CISCO-REPORT-INTERVAL-TC-MIB.db
989. CISCO-FLOW-MONITOR-MIB data/mibs/cisco/CISCO-FLOW-MONITOR-MIB.db
990. CISCO-IGMP-FILTER-MIB data/mibs/cisco/CISCO-IGMP-FILTER-MIB.db
991. CISCO-IP-CBR-METRICS-MIB data/mibs/cisco/CISCO-IP-CBR-METRICS-MIB.db
992. CISCO-IP-UPLINK-REDIRECT-MIB data/mibs/cisco/CISCO-IP-UPLINK-REDIRECT-MIB.db
993. CISCO-IPSEC-MIB data/mibs/cisco/CISCO-IPSEC-MIB.db
994. CISCO-IPSEC-POLICY-MAP-MIB data/mibs/cisco/CISCO-IPSEC-POLICY-MAP-MIB.db
995. CISCO-LOCAL-DIRECTOR-MIB data/mibs/cisco/CISCO-LOCAL-DIRECTOR-MIB.db
996. CISCO-MAC-NOTIFICATION-MIB data/mibs/cisco/CISCO-MAC-NOTIFICATION-MIB.db
997. CISCO-MDI-METRICS-MIB data/mibs/cisco/CISCO-MDI-METRICS-MIB.db
998. CISCO-METRO-PHY-MIB data/mibs/cisco/CISCO-METRO-PHY-MIB.db
999. CISCO-MOBILE-IP-MIB data/mibs/cisco/CISCO-MOBILE-IP-MIB.db
1000. CISCO-NTP-MIB data/mibs/cisco/CISCO-NTP-MIB.db
1001. CISCO-OSCP-MIB data/mibs/cisco/CISCO-OSCP-MIB.db
1002. CISCO-POLICY-GROUP-MIB data/mibs/cisco/CISCO-POLICY-GROUP-MIB.db
1003. CISCO-NAC-TC-MIB data/mibs/cisco/CISCO-NAC-TC-MIB.db
1004. CISCO-PAE-MIB data/mibs/cisco/CISCO-PAE-MIB.db
1005. CISCO-PIM-MIB data/mibs/cisco/CISCO-PIM-MIB.db
1006. CISCO-PORT-QOS-MIB data/mibs/cisco/CISCO-PORT-QOS-MIB.db
1007. CISCO-PPPOE-MIB data/mibs/cisco/CISCO-PPPOE-MIB.db
1008. CISCO-PRIVATE-VLAN-MIB data/mibs/cisco/CISCO-PRIVATE-VLAN-MIB.db
1009. CISCO-PROXY-CONTROL-MIB data/mibs/cisco/CISCO-PROXY-CONTROL-MIB.db
1010. CISCO-QOS-POLICY-CONFIG-MIB data/mibs/cisco/CISCO-QOS-POLICY-CONFIG-MIB.db
1011. CISCO-RF-MIB data/mibs/cisco/CISCO-RF-MIB.db
1012. CISCO-RF-SUPPLEMENTAL-MIB data/mibs/cisco/CISCO-RF-SUPPLEMENTAL-MIB.db
1013. CISCO-RMON-CONFIG-MIB data/mibs/cisco/CISCO-RMON-CONFIG-MIB.db
1014. CISCO-ROUTE-POLICIES-MIB data/mibs/cisco/CISCO-ROUTE-POLICIES-MIB.db
1015. CISCO-RTP-METRICS-MIB data/mibs/cisco/CISCO-RTP-METRICS-MIB.db
1016. CISCO-SAA-APM-MIB data/mibs/cisco/CISCO-SAA-APM-MIB.db
1017. CISCO-SIP-UA-MIB data/mibs/cisco/CISCO-SIP-UA-MIB.db
1018. CISCO-SLB-MIB data/mibs/cisco/CISCO-SLB-MIB.db
1019. CISCO-SONET-MIB data/mibs/cisco/CISCO-SONET-MIB.db
1020. CISCO-SWITCH-CGMP-MIB data/mibs/cisco/CISCO-SWITCH-CGMP-MIB.db
1021. CISCO-SWITCH-USAGE-MIB data/mibs/cisco/CISCO-SWITCH-USAGE-MIB.db
1022. CISCO-THREAT-MITIGATION-SERVICE-MIB data/mibs/cisco/CISCO-THREAT-MITIGATION-SERVICE-MIB.db
1023. CISCO-TS-STACK-MIB data/mibs/cisco/CISCO-TS-STACK-MIB.db
1024. CISCO-UDLD-MIB data/mibs/cisco/CISCO-UDLD-MIB.db
1025. CISCO-VLAN-IPTABLE-RELATIONSHIP-MIB data/mibs/cisco/CISCO-VLAN-IPTABLE-RELATIONSHIP-MIB.db
1026. CISCO-VOICE-APPS-MIB data/mibs/cisco/CISCO-VOICE-APPS-MIB.db

1027. CISCO-VRF-MIB data/mibs/cisco/CISCO-VRF-MIB.db
1028. CISCO-VSI-CONTROLLER-MIB data/mibs/cisco/CISCO-VSI-CONTROLLER-MIB.db
1029. CISCO-WAN-MODULE-MIB data/mibs/cisco/CISCO-WAN-MODULE-MIB.db
1030. CISCO-WAN-RSRC-PART-MIB data/mibs/cisco/CISCO-WAN-RSRC-PART-MIB.db
1031. CISCO-WIRELESS-DOCS-IF-MIB ... data/mibs/cisco/CISCO-WIRELESS-DOCS-IF-MIB.db
1032. CISCO-WIRELESS-DOCS-EXT-MIB .. data/mibs/cisco/CISCO-WIRELESS-DOCS-EXT-MIB.db
1033. CISCO-WIRELESS-P2MP-LINK-METRICS-MIB data/mibs/cisco/CISCO-WIRELESS-P2MP-LINK-METRICS-MIB.db
1034. CISCO-WIRELESS-P2MP-PHY-MIB .. data/mibs/cisco/CISCO-WIRELESS-P2MP-PHY-MIB.db
1035. CISCO-WIRELESS-P2MP-RF-METRICS-MIB data/mibs/cisco/CISCO-WIRELESS-P2MP-RF-METRICS-MIB.db
1036. CISCO-XDSL-LINE-MIB data/mibs/cisco/CISCO-XDSL-LINE-MIB.db
1037. METRO1500-MIB data/mibs/cisco/METRO1500-MIB.db
1038. MPLS-LDP-GENERIC-STD-MIB data/mibs/cisco/MPLS-LDP-GENERIC-STD-MIB.db
1039. MPLS-LDP-MIB data/mibs/cisco/MPLS-LDP-MIB.db
1040. MPLS-LDP-STD-MIB data/mibs/cisco/MPLS-LDP-STD-MIB.db
1041. MPLS-LSR-MIB data/mibs/cisco/MPLS-LSR-MIB.db
1042. MPLS-LSR-STD-MIB data/mibs/cisco/MPLS-LSR-STD-MIB.db
1043. MPLS-TC-STD-MIB data/mibs/cisco/MPLS-TC-STD-MIB.db
1044. MPLS-TE-MIB data/mibs/cisco/MPLS-TE-MIB.db
1045. MPLS-TE-STD-MIB data/mibs/cisco/MPLS-TE-STD-MIB.db
1046. CISCO-ADMISSION-POLICY-MIB ... data/mibs/cisco/CISCO-ADMISSION-POLICY-MIB.db
1047. CISCO-ATM-SWITCH-FR-IWF-MIB .. data/mibs/cisco/CISCO-ATM-SWITCH-FR-IWF-MIB.db
1048. CISCO-ATM-SWITCH-FR-RM-MIB ... data/mibs/cisco/CISCO-ATM-SWITCH-FR-RM-MIB.db
1049. CISCO-C8500-REDUNDANCY-MIB ... data/mibs/cisco/CISCO-C8500-REDUNDANCY-MIB.db
1050. CISCO-DS0BUNDLE-MIB data/mibs/cisco/CISCO-DS0BUNDLE-MIB.db
1051. CISCO-DS0BUNDLE-EXT-MIB data/mibs/cisco/CISCO-DS0BUNDLE-EXT-MIB.db
1052. CISCO-IETF-SCTP-MIB data/mibs/cisco/CISCO-IETF-SCTP-MIB.db
1053. CISCO-IETF-SCTP-EXT-MIB data/mibs/cisco/CISCO-IETF-SCTP-EXT-MIB.db
1054. CISCO-ITP-TC-MIB data/mibs/cisco/CISCO-ITP-TC-MIB.db
1055. CISCO-ITP-ACL-MIB data/mibs/cisco/CISCO-ITP-ACL-MIB.db
1056. CISCO-ITP-ACT-MIB data/mibs/cisco/CISCO-ITP-ACT-MIB.db
1057. CISCO-ITP-MSU-RATES-MIB data/mibs/cisco/CISCO-ITP-MSU-RATES-MIB.db
1058. CISCO-ITP-SP-MIB data/mibs/cisco/CISCO-ITP-SP-MIB.db
1059. CISCO-ITP-RT-MIB data/mibs/cisco/CISCO-ITP-RT-MIB.db
1060. CISCO-ITP-SCCP-MIB data/mibs/cisco/CISCO-ITP-SCCP-MIB.db
1061. CISCO-LEC-DATA-VCC-MIB data/mibs/cisco/CISCO-LEC-DATA-VCC-MIB.db
1062. CISCO-LEC-EXT-MIB data/mibs/cisco/CISCO-LEC-EXT-MIB.db
1063. DRAFT-MSDP-MIB data/mibs/cisco/DRAFT-MSDP-MIB.db
1064. AWC-VX-MIB data/mibs/cisco/AWC-VX-MIB.db
1065. AWC-VLAN-CFG-MIB data/mibs/cisco/AWC-VLAN-CFG-MIB.db
1066. IEEE802dot11-MIB data/mibs/cisco/IEEE802dot11-MIB.db
1067. AIRONET-600-MIB data/mibs/cisco/AIRONET-600-MIB.db
1068. CISCO-CLASS-BASED-QOS-MIB-CAPABILITY data/mibs/cisco/CISCO-CLASS-BASED-QOS-MIB-CAPABILITY.db
1069. CISCOWAN-SMI data/mibs/cisco/CISCOWAN-SMI.db
1070. BASIS-MIB data/mibs/cisco/BASIS-MIB.db
1071. BASIS-GENERIC-MIB data/mibs/cisco/BASIS-GENERIC-MIB.db
1072. BASIS-ONLINE-DIAG-MIB data/mibs/cisco/BASIS-ONLINE-DIAG-MIB.db
1073. BASIS-RAS-DISK-MIB data/mibs/cisco/BASIS-RAS-DISK-MIB.db
1074. BASIS-SERIAL-MIB data/mibs/cisco/BASIS-SERIAL-MIB.db
1075. BASIS-SHELF-MIB data/mibs/cisco/BASIS-SHELF-MIB.db
1076. CALISTA-DPA-MIB data/mibs/cisco/CALISTA-DPA-MIB.db
1077. CISCO-AAL5-EXT-MIB data/mibs/cisco/CISCO-AAL5-EXT-MIB.db
1078. CISCO-ASPP-MIB data/mibs/cisco/CISCO-ASPP-MIB.db
1079. CISCO-ATM-QOS-MIB data/mibs/cisco/CISCO-ATM-QOS-MIB.db
1080. CISCO-AUTHORIZATION-STATS-MIB data/mibs/cisco/CISCO-AUTHORIZATION-STATS-MIB.db
1081. CISCO-BERT-MIB data/mibs/cisco/CISCO-BERT-MIB.db
1082. CISCO-BGP4-MIB data/mibs/cisco/CISCO-BGP4-MIB.db
1083. CISCO-CABLE-AVAILABILITY-MIB .. data/mibs/cisco/CISCO-CABLE-AVAILABILITY-MIB.db
1084. CISCO-CALLHOME-MIB data/mibs/cisco/CISCO-CALLHOME-MIB.db
1085. CISCO-CASA-FA-MIB data/mibs/cisco/CISCO-CASA-FA-MIB.db
1086. CISCO-CASA-MIB data/mibs/cisco/CISCO-CASA-MIB.db
1087. CISCO-CDL-MIB data/mibs/cisco/CISCO-CDL-MIB.db
1088. CISCO-CDMA-AHDLC-MIB data/mibs/cisco/CISCO-CDMA-AHDLC-MIB.db
1089. CISCO-CDMA-PDSN-MIB data/mibs/cisco/CISCO-CDMA-PDSN-MIB.db
1090. CISCO-CNO-SWITCH-MIB data/mibs/cisco/CISCO-CNO-SWITCH-MIB.db
1091. CISCO-CONTENT-ENGINE-MIB data/mibs/cisco/CISCO-CONTENT-ENGINE-MIB.db
1092. CISCO-CONTENT-NETWORK-MIB data/mibs/cisco/CISCO-CONTENT-NETWORK-MIB.db
1093. CISCO-CSG-MIB data/mibs/cisco/CISCO-CSG-MIB.db
1094. CISCO-DDP-IAPP-MIB data/mibs/cisco/CISCO-DDP-IAPP-MIB.db
1095. CISCO-DEVICE-EXCEPTION-REPORTING-MIB data/mibs/cisco/CISCO-DEVICE-EXCEPTION-REPORTING-MIB.db
1096. CISCO-VSAN-MIB data/mibs/cisco/CISCO-VSAN-MIB.db
1097. CISCO-NS-MIB data/mibs/cisco/CISCO-NS-MIB.db
1098. CISCO-ZS-MIB data/mibs/cisco/CISCO-ZS-MIB.db
1099. CISCO-IF-EXTENSION-MIB data/mibs/cisco/CISCO-IF-EXTENSION-MIB.db
1100. CISCO-FC-FE-MIB data/mibs/cisco/CISCO-FC-FE-MIB.db
1101. CISCO-DM-MIB data/mibs/cisco/CISCO-DM-MIB.db
1102. CISCO-DOT11-IF-MIB data/mibs/cisco/CISCO-DOT11-IF-MIB.db
1103. CISCO-DOT11-ANTENNA-MIB data/mibs/cisco/CISCO-DOT11-ANTENNA-MIB.db
1104. CISCO-DOT11-ASSOCIATION-MIB .. data/mibs/cisco/CISCO-DOT11-ASSOCIATION-MIB.db
1105. CISCO-DOT11-CONTEXT-SERVICES-MANAGER-MIB data/mibs/cisco/CISCO-DOT11-CONTEXT-SERVICES-MANAGER-MIB.db
1106. CISCO-ENHANCED-WRED-MIB data/mibs/cisco/CISCO-ENHANCED-WRED-MIB.db
1107. CISCO-ENTITY-PFE-MIB data/mibs/cisco/CISCO-ENTITY-PFE-MIB.db

1108. CISCO-EPM-NOTIFICATION-MIB ... data/mibs/cisco/CISCO-EPM-NOTIFICATION-MIB.db
1109. CISCO-SCSI-MIB data/mibs/cisco/CISCO-SCSI-MIB.db
1110. CISCO-EXT-SCSI-MIB data/mibs/cisco/CISCO-EXT-SCSI-MIB.db
1111. CISCO-FABRIC-CL2K-MIB data/mibs/cisco/CISCO-FABRIC-CL2K-MIB.db
1112. CISCO-FC-ROUTE-MIB data/mibs/cisco/CISCO-FC-ROUTE-MIB.db
1113. CISCO-FCPING-MIB data/mibs/cisco/CISCO-FCPING-MIB.db
1114. CISCO-FCS-MIB data/mibs/cisco/CISCO-FCS-MIB.db
1115. CISCO-FCTRACEROUTE-MIB data/mibs/cisco/CISCO-FCTRACEROUTE-MIB.db
1116. CISCO-FIPS-STATS-MIB data/mibs/cisco/CISCO-FIPS-STATS-MIB.db
1117. CISCO-FSPF-MIB data/mibs/cisco/CISCO-FSPF-MIB.db
1118. CISCO-GTP-MIB data/mibs/cisco/CISCO-GTP-MIB.db
1119. CISCO-GTPV2-MIB data/mibs/cisco/CISCO-GTPV2-MIB.db
1120. CISCO-GGSN-MIB data/mibs/cisco/CISCO-GGSN-MIB.db
1121. CISCO-GGSN-QOS-MIB data/mibs/cisco/CISCO-GGSN-QOS-MIB.db
1122. CISCO-GPRS-ACC-PT-MIB data/mibs/cisco/CISCO-GPRS-ACC-PT-MIB.db
1123. CISCO-GPRS-CHARGING-MIB data/mibs/cisco/CISCO-GPRS-CHARGING-MIB.db
1124. CISCO-GTP-DIRECTOR-MIB data/mibs/cisco/CISCO-GTP-DIRECTOR-MIB.db
1125. CISCO-HC-ALARM-MIB data/mibs/cisco/CISCO-HC-ALARM-MIB.db
1126. CISCO-IETF-DOT11-QOS-MIB data/mibs/cisco/CISCO-IETF-DOT11-QOS-MIB.db
1127. CISCO-WLAN-VLAN-MIB data/mibs/cisco/CISCO-WLAN-VLAN-MIB.db
1128. CISCO-IETF-DOT11-QOS-EXT-MIB . data/mibs/cisco/CISCO-IETF-DOT11-QOS-EXT-MIB.db
1129. CISCO-IETF-IP-FORWARD-MIB data/mibs/cisco/CISCO-IETF-IP-FORWARD-MIB.db
1130. CISCO-IETF-IP-MIB data/mibs/cisco/CISCO-IETF-IP-MIB.db
1131. CISCO-IETF-NAT-MIB data/mibs/cisco/CISCO-IETF-NAT-MIB.db
1132. CISCO-IETF-VDSL-LINE-MIB data/mibs/cisco/CISCO-IETF-VDSL-LINE-MIB.db
1133. CISCO-IP-LINK-CONFIG-MIB data/mibs/cisco/CISCO-IP-LINK-CONFIG-MIB.db
1134. CISCO-IP-LOOPBACK-MIB data/mibs/cisco/CISCO-IP-LOOPBACK-MIB.db
1135. CISCO-IGMP-SNOOPING-MIB data/mibs/cisco/CISCO-IGMP-SNOOPING-MIB.db
1136. CISCO-IMA-MIB data/mibs/cisco/CISCO-IMA-MIB.db
1137. CISCO-IP-IF-MIB data/mibs/cisco/CISCO-IP-IF-MIB.db
1138. CISCO-IP-PROTOCOL-FILTER-MIB . data/mibs/cisco/CISCO-IP-PROTOCOL-FILTER-MIB.db
1139. CISCO-ISCSI-MIB data/mibs/cisco/CISCO-ISCSI-MIB.db
1140. CISCO-ISCSI-GW-MIB data/mibs/cisco/CISCO-ISCSI-GW-MIB.db
1141. CISCO-ITP-GSP-MIB data/mibs/cisco/CISCO-ITP-GSP-MIB.db
1142. CISCO-ITP-GACT-MIB data/mibs/cisco/CISCO-ITP-GACT-MIB.db
1143. CISCO-ITP-GRT-MIB data/mibs/cisco/CISCO-ITP-GRT-MIB.db
1144. CISCO-ITP-GSCCP-MIB data/mibs/cisco/CISCO-ITP-GSCCP-MIB.db
1145. CISCO-ITP-GSP2-MIB data/mibs/cisco/CISCO-ITP-GSP2-MIB.db
1146. CISCO-ITP-SP2-MIB data/mibs/cisco/CISCO-ITP-SP2-MIB.db
1147. CISCO-ITP-XUA-MIB data/mibs/cisco/CISCO-ITP-XUA-MIB.db
1148. CISCO-L2-DEV-MONITORING-MIB .. data/mibs/cisco/CISCO-L2-DEV-MONITORING-MIB.db
1149. CISCO-L2-TUNNEL-CONFIG-MIB ... data/mibs/cisco/CISCO-L2-TUNNEL-CONFIG-MIB.db
1150. IEEE8023-LAG-MIB data/mibs/cisco/IEEE8023-LAG-MIB.db
1151. CISCO-LAG-MIB data/mibs/cisco/CISCO-LAG-MIB.db
1152. CISCO-LRE-CPE-MIB data/mibs/cisco/CISCO-LRE-CPE-MIB.db
1153. CISCO-MGX82XX-ATM-UNI-PORT-MIB data/mibs/cisco/CISCO-MGX82XX-ATM-UNI-PORT-MIB.db
1154. CISCO-MGX82XX-CARD-FEATURE-MIB data/mibs/cisco/CISCO-MGX82XX-CARD-FEATURE-MIB.db
1155. CISCO-MGX82XX-DSX1-MIB data/mibs/cisco/CISCO-MGX82XX-DSX1-MIB.db
1156. CISCO-MGX82XX-DSX3-BERT-MIB .. data/mibs/cisco/CISCO-MGX82XX-DSX3-BERT-MIB.db
1157. CISCO-MGX82XX-DSX3-MIB data/mibs/cisco/CISCO-MGX82XX-DSX3-MIB.db
1158. CISCO-MGX82XX-ENVMON-MIB data/mibs/cisco/CISCO-MGX82XX-ENVMON-MIB.db
1159. CISCO-MGX82XX-MODULE-RSRC-PART-MIB data/mibs/cisco/CISCO-MGX82XX-MODULE-RSRC-PART-MIB.db
1160. CISCO-MGX82XX-PXM-CLOCK-MIB .. data/mibs/cisco/CISCO-MGX82XX-PXM-CLOCK-MIB.db
1161. CISCO-MGX82XX-RPM-CONN-MIB ... data/mibs/cisco/CISCO-MGX82XX-RPM-CONN-MIB.db
1162. CISCO-MGX82XX-RPM-RSRC-PART-MIB data/mibs/cisco/CISCO-MGX82XX-RPM-RSRC-PART-MIB.db
1163. CISCO-MGX82XX-RPM-SUBIF-MIB .. data/mibs/cisco/CISCO-MGX82XX-RPM-SUBIF-MIB.db
1164. CISCO-MGX82XX-VIRTUAL-PORT-MIB data/mibs/cisco/CISCO-MGX82XX-VIRTUAL-PORT-MIB.db
1165. CISCO-MGX8800-IF-MAPPING-MIB . data/mibs/cisco/CISCO-MGX8800-IF-MAPPING-MIB.db
1166. CISCO-NBAR-PROTOCOL-DISCOVERY-MIB data/mibs/cisco/CISCO-NBAR-PROTOCOL-DISCOVERY-MIB.db
1167. CISCO-NDE-MIB data/mibs/cisco/CISCO-NDE-MIB.db
1168. CISCO-NETWORK-REGISTRAR-MIB .. data/mibs/cisco/CISCO-NETWORK-REGISTRAR-MIB.db
1169. CISCO-NMS-APPL-HEALTH-MIB data/mibs/cisco/CISCO-NMS-APPL-HEALTH-MIB.db
1170. CISCO-OPTICAL-IF-CROSS-CONNECT-MIB data/mibs/cisco/CISCO-OPTICAL-IF-CROSS-CONNECT-MIB.db
1171. CISCO-OPTICAL-IF-EXTN-MIB data/mibs/cisco/CISCO-OPTICAL-IF-EXTN-MIB.db
1172. CISCO-OPTICAL-MONITOR-MIB data/mibs/cisco/CISCO-OPTICAL-MONITOR-MIB.db
1173. CISCO-OPTICAL-MONITORING-MIB . data/mibs/cisco/CISCO-OPTICAL-MONITORING-MIB.db
1174. CISCO-OPTICAL-PATCH-MIB data/mibs/cisco/CISCO-OPTICAL-PATCH-MIB.db
1175. CISCO-OSPF-MIB data/mibs/cisco/CISCO-OSPF-MIB.db
1176. CISCO-OSPF-TRAP-MIB data/mibs/cisco/CISCO-OSPF-TRAP-MIB.db
1177. CISCO-PORT-CHANNEL-MIB data/mibs/cisco/CISCO-PORT-CHANNEL-MIB.db
1178. CISCO-PORT-SECURITY-MIB data/mibs/cisco/CISCO-PORT-SECURITY-MIB.db
1179. CISCO-PSA-MICROCODE-MIB data/mibs/cisco/CISCO-PSA-MICROCODE-MIB.db
1180. CISCO-PTOPO-EXTN-MIB data/mibs/cisco/CISCO-PTOPO-EXTN-MIB.db
1181. CISCO-RADIUS-MIB data/mibs/cisco/CISCO-RADIUS-MIB.db
1182. CISCO-RPMS-MIB data/mibs/cisco/CISCO-RPMS-MIB.db
1183. CISCO-RSCN-MIB data/mibs/cisco/CISCO-RSCN-MIB.db
1184. CISCO-SCTP-MIB data/mibs/cisco/CISCO-SCTP-MIB.db
1185. CISCO-SIBU-FLASH-MIB data/mibs/cisco/CISCO-SIBU-FLASH-MIB.db
1186. CISCO-SIBU-MANAGERS-MIB data/mibs/cisco/CISCO-SIBU-MANAGERS-MIB.db
1187. CISCO-SIBU-STACKABLE-DUAL-SPEED-HUB-MIB data/mibs/cisco/CISCO-SIBU-STACKABLE-DUAL-SPEED-HUB-MIB.db
1188. CISCO-SLB-EXT-MIB data/mibs/cisco/CISCO-SLB-EXT-MIB.db

1189. CISCO-SM-FILE-DOWNLOAD-MIB ... data/mibs/cisco/CISCO-SM-FILE-DOWNLOAD-MIB.db
1190. CISCO-SP-MIB data/mibs/cisco/CISCO-SP-MIB.db
1191. CISCO-SSG-MIB data/mibs/cisco/CISCO-SSG-MIB.db
1192. CISCO-STACKMAKER-MIB data/mibs/cisco/CISCO-STACKMAKER-MIB.db
1193. CISCO-SYS-INFO-LOG-MIB data/mibs/cisco/CISCO-SYS-INFO-LOG-MIB.db
1194. CISCO-SYSLOG-EVENT-EXT-MIB ... data/mibs/cisco/CISCO-SYSLOG-EVENT-EXT-MIB.db
1195. CISCO-SYSLOG-EXT-MIB data/mibs/cisco/CISCO-SYSLOG-EXT-MIB.db
1196. CISCO-SYSTEM-EXT-MIB data/mibs/cisco/CISCO-SYSTEM-EXT-MIB.db
1197. CISCO-TBRIDGE-DEV-IF-MIB data/mibs/cisco/CISCO-TBRIDGE-DEV-IF-MIB.db
1198. CISCO-VIRTUAL-NW-IF-MIB data/mibs/cisco/CISCO-VIRTUAL-NW-IF-MIB.db
1199. CISCO-VOA-MIB data/mibs/cisco/CISCO-VOA-MIB.db
1200. CISCO-VOICE-ATM-DIAL-CONTROL-MIB data/mibs/cisco/CISCO-VOICE-ATM-DIAL-CONTROL-MIB.db
1201. CISCO-VOICE-ENABLED-LINK-MIB . data/mibs/cisco/CISCO-VOICE-ENABLED-LINK-MIB.db
1202. CISCO-VOICE-FR-DIAL-CONTROL-MIB data/mibs/cisco/CISCO-VOICE-FR-DIAL-CONTROL-MIB.db
1203. CISCO-VOICE-HDLC-DIAL-CONTROL-MIB data/mibs/cisco/CISCO-VOICE-HDLC-DIAL-CONTROL-MIB.db
1204. CISCO-VPDN-MGMT-EXT-MIB data/mibs/cisco/CISCO-VPDN-MGMT-EXT-MIB.db
1205. CISCO-WAN-ATM-CONN-MIB data/mibs/cisco/CISCO-WAN-ATM-CONN-MIB.db
1206. CISCO-WAN-BBIF-ATM-CONN-MIB .. data/mibs/cisco/CISCO-WAN-BBIF-ATM-CONN-MIB.db
1207. CISCO-WAN-BBIF-ATM-CONN-STAT-MIB data/mibs/cisco/CISCO-WAN-BBIF-ATM-CONN-STAT-MIB.db
1208. CISCO-WAN-BBIF-ILMI-MIB data/mibs/cisco/CISCO-WAN-BBIF-ILMI-MIB.db
1209. CISCO-WAN-BBIF-PORT-MIB data/mibs/cisco/CISCO-WAN-BBIF-PORT-MIB.db
1210. CISCO-WAN-CES-CONN-MIB data/mibs/cisco/CISCO-WAN-CES-CONN-MIB.db
1211. CISCO-WAN-CES-CONN-STAT-MIB .. data/mibs/cisco/CISCO-WAN-CES-CONN-STAT-MIB.db
1212. CISCO-WAN-CES-PORT-MIB data/mibs/cisco/CISCO-WAN-CES-PORT-MIB.db
1213. CISCO-WAN-CES-RSRC-PART-MIB .. data/mibs/cisco/CISCO-WAN-CES-RSRC-PART-MIB.db
1214. CISCO-WAN-FEEDER-MIB data/mibs/cisco/CISCO-WAN-FEEDER-MIB.db
1215. CISCO-WAN-FR-CONN-MIB data/mibs/cisco/CISCO-WAN-FR-CONN-MIB.db
1216. CISCO-WAN-FR-CONN-STAT-MIB ... data/mibs/cisco/CISCO-WAN-FR-CONN-STAT-MIB.db
1217. CISCO-WAN-FR-PORT-MIB data/mibs/cisco/CISCO-WAN-FR-PORT-MIB.db
1218. CISCO-WAN-FR-RSRC-PART-MIB ... data/mibs/cisco/CISCO-WAN-FR-RSRC-PART-MIB.db
1219. CISCO-WAN-FR-X21-MIB data/mibs/cisco/CISCO-WAN-FR-X21-MIB.db
1220. CISCO-WAN-NCDP-MIB data/mibs/cisco/CISCO-WAN-NCDP-MIB.db
1221. CISCO-WAN-PAR-MIB data/mibs/cisco/CISCO-WAN-PAR-MIB.db
1222. CISCO-WAN-SCT-MGMT-MIB data/mibs/cisco/CISCO-WAN-SCT-MGMT-MIB.db
1223. CISCO-WAN-SONET-MIB data/mibs/cisco/CISCO-WAN-SONET-MIB.db
1224. CISCO-WAN-SRM-BERT-MIB data/mibs/cisco/CISCO-WAN-SRM-BERT-MIB.db
1225. CISCO-WAN-SRM-MIB data/mibs/cisco/CISCO-WAN-SRM-MIB.db
1226. CISCO-WAN-SVC-MIB data/mibs/cisco/CISCO-WAN-SVC-MIB.db
1227. CISCO-WAN-TOPOLOGY-MIB data/mibs/cisco/CISCO-WAN-TOPOLOGY-MIB.db
1228. CISCO-WAN-TRAP-VARS-MIB data/mibs/cisco/CISCO-WAN-TRAP-VARS-MIB.db
1229. CISCO-WNMGR-MIB data/mibs/cisco/CISCO-WNMGR-MIB.db
1230. CISCOWORKS-MIB data/mibs/cisco/CISCOWORKS-MIB.db
1231. COMPAT-MIB data/mibs/cisco/COMPAT-MIB.db
1232. CISCO-CABLE-DIAG-MIB data/mibs/cisco/CISCO-CABLE-DIAG-MIB.db
1233. CISCO-CABLE-METERING-MIB data/mibs/cisco/CISCO-CABLE-METERING-MIB.db
1234. CISCO-DHCP-SNOOPING-MIB data/mibs/cisco/CISCO-DHCP-SNOOPING-MIB.db
1235. CISCO-DOT11-RADIO-DIAGNOSTIC-MIB data/mibs/cisco/CISCO-DOT11-RADIO-DIAGNOSTIC-MIB.db
1236. CISCO-DYNAMIC-ARP-INSPECTION-MIB data/mibs/cisco/CISCO-DYNAMIC-ARP-INSPECTION-MIB.db
1237. CISCO-ENTITY-DISPLAY-MIB data/mibs/cisco/CISCO-ENTITY-DISPLAY-MIB.db
1238. CISCO-FABRIC-HFR-MIB data/mibs/cisco/CISCO-FABRIC-HFR-MIB.db
1239. CISCO-FABRIC-MCAST-MIB data/mibs/cisco/CISCO-FABRIC-MCAST-MIB.db
1240. CISCO-FABRIC-MCAST-APPL-MIB .. data/mibs/cisco/CISCO-FABRIC-MCAST-APPL-MIB.db
1241. CISCO-IF-CALL-SERVICE-MIB data/mibs/cisco/CISCO-IF-CALL-SERVICE-MIB.db
1242. CISCO-IP-LOCAL-POOL-MIB data/mibs/cisco/CISCO-IP-LOCAL-POOL-MIB.db
1243. CISCO-ITP-MLR-MIB data/mibs/cisco/CISCO-ITP-MLR-MIB.db
1244. CISCO-ITP-MONITOR-MIB data/mibs/cisco/CISCO-ITP-MONITOR-MIB.db
1245. CISCO-MEDIA-GATEWAY-MIB data/mibs/cisco/CISCO-MEDIA-GATEWAY-MIB.db
1246. CISCO-IPSEC-FLOW-MONITOR-MIB . data/mibs/cisco/CISCO-IPSEC-FLOW-MONITOR-MIB.db
1247. CISCO-MGC-MIB data/mibs/cisco/CISCO-MGC-MIB.db
1248. CISCO-MODULE-AUTO-SHUTDOWN-MIB data/mibs/cisco/CISCO-MODULE-AUTO-SHUTDOWN-MIB.db
1249. CISCO-OUTAGE-MONITOR-MIB data/mibs/cisco/CISCO-OUTAGE-MONITOR-MIB.db
1250. CISCO-SECURE-SHELL-MIB data/mibs/cisco/CISCO-SECURE-SHELL-MIB.db
1251. CISCO-SVI-AUTOSTATE-MIB data/mibs/cisco/CISCO-SVI-AUTOSTATE-MIB.db
1252. CISCO-TAP-MIB data/mibs/cisco/CISCO-TAP-MIB.db
1253. CISCO-VISM-ATM-TRUNK-MIB data/mibs/cisco/CISCO-VISM-ATM-TRUNK-MIB.db
1254. CISCO-VISM-CONN-MIB data/mibs/cisco/CISCO-VISM-CONN-MIB.db
1255. CISCO-VISM-CONN-STAT-MIB data/mibs/cisco/CISCO-VISM-CONN-STAT-MIB.db
1256. CISCO-VISM-DSX0-MIB data/mibs/cisco/CISCO-VISM-DSX0-MIB.db
1257. CISCO-VISM-DSX1-MIB data/mibs/cisco/CISCO-VISM-DSX1-MIB.db
1258. CISCO-VOICE-CAS-MODULE-MIB ... data/mibs/cisco/CISCO-VOICE-CAS-MODULE-MIB.db
1259. CISCO-VOICE-DNIS-MIB data/mibs/cisco/CISCO-VOICE-DNIS-MIB.db
1260. CISCO-VOICE-DIAL-CONTROL-MIB . data/mibs/cisco/CISCO-VOICE-DIAL-CONTROL-MIB.db
1261. CISCO-WAN-CODEC-GEN-PARM-MIB . data/mibs/cisco/CISCO-WAN-CODEC-GEN-PARM-MIB.db
1262. CISCO-NETFLOW-MIB data/mibs/cisco/CISCO-NETFLOW-MIB.db
1263. RTM-MIB data/mibs/cisco/RTM-MIB.db
1264. CISCO-ANNOUNCEMENT-MIB data/mibs/cisco/CISCO-ANNOUNCEMENT-MIB.db
1265. CISCO-APPLIANCE-REDUNDANCY-MIB data/mibs/cisco/CISCO-APPLIANCE-REDUNDANCY-MIB.db
1266. CISCO-BBSM-MIB data/mibs/cisco/CISCO-BBSM-MIB.db
1267. CISCO-BCP-MIB data/mibs/cisco/CISCO-BCP-MIB.db
1268. CISCO-BITS-CLOCK-MIB data/mibs/cisco/CISCO-BITS-CLOCK-MIB.db
1269. CISCO-CABLE-ADMISSION-CTRL-MIB data/mibs/cisco/CISCO-CABLE-ADMISSION-CTRL-MIB.db

1270. CISCO-CABLE-QOS-MONITOR-MIB .. data/mibs/cisco/CISCO-CABLE-QOS-MONITOR-MIB.db
1271. CISCO-CAC-SYSTEM-MIB data/mibs/cisco/CISCO-CAC-SYSTEM-MIB.db
1272. CISCO-CCME-MIB data/mibs/cisco/CISCO-CCME-MIB.db
1273. CISCO-CFS-MIB data/mibs/cisco/CISCO-CFS-MIB.db
1274. CISCO-CIDS-MIB data/mibs/cisco/CISCO-CIDS-MIB.db
1275. CISCO-COMMON-ROLES-MIB data/mibs/cisco/CISCO-COMMON-ROLES-MIB.db
1276. CISCO-CRYPTO-ACCELERATOR-MIB . data/mibs/cisco/CISCO-CRYPTO-ACCELERATOR-MIB.db
1277. CISCO-DNS-CLIENT-MIB data/mibs/cisco/CISCO-DNS-CLIENT-MIB.db
1278. CISCO-DOT11-CONTEXT-SERVICES-CLIENT-MIB data/mibs/cisco/CISCO-DOT11-CONTEXT-SERVICES-CLIENT-MIB.db
1279. CISCO-DOT11-CONTEXT-SERVICES-MIB data/mibs/cisco/CISCO-DOT11-CONTEXT-SERVICES-MIB.db
1280. CISCO-DOT11-QOS-MIB data/mibs/cisco/CISCO-DOT11-QOS-MIB.db
1281. CISCO-DOT11-SSID-SECURITY-MIB data/mibs/cisco/CISCO-DOT11-SSID-SECURITY-MIB.db
1282. CISCO-DOT11-WIDS-MIB data/mibs/cisco/CISCO-DOT11-WIDS-MIB.db
1283. CISCO-DS0-CROSS-CONNECT-MIB .. data/mibs/cisco/CISCO-DS0-CROSS-CONNECT-MIB.db
1284. CISCO-DS1-EXT-MIB data/mibs/cisco/CISCO-DS1-EXT-MIB.db
1285. CISCO-EIGRP-MIB data/mibs/cisco/CISCO-EIGRP-MIB.db
1286. CISCO-EMBEDDED-EVENT-MGR-MIB CISCO-OLD-EMBEDDED-EVENT-MGR-MIB data/mibs/cisco/CISCO-EMBEDDED-EVENT-MGR-MIB CISCO-OLD-EMBEDDED-EVENT-MGR-MIB.db
1287. CISCO-IMAGE-TC data/mibs/cisco/CISCO-IMAGE-TC.db
1288. CISCO-ENHANCED-IMAGE-MIB data/mibs/cisco/CISCO-ENHANCED-IMAGE-MIB.db
1289. CISCO-IPSEC-TC data/mibs/cisco/CISCO-IPSEC-TC.db
1290. CISCO-ENHANCED-IPSEC-FLOW-MIB data/mibs/cisco/CISCO-ENHANCED-IPSEC-FLOW-MIB.db
1291. CISCO-FCC-MIB data/mibs/cisco/CISCO-FCC-MIB.db
1292. CISCO-FCIP-MGMT-MIB data/mibs/cisco/CISCO-FCIP-MGMT-MIB.db
1293. CISCO-FCIP-MGMT-EXT-MIB data/mibs/cisco/CISCO-FCIP-MGMT-EXT-MIB.db
1294. CISCO-FC-MULTICAST-MIB data/mibs/cisco/CISCO-FC-MULTICAST-MIB.db
1295. CISCO-FC-SPAN-MIB data/mibs/cisco/CISCO-FC-SPAN-MIB.db
1296. CISCO-FDMI-MIB data/mibs/cisco/CISCO-FDMI-MIB.db
1297. CISCO-FEATURE-CONTROL-MIB data/mibs/cisco/CISCO-FEATURE-CONTROL-MIB.db
1298. CISCO-FLEX-LINKS-MIB data/mibs/cisco/CISCO-FLEX-LINKS-MIB.db
1299. CISCO-HEALTH-MONITOR-MIB data/mibs/cisco/CISCO-HEALTH-MONITOR-MIB.db
1300. CISCO-IETF-IPROUTE-MIB data/mibs/cisco/CISCO-IETF-IPROUTE-MIB.db
1301. CISCO-IETF-PIM-MIB data/mibs/cisco/CISCO-IETF-PIM-MIB.db
1302. CISCO-IETF-PIM-EXT-MIB data/mibs/cisco/CISCO-IETF-PIM-EXT-MIB.db
1303. CISCO-IETF-ISIS-MIB data/mibs/cisco/CISCO-IETF-ISIS-MIB.db
1304. CISCO-IETF-ISNS-MGMT-MIB data/mibs/cisco/CISCO-IETF-ISNS-MGMT-MIB.db
1305. CISCO-IETF-MEGACO-MIB data/mibs/cisco/CISCO-IETF-MEGACO-MIB.db
1306. CISCO-IETF-PW-TC-MIB data/mibs/cisco/CISCO-IETF-PW-TC-MIB.db
1307. CISCO-IETF-PW-FR-MIB data/mibs/cisco/CISCO-IETF-PW-FR-MIB.db
1308. CISCO-IETF-PW-MIB data/mibs/cisco/CISCO-IETF-PW-MIB.db
1309. CISCO-IPSEC-SIGNALING-MIB data/mibs/cisco/CISCO-IPSEC-SIGNALING-MIB.db
1310. CISCO-IKE-CONFIGURATION-MIB .. data/mibs/cisco/CISCO-IKE-CONFIGURATION-MIB.db
1311. CISCO-IKE-FLOW-EXT-MIB data/mibs/cisco/CISCO-IKE-FLOW-EXT-MIB.db
1312. CISCO-IKE-FLOW-MIB data/mibs/cisco/CISCO-IKE-FLOW-MIB.db
1313. CISCO-IMAGE-CHECK-MIB data/mibs/cisco/CISCO-IMAGE-CHECK-MIB.db
1314. CISCO-IMAGE-UPGRADE-MIB data/mibs/cisco/CISCO-IMAGE-UPGRADE-MIB.db
1315. CISCO-IPSEC-PROVISIONING-MIB . data/mibs/cisco/CISCO-IPSEC-PROVISIONING-MIB.db
1316. CISCO-ISNS-CLIENT-MIB data/mibs/cisco/CISCO-ISNS-CLIENT-MIB.db
1317. CISCO-ISNS-IP-NW-DISCOVERY-MIB data/mibs/cisco/CISCO-ISNS-IP-NW-DISCOVERY-MIB.db
1318. CISCO-IVR-MIB data/mibs/cisco/CISCO-IVR-MIB.db
1319. CISCO-L2-CONTROL-MIB data/mibs/cisco/CISCO-L2-CONTROL-MIB.db
1320. CISCO-LATITUDE-MIB data/mibs/cisco/CISCO-LATITUDE-MIB.db
1321. CISCO-LICENSE-MGR-MIB data/mibs/cisco/CISCO-LICENSE-MGR-MIB.db
1322. CISCO-LICENSE-MIB data/mibs/cisco/CISCO-LICENSE-MIB.db
1323. CISCO-LINK-ERROR-MONITOR-MIB . data/mibs/cisco/CISCO-LINK-ERROR-MONITOR-MIB.db
1324. CISCO-MAU-EXT-MIB data/mibs/cisco/CISCO-MAU-EXT-MIB.db
1325. CISCO-MVPN-MIB data/mibs/cisco/CISCO-MVPN-MIB.db
1326. CISCO-POE-PD-MIB data/mibs/cisco/CISCO-POE-PD-MIB.db
1327. CISCO-PORT-STORM-CONTROL-MIB . data/mibs/cisco/CISCO-PORT-STORM-CONTROL-MIB.db
1328. CISCO-PORT-TRACK-MIB data/mibs/cisco/CISCO-PORT-TRACK-MIB.db
1329. CISCO-POWER-ETHERNET-EXT-MIB . data/mibs/cisco/CISCO-POWER-ETHERNET-EXT-MIB.db
1330. CISCO-PROP-ATM-IF-MIB data/mibs/cisco/CISCO-PROP-ATM-IF-MIB.db
1331. CISCO-PSM-MIB data/mibs/cisco/CISCO-PSM-MIB.db
1332. CISCO-REMOTE-ACCESS-MONITOR-MIB data/mibs/cisco/CISCO-REMOTE-ACCESS-MONITOR-MIB.db
1333. CISCO-SANTAP-MIB data/mibs/cisco/CISCO-SANTAP-MIB.db
1334. CISCO-SNMP-NOTIFICATION-EXT-MIB data/mibs/cisco/CISCO-SNMP-NOTIFICATION-EXT-MIB.db
1335. CISCO-SNMP-VACM-EXT-MIB data/mibs/cisco/CISCO-SNMP-VACM-EXT-MIB.db
1336. CISCO-SRST-MIB data/mibs/cisco/CISCO-SRST-MIB.db
1337. CISCO-SSL-PROXY-MIB data/mibs/cisco/CISCO-SSL-PROXY-MIB.db
1338. CISCO-SVC-INTERFACE-MIB data/mibs/cisco/CISCO-SVC-INTERFACE-MIB.db
1339. CISCO-TPC-MIB data/mibs/cisco/CISCO-TPC-MIB.db
1340. CISCO-VISM-CAC-MIB data/mibs/cisco/CISCO-VISM-CAC-MIB.db
1341. CISCO-VISM-CAS-MIB data/mibs/cisco/CISCO-VISM-CAS-MIB.db
1342. CISCO-VISM-CODEC-MIB data/mibs/cisco/CISCO-VISM-CODEC-MIB.db
1343. CISCO-VISM-HDLC-MIB data/mibs/cisco/CISCO-VISM-HDLC-MIB.db
1344. CISCO-VISM-MODULE-MIB data/mibs/cisco/CISCO-VISM-MODULE-MIB.db
1345. CISCO-VISM-PORT-MIB data/mibs/cisco/CISCO-VISM-PORT-MIB.db
1346. CISCO-VISM-RSRC-PART-MIB data/mibs/cisco/CISCO-VISM-RSRC-PART-MIB.db
1347. CISCO-VISM-SESSION-MIB data/mibs/cisco/CISCO-VISM-SESSION-MIB.db
1348. CISCO-VISM-SVC-MIB data/mibs/cisco/CISCO-VISM-SVC-MIB.db
1349. CISCO-VISM-XGCP-EXT data/mibs/cisco/CISCO-VISM-XGCP-EXT.db
1350. CISCO-WAN-AAL2-PROFILES-MIB .. data/mibs/cisco/CISCO-WAN-AAL2-PROFILES-MIB.db

1351. CISCO-WAN-ANNOUNCEMENT-MIB ... data/mibs/cisco/CISCO-WAN-ANNOUNCEMENT-MIB.db
1352. CISCO-WAN-ATM-COSB-MIB data/mibs/cisco/CISCO-WAN-ATM-COSB-MIB.db
1353. CISCO-WAN-ATM-CUG-MIB data/mibs/cisco/CISCO-WAN-ATM-CUG-MIB.db
1354. CISCO-WAN-ATM-PARTY-MIB data/mibs/cisco/CISCO-WAN-ATM-PARTY-MIB.db
1355. CISCO-WAN-ATM-PREF-ROUTE-MIB . data/mibs/cisco/CISCO-WAN-ATM-PREF-ROUTE-MIB.db
1356. CISCO-WAN-FR-SIGNALING-MIB ... data/mibs/cisco/CISCO-WAN-FR-SIGNALING-MIB.db
1357. CISCO-WAN-LAPD-TRUNK-MIB data/mibs/cisco/CISCO-WAN-LAPD-TRUNK-MIB.db
1358. CISCO-WAN-MG-MIB data/mibs/cisco/CISCO-WAN-MG-MIB.db
1359. CISCO-WAN-MGC-REDUN-MIB data/mibs/cisco/CISCO-WAN-MGC-REDUN-MIB.db
1360. CISCO-WAN-PERSISTENT-XGCP-EVENTS-MIB data/mibs/cisco/CISCO-WAN-PERSISTENT-XGCP-EVENTS-MIB.db
1361. CISCO-WAN-RPM-CONN-EXT-MIB ... data/mibs/cisco/CISCO-WAN-RPM-CONN-EXT-MIB.db
1362. CISCO-WAN-RTP-CONN-MIB data/mibs/cisco/CISCO-WAN-RTP-CONN-MIB.db
1363. CISCO-WAN-SRCP-MIB data/mibs/cisco/CISCO-WAN-SRCP-MIB.db
1364. CISCO-WAN-T38-FAXRELAY-MIB ... data/mibs/cisco/CISCO-WAN-T38-FAXRELAY-MIB.db
1365. CISCO-WAN-VISM-TONE-PLAN-MIB . data/mibs/cisco/CISCO-WAN-VISM-TONE-PLAN-MIB.db
1366. CISCO-WDS-IDS-MIB data/mibs/cisco/CISCO-WDS-IDS-MIB.db
1367. CISCO-WLAN-MAN-MIB data/mibs/cisco/CISCO-WLAN-MAN-MIB.db
1368. CISCO-ZS-EXT-MIB data/mibs/cisco/CISCO-ZS-EXT-MIB.db
1369. CISCO-SNMP-TARGET-EXT-MIB data/mibs/cisco/CISCO-SNMP-TARGET-EXT-MIB.db
1370. CISCO-DIFFSERV-EXT-MIB data/mibs/cisco/CISCO-DIFFSERV-EXT-MIB.db
1371. CISCO-COMMON-MGMT-MIB data/mibs/cisco/CISCO-COMMON-MGMT-MIB.db
1372. CISCO-DNS-SERVER-MIB data/mibs/cisco/CISCO-DNS-SERVER-MIB.db
1373. CISCO-DOT11-LBS-MIB data/mibs/cisco/CISCO-DOT11-LBS-MIB.db
1374. CISCO-SLB-HEALTH-MON-MIB data/mibs/cisco/CISCO-SLB-HEALTH-MON-MIB.db
1375. CISCO-ENHANCED-SLB-MIB data/mibs/cisco/CISCO-ENHANCED-SLB-MIB.db
1376. CISCO-FICON-MIB data/mibs/cisco/CISCO-FICON-MIB.db
1377. CISCO-FIREWALL-TC data/mibs/cisco/CISCO-FIREWALL-TC.db
1378. CISCO-GGSN-SERVICE-AWARE-MIB . data/mibs/cisco/CISCO-GGSN-SERVICE-AWARE-MIB.db
1379. CISCO-L4L7MODULE-RESOURCE-LIMIT-MIB data/mibs/cisco/CISCO-L4L7MODULE-RESOURCE-LIMIT-MIB.db
1380. CISCO-MODULE-VIRTUALIZATION-MIB data/mibs/cisco/CISCO-MODULE-VIRTUALIZATION-MIB.db
1381. CISCO-NAC-NAD-MIB data/mibs/cisco/CISCO-NAC-NAD-MIB.db
1382. CISCO-PKI-PARTICIPATION-MIB .. data/mibs/cisco/CISCO-PKI-PARTICIPATION-MIB.db
1383. CISCO-PSD-CLIENT-MIB data/mibs/cisco/CISCO-PSD-CLIENT-MIB.db
1384. CISCO-RTTMON-ICMP-MIB data/mibs/cisco/CISCO-RTTMON-ICMP-MIB.db
1385. CISCO-RTTMON-RTP-MIB data/mibs/cisco/CISCO-RTTMON-RTP-MIB.db
1386. CISCO-STACKWISE-MIB data/mibs/cisco/CISCO-STACKWISE-MIB.db
1387. CISCO-TAP2-MIB data/mibs/cisco/CISCO-TAP2-MIB.db
1388. CISCO-UNIFIED-FIREWALL-MIB ... data/mibs/cisco/CISCO-UNIFIED-FIREWALL-MIB.db
1389. CISCO-UNITY-EXPRESS-MIB data/mibs/cisco/CISCO-UNITY-EXPRESS-MIB.db
1390. CISCO-USER-CONNECTION-TAP-MIB data/mibs/cisco/CISCO-USER-CONNECTION-TAP-MIB.db
1391. CISCO-VLAN-TRANSLATION-MIB ... data/mibs/cisco/CISCO-VLAN-TRANSLATION-MIB.db
1392. CISCO-VOICE-APPLICATIONS-OID-MIB data/mibs/cisco/CISCO-VOICE-APPLICATIONS-OID-MIB.db
1393. CISCO-VOICE-CONNECTIVITY-MIB *** data/mibs/cisco/CISCO-VOICE-CONNECTIVITY-MIB ***.db
1394. CISCO-VOICE-TONE-CADENCE-MIB . data/mibs/cisco/CISCO-VOICE-TONE-CADENCE-MIB.db
1395. CISCO-WAN-ATM-CONN-STAT-MIB .. data/mibs/cisco/CISCO-WAN-ATM-CONN-STAT-MIB.db
1396. CISCO-WDS-INFO-MIB data/mibs/cisco/CISCO-WDS-INFO-MIB.db
1397. CISCO-XGCP-MIB data/mibs/cisco/CISCO-XGCP-MIB.db
1398. CISCO-XGCP-EXT-MIB data/mibs/cisco/CISCO-XGCP-EXT-MIB.db
1399. CISCO-ATM-PVCTRAP-EXTN-MIB ... data/mibs/cisco/CISCO-ATM-PVCTRAP-EXTN-MIB.db
1400. CISCO-TC-NO-U32 data/mibs/cisco/CISCO-TC-NO-U32.db
1401. CISCO-ATM-CONN-INFO-MIB data/mibs/cisco/CISCO-ATM-CONN-INFO-MIB.db
1402. CISCO-ATM-TRUNK-STAT-MIB data/mibs/cisco/CISCO-ATM-TRUNK-STAT-MIB.db
1403. CISCO-DYNAMIC-PORT-VSAN-MIB .. data/mibs/cisco/CISCO-DYNAMIC-PORT-VSAN-MIB.db
1404. CISCO-IPMCAST-MIB data/mibs/cisco/CISCO-IPMCAST-MIB.db
1405. ONS15501-MIB data/mibs/cisco/ONS15501-MIB.db
1406. EXPRESSION-MIB data/mibs/cisco/EXPRESSION-MIB.db
1407. DOCS-SUBMGMT-MIB data/mibs/cisco/DOCS-SUBMGMT-MIB.db
1408. AIRESPACE-REF-MIB data/mibs/cisco/AIRESPACE-REF-MIB.db
1409. AIRESPACE-SWITCHING-MIB data/mibs/cisco/AIRESPACE-SWITCHING-MIB.db
1410. AIRESPACE-WIRELESS-MIB data/mibs/cisco/AIRESPACE-WIRELESS-MIB.db
1411. CISCO-ATM-NETWORK-CLOCK-MIB .. data/mibs/cisco/CISCO-ATM-NETWORK-CLOCK-MIB.db
1412. CISCO-VOICE-AALX-PROFILE-MIB . data/mibs/cisco/CISCO-VOICE-AALX-PROFILE-MIB.db
1413. CISCO-ATM-TRUNK-MIB data/mibs/cisco/CISCO-ATM-TRUNK-MIB.db
1414. CISCO-BRIDGE-DOMAIN-MIB data/mibs/cisco/CISCO-BRIDGE-DOMAIN-MIB.db
1415. CISCO-BRIDGE-EXT-MIB data/mibs/cisco/CISCO-BRIDGE-EXT-MIB.db
1416. CISCO-CABLE-WIDEBAND-MIB data/mibs/cisco/CISCO-CABLE-WIDEBAND-MIB.db
1417. CISCO-CEF-TC data/mibs/cisco/CISCO-CEF-TC.db
1418. CISCO-CEF-MIB data/mibs/cisco/CISCO-CEF-MIB.db
1419. CISCO-DATA-COLLECTION-MIB data/mibs/cisco/CISCO-DATA-COLLECTION-MIB.db
1420. CISCO-DIST-DIRECTOR-MIB data/mibs/cisco/CISCO-DIST-DIRECTOR-MIB.db
1421. CISCO-DOT11-RADAR-MIB data/mibs/cisco/CISCO-DOT11-RADAR-MIB.db
1422. CISCO-ENTITY-DIAG-TC-MIB data/mibs/cisco/CISCO-ENTITY-DIAG-TC-MIB.db
1423. CISCO-ENTITY-DIAG-MIB data/mibs/cisco/CISCO-ENTITY-DIAG-MIB.db
1424. CISCO-ENTITY-REDUNDANCY-TC-MIB data/mibs/cisco/CISCO-ENTITY-REDUNDANCY-TC-MIB.db
1425. CISCO-ENTITY-REDUNDANCY-MIB .. data/mibs/cisco/CISCO-ENTITY-REDUNDANCY-MIB.db
1426. CISCO-ERR-DISABLE-MIB data/mibs/cisco/CISCO-ERR-DISABLE-MIB.db
1427. CISCO-ETHERNET-ACCESS-MIB data/mibs/cisco/CISCO-ETHERNET-ACCESS-MIB.db
1428. CISCO-FC-DEVICE-ALIAS-MIB data/mibs/cisco/CISCO-FC-DEVICE-ALIAS-MIB.db
1429. CISCO-FC-SDV-MIB data/mibs/cisco/CISCO-FC-SDV-MIB.db
1430. CISCO-FCSP-MIB data/mibs/cisco/CISCO-FCSP-MIB.db
1431. CISCO-FLOW-CLONE-MIB data/mibs/cisco/CISCO-FLOW-CLONE-MIB.db

1432. CISCO-GSLB-TC-MIB data/mibs/cisco/CISCO-GSLB-TC-MIB.db
1433. CISCO-GSLB-DNS-MIB data/mibs/cisco/CISCO-GSLB-DNS-MIB.db
1434. CISCO-GSLB-HEALTH-MON-MIB data/mibs/cisco/CISCO-GSLB-HEALTH-MON-MIB.db
1435. CISCO-GSLB-SYSTEM-MIB data/mibs/cisco/CISCO-GSLB-SYSTEM-MIB.db
1436. CISCO-LWAPP-TC-MIB data/mibs/cisco/CISCO-LWAPP-TC-MIB.db
1437. CISCO-LWAPP-WLAN-MIB data/mibs/cisco/CISCO-LWAPP-WLAN-MIB.db
1438. CISCO-LWAPP-AAA-MIB data/mibs/cisco/CISCO-LWAPP-AAA-MIB.db
1439. CISCO-LWAPP-ACL-MIB data/mibs/cisco/CISCO-LWAPP-ACL-MIB.db
1440. CISCO-LWAPP-DOT11-MIB data/mibs/cisco/CISCO-LWAPP-DOT11-MIB.db
1441. CISCO-LWAPP-RF-MIB data/mibs/cisco/CISCO-LWAPP-RF-MIB.db
1442. CISCO-LWAPP-AP-MIB data/mibs/cisco/CISCO-LWAPP-AP-MIB.db
1443. CISCO-LWAPP-CCX-RM-MIB data/mibs/cisco/CISCO-LWAPP-CCX-RM-MIB.db
1444. CISCO-LWAPP-CDP-MIB data/mibs/cisco/CISCO-LWAPP-CDP-MIB.db
1445. CISCO-LWAPP-CLIENT-ROAMING-MIB data/mibs/cisco/CISCO-LWAPP-CLIENT-ROAMING-MIB.db
1446. CISCO-LWAPP-MOBILITY-EXT-MIB . data/mibs/cisco/CISCO-LWAPP-MOBILITY-EXT-MIB.db
1447. CISCO-LWAPP-DOT11-CLIENT-CALIB-MIB data/mibs/cisco/CISCO-LWAPP-DOT11-CLIENT-CALIB-MIB.db
1448. CISCO-LWAPP-DOT11-CLIENT-CCX-TC-MIB data/mibs/cisco/CISCO-LWAPP-DOT11-CLIENT-CCX-TC-MIB.db
1449. CISCO-LWAPP-DOT11-CLIENT-MIB . data/mibs/cisco/CISCO-LWAPP-DOT11-CLIENT-MIB.db
1450. CISCO-LWAPP-DOT11-LDAP-MIB ... data/mibs/cisco/CISCO-LWAPP-DOT11-LDAP-MIB.db
1451. CISCO-LWAPP-IDS-MIB data/mibs/cisco/CISCO-LWAPP-IDS-MIB.db
1452. CISCO-LWAPP-LINKTEST-MIB data/mibs/cisco/CISCO-LWAPP-LINKTEST-MIB.db
1453. CISCO-LWAPP-LOCAL-AUTH-MIB ... data/mibs/cisco/CISCO-LWAPP-LOCAL-AUTH-MIB.db
1454. CISCO-LWAPP-MESH-BATTERY-MIB . data/mibs/cisco/CISCO-LWAPP-MESH-BATTERY-MIB.db
1455. CISCO-LWAPP-MESH-MIB data/mibs/cisco/CISCO-LWAPP-MESH-MIB.db
1456. CISCO-LWAPP-MESH-STATS-MIB ... data/mibs/cisco/CISCO-LWAPP-MESH-STATS-MIB.db
1457. CISCO-LWAPP-MFP-MIB data/mibs/cisco/CISCO-LWAPP-MFP-MIB.db
1458. CISCO-LWAPP-MOBILITY-MIB data/mibs/cisco/CISCO-LWAPP-MOBILITY-MIB.db
1459. CISCO-LWAPP-QOS-MIB data/mibs/cisco/CISCO-LWAPP-QOS-MIB.db
1460. CISCO-LWAPP-REAP-MIB data/mibs/cisco/CISCO-LWAPP-REAP-MIB.db
1461. CISCO-LWAPP-ROGUE-MIB data/mibs/cisco/CISCO-LWAPP-ROGUE-MIB.db
1462. CISCO-LWAPP-RRM-MIB data/mibs/cisco/CISCO-LWAPP-RRM-MIB.db
1463. CISCO-LWAPP-SYS-MIB data/mibs/cisco/CISCO-LWAPP-SYS-MIB.db
1464. CISCO-LWAPP-TSM-MIB data/mibs/cisco/CISCO-LWAPP-TSM-MIB.db
1465. CISCO-LWAPP-WEBAUTH-MIB data/mibs/cisco/CISCO-LWAPP-WEBAUTH-MIB.db
1466. CISCO-LWAPP-WLAN-SECURITY-MIB data/mibs/cisco/CISCO-LWAPP-WLAN-SECURITY-MIB.db
1467. CISCO-NAT-EXT-MIB data/mibs/cisco/CISCO-NAT-EXT-MIB.db
1468. CISCO-NETINT-MIB data/mibs/cisco/CISCO-NETINT-MIB.db
1469. CISCO-OTN-IF-MIB data/mibs/cisco/CISCO-OTN-IF-MIB.db
1470. CISCO-PACKET-CAPTURE-MIB data/mibs/cisco/CISCO-PACKET-CAPTURE-MIB.db
1471. CISCO-PREFERRED-PATH-MIB data/mibs/cisco/CISCO-PREFERRED-PATH-MIB.db
1472. CISCO-QINQ-VLAN-MIB data/mibs/cisco/CISCO-QINQ-VLAN-MIB.db
1473. CISCO-QOS-TC-MIB data/mibs/cisco/CISCO-QOS-TC-MIB.db
1474. CISCO-RESILIENT-ETHERNET-PROTOCOL-MIB data/mibs/cisco/CISCO-RESILIENT-ETHERNET-PROTOCOL-MIB.db
1475. CISCO-SCSI-FLOW-MIB data/mibs/cisco/CISCO-SCSI-FLOW-MIB.db
1476. CISCO-SME-MIB data/mibs/cisco/CISCO-SME-MIB.db
1477. CISCO-SNMP-USM-OIDS-MIB data/mibs/cisco/CISCO-SNMP-USM-OIDS-MIB.db
1478. CISCO-SSM-PROV-MIB data/mibs/cisco/CISCO-SSM-PROV-MIB.db
1479. CISCO-SWITCH-MULTICAST-MIB ... data/mibs/cisco/CISCO-SWITCH-MULTICAST-MIB.db
1480. CISCO-SWITCH-QOS-MIB data/mibs/cisco/CISCO-SWITCH-QOS-MIB.db
1481. CISCO-VIRTUAL-SWITCH-MIB data/mibs/cisco/CISCO-VIRTUAL-SWITCH-MIB.db
1482. CISCO-AON-STATUS-MIB data/mibs/cisco/CISCO-AON-STATUS-MIB.db
1483. CISCO-APPLICATION-ACCELERATION-MIB data/mibs/cisco/CISCO-APPLICATION-ACCELERATION-MIB.db
1484. CISCO-ASN-GATEWAY-MIB data/mibs/cisco/CISCO-ASN-GATEWAY-MIB.db
1485. CISCO-AUTH-FRAMEWORK-MIB data/mibs/cisco/CISCO-AUTH-FRAMEWORK-MIB.db
1486. CISCO-BOOT-HWDIAGS-MIB data/mibs/cisco/CISCO-BOOT-HWDIAGS-MIB.db
1487. CISCO-CDMA-PDSN-EXT-MIB data/mibs/cisco/CISCO-CDMA-PDSN-EXT-MIB.db
1488. CISCO-COMMON-ROLES-EXT-MIB ... data/mibs/cisco/CISCO-COMMON-ROLES-EXT-MIB.db
1489. CISCO-CONTACT-CENTER-APPS-MIB data/mibs/cisco/CISCO-CONTACT-CENTER-APPS-MIB.db
1490. CISCO-CONTEXT-MAPPING-MIB data/mibs/cisco/CISCO-CONTEXT-MAPPING-MIB.db
1491. CISCO-DIAMETER-BASE-PROTOCOL-MIB data/mibs/cisco/CISCO-DIAMETER-BASE-PROTOCOL-MIB.db
1492. CISCO-DIAMETER-CC-APPL-MIB ... data/mibs/cisco/CISCO-DIAMETER-CC-APPL-MIB.db
1493. CISCO-DIAMETER-SG-MIB data/mibs/cisco/CISCO-DIAMETER-SG-MIB.db
1494. CISCO-DIGITAL-MEDIA-SYSTEMS-MIB data/mibs/cisco/CISCO-DIGITAL-MEDIA-SYSTEMS-MIB.db
1495. CISCO-DOT3-OAM-MIB data/mibs/cisco/CISCO-DOT3-OAM-MIB.db
1496. CISCO-ENERGYWISE-MIB data/mibs/cisco/CISCO-ENERGYWISE-MIB.db
1497. CISCO-ERM-MIB data/mibs/cisco/CISCO-ERM-MIB.db
1498. CISCO-ETHERNET-FABRIC-EXTENDER-MIB data/mibs/cisco/CISCO-ETHERNET-FABRIC-EXTENDER-MIB.db
1499. CISCO-EVC-MIB data/mibs/cisco/CISCO-EVC-MIB.db
1500. CISCO-FCOE-MIB data/mibs/cisco/CISCO-FCOE-MIB.db
1501. CISCO-GGSN-EXT-MIB data/mibs/cisco/CISCO-GGSN-EXT-MIB.db
1502. CISCO-H324-DIAL-CONTROL-MIB .. data/mibs/cisco/CISCO-H324-DIAL-CONTROL-MIB.db
1503. CISCO-IETF-BFD-MIB data/mibs/cisco/CISCO-IETF-BFD-MIB.db
1504. CISCO-IETF-DHCP-SERVER-MIB ... data/mibs/cisco/CISCO-IETF-DHCP-SERVER-MIB.db
1505. CISCO-IETF-DHCP-SERVER-EXT-MIB data/mibs/cisco/CISCO-IETF-DHCP-SERVER-EXT-MIB.db
1506. CISCO-IETF-FRR-MIB data/mibs/cisco/CISCO-IETF-FRR-MIB.db
1507. CISCO-IETF-VPLS-GENERIC-MIB .. data/mibs/cisco/CISCO-IETF-VPLS-GENERIC-MIB.db
1508. CISCO-IETF-VPLS-BGP-EXT-MIB .. data/mibs/cisco/CISCO-IETF-VPLS-BGP-EXT-MIB.db
1509. CISCO-IETF-VPLS-LDP-MIB data/mibs/cisco/CISCO-IETF-VPLS-LDP-MIB.db
1510. INTERFACETOPN-MIB data/mibs/cisco/INTERFACETOPN-MIB.db
1511. CISCO-INTERFACETOPN-EXT-MIB .. data/mibs/cisco/CISCO-INTERFACETOPN-EXT-MIB.db
1512. CISCO-IP-URPF-MIB data/mibs/cisco/CISCO-IP-URPF-MIB.db

1513. CISCO-L4L7MODULE-REDUNDANCY-MIB data/mibs/cisco/CISCO-L4L7MODULE-REDUNDANCY-MIB.db
1514. CISCO-MAC-AUTH-BYPASS-MIB data/mibs/cisco/CISCO-MAC-AUTH-BYPASS-MIB.db
1515. CISCO-MMODAL-CONTACT-APPS-MIB data/mibs/cisco/CISCO-MMODAL-CONTACT-APPS-MIB.db
1516. CISCO-MOBILE-POLICY-CHARGING-CONTROL-MIB data/mibs/cisco/CISCO-MOBILE-POLICY-CHARGING-CONTROL-MIB.db
1517. CISCO-MOBILITY-TAP-MIB data/mibs/cisco/CISCO-MOBILITY-TAP-MIB.db
1518. CISCO-NOTIFICATION-CONTROL-MIB data/mibs/cisco/CISCO-NOTIFICATION-CONTROL-MIB.db
1519. CISCO-OBMI-MIB data/mibs/cisco/CISCO-OBMI-MIB.db
1520. CISCO-P2P-IF-MIB data/mibs/cisco/CISCO-P2P-IF-MIB.db
1521. CISCO-RTTMON-IP-EXT-MIB data/mibs/cisco/CISCO-RTTMON-IP-EXT-MIB.db
1522. CISCO-SAN-BASE-SVC-MIB data/mibs/cisco/CISCO-SAN-BASE-SVC-MIB.db
1523. CISCO-SERVICE-CONTROL-ATTACK-MIB data/mibs/cisco/CISCO-SERVICE-CONTROL-ATTACK-MIB.db
1524. CISCO-SERVICE-CONTROL-LINK-MIB data/mibs/cisco/CISCO-SERVICE-CONTROL-LINK-MIB.db
1525. CISCO-SERVICE-CONTROL-RDR-MIB data/mibs/cisco/CISCO-SERVICE-CONTROL-RDR-MIB.db
1526. CISCO-SERVICE-CONTROL-SUBSCRIBERS-MIB data/mibs/cisco/CISCO-SERVICE-CONTROL-SUBSCRIBERS-MIB.db
1527. CISCO-SERVICE-CONTROL-TP-STATS-MIB data/mibs/cisco/CISCO-SERVICE-CONTROL-TP-STATS-MIB.db
1528. CISCO-SERVICE-CONTROLLER-MIB . data/mibs/cisco/CISCO-SERVICE-CONTROLLER-MIB.db
1529. CISCO-SESS-BORDER-CTRLR-CALL-STATS-MIB data/mibs/cisco/CISCO-SESS-BORDER-CTRLR-CALL-STATS-MIB.db
1530. CISCO-SESS-BORDER-CTRLR-EVENT-MIB data/mibs/cisco/CISCO-SESS-BORDER-CTRLR-EVENT-MIB.db
1531. CISCO-SEU-MITIGATION-MIB data/mibs/cisco/CISCO-SEU-MITIGATION-MIB.db
1532. CISCO-SLB-DFP-MIB data/mibs/cisco/CISCO-SLB-DFP-MIB.db
1533. CISCO-SWITCH-HARDWARE-CAPACITY-MIB data/mibs/cisco/CISCO-SWITCH-HARDWARE-CAPACITY-MIB.db
1534. CISCO-TELEPRESENCE-CALL-MIB . . data/mibs/cisco/CISCO-TELEPRESENCE-CALL-MIB.db
1535. CISCO-TELEPRESENCE-MIB data/mibs/cisco/CISCO-TELEPRESENCE-MIB.db
1536. CISCO-TELNET-SERVER-MIB data/mibs/cisco/CISCO-TELNET-SERVER-MIB.db
1537. CISCO-UNITY-MIB data/mibs/cisco/CISCO-UNITY-MIB.db
1538. CISCO-VIRTUAL-INTERFACE-MIB . . data/mibs/cisco/CISCO-VIRTUAL-INTERFACE-MIB.db
1539. CISCO-WAN-3G-MIB data/mibs/cisco/CISCO-WAN-3G-MIB.db
1540. CLAB-DEF-MIB data/mibs/cisco/CLAB-DEF-MIB.db
1541. CLAB-TOPO-MIB data/mibs/cisco/CLAB-TOPO-MIB.db
1542. DOCS-IF3-MIB data/mibs/cisco/DOCS-IF3-MIB.db
1543. DOCS-DIAG-MIB data/mibs/cisco/DOCS-DIAG-MIB.db
1544. DOCS-DRF-MIB data/mibs/cisco/DOCS-DRF-MIB.db
1545. DOCS-IFEXT2-MIB data/mibs/cisco/DOCS-IFEXT2-MIB.db
1546. DOCS-L2VPN-MIB data/mibs/cisco/DOCS-L2VPN-MIB.db
1547. DOCS-QOS3-MIB data/mibs/cisco/DOCS-QOS3-MIB.db
1548. DOCS-TEST-MIB data/mibs/cisco/DOCS-TEST-MIB.db
1549. LLDP-MIB data/mibs/cisco/LLDP-MIB.db
1550. LLDP-EXT-MED-MIB data/mibs/cisco/LLDP-EXT-MED-MIB.db
1551. CISCO-CABLE-L2VPN-MIB data/mibs/cisco/CISCO-CABLE-L2VPN-MIB.db
1552. CISCO-CDMA-PDSN-CRP-MIB data/mibs/cisco/CISCO-CDMA-PDSN-CRP-MIB.db
1553. CISCO-CDSTV-AUTHMGR-MIB data/mibs/cisco/CISCO-CDSTV-AUTHMGR-MIB.db
1554. CISCO-CDSTV-BWMGR-MIB data/mibs/cisco/CISCO-CDSTV-BWMGR-MIB.db
1555. CISCO-CDSTV-CS-STATS-MIB data/mibs/cisco/CISCO-CDSTV-CS-STATS-MIB.db
1556. CISCO-CDSTV-FSI-MIB data/mibs/cisco/CISCO-CDSTV-FSI-MIB.db
1557. CISCO-CDSTV-INGEST-TUNING-MIB data/mibs/cisco/CISCO-CDSTV-INGEST-TUNING-MIB.db
1558. CISCO-CDSTV-INGESTMGR-MIB data/mibs/cisco/CISCO-CDSTV-INGESTMGR-MIB.db
1559. CISCO-CDSTV-ISA-MIB data/mibs/cisco/CISCO-CDSTV-ISA-MIB.db
1560. CISCO-CDSTV-SERVER-MIB data/mibs/cisco/CISCO-CDSTV-SERVER-MIB.db
1561. CISCO-CDSTV-SERVICES-MIB data/mibs/cisco/CISCO-CDSTV-SERVICES-MIB.db
1562. CISCO-CONTENT-DELIVERY-STREAMING-MIB data/mibs/cisco/CISCO-CONTENT-DELIVERY-STREAMING-MIB.db
1563. CISCO-CONTENT-SERVICES-MIB . . . data/mibs/cisco/CISCO-CONTENT-SERVICES-MIB.db
1564. CISCO-CUICAPPS-MIB data/mibs/cisco/CISCO-CUICAPPS-MIB.db
1565. CISCO-CVP-MIB data/mibs/cisco/CISCO-CVP-MIB.db
1566. CISCO-DOCS-EXT-MIB data/mibs/cisco/CISCO-DOCS-EXT-MIB.db
1567. CISCO-DOCS-REMOTE-QUERY-MIB . . data/mibs/cisco/CISCO-DOCS-REMOTE-QUERY-MIB.db
1568. CISCO-ENTITY-QFP-MIB data/mibs/cisco/CISCO-ENTITY-QFP-MIB.db
1569. CISCO-ENTITY-SENSOR-HISTORY-MIB data/mibs/cisco/CISCO-ENTITY-SENSOR-HISTORY-MIB.db
1570. CISCO-ETHERLIKE-EXT-MIB data/mibs/cisco/CISCO-ETHERLIKE-EXT-MIB.db
1571. CISCO-GGSN-GEO-MIB data/mibs/cisco/CISCO-GGSN-GEO-MIB.db
1572. CISCO-GPRS-ISGSN-MIB data/mibs/cisco/CISCO-GPRS-ISGSN-MIB.db
1573. CISCO-H225-MIB data/mibs/cisco/CISCO-H225-MIB.db
1574. CISCO-H320-DIAL-CONTROL-MIB . . data/mibs/cisco/CISCO-H320-DIAL-CONTROL-MIB.db
1575. CISCO-HW-MODULE-CONTROL-MIB . . data/mibs/cisco/CISCO-HW-MODULE-CONTROL-MIB.db
1576. IEEEB8021-CFM-MIB data/mibs/cisco/IEEB8021-CFM-MIB.db
1577. CISCO-IEEB8021-CFM-EXT-MIB . . . data/mibs/cisco/CISCO-IEEB8021-CFM-EXT-MIB.db
1578. CISCO-IETF-MPLS-TE-P2MP-STD-MIB data/mibs/cisco/CISCO-IETF-MPLS-TE-P2MP-STD-MIB.db
1579. CISCO-IETF-PW-ATM-MIB data/mibs/cisco/CISCO-IETF-PW-ATM-MIB.db
1580. CISCO-IETF-PW-TDM-MIB data/mibs/cisco/CISCO-IETF-PW-TDM-MIB.db
1581. CISCO-IMAGE-LICENSE-MGMT-MIB . data/mibs/cisco/CISCO-IMAGE-LICENSE-MGMT-MIB.db
1582. CISCO-INTERFACE-XCVR-MONITOR-MIB data/mibs/cisco/CISCO-INTERFACE-XCVR-MONITOR-MIB.db
1583. CISCO-IP-RAN-BACKHAUL-MIB data/mibs/cisco/CISCO-IP-RAN-BACKHAUL-MIB.db
1584. CISCO-IPSLA-TC-MIB data/mibs/cisco/CISCO-IPSLA-TC-MIB.db
1585. CISCO-IPSLA-AUTOMEASURE-MIB . . data/mibs/cisco/CISCO-IPSLA-AUTOMEASURE-MIB.db
1586. CISCO-IPSLA-ECHO-MIB data/mibs/cisco/CISCO-IPSLA-ECHO-MIB.db
1587. CISCO-IPSLA-ETHERNET-MIB data/mibs/cisco/CISCO-IPSLA-ETHERNET-MIB.db
1588. CISCO-IPSLA-JITTER-MIB data/mibs/cisco/CISCO-IPSLA-JITTER-MIB.db
1589. CISCO-IPSLA-VIDEO-MIB data/mibs/cisco/CISCO-IPSLA-VIDEO-MIB.db
1590. CISCO-LICENSE-MGMT-MIB data/mibs/cisco/CISCO-LICENSE-MGMT-MIB.db
1591. CISCO-NHRP-EXT-MIB data/mibs/cisco/CISCO-NHRP-EXT-MIB.db
1592. CISCO-NPORT-VIRTUALIZATION-MIB data/mibs/cisco/CISCO-NPORT-VIRTUALIZATION-MIB.db
1593. CISCO-SIP-CALLS-MIB data/mibs/cisco/CISCO-SIP-CALLS-MIB.db

1594. CISCO-SMART-INSTALL-MIB data/mibs/cisco/CISCO-SMART-INSTALL-MIB.db
1595. CISCO-TM data/mibs/cisco/CISCO-TM.db
1596. CISCO-TRUSTSEC-TC-MIB data/mibs/cisco/CISCO-TRUSTSEC-TC-MIB.db
1597. CISCO-TRUSTSEC-MIB data/mibs/cisco/CISCO-TRUSTSEC-MIB.db
1598. CISCO-TRUSTSEC-INTERFACE-MIB . data/mibs/cisco/CISCO-TRUSTSEC-INTERFACE-MIB.db
1599. CISCO-TRUSTSEC-SXP-MIB data/mibs/cisco/CISCO-TRUSTSEC-SXP-MIB.db
1600. CISCO-TRUSTSEC-POLICY-MIB data/mibs/cisco/CISCO-TRUSTSEC-POLICY-MIB.db
1601. CISCO-TRUSTSEC-SERVER-MIB data/mibs/cisco/CISCO-TRUSTSEC-SERVER-MIB.db
1602. CISCO-UBE-MIB data/mibs/cisco/CISCO-UBE-MIB.db
1603. CISCO-UNIFIED-COMPUTING-MIB .. data/mibs/cisco/CISCO-UNIFIED-COMPUTING-MIB.db
1604. CISCO-UNIFIED-COMPUTING-TC-MIB data/mibs/cisco/CISCO-UNIFIED-COMPUTING-TC-MIB.db
1605. CISCO-UNIFIED-COMPUTING-FAULT-MIB data/mibs/cisco/CISCO-UNIFIED-COMPUTING-FAULT-MIB.db
1606. CISCO-VIDEO-TC data/mibs/cisco/CISCO-VIDEO-TC.db
1607. CISCO-VIDEO-SESSION-MIB data/mibs/cisco/CISCO-VIDEO-SESSION-MIB.db
1608. CISCO-VIRTUAL-NIC-MIB data/mibs/cisco/CISCO-VIRTUAL-NIC-MIB.db
1609. CISCO-VLAN-GROUP-MIB data/mibs/cisco/CISCO-VLAN-GROUP-MIB.db
1610. CISCO-VOICE-CARD-MIB data/mibs/cisco/CISCO-VOICE-CARD-MIB.db
1611. CISCO-VOICE-LMR-MIB data/mibs/cisco/CISCO-VOICE-LMR-MIB.db
1612. CISCO-VOIP-TAP-MIB data/mibs/cisco/CISCO-VOIP-TAP-MIB.db
1613. CISCO-VQE-TOOLS-MIB data/mibs/cisco/CISCO-VQE-TOOLS-MIB.db
1614. CISCO-VQES-MIB data/mibs/cisco/CISCO-VQES-MIB.db
1615. DOCS-IF-M-CMTS-MIB data/mibs/cisco/DOCS-IF-M-CMTS-MIB.db
1616. DOCS-LOADBAL3-MIB data/mibs/cisco/DOCS-LOADBAL3-MIB.db
1617. DOCS-LOADBALANCING-MIB data/mibs/cisco/DOCS-LOADBALANCING-MIB.db
1618. DOCS-MCAST-AUTH-MIB data/mibs/cisco/DOCS-MCAST-AUTH-MIB.db
1619. DOCS-MCAST-MIB data/mibs/cisco/DOCS-MCAST-MIB.db
1620. DOCS-SEC-MIB data/mibs/cisco/DOCS-SEC-MIB.db
1621. DOCS-SUBMGT3-MIB data/mibs/cisco/DOCS-SUBMGT3-MIB.db
1622. DTI-MIB data/mibs/cisco/DTI-MIB.db
1623. IEEEB8021-TC-MIB data/mibs/cisco/IEEB8021-TC-MIB.db
1624. IEEEB8021-CFM-V2-MIB data/mibs/cisco/IEEB8021-CFM-V2-MIB.db
1625. CISCO-PMON-MIB data/mibs/cisco/CISCO-PMON-MIB.db
1626. CISCO-ITP-GTCAP-MIB data/mibs/cisco/CISCO-ITP-GTCAP-MIB.db
1627. CISCO-ITP-DSMR-MIB data/mibs/cisco/CISCO-ITP-DSMR-MIB.db
1628. CISCO-ITP-DSMR-SMPP-MIB data/mibs/cisco/CISCO-ITP-DSMR-SMPP-MIB.db
1629. CISCO-ITP-DSMR-UCP-MIB data/mibs/cisco/CISCO-ITP-DSMR-UCP-MIB.db
1630. CISCO-IPSLA-VIDEO-PROFILE-MIB data/mibs/cisco/CISCO-IPSLA-VIDEO-PROFILE-MIB.db
1631. CISCO-SELECTIVE-VRF-DOWNLOAD-MIB data/mibs/cisco/CISCO-SELECTIVE-VRF-DOWNLOAD-MIB.db
1632. CISCO-MEETINGPLACE-MIB data/mibs/cisco/CISCO-MEETINGPLACE-MIB.db
1633. CISCO-SWITCH-RATE-LIMITER-MIB data/mibs/cisco/CISCO-SWITCH-RATE-LIMITER-MIB.db
1634. CISCO-SESS-BORDER-CTRLR-STATS-MIB data/mibs/cisco/CISCO-SESS-BORDER-CTRLR-STATS-MIB.db
1635. CISCO-GDOI-MIB data/mibs/cisco/CISCO-GDOI-MIB.db
1636. CISCO-MEDIA-QUALITY-MIB data/mibs/cisco/CISCO-MEDIA-QUALITY-MIB.db
1637. CISCO-SWITCH-NETFLOW-MIB data/mibs/cisco/CISCO-SWITCH-NETFLOW-MIB.db
1638. CISCO-SWITCH-STATS-MIB data/mibs/cisco/CISCO-SWITCH-STATS-MIB.db
1639. CISCO-MPLS-TE-STD-EXT-MIB data/mibs/cisco/CISCO-MPLS-TE-STD-EXT-MIB.db
1640. CISCO-WAN-OPTIMIZATION-MIB ... data/mibs/cisco/CISCO-WAN-OPTIMIZATION-MIB.db
1641. CISCO-ACL-MIB data/mibs/cisco/CISCO-ACL-MIB.db
1642. CISCO-APPNAV-MIB data/mibs/cisco/CISCO-APPNAV-MIB.db
1643. CISCO-CABLE-DSG-IF-MIB data/mibs/cisco/CISCO-CABLE-DSG-IF-MIB.db
1644. CISCO-MEGACO-EXT-MIB data/mibs/cisco/CISCO-MEGACO-EXT-MIB.db
1645. CISCO-CAS-IF-EXT-MIB data/mibs/cisco/CISCO-CAS-IF-EXT-MIB.db
1646. CISCO-CBP-TC-MIB data/mibs/cisco/CISCO-CBP-TC-MIB.db
1647. CISCO-CBP-TARGET-TC-MIB data/mibs/cisco/CISCO-CBP-TARGET-TC-MIB.db
1648. CISCO-CBP-TARGET-MIB data/mibs/cisco/CISCO-CBP-TARGET-MIB.db
1649. CISCO-DOT11-HT-MAC-MIB data/mibs/cisco/CISCO-DOT11-HT-MAC-MIB.db
1650. CISCO-DOT11-HT-PHY-MIB data/mibs/cisco/CISCO-DOT11-HT-PHY-MIB.db
1651. DSG-IF-MIB data/mibs/cisco/DSG-IF-MIB.db
1652. CISCO-DSG-IF-EXT-MIB data/mibs/cisco/CISCO-DSG-IF-EXT-MIB.db
1653. CISCO-DYNAMIC-TEMPLATE-TC-MIB data/mibs/cisco/CISCO-DYNAMIC-TEMPLATE-TC-MIB.db
1654. CISCO-FABRICPATH-TOPOLOGY-MIB data/mibs/cisco/CISCO-FABRICPATH-TOPOLOGY-MIB.db
1655. CISCO-FC-MGMT-MIB data/mibs/cisco/CISCO-FC-MGMT-MIB.db
1656. CISCO-FLOW-METADATA-MIB data/mibs/cisco/CISCO-FLOW-METADATA-MIB.db
1657. CISCO-GPRS-L2RLY-MIB data/mibs/cisco/CISCO-GPRS-L2RLY-MIB.db
1658. CISCO-HARDWARE-IP-VERIFY-MIB . data/mibs/cisco/CISCO-HARDWARE-IP-VERIFY-MIB.db
1659. CISCO-MPLS-TC-EXT-STD-MIB data/mibs/cisco/CISCO-MPLS-TC-EXT-STD-MIB.db
1660. CISCO-IETF-MPLS-TE-EXT-STD-03-MIB data/mibs/cisco/CISCO-IETF-MPLS-TE-EXT-STD-03-MIB.db
1661. CISCO-IETF-MPLS-ID-STD-03-MIB data/mibs/cisco/CISCO-IETF-MPLS-ID-STD-03-MIB.db
1662. CISCO-IETF-MSDP-MIB data/mibs/cisco/CISCO-IETF-MSDP-MIB.db
1663. CISCO-IETF-PW-MPLS-MIB data/mibs/cisco/CISCO-IETF-PW-MPLS-MIB.db
1664. CISCO-IETF-PW-ENET-MIB data/mibs/cisco/CISCO-IETF-PW-ENET-MIB.db
1665. CISCO-LIVEDATA-MIB data/mibs/cisco/CISCO-LIVEDATA-MIB.db
1666. CISCO-LOCAL-AUTH-USER-MIB data/mibs/cisco/CISCO-LOCAL-AUTH-USER-MIB.db
1667. CISCO-LWAPP-DOT11-CLIENT-CCX-REPORTS-MIB data/mibs/cisco/CISCO-LWAPP-DOT11-CLIENT-CCX-REPORTS-MIB.db
1668. CISCO-LWAPP-INTERFACE-MIB data/mibs/cisco/CISCO-LWAPP-INTERFACE-MIB.db
1669. CISCO-LWAPP-MESH-LINKTEST-MIB data/mibs/cisco/CISCO-LWAPP-MESH-LINKTEST-MIB.db
1670. CISCO-MEDIATRACE-MIB DEFINITIONS data/mibs/cisco/CISCO-MEDIATRACE-MIB DEFINITIONS.db
1671. CISCO-MPLS-LSR-EXT-STD-MIB ... data/mibs/cisco/CISCO-MPLS-LSR-EXT-STD-MIB.db
1672. CISCO-MSP-MIB data/mibs/cisco/CISCO-MSP-MIB.db
1673. CISCO-NETFLOW-LITE-MIB data/mibs/cisco/CISCO-NETFLOW-LITE-MIB.db
1674. CISCO-OTV-MIB data/mibs/cisco/CISCO-OTV-MIB.db

1675. CISCO-PFR-MIB data/mibs/cisco/CISCO-PFR-MIB.db
1676. CISCO-RADIUS-EXT-MIB data/mibs/cisco/CISCO-RADIUS-EXT-MIB.db
1677. CISCO-SUBSCRIBER-IDENTITY-TC-MIB data/mibs/cisco/CISCO-SUBSCRIBER-IDENTITY-TC-MIB.db
1678. CISCO-SUBSCRIBER-SESSION-TC-MIB data/mibs/cisco/CISCO-SUBSCRIBER-SESSION-TC-MIB.db
1679. CISCO-SUBSCRIBER-SESSION-MIB . data/mibs/cisco/CISCO-SUBSCRIBER-SESSION-MIB.db
1680. CISCO-SWITCH-CEF-MIB data/mibs/cisco/CISCO-SWITCH-CEF-MIB.db
1681. CISCO-SWITCH-FABRIC-MIB data/mibs/cisco/CISCO-SWITCH-FABRIC-MIB.db
1682. CISCO-VDC-MIB data/mibs/cisco/CISCO-VDC-MIB.db
1683. CISCO-VOICE-URI-CLASS-MIB data/mibs/cisco/CISCO-VOICE-URI-CLASS-MIB.db
1684. CISCO-VPC-MIB data/mibs/cisco/CISCO-VPC-MIB.db
1685. CISCO-VPN-LIC-USAGE-MONITOR-MIB data/mibs/cisco/CISCO-VPN-LIC-USAGE-MONITOR-MIB.db
1686. CISCO-WBX-MEETING-MIB data/mibs/cisco/CISCO-WBX-MEETING-MIB.db
1687. CISCO-PFC-EXT-MIB data/mibs/cisco/CISCO-PFC-EXT-MIB.db
1688. CISCO-WIRELESS-NOTIFICATION-MIB data/mibs/cisco/CISCO-WIRELESS-NOTIFICATION-MIB.db
1689. CISCO-IETF-VRRP-07-MIB data/mibs/cisco/CISCO-IETF-VRRP-07-MIB.db
1690. CISCO-MEDIA-METRICS-MIB data/mibs/cisco/CISCO-MEDIA-METRICS-MIB.db
1691. CISCO-TCP-METRICS-MIB data/mibs/cisco/CISCO-TCP-METRICS-MIB.db
1692. CISCO-DTI-EXT-MIB data/mibs/cisco/CISCO-DTI-EXT-MIB.db
1693. CISCO-WAN-CELL-EXT-MIB data/mibs/cisco/CISCO-WAN-CELL-EXT-MIB.db
1694. CISCO-QP-LBG-MIB data/mibs/cisco/CISCO-QP-LBG-MIB.db
1695. CISCO-IP-ADDRESS-POOL-TC-MIB . data/mibs/cisco/CISCO-IP-ADDRESS-POOL-TC-MIB.db
1696. CISCO-IP-ADDRESS-POOL-MIB data/mibs/cisco/CISCO-IP-ADDRESS-POOL-MIB.db
1697. CISCO-NETWORK-VIRTUALIZATION-OVERLAY-MIB data/mibs/cisco/CISCO-NETWORK-VIRTUALIZATION-OVERLAY-MIB.db
1698. CISCO-L2NAT-MIB data/mibs/cisco/CISCO-L2NAT-MIB.db
1699. CISCO-WPAN-MIB data/mibs/cisco/CISCO-WPAN-MIB.db
1700. CISCO-CABLE-LICENSE-MIB data/mibs/cisco/CISCO-CABLE-LICENSE-MIB.db
1701. CISCO-SECY-EXT-MIB data/mibs/cisco/CISCO-SECY-EXT-MIB.db
1702. CISCO-OPTICAL-MIB data/mibs/cisco/CISCO-OPTICAL-MIB.db
1703. CISCO-LWAPP-DOT11-CLIENT-TS-MIB data/mibs/cisco/CISCO-LWAPP-DOT11-CLIENT-TS-MIB.db
1704. CISCO-LWAPP-SI-MIB data/mibs/cisco/CISCO-LWAPP-SI-MIB.db
1705. CISCO-CABLE-IRON-BUS-STAT-MIB data/mibs/cisco/CISCO-CABLE-IRON-BUS-STAT-MIB.db
1706. CISCO-SMART-LIC-MIB data/mibs/cisco/CISCO-SMART-LIC-MIB.db
1707. CISCO-SSLVPN-MIB data/mibs/cisco/CISCO-SSLVPN-MIB.db
1708. CISCO-LPTS-MIB data/mibs/cisco/CISCO-LPTS-MIB.db
1709. CISCO-LISP-EXT-MIB data/mibs/cisco/CISCO-LISP-EXT-MIB.db
1710. CISCO-USP-MIB data/mibs/cisco/CISCO-USP-MIB.db
1711. CISCO-DEVICE-LOCATION-MIB data/mibs/cisco/CISCO-DEVICE-LOCATION-MIB.db
1712. CISCO-LWAPP-CLIENT-ROAMING-CAPABILITY data/mibs/cisco/CISCO-LWAPP-CLIENT-ROAMING-CAPABILITY.db
1713. CISCO-LWAPP-DOT11-CLIENT-CALIB-CAPABILITY data/mibs/cisco/CISCO-LWAPP-DOT11-CLIENT-CALIB-CAPABILITY.db
1714. CISCO-LWAPP-MFP-CAPABILITY ... data/mibs/cisco/CISCO-LWAPP-MFP-CAPABILITY.db
1715. CISCO-LWAPP-QOS-CAPABILITY ... data/mibs/cisco/CISCO-LWAPP-QOS-CAPABILITY.db
1716. CISCO-LWAPP-TSM-CAPABILITY ... data/mibs/cisco/CISCO-LWAPP-TSM-CAPABILITY.db
1717. CISCO-LWAPP-DHCP-MIB data/mibs/cisco/CISCO-LWAPP-DHCP-MIB.db
1718. CISCO-LWAPP-DOT11-CCX-CLIENT-MIB data/mibs/cisco/CISCO-LWAPP-DOT11-CCX-CLIENT-MIB.db
1719. CISCO-LWAPP-PMIP-MIB data/mibs/cisco/CISCO-LWAPP-PMIP-MIB.db
1720. CISCO-LWAPP-EXT-MIB data/mibs/cisco/CISCO-LWAPP-EXT-MIB.db
1721. CISCO-LWAPP-WAPI-MIB data/mibs/cisco/CISCO-LWAPP-WAPI-MIB.db
1722. GENERICOBJECT-MIB data/mibs/cisco/GENERICOBJECT-MIB.db
1723. MPOA-MIB data/mibs/cisco/MPOA-MIB.db
1724. NETRANGER data/mibs/cisco/NETRANGER.db
1725. CPQHOST-MIB data/mibs/compaq/CPQHOST-MIB.db
1726. CPQSINFO-MIB data/mibs/compaq/CPQSINFO-MIB.db
1727. CPQHLTH-MIB data/mibs/compaq/CPQHLTH-MIB.db
1728. CPQIDA-MIB data/mibs/compaq/CPQIDA-MIB.db
1729. CPQSCSI-MIB data/mibs/compaq/CPQSCSI-MIB.db
1730. -- CPQSRVMN-MIB data/mibs/compaq/-- CPQSRVMN-MIB.db
1731. CPQSTDEQ-MIB data/mibs/compaq/CPQSTDEQ-MIB.db
1732. CPQSTSYS-MIB data/mibs/compaq/CPQSTSYS-MIB.db
1733. CPQTHRSH-MIB data/mibs/compaq/CPQTHRSH-MIB.db
1734. CPQUPS-MIB data/mibs/compaq/CPQUPS-MIB.db
1735. CPQHVS-MIB data/mibs/compaq/CPQHVS-MIB.db
1736. COMPAQ-ID-REC-MIB data/mibs/compaq/COMPAQ-ID-REC-MIB.db
1737. COMPAQ-L2MGMT-MIB data/mibs/compaq/COMPAQ-L2MGMT-MIB.db
1738. DEFINITIONS data/mibs/compaq/DEFINITIONS.db
1739. DEFINITIONS data/mibs/compaq/DEFINITIONS.db
1740. CPQSANAPP-MIB data/mibs/compaq/CPQSANAPP-MIB.db
1741. CPQSANEVENT-MIB data/mibs/compaq/CPQSANEVENT-MIB.db
1742. CPQRACK-MIB data/mibs/compaq/CPQRACK-MIB.db
1743. CPQN54NN-MIB data/mibs/compaq/CPQN54NN-MIB.db
1744. COMPAQ-AGENT-MIB data/mibs/compaq/COMPAQ-AGENT-MIB.db
1745. CPQAPPLIANCE-MIB data/mibs/compaq/CPQAPPLIANCE-MIB.db
1746. CPQAPPG80-MIB data/mibs/compaq/CPQAPPG80-MIB.db
1747. CPQCLUSTER-MIB data/mibs/compaq/CPQCLUSTER-MIB.db
1748. CPQCMC-MIB data/mibs/compaq/CPQCMC-MIB.db
1749. CPQCR-MIB data/mibs/compaq/CPQCR-MIB.db
1750. CPQDCEO-MIB data/mibs/compaq/CPQDCEO-MIB.db
1751. CPQDMII-MIB data/mibs/compaq/CPQDMII-MIB.db
1752. CPQDSCCS-MIB data/mibs/compaq/CPQDSCCS-MIB.db
1753. CPQFCA-MIB data/mibs/compaq/CPQFCA-MIB.db
1754. -- CPQGEN-MIB data/mibs/compaq/-- CPQGEN-MIB.db
1755. -- CPQHOST2-MIB data/mibs/compaq/-- CPQHOST2-MIB.db

1756.	CPQICA-MIB	data/mibs/compaq/CPQICA-MIB.db
1757.	CPQIDE-MIB	data/mibs/compaq/CPQIDE-MIB.db
1758.	CPQIODRV-MIB	data/mibs/compaq/CPQIODRV-MIB.db
1759.	CPQISCSI-MIB	data/mibs/compaq/CPQISCSI-MIB.db
1760.	CPQLINOS-MIB	data/mibs/compaq/CPQLINOS-MIB.db
1761.	DEFINITIONS	data/mibs/compaq/DEFINITIONS.db
1762.	CPQNIC-MIB	data/mibs/compaq/CPQNIC-MIB.db
1763.	CPQNODE-MIB	data/mibs/compaq/CPQNODE-MIB.db
1764.	CPQOneView-MIB	data/mibs/compaq/CPQOneView-MIB.db
1765.	CPQPOWER-MIB	data/mibs/compaq/CPQPOWER-MIB.db
1766.	CPQRECOV-MIB	data/mibs/compaq/CPQRECOV-MIB.db
1767.	CPQSASSWITCH-MIB	data/mibs/compaq/CPQSASSWITCH-MIB.db
1768.	CPQSERVICE-MIB	data/mibs/compaq/CPQSERVICE-MIB.db
1769.	CPQSM2-MIB	data/mibs/compaq/CPQSM2-MIB.db
1770.	CPQEXTERNAL-MIB	data/mibs/compaq/CPQEXTERNAL-MIB.db
1771.	CPQSWCC-MIB	data/mibs/compaq/CPQSWCC-MIB.db
1772.	CPQ-TRAPS-MIB	data/mibs/compaq/CPQ-TRAPS-MIB.db
1773.	CPQWCRM-MIB	data/mibs/compaq/CPQWCRM-MIB.db
1774.	CPQOS-MIB	data/mibs/compaq/CPQOS-MIB.db
1775.	DECHUB90-MIB	data/mibs/dec/DECHUB90-MIB.db
1776.	DEC-MDS-MIB	data/mibs/dec/DEC-MDS-MIB.db
1777.	DECserver-Accounting-MIB	data/mibs/dec/DECserver-Accounting-MIB.db
1778.	ATM-MIB	data/mibs/fore/ATM-MIB.db
1779.	Fore-Common-MIB -- --	data/mibs/fore/Fore-Common-MIB -- --.db
1780.	Fore-If-MIB	data/mibs/fore/Fore-If-MIB.db
1781.	Fore-TrapLog-MIB	data/mibs/fore/Fore-TrapLog-MIB.db
1782.	Fore-Adapter-MIB	data/mibs/fore/Fore-Adapter-MIB.db
1783.	Fore-frf8-MIB	data/mibs/fore/Fore-frf8-MIB.db
1784.	Fore-Errors-MIB	data/mibs/fore/Fore-Errors-MIB.db
1785.	Fore-Callrecord-MIB	data/mibs/fore/Fore-Callrecord-MIB.db
1786.	Fore-Switch-MIB	data/mibs/fore/Fore-Switch-MIB.db
1787.	Fore-Assembly-MIB	data/mibs/fore/Fore-Assembly-MIB.db
1788.	Fore-pre802dot1Q-VLAN-MIB	data/mibs/fore/Fore-pre802dot1Q-VLAN-MIB.db
1789.	Fore-Bridge-Extensions-MIB	data/mibs/fore/Fore-Bridge-Extensions-MIB.db
1790.	Fore-Lane-MIB	data/mibs/fore/Fore-Lane-MIB.db
1791.	Fore-FileXfr-MIB	data/mibs/fore/Fore-FileXfr-MIB.db
1792.	Fore-DS1-MIB	data/mibs/fore/Fore-DS1-MIB.db
1793.	Fore-DS3-MIB	data/mibs/fore/Fore-DS3-MIB.db
1794.	Fore-DSX1-MIB	data/mibs/fore/Fore-DSX1-MIB.db
1795.	Fore-SONET-MIB	data/mibs/fore/Fore-SONET-MIB.db
1796.	Fore-E1-MIB	data/mibs/fore/Fore-E1-MIB.db
1797.	Fore-E3-MIB	data/mibs/fore/Fore-E3-MIB.db
1798.	Fore-J2-MIB	data/mibs/fore/Fore-J2-MIB.db
1799.	Fore-tp25-MIB	data/mibs/fore/Fore-tp25-MIB.db
1800.	Fore-Framnetmod-MIB	data/mibs/fore/Fore-Framnetmod-MIB.db
1801.	Fore-aal5-MIB	data/mibs/fore/Fore-aal5-MIB.db
1802.	Fore-frs-MIB	data/mibs/fore/Fore-frs-MIB.db
1803.	Fore-Profile-MIB	data/mibs/fore/Fore-Profile-MIB.db
1804.	ATM-FORUM-MIB -- -- --	data/mibs/fore/ATM-FORUM-MIB -- -- --.db
1805.	Fore-IlmiSnmpProxy-MIB	data/mibs/fore/Fore-IlmiSnmpProxy-MIB.db
1806.	Fore-IlmiRegistry-MIB	data/mibs/fore/Fore-IlmiRegistry-MIB.db
1807.	Fore-ES-System-MIB	data/mibs/fore/Fore-ES-System-MIB.db
1808.	Fore-RMON-Extensions-MIB	data/mibs/fore/Fore-RMON-Extensions-MIB.db
1809.	SEC10	data/mibs/fore/SEC10.db
1810.	SEC100	data/mibs/fore/SEC100.db
1811.	Fore-TCM-MIB	data/mibs/fore/Fore-TCM-MIB.db
1812.	ATMF-CES-MIB	data/mibs/fore/ATMF-CES-MIB.db
1813.	PNNI-MIB	data/mibs/fore/PNNI-MIB.db
1814.	AAC3-MIB	data/mibs/fore/AAC3-MIB.db
1815.	CELLPATH90-MIB	data/mibs/fore/CELLPATH90-MIB.db
1816.	Fore-Funi-MIB	data/mibs/fore/Fore-Funi-MIB.db
1817.	Fore-PNNI-MIB	data/mibs/fore/Fore-PNNI-MIB.db
1818.	POWERHUB-CORE-MIB	data/mibs/fore/POWERHUB-CORE-MIB.db
1819.	POWERHUB-ATM-MIB	data/mibs/fore/POWERHUB-ATM-MIB.db
1820.	POWERHUB-TCPIP-MIB	data/mibs/fore/POWERHUB-TCPIP-MIB.db
1821.	POWERHUB-VLAN-MIB	data/mibs/fore/POWERHUB-VLAN-MIB.db
1822.	HP-ICF-OID	data/mibs/hp/HP-ICF-OID.db
1823.	NETSWITCH-MIB	data/mibs/hp/NETSWITCH-MIB.db
1824.	PROBE-MIB	data/mibs/hp/PROBE-MIB.db
1825.	HP700RX-MIB	data/mibs/hp/HP700RX-MIB.db
1826.	HP-MPE-XL	data/mibs/hp/HP-MPE-XL.db
1827.	JETDIRECT3-MIB	data/mibs/hp/JETDIRECT3-MIB.db
1828.	HPSWITCH-MIB	data/mibs/hp/HPSWITCH-MIB.db
1829.	HP-UNIX	data/mibs/hp/HP-UNIX.db
1830.	HPNSAEISA-MIB	data/mibs/hp/HPNSAEISA-MIB.db
1831.	HPNSAENV-MIB	data/mibs/hp/HPNSAENV-MIB.db
1832.	HPNSAEVENT-MIB	data/mibs/hp/HPNSAEVENT-MIB.db
1833.	HPNSAHOTSWAPSUBSYSTEM-MIB	data/mibs/hp/HPNSAHOTSWAPSUBSYSTEM-MIB.db
1834.	HPNSAPCI-MIB	data/mibs/hp/HPNSAPCI-MIB.db
1835.	HPNSARPS-MIB	data/mibs/hp/HPNSARPS-MIB.db
1836.	HPNSASCSI-MIB	data/mibs/hp/HPNSASCSI-MIB.db

1837.	HPNSATRAP-MIB	data/mibs/hp/HPNSATRAP-MIB.db
1838.	HPNSATRAPCFG-MIB	data/mibs/hp/HPNSATRAPCFG-MIB.db
1839.	HPNSASTORAGECAP-MIB	data/mibs/hp/HPNSASTORAGECAP-MIB.db
1840.	HPNSAECC-MIB	data/mibs/hp/HPNSAECC-MIB.db
1841.	HP-OV-NETMON	data/mibs/hp/HP-OV-NETMON.db
1842.	HP-OV-TOPO-DB	data/mibs/hp/HP-OV-TOPO-DB.db
1843.	HPNSAREMOTEAASSIST-MIB	data/mibs/hp/HPNSAREMOTEAASSIST-MIB.db
1844.	HP-ROUTER-MIB	data/mibs/hp/HP-ROUTER-MIB.db
1845.	HP-SN-ROOT-MIB	data/mibs/hp/HP-SN-ROOT-MIB.db
1846.	HP-SN-AGENT-MIB	data/mibs/hp/HP-SN-AGENT-MIB.db
1847.	HP-SN-IP-MIB	data/mibs/hp/HP-SN-IP-MIB.db
1848.	HP-SN-APPLETALK-MIB	data/mibs/hp/HP-SN-APPLETALK-MIB.db
1849.	HP-SN-BGP4-GROUP-MIB	data/mibs/hp/HP-SN-BGP4-GROUP-MIB.db
1850.	HP-SN-IGMP-MIB	data/mibs/hp/HP-SN-IGMP-MIB.db
1851.	HP-SN-IPX-MIB	data/mibs/hp/HP-SN-IPX-MIB.db
1852.	HP-SN-OSPF-GROUP-MIB	data/mibs/hp/HP-SN-OSPF-GROUP-MIB.db
1853.	HP-SN-POS-GROUP-MIB	data/mibs/hp/HP-SN-POS-GROUP-MIB.db
1854.	HP-SN-SW-L4-SWITCH-GROUP-MIB	data/mibs/hp/HP-SN-SW-L4-SWITCH-GROUP-MIB.db
1855.	HP-SN-SWITCH-GROUP-MIB	data/mibs/hp/HP-SN-SWITCH-GROUP-MIB.db
1856.	HP-ENTITY-MIB	data/mibs/hp/HP-ENTITY-MIB.db
1857.	SEMI-MIB	data/mibs/hp/SEMI-MIB.db
1858.	NETSWITCH-DMA-MIB	data/mibs/hp/NETSWITCH-DMA-MIB.db
1859.	NETSWITCH-DRIVERS-MIB	data/mibs/hp/NETSWITCH-DRIVERS-MIB.db
1860.	HP-SwitchStack-MIB	data/mibs/hp/HP-SwitchStack-MIB.db
1861.	HP-VLAN	data/mibs/hp/HP-VLAN.db
1862.	HP-ICF-TC	data/mibs/hp/HP-ICF-TC.db
1863.	STATISTICS-MIB	data/mibs/hp/STATISTICS-MIB.db
1864.	HP-ICF-8023-RPTR	data/mibs/hp/HP-ICF-8023-RPTR.db
1865.	HP-ICF-BASIC	data/mibs/hp/HP-ICF-BASIC.db
1866.	HP-ICF-BRIDGE	data/mibs/hp/HP-ICF-BRIDGE.db
1867.	HP-ICF-CHAIN	data/mibs/hp/HP-ICF-CHAIN.db
1868.	HP-ICF-CHASSIS	data/mibs/hp/HP-ICF-CHASSIS.db
1869.	HP-ICF-DOWNLOAD	data/mibs/hp/HP-ICF-DOWNLOAD.db
1870.	HP-ICF-GENERIC-RPTR	data/mibs/hp/HP-ICF-GENERIC-RPTR.db
1871.	ICF-ETWIST	data/mibs/hp/ICF-ETWIST.db
1872.	HP-ICF-FAULT-FINDER-MIB	data/mibs/hp/HP-ICF-FAULT-FINDER-MIB.db
1873.	HP-ICF-LINKTEST	data/mibs/hp/HP-ICF-LINKTEST.db
1874.	HP-ICF-SECURITY	data/mibs/hp/HP-ICF-SECURITY.db
1875.	HP-SNTPClientConfiguration-MIB	data/mibs/hp/HP-SNTPClientConfiguration-MIB.db
1876.	HP-ICF-STACK	data/mibs/hp/HP-ICF-STACK.db
1877.	ICF-VG-RPTR	data/mibs/hp/ICF-VG-RPTR.db
1878.	HP-ICF-VG-RPTR	data/mibs/hp/HP-ICF-VG-RPTR.db
1879.	CONFIG-MIB	data/mibs/hp/CONFIG-MIB.db
1880.	IPX	data/mibs/hp/IPX.db
1881.	HP-ICF-WSM-SMI	data/mibs/hp/HP-ICF-WSM-SMI.db
1882.	HP-ICF-WSM-CC-SMI	data/mibs/hp/HP-ICF-WSM-CC-SMI.db
1883.	HP-ICF-WSM-TYPE-MIB	data/mibs/hp/HP-ICF-WSM-TYPE-MIB.db
1884.	HP-ICF-WSM-CC-RF-MIB	data/mibs/hp/HP-ICF-WSM-CC-RF-MIB.db
1885.	HP-ICF-WSM-CC-STATS-MIB	data/mibs/hp/HP-ICF-WSM-CC-STATS-MIB.db
1886.	HP-ICF-WSM-INFRA-SMI-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-SMI-MIB.db
1887.	HP-ICF-WSM-INFRA-AUTO-UPDATE-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-AUTO-UPDATE-MIB.db
1888.	HP-ICF-WSM-INFRA-DIAG-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-DIAG-MIB.db
1889.	HP-ICF-WSM-INFRA-FILE-MGMT-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-FILE-MGMT-MIB.db
1890.	HP-ICF-WSM-INFRA-LIC-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-LIC-MIB.db
1891.	HP-ICF-WSM-INFRA-MODULE-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-MODULE-MIB.db
1892.	HP-ICF-WSM-INFRA-NTP-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-NTP-MIB.db
1893.	HP-ICF-WSM-INFRA-PM-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-PM-MIB.db
1894.	HP-ICF-WSM-INFRA-REDUNDANCY-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-REDUNDANCY-MIB.db
1895.	HP-ICF-WSM-INFRA-SYMSMG-MIB	data/mibs/hp/HP-ICF-WSM-INFRA-SYMSMG-MIB.db
1896.	HP-ICF-WSM-LICENSE-MIB	data/mibs/hp/HP-ICF-WSM-LICENSE-MIB.db
1897.	HP-ICF-WSM-MGMT-MIB	data/mibs/hp/HP-ICF-WSM-MGMT-MIB.db
1898.	HP-ICF-WSM-SW-SMI	data/mibs/hp/HP-ICF-WSM-SW-SMI.db
1899.	HP-ICF-WSM-SW-ACL-MIB	data/mibs/hp/HP-ICF-WSM-SW-ACL-MIB.db
1900.	HP-ICF-WSM-SW-ARP-MIB	data/mibs/hp/HP-ICF-WSM-SW-ARP-MIB.db
1901.	HP-ICF-WSM-SW-AUTHENTICATEDNTP-MIB	data/mibs/hp/HP-ICF-WSM-SW-AUTHENTICATEDNTP-MIB.db
1902.	HP-ICF-WSM-SW-DHCP-MIB	data/mibs/hp/HP-ICF-WSM-SW-DHCP-MIB.db
1903.	HP-ICF-WSM-SW-DHCPSEVER-MIB	data/mibs/hp/HP-ICF-WSM-SW-DHCPSEVER-MIB.db
1904.	HP-ICF-WSM-SW-DNS-MIB	data/mibs/hp/HP-ICF-WSM-SW-DNS-MIB.db
1905.	HP-ICF-WSM-SW-IF-MIB	data/mibs/hp/HP-ICF-WSM-SW-IF-MIB.db
1906.	HP-ICF-WSM-SW-GRE-MIB	data/mibs/hp/HP-ICF-WSM-SW-GRE-MIB.db
1907.	HP-ICF-WSM-SW-GUEST-USER-RAD-MIB	data/mibs/hp/HP-ICF-WSM-SW-GUEST-USER-RAD-MIB.db
1908.	HP-ICF-WSM-SW-IP-MIB	data/mibs/hp/HP-ICF-WSM-SW-IP-MIB.db
1909.	HP-ICF-WSM-SW-MOBILITY-MIB	data/mibs/hp/HP-ICF-WSM-SW-MOBILITY-MIB.db
1910.	HP-ICF-WSM-SW-MSTP-MIB	data/mibs/hp/HP-ICF-WSM-SW-MSTP-MIB.db
1911.	HP-ICF-WSM-SW-PORTVLAN-MIB	data/mibs/hp/HP-ICF-WSM-SW-PORTVLAN-MIB.db
1912.	HP-ICF-WSM-SW-RADIUS-MIB	data/mibs/hp/HP-ICF-WSM-SW-RADIUS-MIB.db
1913.	HP-ICF-WSM-SW-SNMP-SAVE-MIB	data/mibs/hp/HP-ICF-WSM-SW-SNMP-SAVE-MIB.db
1914.	HP-ICF-WSM-TRAP-MIB	data/mibs/hp/HP-ICF-WSM-TRAP-MIB.db
1915.	HP-ICF-WSM-USER-MGMT-MIB	data/mibs/hp/HP-ICF-WSM-USER-MGMT-MIB.db
1916.	HP-PROCURVE-WLAN-SMI	data/mibs/hp/HP-PROCURVE-WLAN-SMI.db
1917.	HP-WLAN-SFLOW-EXTENSIONS-MIB	data/mibs/hp/HP-WLAN-SFLOW-EXTENSIONS-MIB.db

1918.	HP-BASE-MIB	data/mibs/hp/HP-BASE-MIB.db
1919.	HP-IP-EXT-MIB	data/mibs/hp/HP-IP-EXT-MIB.db
1920.	HP-MEMPROC-MIB	data/mibs/hp/HP-MEMPROC-MIB.db
1921.	HP-SYSTEM-MIB	data/mibs/hp/HP-SYSTEM-MIB.db
1922.	HP-ICF-ARP-PROTECT	data/mibs/hp/HP-ICF-ARP-PROTECT.db
1923.	HP-AUTZ-MIB	data/mibs/hp/HP-AUTZ-MIB.db
1924.	HP-ICF-CONNECTION-RATE-FILTER	data/mibs/hp/HP-ICF-CONNECTION-RATE-FILTER.db
1925.	HpicHighAvailability-MIB	data/mibs/hp/HpicHighAvailability-MIB.db
1926.	HP-ICF-INST-MON	data/mibs/hp/HP-ICF-INST-MON.db
1927.	HP-ICF-IPCONFIG	data/mibs/hp/HP-ICF-IPCONFIG.db
1928.	HP-ICF-IP-ROUTING	data/mibs/hp/HP-ICF-IP-ROUTING.db
1929.	HP-ICF-JUMBO-MIB	data/mibs/hp/HP-ICF-JUMBO-MIB.db
1930.	HP-ICF-L3MAC-MIB	data/mibs/hp/HP-ICF-L3MAC-MIB.db
1931.	HP-ICF-OSPF	data/mibs/hp/HP-ICF-OSPF.db
1932.	HP-ICF-PIM	data/mibs/hp/HP-ICF-PIM.db
1933.	HP-ICF-RATE-LIMIT-MIB	data/mibs/hp/HP-ICF-RATE-LIMIT-MIB.db
1934.	HP-ICF-RIP	data/mibs/hp/HP-ICF-RIP.db
1935.	HP-ICF-SNMP-MIB	data/mibs/hp/HP-ICF-SNMP-MIB.db
1936.	HP-ICF-UDLD-MIB	data/mibs/hp/HP-ICF-UDLD-MIB.db
1937.	HP-ICF-UDP-FORWARD	data/mibs/hp/HP-ICF-UDP-FORWARD.db
1938.	HP-USER-AUTH	data/mibs/hp/HP-USER-AUTH.db
1939.	HP-ICF-VRRP-MIB	data/mibs/hp/HP-ICF-VRRP-MIB.db
1940.	HP-ICF-XRRP	data/mibs/hp/HP-ICF-XRRP.db
1941.	HP-PROCURVE-WLAN-TC	data/mibs/hp/HP-PROCURVE-WLAN-TC.db
1942.	HP-PROCURVE-WLAN-AP-MIB	data/mibs/hp/HP-PROCURVE-WLAN-AP-MIB.db
1943.	HP-PROCURVE-WLAN-AR-MIB	data/mibs/hp/HP-PROCURVE-WLAN-AR-MIB.db
1944.	HP-PROCURVE-WLAN-NOTIFY-MIB	data/mibs/hp/HP-PROCURVE-WLAN-NOTIFY-MIB.db
1945.	HP-PROCURVE-WLAN-SYSTEM-MIB	data/mibs/hp/HP-PROCURVE-WLAN-SYSTEM-MIB.db
1946.	IPV6-MLD-MIB	data/mibs/hp/IPV6-MLD-MIB.db
1947.	DEFINITIONS	data/mibs/hp/DEFINITIONS.db
1948.	HP-AUTH-MIB	data/mibs/hp/HP-AUTH-MIB.db
1949.	HP-ICF-AUTORUN	data/mibs/hp/HP-ICF-AUTORUN.db
1950.	FAN-MIB	data/mibs/hp/FAN-MIB.db
1951.	POWERSUPPLY-MIB	data/mibs/hp/POWERSUPPLY-MIB.db
1952.	HP-ICF-DHCP-SNOOP-MIB	data/mibs/hp/HP-ICF-DHCP-SNOOP-MIB.db
1953.	HP-ICF-DHCPv6-RELAY	data/mibs/hp/HP-ICF-DHCPv6-RELAY.db
1954.	HP-ICF-DOS-FILTER-MIB	data/mibs/hp/HP-ICF-DOS-FILTER-MIB.db
1955.	IEEE8021-PAE-MIB	data/mibs/hp/IEEE8021-PAE-MIB.db
1956.	HP-DOT1X-EXTENSIONS-MIB	data/mibs/hp/HP-DOT1X-EXTENSIONS-MIB.db
1957.	HP-ICF-FTRCO	data/mibs/hp/HP-ICF-FTRCO.db
1958.	HP-ICF-GPPC-MIB	data/mibs/hp/HP-ICF-GPPC-MIB.db
1959.	HP-ICF-IP-LOCKDOWN-MIB	data/mibs/hp/HP-ICF-IP-LOCKDOWN-MIB.db
1960.	HP-ICF-MLD-MIB	data/mibs/hp/HP-ICF-MLD-MIB.db
1961.	HP-ICF-OOBM-MIB	data/mibs/hp/HP-ICF-OOBM-MIB.db
1962.	HP-ICF-POE-MIB	data/mibs/hp/HP-ICF-POE-MIB.db
1963.	HP-ICF-PROVIDER-BRIDGE	data/mibs/hp/HP-ICF-PROVIDER-BRIDGE.db
1964.	SAVEPOWER-MIB	data/mibs/hp/SAVEPOWER-MIB.db
1965.	HP-ICF-SYSLOG-MIB	data/mibs/hp/HP-ICF-SYSLOG-MIB.db
1966.	HP-ICF-USBPORT	data/mibs/hp/HP-ICF-USBPORT.db
1967.	HP-ICF-USER-PROFILE-MIB	data/mibs/hp/HP-ICF-USER-PROFILE-MIB.db
1968.	HP-ICF-WAN-SNMP	data/mibs/hp/HP-ICF-WAN-SNMP.db
1969.	HP-ICF-WAN-UNIT	data/mibs/hp/HP-ICF-WAN-UNIT.db
1970.	DT-MIB	data/mibs/hp/DT-MIB.db
1971.	HP-SWITCH-PL-MIB	data/mibs/hp/HP-SWITCH-PL-MIB.db
1972.	ATM-FORUM-TC	data/mibs/ibm/ATM-FORUM-TC.db
1973.	ATM-FORUM-TC-MIB	data/mibs/ibm/ATM-FORUM-TC-MIB.db
1974.	PNNI-MIB	data/mibs/ibm/PNNI-MIB.db
1975.	ATM-SOFT-PVC-MIB	data/mibs/ibm/ATM-SOFT-PVC-MIB.db
1976.	IBM-MIB	data/mibs/ibm/IBM-MIB.db
1977.	IBM2210-V1R2-PART1-MIB	data/mibs/ibm/IBM2210-V1R2-PART1-MIB.db
1978.	IBM2210-V1R2-PART2-MIB	data/mibs/ibm/IBM2210-V1R2-PART2-MIB.db
1979.	MSSCOMMON-MIB	data/mibs/ibm/MSSCOMMON-MIB.db
1980.	PROTEON-MIB	data/mibs/ibm/PROTEON-MIB.db
1981.	IBM-8271-EtherStreamer-Switch-MIB	data/mibs/ibm/IBM-8271-EtherStreamer-Switch-MIB.db
1982.	LAN-EMULATION-CLIENT-MIB	data/mibs/ibm/LAN-EMULATION-CLIENT-MIB.db
1983.	ELAN-MIB	data/mibs/ibm/ELAN-MIB.db
1984.	LES-MIB	data/mibs/ibm/LES-MIB.db
1985.	BUS-MIB	data/mibs/ibm/BUS-MIB.db
1986.	LAN-EMULATION-BUS-MIB	data/mibs/ibm/LAN-EMULATION-BUS-MIB.db
1987.	LAN-EMULATION-ELAN-MIB	data/mibs/ibm/LAN-EMULATION-ELAN-MIB.db
1988.	LAN-EMULATION-LES-MIB	data/mibs/ibm/LAN-EMULATION-LES-MIB.db
1989.	MSSSERVER8210-MIB	data/mibs/ibm/MSSSERVER8210-MIB.db
1990.	MSSSERVER8260-MIB	data/mibs/ibm/MSSSERVER8260-MIB.db
1991.	NWAYSMS-MIB	data/mibs/ibm/NWAYSMS-MIB.db
1992.	IBM-LAN-EMULATION-EXTENSION-MIB	data/mibs/ibm/IBM-LAN-EMULATION-EXTENSION-MIB.db
1993.	IBM-BCM-MIB	data/mibs/ibm/IBM-BCM-MIB.db
1994.	IBM-ELAN-MIB	data/mibs/ibm/IBM-ELAN-MIB.db
1995.	IBM-LES-BUS-MIB	data/mibs/ibm/IBM-LES-BUS-MIB.db
1996.	IBM-LES-LECS-MIB	data/mibs/ibm/IBM-LES-LECS-MIB.db
1997.	NV6000-MIB	data/mibs/ibm/NV6000-MIB.db
1998.	MIDLEVELMGR-MIB	data/mibs/ibm/MIDLEVELMGR-MIB.db

1999.	SYSINFO-MIB	data/mibs/ibm/SYSINFO-MIB.db
2000.	SYSMON6K-MIB	data/mibs/ibm/SYSMON6K-MIB.db
2001.	IBM3172-MIB	data/mibs/ibm/IBM3172-MIB.db
2002.	IBM-3174-PRIVATE-MIB	data/mibs/ibm/IBM-3174-PRIVATE-MIB.db
2003.	IBM-8224-MIB	data/mibs/ibm/IBM-8224-MIB.db
2004.	SRTB-MIB	data/mibs/ibm/SRTB-MIB.db
2005.	CAU-MIB	data/mibs/ibm/CAU-MIB.db
2006.	BRIDGE-IBMSRTB-MIB	data/mibs/ibm/BRIDGE-IBMSRTB-MIB.db
2007.	ATM-SWITCHING-NODE-MIB	data/mibs/ibm/ATM-SWITCHING-NODE-MIB.db
2008.	IBM6611-MIB	data/mibs/ibm/IBM6611-MIB.db
2009.	IBM2216-MIB	data/mibs/ibm/IBM2216-MIB.db
2010.	IBMIROC-MIB	data/mibs/ibm/IBMIROC-MIB.db
2011.	IBMIROCLAN-MIB	data/mibs/ibm/IBMIROCLAN-MIB.db
2012.	Intel-Common-MIB	data/mibs/intel/Intel-Common-MIB.db
2013.	INTEL-GEN-MIB	data/mibs/intel/INTEL-GEN-MIB.db
2014.	INTEL-EXPRESS110-MIB	data/mibs/intel/INTEL-EXPRESS110-MIB.db
2015.	INTEL-ES400-MIB	data/mibs/intel/INTEL-ES400-MIB.db
2016.	INTEL-SYS-MIB	data/mibs/intel/INTEL-SYS-MIB.db
2017.	INTEL-S500-MIB	data/mibs/intel/INTEL-S500-MIB.db
2018.	INTEL-VLAN-MIB	data/mibs/intel/INTEL-VLAN-MIB.db
2019.	INTEL-L3LINK-MIB	data/mibs/intel/INTEL-L3LINK-MIB.db
2020.	INTEL-IPX-MIB	data/mibs/intel/INTEL-IPX-MIB.db
2021.	INTEL-IPROUTER-MIB	data/mibs/intel/INTEL-IPROUTER-MIB.db
2022.	INTEL-DIAG-MIB	data/mibs/intel/INTEL-DIAG-MIB.db
2023.	INTEL-FREXT-MIB	data/mibs/intel/INTEL-FREXT-MIB.db
2024.	INT-HDLC-MIB	data/mibs/intel/INT-HDLC-MIB.db
2025.	IPMROUTE-MIB	data/mibs/intel/IPMROUTE-MIB.db
2026.	NOVELL-IPX-MIB	data/mibs/intel/NOVELL-IPX-MIB.db
2027.	INTEL-ISDN-MIB	data/mibs/intel/INTEL-ISDN-MIB.db
2028.	INTEL-LAPB-MIB	data/mibs/intel/INTEL-LAPB-MIB.db
2029.	INTEL-MPLINK-MIB	data/mibs/intel/INTEL-MPLINK-MIB.db
2030.	INTEL-PPP-MIB	data/mibs/intel/INTEL-PPP-MIB.db
2031.	INTEL-RMODEXT-MIB	data/mibs/intel/INTEL-RMODEXT-MIB.db
2032.	NOVELL-RIPSAP-MIB	data/mibs/intel/NOVELL-RIPSAP-MIB.db
2033.	INTEL-RSVP-MIB	data/mibs/intel/INTEL-RSVP-MIB.db
2034.	INTEL-TUNNEL-MIB	data/mibs/intel/INTEL-TUNNEL-MIB.db
2035.	INTEL-WBR-MIB	data/mibs/intel/INTEL-WBR-MIB.db
2036.	INTEL-X25EXT-MIB	data/mibs/intel/INTEL-X25EXT-MIB.db
2037.	INTEL-IPF-MIB	data/mibs/intel/INTEL-IPF-MIB.db
2038.	INTEL-IGMP-PRUNING-MIB	data/mibs/intel/INTEL-IGMP-PRUNING-MIB.db
2039.	SCANET-MIB	data/mibs/intel/SCANET-MIB.db
2040.	SCA-BOX-MIB	data/mibs/intel/SCA-BOX-MIB.db
2041.	ETH-SWITCH-MIB	data/mibs/intel/ETH-SWITCH-MIB.db
2042.	SCA-IPROUTER-MIB	data/mibs/intel/SCA-IPROUTER-MIB.db
2043.	ISD2-MIB	data/mibs/intel/ISD2-MIB.db
2044.	SCA-MPLINK2-MIB	data/mibs/intel/SCA-MPLINK2-MIB.db
2045.	SCA-MPLINK-MIB	data/mibs/intel/SCA-MPLINK-MIB.db
2046.	SCA-TUNNEL-MIB	data/mibs/intel/SCA-TUNNEL-MIB.db
2047.	NBASE-G1-MIB	data/mibs/intel/NBASE-G1-MIB.db
2048.	EXPRESSGS-MIB	data/mibs/intel/EXPRESSGS-MIB.db
2049.	AAC3-MIB	data/mibs/kentrox/AAC3-MIB.db
2050.	FIBERMUX-BRIDGE-MIB	data/mibs/kentrox/FIBERMUX-BRIDGE-MIB.db
2051.	ATMF-CES-MIB	data/mibs/kentrox/ATMF-CES-MIB.db
2052.	DATASmart-MIB	data/mibs/kentrox/DATASmart-MIB.db
2053.	DSMAX-MIB	data/mibs/kentrox/DSMAX-MIB.db
2054.	ELMAX-MIB	data/mibs/kentrox/ELMAX-MIB.db
2055.	FIBERMUX-MAGNUM3-MIB	data/mibs/kentrox/FIBERMUX-MAGNUM3-MIB.db
2056.	PACSEtTER-COMMON-MIB	data/mibs/kentrox/PACSEtTER-COMMON-MIB.db
2057.	KENTROX-MIB	data/mibs/kentrox/KENTROX-MIB.db
2058.	AIIDEFS-MIB	data/mibs/kentrox/AIIDEFS-MIB.db
2059.	AISYSTEM-MIB	data/mibs/kentrox/AISYSTEM-MIB.db
2060.	AIOPENVPN-MIB	data/mibs/kentrox/AIOPENVPN-MIB.db
2061.	ACMIB	data/mibs/lucent/ACMIB.db
2062.	LDR200MIB	data/mibs/lucent/LDR200MIB.db
2063.	GEN-MIB	data/mibs/lucent/GEN-MIB.db
2064.	ATMEDGE-MIB	data/mibs/lucent/ATMEDGE-MIB.db
2065.	APPLIC-MIB	data/mibs/lucent/APPLIC-MIB.db
2066.	ATMSWCH-MIB	data/mibs/lucent/ATMSWCH-MIB.db
2067.	CROUTE-MIB	data/mibs/lucent/CROUTE-MIB.db
2068.	CONFIG-MIB	data/mibs/lucent/CONFIG-MIB.db
2069.	MODULES-MIB	data/mibs/lucent/MODULES-MIB.db
2070.	MERITAGE-MIB	data/mibs/lucent/MERITAGE-MIB.db
2071.	PROPLANE-MIB	data/mibs/lucent/PROPLANE-MIB.db
2072.	PROMINET-MIB	data/mibs/lucent/PROMINET-MIB.db
2073.	SMARTD-MIB	data/mibs/lucent/SMARTD-MIB.db
2074.	RMON2-MIB	data/mibs/lucent/RMON2-MIB.db
2075.	SMON2-MIB	data/mibs/lucent/SMON2-MIB.db
2076.	XSWITCH-MIB	data/mibs/lucent/XSWITCH-MIB.db
2077.	CAJUN-POLICY-CAPABILITIES	data/mibs/lucent/CAJUN-POLICY-CAPABILITIES.db
2078.	WaveLAN-MIB	data/mibs/lucent/WaveLAN-MIB.db
2079.	WaveLAN-Roaming-MIB	data/mibs/lucent/WaveLAN-Roaming-MIB.db

2080. LanMgr-Mib-II-MIB data/mibs/microsoft/LanMgr-Mib-II-MIB.db
2081. WINS-MIB data/mibs/microsoft/WINS-MIB.db
2082. DHCP-MIB data/mibs/microsoft/DHCP-MIB.db
2083. MIMIC-UNKNOWN-MICROSOFT-TRAP-MIB data/mibs/microsoft/MIMIC-UNKNOWN-MICROSOFT-TRAP-MIB.db
2084. NORTEL-MIB data/mibs/nortel/NORTEL-MIB.db
2085. NORTEL-GENERIC-MIB data/mibs/nortel/NORTEL-GENERIC-MIB.db
2086. NORTEL-NMI-CONFORMANCE-MIB data/mibs/nortel/NORTEL-NMI-CONFORMANCE-MIB.db
2087. NORTEL-NMI-APP-COMPLIANCE-INDICATION-MIB data/mibs/nortel/NORTEL-NMI-APP-COMPLIANCE-INDICATION-MIB.db
2088. NORTEL-NMI-GROUPS-MIB data/mibs/nortel/NORTEL-NMI-GROUPS-MIB.db
2089. NORTEL-NMI-SYSTEM-MIB data/mibs/nortel/NORTEL-NMI-SYSTEM-MIB.db
2090. NORTEL-NMI-TC-MIB data/mibs/nortel/NORTEL-NMI-TC-MIB.db
2091. NORTEL-NMI-CONFIG-MIB data/mibs/nortel/NORTEL-NMI-CONFIG-MIB.db
2092. NORTEL-NMI-RESOURCE-MGMT-MIB data/mibs/nortel/NORTEL-NMI-RESOURCE-MGMT-MIB.db
2093. NORTEL-NMI-NE-INVENTORY-MIB data/mibs/nortel/NORTEL-NMI-NE-INVENTORY-MIB.db
2094. NORTEL-NMI-FAULT-MGMT-MIB data/mibs/nortel/NORTEL-NMI-FAULT-MGMT-MIB.db
2095. NORTEL-NMI-ALARM-SURV-MIB data/mibs/nortel/NORTEL-NMI-ALARM-SURV-MIB.db
2096. NORTEL-NMI-OSI-STATE-MIB data/mibs/nortel/NORTEL-NMI-OSI-STATE-MIB.db
2097. NORTEL-NMI-NOTIFICATION-LOG-MIB data/mibs/nortel/NORTEL-NMI-NOTIFICATION-LOG-MIB.db
2098. NORTEL-NMI-NOTIFICATIONS-MIB data/mibs/nortel/NORTEL-NMI-NOTIFICATIONS-MIB.db
2099. NORTEL-NMI-CONFIG-NOTI-MIB data/mibs/nortel/NORTEL-NMI-CONFIG-NOTI-MIB.db
2100. NORTEL-NMI-FAULT-NOTI-MIB data/mibs/nortel/NORTEL-NMI-FAULT-NOTI-MIB.db
2101. NORTEL-CSMOA-APP-COMPLIANCE-INDICATION-MIB data/mibs/nortel/NORTEL-CSMOA-APP-COMPLIANCE-INDICATION-MIB.db
2102. NORTEL-NMI-APP-REQUIREMENTS-MIB data/mibs/nortel/NORTEL-NMI-APP-REQUIREMENTS-MIB.db
2103. NORTEL-CSMOA-APP-REQUIREMENTS-MIB data/mibs/nortel/NORTEL-CSMOA-APP-REQUIREMENTS-MIB.db
2104. IPT data/mibs/nortel/IPT.db
2105. TLS data/mibs/nortel/TLS.db
2106. MICOM-OSCAR-MIB data/mibs/nortel/MICOM-OSCAR-MIB.db
2107. MICOM-SYS-MIB data/mibs/nortel/MICOM-SYS-MIB.db
2108. MICOM-4400-T1E1-MIB data/mibs/nortel/MICOM-4400-T1E1-MIB.db
2109. MICOM-TRAFFIC-MGMT-MIB data/mibs/nortel/MICOM-TRAFFIC-MGMT-MIB.db
2110. MICOM-4400-VOICE-NETWORK-MIB data/mibs/nortel/MICOM-4400-VOICE-NETWORK-MIB.db
2111. MICOM-56KCSU-MIB data/mibs/nortel/MICOM-56KCSU-MIB.db
2112. MICOMBRGEXT data/mibs/nortel/MICOMBRGEXT.db
2113. MICOM-CSU-MIB data/mibs/nortel/MICOM-CSU-MIB.db
2114. TFTP-MIB data/mibs/nortel/TFTP-MIB.db
2115. MICOMETHER data/mibs/nortel/MICOMETHER.db
2116. MICOMFLTR data/mibs/nortel/MICOMFLTR.db
2117. MICOM-FRAMERELAY-MIB data/mibs/nortel/MICOM-FRAMERELAY-MIB.db
2118. MICOM-FRDCE-MIB data/mibs/nortel/MICOM-FRDCE-MIB.db
2119. MICOM-GCM-MIB data/mibs/nortel/MICOM-GCM-MIB.db
2120. MICOM-IPDNA-MIB data/mibs/nortel/MICOM-IPDNA-MIB.db
2121. MICOMIPRIP-MIB data/mibs/nortel/MICOMIPRIP-MIB.db
2122. IPX data/mibs/nortel/IPX.db
2123. MICOM-ISDN-MIB data/mibs/nortel/MICOM-ISDN-MIB.db
2124. MICOM-MPANL-LMI-MIB data/mibs/nortel/MICOM-MPANL-LMI-MIB.db
2125. MICOM-MPANL-SIGNALING-MIB data/mibs/nortel/MICOM-MPANL-SIGNALING-MIB.db
2126. MICOM-NAC-MIB data/mibs/nortel/MICOM-NAC-MIB.db
2127. MICOM-NAS-MIB data/mibs/nortel/MICOM-NAS-MIB.db
2128. MICOM-RSI-MIB data/mibs/nortel/MICOM-RSI-MIB.db
2129. MICOM-T1CSU-MIB data/mibs/nortel/MICOM-T1CSU-MIB.db
2130. MICOM-WAN-MIB data/mibs/nortel/MICOM-WAN-MIB.db
2131. MICOM-NODE-MIB data/mibs/nortel/MICOM-NODE-MIB.db
2132. Nortel-Magellan-Passport-UsefulDefinitionsMIB data/mibs/nortel/Nortel-Magellan-Passport-UsefulDefinitionsMIB.db
2133. Nortel-Magellan-Passport-StandardTextualConventionsMIB data/mibs/nortel/Nortel-Magellan-Passport-StandardTextualConventionsMIB.db
2134. Nortel-Magellan-Passport-TextualConventionsMIB data/mibs/nortel/Nortel-Magellan-Passport-TextualConventionsMIB.db
2135. Nortel-Magellan-Passport-AlarmMIB data/mibs/nortel/Nortel-Magellan-Passport-AlarmMIB.db
2136. Nortel-Magellan-Passport-AppnMIB data/mibs/nortel/Nortel-Magellan-Passport-AppnMIB.db
2137. Nortel-Magellan-Passport-ApsMIB data/mibs/nortel/Nortel-Magellan-Passport-ApsMIB.db
2138. Nortel-Magellan-Passport-LogicalProcessorMIB data/mibs/nortel/Nortel-Magellan-Passport-LogicalProcessorMIB.db
2139. Nortel-Magellan-Passport-AtmBaseMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmBaseMIB.db
2140. Nortel-Magellan-Passport-AtmCoreMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmCoreMIB.db
2141. Nortel-Magellan-Passport-AtmBearerServiceMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmBearerServiceMIB.db
2142. Nortel-Magellan-Passport-AtmIispMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmIispMIB.db
2143. Nortel-Magellan-Passport-AtmMpeMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmMpeMIB.db
2144. Nortel-Magellan-Passport-AtmNetworkingMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmNetworkingMIB.db
2145. Nortel-Magellan-Passport-AtmPnniMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmPnniMIB.db
2146. Nortel-Magellan-Passport-TrunksMIB data/mibs/nortel/Nortel-Magellan-Passport-TrunksMIB.db
2147. Nortel-Magellan-Passport-AtmTrunksMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmTrunksMIB.db
2148. Nortel-Magellan-Passport-AtmUniMIB data/mibs/nortel/Nortel-Magellan-Passport-AtmUniMIB.db
2149. Nortel-Magellan-Passport-BaseRoutingMIB data/mibs/nortel/Nortel-Magellan-Passport-BaseRoutingMIB.db
2150. Nortel-Magellan-Passport-VirtualRouterMIB data/mibs/nortel/Nortel-Magellan-Passport-VirtualRouterMIB.db
2151. Nortel-Magellan-Passport-BaseSnmpMIB data/mibs/nortel/Nortel-Magellan-Passport-BaseSnmpMIB.db
2152. Nortel-Magellan-Passport-IpMIB data/mibs/nortel/Nortel-Magellan-Passport-IpMIB.db
2153. Nortel-Magellan-Passport-BgpMIB data/mibs/nortel/Nortel-Magellan-Passport-BgpMIB.db
2154. Nortel-Magellan-Passport-BitTransparentMIB data/mibs/nortel/Nortel-Magellan-Passport-BitTransparentMIB.db
2155. Nortel-Magellan-Passport-BridgeMIB data/mibs/nortel/Nortel-Magellan-Passport-BridgeMIB.db
2156. Nortel-Magellan-Passport-CallRedirectionMIB data/mibs/nortel/Nortel-Magellan-Passport-CallRedirectionMIB.db
2157. Nortel-Magellan-Passport-CallServerMIB data/mibs/nortel/Nortel-Magellan-Passport-CallServerMIB.db
2158. Nortel-Magellan-Passport-CasTestMIB data/mibs/nortel/Nortel-Magellan-Passport-CasTestMIB.db
2159. Nortel-Magellan-Passport-CircuitEmulationServiceMIB data/mibs/nortel/Nortel-Magellan-Passport-CircuitEmulationServiceMIB.db
2160. Nortel-Magellan-Passport-DataCollectionMIB data/mibs/nortel/Nortel-Magellan-Passport-DataCollectionMIB.db

2161. Nortel-Magellan-Passport-DataIsdnMIB data/mibs/nortel/Nortel-Magellan-Passport-DataIsdnMIB.db
2162. Nortel-Magellan-Passport-DcmeMIB data/mibs/nortel/Nortel-Magellan-Passport-DcmeMIB.db
2163. Nortel-Magellan-Passport-DisdnETSIMIB data/mibs/nortel/Nortel-Magellan-Passport-DisdnETSIMIB.db
2164. Nortel-Magellan-Passport-DisdnJapanInsMIB data/mibs/nortel/Nortel-Magellan-Passport-DisdnJapanInsMIB.db
2165. Nortel-Magellan-Passport-DisdnNI2MIB data/mibs/nortel/Nortel-Magellan-Passport-DisdnNI2MIB.db
2166. Nortel-Magellan-Passport-DisdnNISMIB data/mibs/nortel/Nortel-Magellan-Passport-DisdnNISMIB.db
2167. Nortel-Magellan-Passport-DisdnTS014MIB data/mibs/nortel/Nortel-Magellan-Passport-DisdnTS014MIB.db
2168. Nortel-Magellan-Passport-DpnRoutingMIB data/mibs/nortel/Nortel-Magellan-Passport-DpnRoutingMIB.db
2169. Nortel-Magellan-Passport-DpnTrunksMIB data/mibs/nortel/Nortel-Magellan-Passport-DpnTrunksMIB.db
2170. Nortel-Magellan-Passport-McsMgrMIB data/mibs/nortel/Nortel-Magellan-Passport-McsMgrMIB.db
2171. Nortel-Magellan-Passport-DprsMcsEpMIB data/mibs/nortel/Nortel-Magellan-Passport-DprsMcsEpMIB.db
2172. Nortel-Magellan-Passport-ExtensionsMIB data/mibs/nortel/Nortel-Magellan-Passport-ExtensionsMIB.db
2173. Nortel-Magellan-Passport-FileSystemMIB data/mibs/nortel/Nortel-Magellan-Passport-FileSystemMIB.db
2174. Nortel-Magellan-Passport-TraceBaseMIB data/mibs/nortel/Nortel-Magellan-Passport-TraceBaseMIB.db
2175. Nortel-Magellan-Passport-FrTraceRcvrMIB data/mibs/nortel/Nortel-Magellan-Passport-FrTraceRcvrMIB.db
2176. Nortel-Magellan-Passport-FraDpnTrunksMIB data/mibs/nortel/Nortel-Magellan-Passport-FraDpnTrunksMIB.db
2177. Nortel-Magellan-Passport-FrameRelayAtmMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayAtmMIB.db
2178. Nortel-Magellan-Passport-FrameRelayDteMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayDteMIB.db
2179. Nortel-Magellan-Passport-FrameRelayEngMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayEngMIB.db
2180. Nortel-Magellan-Passport-FrameRelayUniMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayUniMIB.db
2181. Nortel-Magellan-Passport-FrameRelayIsdnMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayIsdnMIB.db
2182. Nortel-Magellan-Passport-FrameRelayMuxMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayMuxMIB.db
2183. Nortel-Magellan-Passport-FrameRelayNniMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayNniMIB.db
2184. Nortel-Magellan-Passport-FrameRelayNniTraceMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayNniTraceMIB.db
2185. Nortel-Magellan-Passport-FrameRelayUniTraceMIB data/mibs/nortel/Nortel-Magellan-Passport-FrameRelayUniTraceMIB.db
2186. Nortel-Magellan-Passport-Frf5EpMIB data/mibs/nortel/Nortel-Magellan-Passport-Frf5EpMIB.db
2187. Nortel-Magellan-Passport-GeneralVcInterfaceMIB data/mibs/nortel/Nortel-Magellan-Passport-GeneralVcInterfaceMIB.db
2188. Nortel-Magellan-Passport-GsmIwFMIB data/mibs/nortel/Nortel-Magellan-Passport-GsmIwFMIB.db
2189. Nortel-Magellan-Passport-HdlcTransparentMIB data/mibs/nortel/Nortel-Magellan-Passport-HdlcTransparentMIB.db
2190. Nortel-Magellan-Passport-HuntGroupEngMIB data/mibs/nortel/Nortel-Magellan-Passport-HuntGroupEngMIB.db
2191. Nortel-Magellan-Passport-HuntGroupMIB data/mibs/nortel/Nortel-Magellan-Passport-HuntGroupMIB.db
2192. Nortel-Magellan-Passport-ImaMIB data/mibs/nortel/Nortel-Magellan-Passport-ImaMIB.db
2193. Nortel-Magellan-Passport-IpCosMIB data/mibs/nortel/Nortel-Magellan-Passport-IpCosMIB.db
2194. Nortel-Magellan-Passport-IpNhrpMIB data/mibs/nortel/Nortel-Magellan-Passport-IpNhrpMIB.db
2195. Nortel-Magellan-Passport-IpVrrpMIB data/mibs/nortel/Nortel-Magellan-Passport-IpVrrpMIB.db
2196. Nortel-Magellan-Passport-IpiFrMIB data/mibs/nortel/Nortel-Magellan-Passport-IpiFrMIB.db
2197. Nortel-Magellan-Passport-IpiVcMIB data/mibs/nortel/Nortel-Magellan-Passport-IpiVcMIB.db
2198. Nortel-Magellan-Passport-IpxMIB data/mibs/nortel/Nortel-Magellan-Passport-IpxMIB.db
2199. Nortel-Magellan-Passport-KurtNetMIB data/mibs/nortel/Nortel-Magellan-Passport-KurtNetMIB.db
2200. Nortel-Magellan-Passport-LanDriversMIB data/mibs/nortel/Nortel-Magellan-Passport-LanDriversMIB.db
2201. Nortel-Magellan-Passport-LaneClientMIB data/mibs/nortel/Nortel-Magellan-Passport-LaneClientMIB.db
2202. Nortel-Magellan-Passport-MgmtInterfacesMIB data/mibs/nortel/Nortel-Magellan-Passport-MgmtInterfacesMIB.db
2203. Nortel-Magellan-Passport-ShelfMIB data/mibs/nortel/Nortel-Magellan-Passport-ShelfMIB.db
2204. Nortel-Magellan-Passport-ModCommonMIB data/mibs/nortel/Nortel-Magellan-Passport-ModCommonMIB.db
2205. Nortel-Magellan-Passport-ModAtmQosMIB data/mibs/nortel/Nortel-Magellan-Passport-ModAtmQosMIB.db
2206. Nortel-Magellan-Passport-ModDprsQosMIB data/mibs/nortel/Nortel-Magellan-Passport-ModDprsQosMIB.db
2207. Nortel-Magellan-Passport-MpaNetworkLinkMIB data/mibs/nortel/Nortel-Magellan-Passport-MpaNetworkLinkMIB.db
2208. Nortel-Magellan-Passport-NetSentryMIB data/mibs/nortel/Nortel-Magellan-Passport-NetSentryMIB.db
2209. Nortel-Magellan-Passport-OamEthernetMIB data/mibs/nortel/Nortel-Magellan-Passport-OamEthernetMIB.db
2210. Nortel-Magellan-Passport-PorsTrunksMIB data/mibs/nortel/Nortel-Magellan-Passport-PorsTrunksMIB.db
2211. Nortel-Magellan-Passport-PorsAtmTrunksMIB data/mibs/nortel/Nortel-Magellan-Passport-PorsAtmTrunksMIB.db
2212. Nortel-Magellan-Passport-PorsVcMIB data/mibs/nortel/Nortel-Magellan-Passport-PorsVcMIB.db
2213. Nortel-Magellan-Passport-PppMIB data/mibs/nortel/Nortel-Magellan-Passport-PppMIB.db
2214. Nortel-Magellan-Passport-ProvisioningMIB data/mibs/nortel/Nortel-Magellan-Passport-ProvisioningMIB.db
2215. Nortel-Magellan-Passport-StateSummaryMIB data/mibs/nortel/Nortel-Magellan-Passport-StateSummaryMIB.db
2216. Nortel-Magellan-Passport-ServerAccessRsMIB data/mibs/nortel/Nortel-Magellan-Passport-ServerAccessRsMIB.db
2217. Nortel-Magellan-Passport-ShortcutConnectionMIB data/mibs/nortel/Nortel-Magellan-Passport-ShortcutConnectionMIB.db
2218. Nortel-Magellan-Passport-SnaMIB data/mibs/nortel/Nortel-Magellan-Passport-SnaMIB.db
2219. Nortel-Magellan-Passport-SoftwareMIB data/mibs/nortel/Nortel-Magellan-Passport-SoftwareMIB.db
2220. Nortel-Magellan-Passport-SourceRouteEndStationMIB data/mibs/nortel/Nortel-Magellan-Passport-SourceRouteEndStationMIB.db
2221. Nortel-Magellan-Passport-SubnetInterfaceMIB data/mibs/nortel/Nortel-Magellan-Passport-SubnetInterfaceMIB.db
2222. Nortel-Magellan-Passport-TdmaIwfMIB data/mibs/nortel/Nortel-Magellan-Passport-TdmaIwfMIB.db
2223. Nortel-Magellan-Passport-TimeMIB data/mibs/nortel/Nortel-Magellan-Passport-TimeMIB.db
2224. Nortel-Magellan-Passport-UnackTrunksMIB data/mibs/nortel/Nortel-Magellan-Passport-UnackTrunksMIB.db
2225. Nortel-Magellan-Passport-UtpDpnTrunksMIB data/mibs/nortel/Nortel-Magellan-Passport-UtpDpnTrunksMIB.db
2226. Nortel-Magellan-Passport-VcTesterMIB data/mibs/nortel/Nortel-Magellan-Passport-VcTesterMIB.db
2227. Nortel-Magellan-Passport-VirtualMediaMIB data/mibs/nortel/Nortel-Magellan-Passport-VirtualMediaMIB.db
2228. Nortel-Magellan-Passport-VncsCallServerMIB data/mibs/nortel/Nortel-Magellan-Passport-VncsCallServerMIB.db
2229. Nortel-Magellan-Passport-VoiceNetworkingMIB data/mibs/nortel/Nortel-Magellan-Passport-VoiceNetworkingMIB.db
2230. Nortel-Magellan-Passport-VnetEtsiQsigMIB data/mibs/nortel/Nortel-Magellan-Passport-VnetEtsiQsigMIB.db
2231. Nortel-Magellan-Passport-VnetEuroIsdnMIB data/mibs/nortel/Nortel-Magellan-Passport-VnetEuroIsdnMIB.db
2232. Nortel-Magellan-Passport-VnetMcdnSigMIB data/mibs/nortel/Nortel-Magellan-Passport-VnetMcdnSigMIB.db
2233. Nortel-Magellan-Passport-VnetNisSigMIB data/mibs/nortel/Nortel-Magellan-Passport-VnetNisSigMIB.db
2234. Nortel-Magellan-Passport-VnsMIB data/mibs/nortel/Nortel-Magellan-Passport-VnsMIB.db
2235. Nortel-Magellan-Passport-VoiceMIB data/mibs/nortel/Nortel-Magellan-Passport-VoiceMIB.db
2236. Nortel-Magellan-Passport-X25DteMIB data/mibs/nortel/Nortel-Magellan-Passport-X25DteMIB.db
2237. Nortel-Magellan-Passport-X25TraceRcvrMIB data/mibs/nortel/Nortel-Magellan-Passport-X25TraceRcvrMIB.db
2238. NTWS-ROOT-MIB data/mibs/nortel/NTWS-ROOT-MIB.db
2239. NTWS-AP-TC data/mibs/nortel/NTWS-AP-TC.db
2240. NTWS-AP-STATUS-MIB data/mibs/nortel/NTWS-AP-STATUS-MIB.db
2241. NTWS-BASIC-MIB data/mibs/nortel/NTWS-BASIC-MIB.db

2242. NTWS-CLIENT-SESSION-TC data/mibs/nortel/NTWS-CLIENT-SESSION-TC.db
2243. NTWS-CLIENT-SESSION-MIB data/mibs/nortel/NTWS-CLIENT-SESSION-MIB.db
2244. NTWS-EXTERNAL-SERVER-MIB data/mibs/nortel/NTWS-EXTERNAL-SERVER-MIB.db
2245. NTWS-RF-DETECT-TC data/mibs/nortel/NTWS-RF-DETECT-TC.db
2246. NTWS-INFO-RF-DETECT-MIB data/mibs/nortel/NTWS-INFO-RF-DETECT-MIB.db
2247. NTWS-PORT-MIB data/mibs/nortel/NTWS-PORT-MIB.db
2248. NTWS-REGISTRATION-CHASSIS-MIB data/mibs/nortel/NTWS-REGISTRATION-CHASSIS-MIB.db
2249. NTWS-REGISTRATION-DEVICES-MIB data/mibs/nortel/NTWS-REGISTRATION-DEVICES-MIB.db
2250. NTWS-SYSTEM-MIB data/mibs/nortel/NTWS-SYSTEM-MIB.db
2251. NTWS-TRAP-MIB data/mibs/nortel/NTWS-TRAP-MIB.db
2252. SYNOPTICS-ROOT-MIB data/mibs/nortel/SYNOPTICS-ROOT-MIB.db
2253. NORTEL-NETWORKS-MULTIPLE-SPANNING-TREE-MIB data/mibs/nortel/NORTEL-NETWORKS-MULTIPLE-SPANNING-TREE-MIB.db
2254. NORTEL-NETWORKS-RAPID-SPANNING-TREE-MIB data/mibs/nortel/NORTEL-NETWORKS-RAPID-SPANNING-TREE-MIB.db
2255. NORTEL-SECURE-NETWORK-ACCESS-MIB data/mibs/nortel/NORTEL-SECURE-NETWORK-ACCESS-MIB.db
2256. ACUMEN-MIB data/mibs/nortel/ACUMEN-MIB.db
2257. WARP-REG-MIB data/mibs/nortel/WARP-REG-MIB.db
2258. WARP-MIB data/mibs/nortel/WARP-MIB.db
2259. WARP-DESIGNER-MIB data/mibs/nortel/WARP-DESIGNER-MIB.db
2260. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2261. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2262. SMALLSITE-COMMON-MIB data/mibs/nortel/SMALLSITE-COMMON-MIB.db
2263. SMALLSITE-EVENTS-MIB data/mibs/nortel/SMALLSITE-EVENTS-MIB.db
2264. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2265. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2266. NT-ENTERPRISE-DATA-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-MIB.db
2267. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2268. QOSSLA-MIB data/mibs/nortel/QOSSLA-MIB.db
2269. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2270. NT-ENTERPRISE-DATA-TASMAN-MGMT-SYSTEM-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-SYSTEM-MIB.db
2271. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2272. NT-ENTERPRISE-DATA-TASMAN-MGMT-SNMP-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-SNMP-MIB.db
2273. NT-ENTERPRISE-DATA-TASMAN-MGMT-SNAG-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-SNAG-MIB.db
2274. NT-ENTERPRISE-DATA-TASMAN-MGMT-SERIAL-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-SERIAL-MIB.db
2275. NT-ENTERPRISE-DATA-TASMAN-MGMT-QOS-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-QOS-MIB.db
2276. NT-ENTERPRISE-DATA-TASMAN-MGMT-PPP-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-PPP-MIB.db
2277. NT-ENTERPRISE-DATA-TASMAN-MGMT-IP-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-IP-MIB.db
2278. NT-ENTERPRISE-DATA-TASMAN-MGMT-GENERIC-HDLC-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-GENERIC-HDLC-MIB.db
2279. NT-ENTERPRISE-DATA-TASMAN-MGMT-FR-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-FR-MIB.db
2280. NT-ENTERPRISE-DATA-TASMAN-MGMT-ETHERNET-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-ETHERNET-MIB.db
2281. NT-ENTERPRISE-DATA-TASMAN-MGMT-ENVIRONMENT-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-ENVIRONMENT-MIB.db
2282. NT-ENTERPRISE-DATA-TASMAN-MGMT-DSX-TE3-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-DSX-TE3-MIB.db
2283. NT-ENTERPRISE-DATA-TASMAN-MGMT-DSX-TE1-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-DSX-TE1-MIB.db
2284. NT-ENTERPRISE-DATA-TASMAN-MGMT-DSX-TC-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-DSX-TC-MIB.db
2285. NT-ENTERPRISE-DATA-TASMAN-MGMT-DOS-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-DOS-MIB.db
2286. NT-ENTERPRISE-DATA-TASMAN-MGMT-CONFIG-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-CONFIG-MIB.db
2287. NT-ENTERPRISE-DATA-TASMAN-MGMT-CHASSIS-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-CHASSIS-MIB.db
2288. NT-ENTERPRISE-DATA-TASMAN-MGMT-BUNDLE-MIB data/mibs/nortel/NT-ENTERPRISE-DATA-TASMAN-MGMT-BUNDLE-MIB.db
2289. NEWOAK-MIB data/mibs/nortel/NEWOAK-MIB.db
2290. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2291. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2292. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2293. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2294. CONTIVITY-TRAPS-V2-MIB data/mibs/nortel/CONTIVITY-TRAPS-V2-MIB.db
2295. CONTIVITY-TRAPS-V1-MIB data/mibs/nortel/CONTIVITY-TRAPS-V1-MIB.db
2296. CONTIVITY-INFO-V1-MIB data/mibs/nortel/CONTIVITY-INFO-V1-MIB.db
2297. CONTIVITY-TRAP-ACKNOWLEDGMENT-MIB data/mibs/nortel/CONTIVITY-TRAP-ACKNOWLEDGMENT-MIB.db
2298. CONTIVITY-ID-V1-MIB data/mibs/nortel/CONTIVITY-ID-V1-MIB.db
2299. DEFINITIONS data/mibs/nortel/DEFINITIONS.db
2300. ACUMEN-WB-MIB data/mibs/nortel/ACUMEN-WB-MIB.db
2301. WARP-TRAP-MIB data/mibs/nortel/WARP-TRAP-MIB.db
2302. RBN-SMI data/mibs/redback/RBN-SMI.db
2303. RBN-X-AAL5-VCL-STAT-MIB data/mibs/redback/RBN-X-AAL5-VCL-STAT-MIB.db
2304. RBN-AAL5-VCL-STAT-MIB data/mibs/redback/RBN-AAL5-VCL-STAT-MIB.db
2305. RBN-ATM-PROFILE-MIB data/mibs/redback/RBN-ATM-PROFILE-MIB.db
2306. RBN-PRODUCT-MIB data/mibs/redback/RBN-PRODUCT-MIB.db
2307. RBN-SMS1000-ENVMON-MIB data/mibs/redback/RBN-SMS1000-ENVMON-MIB.db
2308. RBN-X-ATM-PROFILE-MIB data/mibs/redback/RBN-X-ATM-PROFILE-MIB.db
2309. SHIVA-MIB data/mibs/shiva/SHIVA-MIB.db
2310. SHIVA-MLOG-MIB data/mibs/shiva/SHIVA-MLOG-MIB.db
2311. SHIVA-SCC-MIB data/mibs/shiva/SHIVA-SCC-MIB.db
2312. SHIVA-SAM-MIB data/mibs/shiva/SHIVA-SAM-MIB.db
2313. SHIVA-USERS-MIB data/mibs/shiva/SHIVA-USERS-MIB.db
2314. SHIVA-LOG-MIB data/mibs/shiva/SHIVA-LOG-MIB.db
2315. SHIVA-VERS-MIB data/mibs/shiva/SHIVA-VERS-MIB.db
2316. SHIVA-MEM-MIB data/mibs/shiva/SHIVA-MEM-MIB.db
2317. SHIVA-TIME-MIB data/mibs/shiva/SHIVA-TIME-MIB.db
2318. SHIVA-FEATURES-MIB data/mibs/shiva/SHIVA-FEATURES-MIB.db
2319. SHIVA-CONF-MIB data/mibs/shiva/SHIVA-CONF-MIB.db
2320. SHIVA-SERIAL-MIB data/mibs/shiva/SHIVA-SERIAL-MIB.db
2321. SHIVA-SESS-MIB data/mibs/shiva/SHIVA-SESS-MIB.db
2322. SHIVA-LTL-MIB data/mibs/shiva/SHIVA-LTL-MIB.db

2323. SHIVA-MODEM-MIB data/mibs/shiva/SHIVA-MODEM-MIB.db
2324. SHIVA-LINE-MIB data/mibs/shiva/SHIVA-LINE-MIB.db
2325. SHIVA-SLOT-MIB data/mibs/shiva/SHIVA-SLOT-MIB.db
2326. SHIVA-PERFORMANCE-ACCOUNTING-MIB data/mibs/shiva/SHIVA-PERFORMANCE-ACCOUNTING-MIB.db
2327. SHIVA-CALL-MIB data/mibs/shiva/SHIVA-CALL-MIB.db
2328. SHIVA-ACCT-MIB data/mibs/shiva/SHIVA-ACCT-MIB.db
2329. SHIVA-DMC-MIB data/mibs/shiva/SHIVA-DMC-MIB.db
2330. SHIVA-FP-MIB data/mibs/shiva/SHIVA-FP-MIB.db
2331. SHIVA-NME-MIB data/mibs/shiva/SHIVA-NME-MIB.db
2332. SHIVA-AT-MIB data/mibs/shiva/SHIVA-AT-MIB.db
2333. SHIVA-IP-MIB data/mibs/shiva/SHIVA-IP-MIB.db
2334. SHIVA-COMM-MIB data/mibs/shiva/SHIVA-COMM-MIB.db
2335. SHIVA-ETHER-MIB data/mibs/shiva/SHIVA-ETHER-MIB.db
2336. SHIVA-NW-MIB data/mibs/shiva/SHIVA-NW-MIB.db
2337. SHIVA-CHIPS-MIB data/mibs/shiva/SHIVA-CHIPS-MIB.db
2338. SHIVA-FILTER-MIB data/mibs/shiva/SHIVA-FILTER-MIB.db
2339. SHIVA-TPPP-MIB data/mibs/shiva/SHIVA-TPPP-MIB.db
2340. SHIVA-RADIUS-MIB data/mibs/shiva/SHIVA-RADIUS-MIB.db
2341. SHIVA-FLASH-MIB data/mibs/shiva/SHIVA-FLASH-MIB.db
2342. SHIVA-PBURST-MIB data/mibs/shiva/SHIVA-PBURST-MIB.db
2343. SONOMASYSTEMS-SONOMA-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-MIB.db
2344. SONOMASYSTEMS-SONOMA-ATM-GENERIC-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-ATM-GENERIC-MIB.db
2345. SONOMASYSTEMS-SONOMA-ATM-T1-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-ATM-T1-MIB.db
2346. SONOMASYSTEMS-SONOMA-ATM-DS3-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-ATM-DS3-MIB.db
2347. SONOMASYSTEMS-SONOMA-ATM-E1-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-ATM-E1-MIB .db
2348. SONOMASYSTEMS-SONOMA-ATM-E3-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-ATM-E3-MIB.db
2349. SONOMASYSTEMS-SONOMA-ATM-IMA-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-ATM-IMA-MIB.db
2350. SONOMASYSTEMS-SONOMA-IPAPPS-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-IPAPPS-MIB .db
2351. SONOMASYSTEMS-SONOMA-IPPROP-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-IPPROP-MIB .db
2352. SONOMASYSTEMS-SONOMA-ATM-OC3c-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-ATM-OC3c-MIB .db
2353. SONOMASYSTEMS-SONOMA-SLIP-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-SLIP-MIB .db
2354. SONOMASYSTEMS-SONOMA-ETHERNET-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-ETHERNET-MIB .db
2355. SONOMASYSTEMS-SONOMA-SRTB-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-SRTB-MIB .db
2356. SONOMASYSTEMS-SONOMA-TRNCPQ-MIB data/mibs/sonoma/SONOMASYSTEMS-SONOMA-TRNCPQ-MIB .db
2357. SONOMASYSTEMS-TRAPS-MIB data/mibs/sonoma/SONOMASYSTEMS-TRAPS-MIB.db
2358. FASTCOMM-MIB data/mibs/timeplex/FASTCOMM-MIB.db
2359. FASTCOMM-F82x4-MIB data/mibs/timeplex/FASTCOMM-F82x4-MIB.db
2360. RFC1229-MIB data/mibs/timeplex/RFC1229-MIB.db
2361. FASTCOMM-F2x41-MIB data/mibs/timeplex/FASTCOMM-F2x41-MIB.db
2362. FASTCOMM-F2x44-MIB data/mibs/timeplex/FASTCOMM-F2x44-MIB.db
2363. FASTCOMM-F2x44-STATS-MIB data/mibs/timeplex/FASTCOMM-F2x44-STATS-MIB.db
2364. VERILINK-ENTERPRISE-NCMALARM-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMALARM-MIB.db
2365. VERILINK-ENTERPRISE-NCMGENERIC-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMGENERIC-MIB.db
2366. VERILINK-ENTERPRISE-NCMQUAD-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMQUAD-MIB.db
2367. VERILINK-ENTERPRISE-NCMDSU-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMDSU-MIB.db
2368. VERILINK-ENTERPRISE-CSUNCM-MIB data/mibs/verilink/VERILINK-ENTERPRISE-CSUNCM-MIB.db
2369. VERILINK-ENTERPRISE-NCMIDCSU-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMIDCSU-MIB.db
2370. VERILINK-ENTERPRISE-NCMIMUX-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMIMUX-MIB.db
2371. VERILINK-ENTERPRISE-NCMISDN-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMISDN-MIB.db
2372. VERILINK-ENTERPRISE-NCMDS3-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMDS3-MIB.db
2373. VERILINK-ENTERPRISE-NCMM13-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMM13-MIB.db
2374. VERILINK-ENTERPRISE-NCMJAPISDN-MIB data/mibs/verilink/VERILINK-ENTERPRISE-NCMJAPISDN-MIB.db
2375. Wellfleet-COMMON-MIB data/mibs/wellfleet/Wellfleet-COMMON-MIB.db
2376. BayNetworks-AHB-MIB data/mibs/wellfleet/BayNetworks-AHB-MIB.db
2377. BayNetworks-DNS-MIB data/mibs/wellfleet/BayNetworks-DNS-MIB.db
2378. BayNetworks-NTP-MIB data/mibs/wellfleet/BayNetworks-NTP-MIB.db
2379. BayNetworks-RCMDS-MIB data/mibs/wellfleet/BayNetworks-RCMDS-MIB.db
2380. Wellfleet-AOT-MIB data/mibs/wellfleet/Wellfleet-AOT-MIB.db
2381. Wellfleet-APPN-MIB data/mibs/wellfleet/Wellfleet-APPN-MIB.db
2382. Wellfleet-5000-CHASSIS-MIB ... data/mibs/wellfleet/Wellfleet-5000-CHASSIS-MIB.db
2383. Wellfleet-ARP-MIB data/mibs/wellfleet/Wellfleet-ARP-MIB.db
2384. Wellfleet-ASR-MIB data/mibs/wellfleet/Wellfleet-ASR-MIB.db
2385. Wellfleet-ASync-MIB data/mibs/wellfleet/Wellfleet-ASync-MIB.db
2386. Wellfleet-AT-MIB data/mibs/wellfleet/Wellfleet-AT-MIB.db
2387. Wellfleet-ATM-LE-MIB data/mibs/wellfleet/Wellfleet-ATM-LE-MIB.db
2388. Wellfleet-ATM-MIB data/mibs/wellfleet/Wellfleet-ATM-MIB.db
2389. Wellfleet-BB-MIB-MIB data/mibs/wellfleet/Wellfleet-BB-MIB-MIB.db
2390. Wellfleet-BGP-MIB data/mibs/wellfleet/Wellfleet-BGP-MIB.db
2391. Wellfleet-BISync-MIB data/mibs/wellfleet/Wellfleet-BISync-MIB.db
2392. Wellfleet-BOOTP-MIB data/mibs/wellfleet/Wellfleet-BOOTP-MIB.db
2393. Wellfleet-BOT-MIB data/mibs/wellfleet/Wellfleet-BOT-MIB.db
2394. Wellfleet-BRIDGE-MIB data/mibs/wellfleet/Wellfleet-BRIDGE-MIB.db
2395. Wellfleet-CCT-NAME-MIB data/mibs/wellfleet/Wellfleet-CCT-NAME-MIB.db
2396. Wellfleet-CCTOPTS-MIB data/mibs/wellfleet/Wellfleet-CCTOPTS-MIB.db
2397. Wellfleet-CONSOLE-MIB data/mibs/wellfleet/Wellfleet-CONSOLE-MIB.db
2398. Wellfleet-CSMACD-MIB data/mibs/wellfleet/Wellfleet-CSMACD-MIB.db
2399. Wellfleet-DCMMW-MIB data/mibs/wellfleet/Wellfleet-DCMMW-MIB.db
2400. Wellfleet-DECNET-MIB data/mibs/wellfleet/Wellfleet-DECNET-MIB.db
2401. Wellfleet-DLS-MIB data/mibs/wellfleet/Wellfleet-DLS-MIB.db
2402. Wellfleet-DOT1QTAG-CONFIG-MIB data/mibs/wellfleet/Wellfleet-DOT1QTAG-CONFIG-MIB.db
2403. Wellfleet-DS1-MIB data/mibs/wellfleet/Wellfleet-DS1-MIB.db

2404.	Wellfleet-DS1E1-MIB	data/mibs/wellfleet/Wellfleet-DS1E1-MIB.db
2405.	Wellfleet-DS3-MIB	data/mibs/wellfleet/Wellfleet-DS3-MIB.db
2406.	Wellfleet-DSUCSU-MIB	data/mibs/wellfleet/Wellfleet-DSUCSU-MIB.db
2407.	Wellfleet-DSX3-MIB	data/mibs/wellfleet/Wellfleet-DSX3-MIB.db
2408.	Wellfleet-DS3E3-MIB	data/mibs/wellfleet/Wellfleet-DS3E3-MIB.db
2409.	Wellfleet-DVMRP-MIB	data/mibs/wellfleet/Wellfleet-DVMRP-MIB.db
2410.	Wellfleet-E1-MIB	data/mibs/wellfleet/Wellfleet-E1-MIB.db
2411.	Wellfleet-EGP-MIB	data/mibs/wellfleet/Wellfleet-EGP-MIB.db
2412.	Wellfleet-FAKE-EVENT-MIB	data/mibs/wellfleet/Wellfleet-FAKE-EVENT-MIB.db
2413.	Wellfleet-FDDI-MIB	data/mibs/wellfleet/Wellfleet-FDDI-MIB.db
2414.	Wellfleet-FNFS-ATM-MIB	data/mibs/wellfleet/Wellfleet-FNFS-ATM-MIB.db
2415.	Wellfleet-FR-MIB	data/mibs/wellfleet/Wellfleet-FR-MIB.db
2416.	Wellfleet-FR2-MIB	data/mibs/wellfleet/Wellfleet-FR2-MIB.db
2417.	Wellfleet-FRSW-MIB	data/mibs/wellfleet/Wellfleet-FRSW-MIB.db
2418.	Wellfleet-FSM-MIB	data/mibs/wellfleet/Wellfleet-FSM-MIB.db
2419.	Wellfleet-FTP-MIB	data/mibs/wellfleet/Wellfleet-FTP-MIB.db
2420.	Wellfleet-GAME-STATS-MIB	data/mibs/wellfleet/Wellfleet-GAME-STATS-MIB.db
2421.	Wellfleet-GRE-MIB	data/mibs/wellfleet/Wellfleet-GRE-MIB.db
2422.	Wellfleet-HARDWARE-MIB	data/mibs/wellfleet/Wellfleet-HARDWARE-MIB.db
2423.	Wellfleet-HSSI-MIB	data/mibs/wellfleet/Wellfleet-HSSI-MIB.db
2424.	Wellfleet-HWF-MIB	data/mibs/wellfleet/Wellfleet-HWF-MIB.db
2425.	Wellfleet-HWMO-MIB	data/mibs/wellfleet/Wellfleet-HWMO-MIB.db
2426.	WF-HTTP-MIB	data/mibs/wellfleet/WF-HTTP-MIB.db
2427.	Wellfleet-IF-MIB	data/mibs/wellfleet/Wellfleet-IF-MIB.db
2428.	Wellfleet-IFWALL-MIB	data/mibs/wellfleet/Wellfleet-IFWALL-MIB.db
2429.	Wellfleet-IGMP-MIB	data/mibs/wellfleet/Wellfleet-IGMP-MIB.db
2430.	Wellfleet-INT-SERV-MIB	data/mibs/wellfleet/Wellfleet-INT-SERV-MIB.db
2431.	Wellfleet-IP-MIB	data/mibs/wellfleet/Wellfleet-IP-MIB.db
2432.	Wellfleet-IPACCT-MIB	data/mibs/wellfleet/Wellfleet-IPACCT-MIB.db
2433.	Wellfleet-IPEX-MIB	data/mibs/wellfleet/Wellfleet-IPEX-MIB.db
2434.	Wellfleet-IPPOLICY-MIB	data/mibs/wellfleet/Wellfleet-IPPOLICY-MIB.db
2435.	Wellfleet-IPV6-MIB	data/mibs/wellfleet/Wellfleet-IPV6-MIB.db
2436.	Wellfleet-IPX-MIB	data/mibs/wellfleet/Wellfleet-IPX-MIB.db
2437.	Wellfleet-IPXA-MIB	data/mibs/wellfleet/Wellfleet-IPXA-MIB.db
2438.	Wellfleet-IREDUND-MIB	data/mibs/wellfleet/Wellfleet-IREDUND-MIB.db
2439.	Wellfleet-ISDB-MIB	data/mibs/wellfleet/Wellfleet-ISDB-MIB.db
2440.	Wellfleet-ISDN-MIB	data/mibs/wellfleet/Wellfleet-ISDN-MIB.db
2441.	Wellfleet-L2TP-MIB	data/mibs/wellfleet/Wellfleet-L2TP-MIB.db
2442.	Wellfleet-LAPB-MIB	data/mibs/wellfleet/Wellfleet-LAPB-MIB.db
2443.	Wellfleet-LLC-MIB	data/mibs/wellfleet/Wellfleet-LLC-MIB.db
2444.	Wellfleet-LNM-MIB	data/mibs/wellfleet/Wellfleet-LNM-MIB.db
2445.	Wellfleet-LOADER-MIB	data/mibs/wellfleet/Wellfleet-LOADER-MIB.db
2446.	Wellfleet-MCT1-MIB	data/mibs/wellfleet/Wellfleet-MCT1-MIB.db
2447.	Wellfleet-MIB-HEAP-STATS-MIB	data/mibs/wellfleet/Wellfleet-MIB-HEAP-STATS-MIB.db
2448.	Wellfleet-MIP-MIB	data/mibs/wellfleet/Wellfleet-MIP-MIB.db
2449.	Wellfleet-MODULE-MIB	data/mibs/wellfleet/Wellfleet-MODULE-MIB.db
2450.	Wellfleet-Modem-MIB	data/mibs/wellfleet/Wellfleet-Modem-MIB.db
2451.	Wellfleet-NAME-TABLE-MIB	data/mibs/wellfleet/Wellfleet-NAME-TABLE-MIB.db
2452.	Wellfleet-NAT-MIB	data/mibs/wellfleet/Wellfleet-NAT-MIB.db
2453.	Wellfleet-NBIP-MIB	data/mibs/wellfleet/Wellfleet-NBIP-MIB.db
2454.	Wellfleet-NHRP-MIB	data/mibs/wellfleet/Wellfleet-NHRP-MIB.db
2455.	Wellfleet-NLSP-MIB	data/mibs/wellfleet/Wellfleet-NLSP-MIB.db
2456.	Wellfleet-NML-MIB	data/mibs/wellfleet/Wellfleet-NML-MIB.db
2457.	Wellfleet-NPK-MIB	data/mibs/wellfleet/Wellfleet-NPK-MIB.db
2458.	Wellfleet-OSI-MIB	data/mibs/wellfleet/Wellfleet-OSI-MIB.db
2459.	Wellfleet-OSPF-MIB	data/mibs/wellfleet/Wellfleet-OSPF-MIB.db
2460.	Wellfleet-PCAP-MIB	data/mibs/wellfleet/Wellfleet-PCAP-MIB.db
2461.	Wellfleet-PIM-MIB	data/mibs/wellfleet/Wellfleet-PIM-MIB.db
2462.	Wellfleet-PING-MIB	data/mibs/wellfleet/Wellfleet-PING-MIB.db
2463.	Wellfleet-PPP-MIB	data/mibs/wellfleet/Wellfleet-PPP-MIB.db
2464.	Wellfleet-PROTOPRI-MIB	data/mibs/wellfleet/Wellfleet-PROTOPRI-MIB.db
2465.	Wellfleet-RADIUS-MIB	data/mibs/wellfleet/Wellfleet-RADIUS-MIB.db
2466.	Wellfleet-RARP-MIB	data/mibs/wellfleet/Wellfleet-RARP-MIB.db
2467.	Wellfleet-RESOURCE-MIB	data/mibs/wellfleet/Wellfleet-RESOURCE-MIB.db
2468.	Wellfleet-RF-MIB	data/mibs/wellfleet/Wellfleet-RF-MIB.db
2469.	Wellfleet-RFWALL-MIB	data/mibs/wellfleet/Wellfleet-RFWALL-MIB.db
2470.	Wellfleet-RIP6-MIB	data/mibs/wellfleet/Wellfleet-RIP6-MIB.db
2471.	Wellfleet-RREDUND-MIB	data/mibs/wellfleet/Wellfleet-RREDUND-MIB.db
2472.	Wellfleet-RSVP-MIB	data/mibs/wellfleet/Wellfleet-RSVP-MIB.db
2473.	Wellfleet-RUIBOOT-MIB	data/mibs/wellfleet/Wellfleet-RUIBOOT-MIB.db
2474.	Wellfleet-SDLC-MIB	data/mibs/wellfleet/Wellfleet-SDLC-MIB.db
2475.	Wellfleet-SIP-MIB	data/mibs/wellfleet/Wellfleet-SIP-MIB.db
2476.	Wellfleet-SMDS-MIB	data/mibs/wellfleet/Wellfleet-SMDS-MIB.db
2477.	Wellfleet-SNMPEXT-MIB	data/mibs/wellfleet/Wellfleet-SNMPEXT-MIB.db
2478.	Wellfleet-SNMP-MIB	data/mibs/wellfleet/Wellfleet-SNMP-MIB.db
2479.	Wellfleet-SPAN-MIB	data/mibs/wellfleet/Wellfleet-SPAN-MIB.db
2480.	Wellfleet-SR-MIB	data/mibs/wellfleet/Wellfleet-SR-MIB.db
2481.	Wellfleet-ST2-MIB	data/mibs/wellfleet/Wellfleet-ST2-MIB.db
2482.	Wellfleet-STA-MIB	data/mibs/wellfleet/Wellfleet-STA-MIB.db
2483.	Wellfleet-STATS-MIB	data/mibs/wellfleet/Wellfleet-STATS-MIB.db
2484.	Wellfleet-SWSMDS-MIB	data/mibs/wellfleet/Wellfleet-SWSMDS-MIB.db

2485. Wellfleet-SYNC-MIB data/mibs/wellfleet/Wellfleet-SYNC-MIB.db
2486. Wellfleet-SYS-MIB data/mibs/wellfleet/Wellfleet-SYS-MIB.db
2487. Wellfleet-SYS-SVC-MIB data/mibs/wellfleet/Wellfleet-SYS-SVC-MIB.db
2488. Wellfleet-SYSL-MIB data/mibs/wellfleet/Wellfleet-SYSL-MIB.db
2489. Wellfleet-T1-MIB data/mibs/wellfleet/Wellfleet-T1-MIB.db
2490. Wellfleet-TCP-MIB data/mibs/wellfleet/Wellfleet-TCP-MIB.db
2491. Wellfleet-TCP-ECHOCLI-MIB data/mibs/wellfleet/Wellfleet-TCP-ECHOCLI-MIB.db
2492. Wellfleet-TCP-ECHOSRV-MIB data/mibs/wellfleet/Wellfleet-TCP-ECHOSRV-MIB.db
2493. Wellfleet-TELNET-MIB data/mibs/wellfleet/Wellfleet-TELNET-MIB.db
2494. Wellfleet-TFTP-MIB data/mibs/wellfleet/Wellfleet-TFTP-MIB.db
2495. Wellfleet-TI-RUI-MIB data/mibs/wellfleet/Wellfleet-TI-RUI-MIB.db
2496. Wellfleet-TNC-MIB data/mibs/wellfleet/Wellfleet-TNC-MIB.db
2497. Wellfleet-TOKEN-RING-MIB data/mibs/wellfleet/Wellfleet-TOKEN-RING-MIB.db
2498. Wellfleet-VCCT-MIB data/mibs/wellfleet/Wellfleet-VCCT-MIB.db
2499. Wellfleet-VINES-MIB data/mibs/wellfleet/Wellfleet-VINES-MIB.db
2500. Wellfleet-WCP-MIB data/mibs/wellfleet/Wellfleet-WCP-MIB.db
2501. Wellfleet-WEP-MIB data/mibs/wellfleet/Wellfleet-WEP-MIB.db
2502. Wellfleet-WFDDOT1D-MIB data/mibs/wellfleet/Wellfleet-WFDDOT1D-MIB.db
2503. Wellfleet-WFMPCC-MIB data/mibs/wellfleet/Wellfleet-WFMPCC-MIB.db
2504. Wellfleet-WFMPSS-MIB data/mibs/wellfleet/Wellfleet-WFMPSS-MIB.db
2505. Wellfleet-X25-MIB data/mibs/wellfleet/Wellfleet-X25-MIB.db
2506. Wellfleet-X25PAD-MIB data/mibs/wellfleet/Wellfleet-X25PAD-MIB.db
2507. Wellfleet-XNS-MIB data/mibs/wellfleet/Wellfleet-XNS-MIB.db
2508. ST2-MIB data/mibs/wellfleet/ST2-MIB.db
2509. Wellfleet-Series7-Traps-MIB data/mibs/wellfleet/Wellfleet-Series7-Traps-MIB.db
2510. Wellfleet-OC3-MIB data/mibs/wellfleet/Wellfleet-OC3-MIB.db
2511. Wellfleet-PGM-MIB data/mibs/wellfleet/Wellfleet-PGM-MIB.db
2512. Wellfleet-VRPP-MIB data/mibs/wellfleet/Wellfleet-VRPP-MIB.db
2513. RFC1406-MIB data/mibs/xylan/RFC1406-MIB.db
2514. RFC1407-MIB data/mibs/xylan/RFC1407-MIB.db
2515. RS-232-MIB data/mibs/xylan/RS-232-MIB.db
2516. OSPF-MIB data/mibs/xylan/OSPF-MIB.db
2517. DIAL-CONTROL-MIB data/mibs/xylan/DIAL-CONTROL-MIB.db
2518. XYLAN-BASE-MIB data/mibs/xylan/XYLAN-BASE-MIB.db
2519. ATM-FORUM-ILMI40-MIB data/mibs/xylan/ATM-FORUM-ILMI40-MIB.db
2520. LAN-EMULATION-CLIENT-MIB data/mibs/xylan/LAN-EMULATION-CLIENT-MIB.db
2521. IPX data/mibs/xylan/IPX.db
2522. RIPSAP data/mibs/xylan/RIPSAP.db
2523. MWORKS-MIB data/mibs/xylan/MWORKS-MIB.db
2524. MWORKS-E-MIB data/mibs/xylan/MWORKS-E-MIB.db
2525. NHRP-MIB data/mibs/xylan/NHRP-MIB.db
2526. XYLAN-ATM-MIB data/mibs/xylan/XYLAN-ATM-MIB.db
2527. XYLAN-AVL-MIB data/mibs/xylan/XYLAN-AVL-MIB.db
2528. XYLAN-BACKUP-MIB data/mibs/xylan/XYLAN-BACKUP-MIB.db
2529. XYLAN-ATM-CE-MIB data/mibs/xylan/XYLAN-ATM-CE-MIB.db
2530. CHASSIS-MIB data/mibs/xylan/CHASSIS-MIB.db
2531. XYLAN-CSM-MIB data/mibs/xylan/XYLAN-CSM-MIB.db
2532. XYLAN-DS1-MIB data/mibs/xylan/XYLAN-DS1-MIB.db
2533. XYLAN-DS3-MIB data/mibs/xylan/XYLAN-DS3-MIB.db
2534. ECHANNEL data/mibs/xylan/ECHANNEL.db
2535. XYLAN-FRAME-RELAY-MIB data/mibs/xylan/XYLAN-FRAME-RELAY-MIB.db
2536. XYLAN-FW1-MIB data/mibs/xylan/XYLAN-FW1-MIB.db
2537. XYLAN-FWCONF-MIB data/mibs/xylan/XYLAN-FWCONF-MIB.db
2538. XYLAN-IP-MIB data/mibs/xylan/XYLAN-IP-MIB.db
2539. IPMS-MIB data/mibs/xylan/IPMS-MIB.db
2540. XYLAN-IPX-MIB data/mibs/xylan/XYLAN-IPX-MIB.db
2541. MRD-MIB data/mibs/xylan/MRD-MIB.db
2542. PNHRP-MIB data/mibs/xylan/PNHRP-MIB.db
2543. PNNI-MIB data/mibs/xylan/PNNI-MIB.db
2544. PORT-MIB data/mibs/xylan/PORT-MIB.db
2545. XYLAN-PPP-MIB data/mibs/xylan/XYLAN-PPP-MIB.db
2546. XYLAN-SOFT-PVC-MIB data/mibs/xylan/XYLAN-SOFT-PVC-MIB.db
2547. XYLAN-VAP-MIB data/mibs/xylan/XYLAN-VAP-MIB.db
2548. XYLAN-VLAN-MIB data/mibs/xylan/XYLAN-VLAN-MIB.db
2549. XYLAN-WSM-MIB data/mibs/xylan/XYLAN-WSM-MIB.db
2550. XYLAN-MGMTSTN-MIB data/mibs/xylan/XYLAN-MGMTSTN-MIB.db
2551. XYLANTRAP-MIB data/mibs/xylan/XYLANTRAP-MIB.db
2552. XYLANTRAP-M-MIB data/mibs/xylan/XYLANTRAP-M-MIB.db
2553. XYLANTRAP-3-MIB data/mibs/xylan/XYLANTRAP-3-MIB.db
2554. XYLANTRAP-5-MIB data/mibs/xylan/XYLANTRAP-5-MIB.db
2555. XYLANTRAP-9-MIB data/mibs/xylan/XYLANTRAP-9-MIB.db
2556. XYPLEX-MIB data/mibs/xyplex/XYPLEX-MIB.db
2557. XYPLEX-APPLETALK-MIB data/mibs/xyplex/XYPLEX-APPLETALK-MIB.db
2558. XYPLEX-BOOT-CLIENT-MIB data/mibs/xyplex/XYPLEX-BOOT-CLIENT-MIB.db
2559. XYPLEX-BOOT-SERVER-MIB data/mibs/xyplex/XYPLEX-BOOT-SERVER-MIB.db
2560. XYPLEX-CHARACTER-MIB data/mibs/xyplex/XYPLEX-CHARACTER-MIB.db
2561. XYPLEX-CHASSIS-MIB data/mibs/xyplex/XYPLEX-CHASSIS-MIB.db
2562. XYPLEX-CONCENTRATOR-MIB data/mibs/xyplex/XYPLEX-CONCENTRATOR-MIB.db
2563. XYPLEX-DECNET-MIB data/mibs/xyplex/XYPLEX-DECNET-MIB.db
2564. XYPLEX-ETHERNET-MIB data/mibs/xyplex/XYPLEX-ETHERNET-MIB.db
2565. XYPLEX-IETF-FDDI-MIB data/mibs/xyplex/XYPLEX-IETF-FDDI-MIB.db

2566. [XYPLEX-FRAME-RELAY-MIB](#) data/mibs/xyplex/XYPLEX-FRAME-RELAY-MIB.db
2567. [XYPLEX-IEEE-HUB-MIB](#) data/mibs/xyplex/XYPLEX-IEEE-HUB-MIB.db
2568. [XYPLEX-INTERNET-MIB](#) data/mibs/xyplex/XYPLEX-INTERNET-MIB.db
2569. [XYPLEX-IPX-MIB](#) data/mibs/xyplex/XYPLEX-IPX-MIB.db
2570. [XYPLEX-ISIS-MIB](#) data/mibs/xyplex/XYPLEX-ISIS-MIB.db
2571. [XYPLEX-LAT-MIB](#) data/mibs/xyplex/XYPLEX-LAT-MIB.db
2572. [XYPLEX-LINK-MIB](#) data/mibs/xyplex/XYPLEX-LINK-MIB.db
2573. [XYPLEX-PARAM-CLIENT-MIB](#) data/mibs/xyplex/XYPLEX-PARAM-CLIENT-MIB.db
2574. [XYPLEX-PPP-MIB](#) data/mibs/xyplex/XYPLEX-PPP-MIB.db
2575. [XYPLEX-IETF-PPP-LCP-MIB](#) data/mibs/xyplex/XYPLEX-IETF-PPP-LCP-MIB.db
2576. [XYPLEX-IETF-PPP-BRIDGE-NCP-MIB](#) data/mibs/xyplex/XYPLEX-IETF-PPP-BRIDGE-NCP-MIB.db
2577. [XYPLEX-REPEATER-MIB](#) data/mibs/xyplex/XYPLEX-REPEATER-MIB.db
2578. [XYPLEX-SYSTEM-MIB](#) data/mibs/xyplex/XYPLEX-SYSTEM-MIB.db
2579. [XYPLEX-X25-MIB](#) data/mibs/xyplex/XYPLEX-X25-MIB.db
2580. [XYPLEX-IETF-PPP-IP-NCP-MIB](#) ... data/mibs/xyplex/XYPLEX-IETF-PPP-IP-NCP-MIB.db

[<< Previous Section](#) [Next Section >>](#)



Appendix C: Common Error Messages

Since MIMIC is an extremely complex tool, it was designed to contain extensive diagnostic information. All events of interest are logged and displayed in the [log window](#). Since we cannot display a detailed explanation with every message at runtime, they are listed here.

Information that can change from message to message is shown in **BOLD**.

To use the list, search it by some unique words in the message (this should stay the same across releases).

If you don't see a message explained here, please cut and paste it into an e-mail message and send it to support@gambitcomm.com. We will include it in the next rev of the documentation.

Agent Simulator

```
1. WARN DATE - index simulation load failed TABLE, continuing...  
DATE - cannot find index file for TABLE.  
DATE - data/sim/SIMULATION/MIB/TABLE.idb not in search path
```

Cause

This error occurs when an instance database file (.idb) has not been created for the specified table. The recorder may generate no .idb file if it does not detect instances in a particular table at record time.

Action

The simulator will return no instances for this table.

```
2. WARN DATE - simulation failed OID (OBJECT)  
DATE - agent[NUMBER] ADDRESS: lookup failed  
DATE - value space lookup failed  
DATE - no such object OBJECT  
DATE - scenario load failed data/sim/SIMULATION/MIB/SCENARIO/OBJECT  
DATE - cannot find scenario file for OBJECT.  
DATE - data/sim/SIMULATION/MIB/SCENARIO/OBJECT.var not in search path
```

Cause

This error occurs upon MIB object access when a scenario file (.var) has not been created for the specified object.

Action

The simulator will return 0 for any variables it cannot find.

```
3. WARN DATE - simulation failed OID (OBJECT)  
DATE - cannot exec arg 0 for FUNCTION  
DATE - agent[NUMBER] ADDRESS: lookup failed  
DATE - value space lookup failed  
DATE - no such index
```

Cause

This error occurs when a scenario file (.var) does not contain a variable for the specified instance for the specified object. This can happen on recording of SNMP walk files (produced by a third-party SNMP walk program that does simple GETNEXT traversal), when a table is dynamic and large. Walking one column will retrieve object instances that don't exist for later columns. Creating a simulation of a target device via a live recording solves this problem, since the recorder walks tables entire rows at a time.

Action

The simulator will return 0 for any variables it cannot find.

```
4. WARN DATE - simulation failed OBJECT  
DATE - cannot access trap variable OID  
DATE - not a leaf OBJECT
```

Cause

This error occurs while generating a trap. One of the MIB objects in the VARIABLES clause of the trap is not being simulated by the agent. This usually happens if the agent is not including the MIB which defines the particular variable.

Action

The simulator will not generate the trap because it does not know the syntax of the object. In order to generate this trap, you need to modify the simulation to include the MIB which defines this variable. You can do that with [Simulation->Devices](#) in MIMICView. If you need to find the MIB which defines this variable, use the [oidinfo](#) utility.

```
5. ERROR DATE - agent NUMBER cannot read PDU from ADDRESS  
DATE - snmp_agent_parse failed  
DATE - unknown community: COMMUNITY
```

Cause

An SNMP request with a community string was received which differs from the configured community string for the agent instance.

Action

The SNMP request is ignored. To accept requests at the community, change the agent instance configuration.

6. ERROR **DATE** - agent **NUMBER** cannot read PDU from **ADDRESS**

DATE - snmp_agent_parse failed

DATE - unknown protocol

Cause

An unknown protocol is being used for the agent. MIMIC currently supports SNMPv1, v2c, v2 and v3, but not all may be configured for the particular agent. Some management applications (like HP/OpenView) may want to talk a different protocol, such as SNMPv2c and usually first try it, then fall back to SNMPv1.

Action

The SNMP request is ignored.

7. ERROR **DATE** - agent **NUMBER** cannot read PDU from **ADDRESS**

DATE - snmp_agent_parse failed

DATE - bad authentication. error = -7 (notReportable)

DATE - cannot switch to matching agent.

DATE - no outstanding INFORM for engine id: >**ENGINE-ID**<

Cause

This error usually happens on receiving a duplicate INFORM RESPONSE from the trap receiver.

Action

The INFORM RESPONSE is ignored.

8. ERROR **DATE** - no receiver at **ADDRESS**

Cause

This message means an SNMP request was received for an address that has no agent instance running. This request could be for the host's real IP address, in which case this message happens occasionally.

Action

The SNMP request is ignored.

9. WARN **DATE** - cannot receive from **ADDRESS**. continuing...

DATE - rcvfrom: Connection refused

WARN **DATE** - cannot receive from **ADDRESS**. continuing...

DATE - rcvfrom: Resource temporarily unavailable

WARN **DATE** - cannot receive from **ADDRESS**. continuing...

DATE - rcv: No error

ERROR **DATE** - agent **NUMBER** send failed from **ADDRESS** to **ADDRESS**

DATE - type=**NUMBER** size=**NUMBER**

DATE - sendto: Connection refused

WARN **DATE** - cannot accept connection from **ADDRESS**. continuing...

DATE - accept: Resource temporarily unavailable

Cause

Any of these messages means a protocol request (SNMP, Telnet, TFTP, etc) or connection was aborted by the management application. This happens occasionally, and could be an indication of a faulty management application.

Action

The request is ignored.

10. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**

DATE - possibly caused by another SNMP agent running

DATE - Please refer to Appendix C for more details

DATE - cannot bind receive IP address **ADDRESS** port **PORT**

DATE - bind: Permission denied

Cause

This message means that you are running without sufficient privileges to bind to the selected SNMP socket.

Action

You need to run the MIMIC daemon mimicd with sufficient privileges.

On Unix, this means running mimicd as root, or with setuid-root. The installation by default installs mimicd as setuid-root.

On Windows NT, you need to run MIMIC as a user with Administrator privileges.

11. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**

DATE - possibly caused by another SNMP agent running

DATE - Please refer to Appendix C for more details

DATE - cannot bind receive IP address **ADDRESS** port **PORT**

DATE - bind: no error

```
ERROR DATE - AGNT[x]: cannot start(2) agent NUMBER
DATE - possibly caused by another SNMP agent running
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address ADDRESS port PORT
DATE - bind: No such file or directory
```

```
ERROR DATE - AGNT[x]: cannot start(2) agent NUMBER
DATE - possibly caused by another SNMP agent running
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address ADDRESS port PORT
DATE - bind: Only one usage of each socket address (protocol/network address/port) is normally permitted.
```

```
ERROR DATE - AGNT[x]: cannot start(2) agent NUMBER
DATE - possibly caused by another SNMP agent running
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address ADDRESS port PORT
DATE - bind: the requested address is not valid in its context.
```

Cause

On Windows Vista or later it could also be due to duplicate address detection. See the [Windows Installation Guide](#) for more details.

Action

Only one process can simultaneously use the selected SNMP port on a host.

To verify if there is such a program, stop MIMIC with File->Terminate and use the netstat utility from the DOS command line prompt, for example:

```
C> netstat -a -n | find "161"
TCP 0.0.0.0:161 0.0.0.0:0 LISTENING
UDP 0.0.0.0:161 *:*
```

If these lines show, then start the Windows task manager and see if there is an SNMP agent process running, e.g. snmp.exe, that you need to kill.

Otherwise, contact Technical Support on how to find any other programs using this port.

The Windows NT SNMP service can only be killed from the **Services** control panel.

```
12. ERROR DATE - AGNT[x]: cannot start(2) agent NUMBER
DATE - possibly caused by another SNMP agent running
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address ADDRESS port PORT
DATE - bind: Error 0
```

Cause

This message on Solaris means there is already another process running that uses the selected SNMP port. This is very likely an already-running instance of an SNMP agent.

Action

Only one process can simultaneously use the selected SNMP port on a host.

To verify if there is such a program, stop MIMIC with File->Terminate and use the netstat utility, for example:

```
# netstat -a -n | grep 161
*.161 Idle
#
```

If this gives you any output as above, you already have some process bound to the SNMP port, preventing MIMIC from binding to the port. On Solaris, you should be able to find this process with:

```
# ps -edf | grep snmp
root 300      1 0 Aug 24  ?      0:00 /usr/lib/dmi/snmpXdmid -s ultra5
root 14097 14089 0 09:15:56 pts/10 0:00 grep snmp
root 14094    1 0 09:15:14 ?      0:00 /usr/lib/snmp/snmpdx -y -c /etc/snmp/conf
#
```

(depending on which management software you have loaded on the system). The above shows the default SNMP agent (Process 14094) that ships with Solaris, but HP/Openview has its own, and so do the others.

Otherwise, contact Technical Support on how to find any other programs using this port.

There is no way (that we know) to find out exactly which process has bound to that port, so the only resort is to keep killing processes that have snmp in their name until netstat gives you a clean bill.

```
# kill 14094
# netstat -a -n | grep 161
#
```

```
13. ERROR DATE - AGNT[x]: cannot start(2) agent NUMBER
DATE - possibly caused by another SNMP agent running
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address ADDRESS port PORT
DATE - bind: Address already in use.
```

Cause

This message on Solaris or Linux means there is already another process running that uses the selected SNMP port. This is very likely an already-running

instance of an SNMP agent.

Action

Only one process can simultaneously use the selected SNMP port on a host.

To verify if there is such a program, stop MIMIC with File->Terminate and use the netstat utility, for example:

```
# netstat -a -n | grep 161
*.161 Idle
#
```

If this gives you any output as above, you already have some process bound to the SNMP port, preventing MIMIC from binding to the port. On Linux, you should be able to find this process with:

```
# ps -edf | grep snmp
root 300      1 0 Aug 24  ?        0:00 /usr/sbin/snmpd
root 14097 14089 0 09:15:56 pts/10 0:00 grep snmp
#
```

(depending on which management software you have loaded on the system). The above shows the default SNMP agent (Process 300) that ships with Linux, but HP/Openview has its own, and so do the others.

Otherwise, contact Technical Support on how to find any other programs using this port.

Even with the /proc filesystem, there is no way (that we know) to find out exactly which process has bound to that port, so the only resort is to keep killing processes that have snmp in their name until netstat gives you a clean bill.

```
# kill 14094
# netstat -a -n | grep 161
#
```

14. ERROR **DATE** - agent **NUMBER** cannot start ipalias, continuing...

```
DATE - cannot open socket.
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address ADDRESS port PORT
DATE - bind: Cannot assign requested address
```

Cause

If you receive this while configuring an IPv6 address, then it likely means that duplicate address detection has discovered another system running with the same IPv6 address.

Action

You cannot assign the same IPv6 address to more than one end system. Configure a different address.

15. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**

```
DATE - cannot set address ADDRESS
DATE - ioctl SIOCGIFADDR: No such device or address
```

ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**

```
DATE - cannot set address ADDRESS
DATE - ioctl SIOCSIFADDR: No such device or address
```

Cause

This message on Solaris means that you cannot configure the IP address alias.

Action

You need to perform an extra kernel configuration step on Solaris prior to running MIMIC to enable more than 255 addresses for agent instances. On Solaris 2.6 and later you can, as root, use ndd(1M) to set the necessary parameter.

For details, see the [Solaris Installation Instructions](#).

16. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**

```
DATE - cannot set address ADDRESS
DATE - ioctl SIOCLIFADDIF: No buffer space available
```

Cause

This message on Solaris means that you cannot configure the IP address alias because you have reached the maximum of 8192 per network interface card.

Action

You can only configure 8192 addresses per network interface. To configure more, you need an additional network interface.

For details, see the [Solaris Installation Instructions](#).

17. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**

```
DATE - cannot set address ADDRESS
DATE - cannot add address ADDRESS
DATE - Cannot find unused slot to add ADDRESS
```

Cause

This means that the installation step that pads addresses on Windows NT (before SP4) failed. Consult the [Windows Installation Instructions](#) section for details.

Action

You need to stop MIMIC (using File->Terminate), and perform this step manually:

Login to an account with Administrator privileges.

From the DOS command prompt, change to the BIN\ subdirectory under the MIMIC installation directory, and invoke:

```
C> padaddr 250
```

If you see any error messages at this point, please contact Gambit Communications Technical Support (support@gambitcomm.com).

Otherwise, you need to reboot your machine for this change to take effect.

18. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address **ADDRESS**
DATE - cannot add address **ADDRESS**
DATE - Cannot add IPAddress **ADDRESS**
DATE - Failed DhcpNotifyConfigChange.

Cause
You can only run MIMIC from an account with Administrator privileges. Consult the [Windows Installation Instructions](#) section for details.
Action
Login to an account with Administrator privileges and run MIMIC.

19. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address **ADDRESS**
DATE - cannot add address **ADDRESS**
DATE - AddIPAddress failed. Reason=A device attached to the system is not functioning.

ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address **ADDRESS**
DATE - cannot add address **ADDRESS**
DATE - AddIPAddress failed. Reason=The specified network resource or device is no longer available.

ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address **ADDRESS**
DATE - cannot add address **ADDRESS**
DATE - AddIPAddress failed. Reason=Element not found.

Cause
This error means that the network interface configured for the agent is not available, and usually happens on laptop computers, where you can pull out network cards (PCMCIA, docking stations), or on newer versions of Windows, when the network is unplugged. Consult the [Windows Installation Instructions](#) section for details.
Action
Configure a different network interface for the agent. This can be done interactively with the Interface field in the Advanced tab of the Agent Configuration dialog.

You can also change the default network device (NIC) to be used for all agents where the interface is not explicitly set. To do so, set the MIMIC_DEFAULT_NETDEV environment variable to the desired interface name (eg. as listed in the Interface field in the Agent Configuration dialog) prior to running MIMIC (if it is running, you must terminate it with File->Terminate from MIMICView).

For example, to set the default interface to eth1, in the C shell, do:
% setenv MIMIC_DEFAULT_NETDEV eth1

In the Bourne shell, do:
MIMIC_DEFAULT_NETDEV=eth1; export MIMIC_DEFAULT_NETDEV

On Windows, use the System Control Panel to set this environment variable. For more details, see [Microsoft's instructions](#).

20. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address **ADDRESS**
DATE - cannot add address **ADDRESS**
DATE - AddIPAddress failed. Reason=Access is denied.

Cause
On Windows Vista or later this means you don't have access permissions to run MIMIC. Consult the [Windows Installation Instructions](#) section for details.
Action
Enable the user access level correctly as detailed in the [Windows Installation Instructions](#).

21. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - possibly caused by another SNMP agent running
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address **ADDRESS** port **PORT**
DATE - bind: No such file or directory

Cause
On Windows Vista or later this means you are trying to start agents with "duplicate address detection" enabled by default. Consult the [Windows Installation Instructions](#) section for details.
Action
Disable "duplicate address detection" as detailed in the [Windows Installation Instructions](#).

22. ERROR **DATE** - cannot listen on remote management socket

DATE - listen: No error

Cause

On Windows, this likely means you are running a software firewall which is preventing MIMIC from running properly.

Action

Configure your software firewall to allow MIMIC to access the network. See also the [Windows Installation Instructions](#).

23. ERROR **DATE** - cannot bind management socket
DATE - bind: Address already in use

Cause

This message means there is already another process running that uses the MIMIC management port. This is very likely an already-running instance of mimicd, the MIMIC daemon.

Action

Only one instance of the MIMIC daemon can simultaneously be running on a host.

To verify if there is another MIMIC daemon running, do:

```
% ps ax | grep mimic
593 p0 S 0:03 /usr/local/mimic/bin/mimicd
```

If you see something like the last line above, there is another instance of mimicd running that you need to kill.

Otherwise, contact Technical Support on how to find any other programs using this port.

24. ERROR **DATE** - initialization failed
DATE - Cannot read license

ERROR **DATE** - initialization failed
DATE - License expired

ERROR **DATE** - initialization failed
DATE - cannot get license
DATE - Invalid license key

ERROR **DATE** - initialization failed
DATE - cannot get license
DATE - License corrupt

ERROR **DATE** - initialization failed
DATE - Incorrect license version

Cause

The licensing information in the license file mimicd.lic is incorrect.

Action

MIMIC will not run without correct license keys which you can obtain from support@gambitcomm.com.

You can copy/paste the keys when prompted by the installation program. Or, if you have already installed MIMIC, then edit the `config/*.lic` files to paste the correct key (also see [FAQ](#)).

If you still cannot get it working, please send us the contents of the `config/*.lic` files to expedite the process.

25. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address **ADDRESS**
DATE - ioctl SIOCSIFADDR : unknown error

ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address **ADDRESS**
DATE - ioctl SIOCSIFADDR : Permission denied

Cause

There can be various reasons for this error:

- mimicd is not setuid root. The installation script sets the correct permissions. The output of `ls -l mimicd` should be something like `-rwsr-xr-x 1 root staff 1466311 Aug 14 11:24 mimicd`. If the permissions are not as shown, you need to do as root

```
# chown root mimicd
# chmod 4755 mimicd
```
- You are setting an incorrect IP address. Try another address, like 192.9.200.200 .
- If you are running on a Linux with loadable kernel modules, the first access to IP aliasing will result in this error. It should succeed the second time.
- If you are running on an older Linux, you are not running a Linux kernel with the network aliasing and transparent proxy options enabled (`CONFIG_NET_ALIAS=y`, `CONFIG_IP_ALIAS=m` and `CONFIG_IP_TRANSPARENT_PROXY=y`).

Action

MIMIC will not simulate agents if it cannot set the IP address. You need to resolve this error.

26. ERROR **DATE** - agent **NUMBER** cannot start ipalias, continuing...
DATE - cannot open socket.
DATE - cannot set address **ADDRESS**
DATE - ioctl SIOCSIFADDR : Cannot allocate memory

Cause

The Linux kernel configuration limits the number of network interfaces.

Action

Increase the following Linux kernel parameter from a root shell:

```
# sysctl -w net.ipv6.route.max_size=16384
```

27. ERROR: oid.cc:47: assertion failed - constructor failed

Cause

Any assertion failure is fatal, and should be reported to support@gambitcomm.com. Any message with "constructor failed" is likely a lack of virtual memory. You need to increase your swap space as detailed in the OS-specific installation instructions.

Action

Send email to support@gambitcomm.com.

28. ERROR **DATE** - buffer full from ADDRESS to ADDRESS

Cause

The "buffer full" message is displayed when the management application sends too many requests at once. MIMIC cannot service them all, and buffers them (as all real SNMP agents do). The message alerts you when the buffer overflows, and messages are discarded (as all real SNMP agents do, except they do it silently).

Action

This condition can be caused due to 2 reasons:

a) MIMIC is too slow (running on a underpowered machine). If this message occurs occasionally, you can overcome this problem by either putting MIMIC in overdrive by disabling action scripts, and/or increasing the buffer size for each agent.

If you are not using [action scripts](#), you can disable the extra processing on every received request, resulting in significant performance gain. To disable actions, set the MIMIC_DISABLE_ACTION environment variable to any value prior to running MIMIC (if it is running, you must terminate it with File->Terminate from MIMICView).

In the C shell, do:

```
% setenv MIMIC_DISABLE_ACTION 1
```

In the Bourne shell, do:

```
# MIMIC_DISABLE_ACTION=1; export MIMIC_DISABLE_ACTION
```

On Windows, use the System Control Panel to set this environment variable. For more details, see [Microsoft's instructions](#).

To increase the buffer size for each agent, edit the config/mimicd.cfg file and add a line

```
agent_qsize = value
```

where "value" is a number larger than 10 (the default). Try 20.

If this problem persists, you may want to run MIMIC on a more powerful machine.

b) the management application has a performance bug, ie. it sends too many requests simultaneously (as we have seen). A real agent will never alert you to this condition, except that performance suffers, since the app will retransmit the discarded requests.

A common bug in management applications is the issuance of too many simultaneous requests in a "burst". Performance bugs are violations of the performance requirements for managing devices. These are subtle bugs, since their only symptoms are degraded performance, which is hard to measure. MIMIC helps you detect these violations.

An example of this condition is "aggressive retransmission policy", which could trigger this effect: the app is sending a request, which may be delayed. The app times out, and retransmits. If this happens more than a certain number of times consecutively, the buffer overflows. This is independent of the overall rate.

A short protocol analyzer session would verify this: Gambit ships a free (unsupported) protocol analyzer called tcpdump downloadable from <http://www.gambitcomm.com/unsupported>. If you run tcpdump as follows from root:

```
# tcpdump -s 256 -n host agent-IP-address and port 161
```

it will dump all SNMP packets to/from that agent IP address. Run this analyzer until the buffer full error happens. Then send us the output.

An aggressive retransmit policy could be a bug in the application, ie. it will have performance problems interacting with any agent, whether MIMIC or anybody else's.

29. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**

```
DATE - cannot set address ADDRESS
```

```
DATE - socket: Too many open files in system
```

Cause

This message on Linux means that the global file descriptor table is full.

Action

Consult the [Linux Installation Instructions](#) "1000 / 2000 / 5000 / 10000 agent support" section for details.

30. WARN **DATE** - User changed log file

```
DATE - Continued to FILE.....
```

```
WARN DATE - User changed log file
```

DATE - Continued from **FILE**....

Cause

The log file was changed at the specified time.

Action

In order to manage large log files, MIMICView periodically switches the file where the Simulator diagnostic messages are stored. This happens around midnight, and whenever the user saves the log. Use **File->Log->View...** to view the current log file.

31. WARN **DATE** - USM Error: sending report PDU
DATE - unknown engine id: ""

Cause

This warning is an indication that the management application did a discovery of the SNMPv3 engine ID from the agent: in order to detect the engine ID it sends an illegal engine ID, causing the agent to respond with an unknownEngineID REPORT PDU.

Action

None necessary, if your SNMPv3 management application works correctly. Otherwise, you may have to do manual configuration of the engine ID.

32. WARN **DATE** - agent **NUMBER** cannot remove primary alias
DATE - no receiver thread for socket **NUMBER**

WARN **DATE** - cannot clear pollfd **NUMBER**
DATE - T[**x**], socket **NUMBER** not registered

WARN **DATE** - cannot clear pollfd **NUMBER**
DATE - T[**x**], socket **NUMBER** not found in map

WARN **DATE** - NR[**x**]: continuing poll on agents...
DATE - socket **NUMBER** not registered

WARN **DATE** - cannot switch socket **NUMBER** to inactive thread, continuing...
DATE - command buffer overflow

Cause

Any of these messages indicates temporary problems in the protocol dispatcher in the simulator. These messages happen occasionally, under extreme stress of the simulator.

Action

The messages indicate a recovery action by the simulator, and may result in dropped messages to the agents. No action is necessary, unless the messages happen frequently. If they do, please contact support@gambitcomm.com to remedy the problem.

33. ERROR **DATE** - **PROTOCOL** [AGT=**NUMBER**]: cannot start server
DATE - cannot enable IP address for
DATE - cannot start ipalias
DATE - cannot open socket
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address **ADDRESS** port **PORT**
DATE - bind: Address already in use.

Cause

This error means that there is already a service running on the indicated port, eg. there is already a Telnet service bound on port 23 or SSH on port 22. By default, services bind a listening port to address INADDR_ANY, preventing any further bind operations to the same port. On Linux systems, the Telnet service is managed with xinetd. Some services can change the listening address through server configuration.

Action

You have 3 options:

1. start the MIMIC protocol server on a different port. Eg. start the telnet server on port 2423.
2. configure the server to bind to a single IP address rather than the default INADDR_ANY. Eg. for SSH on Linux, edit the ListenAddress entry in /etc/ssh/sshd_config.
3. disable the platform-native Telnet service. On Linux systems, this can be accomplished with /usr/sbin/setup, in the System Services menu, or via chkconfig.

34. WARN **DATE** - agent **NUMBER** cannot stop, continuing...
DATE - agent **NUMBER** cannot remove interface
DATE - address <**ADDRESS**> was created by another application.

Cause

This warning on Windows means that MIMIC refuses to remove an IP alias that it did not create. If MIMIC is the only software creating IP aliases on this system, then the IP alias was probably left over from some previous run of MIMIC, maybe due to a crash.

Action

To remove the IP alias, stop MIMIC, then disable the network interface on which the IP aliases are left over, for example through the System Control Panel. After enabling the network interface again, the IP aliases are removed.

35. ERROR **DATE** - saw this message 100 times, skipping future occurrences...

Cause

The error message was seen too many times. Future occurrences are skipped.

Action

Rather than filling the log with the same error an unlimited number of times, MIMIC limits the number of a particular message to 100 by default. You should fix the problem, rather than ignoring error messages.

To remove this limit, edit the config/mimicd.cfg file and add a line

```
error_occurrences = 0
```

Once you restart MIMIC, the limit is removed.

Recorder

1. WARN **DATE** - setting counter rate **OBJECT** to 0
DATE - sysUptime **SYSUPTIME**, value = **VALUE**

Cause

When an agent has been running for an extended period of time, and a statistics counter is too low to be able to compute a rate, the Recorder sets the rate to 0.

Action

The rate of the counter is set to 0.

2. ERROR **DATE** - failed line **LINE** object **OID**
DATE - unknown object **OID**

```
ERROR DATE - errors from line LINE to LINE
```

Cause

This set of messages means a group of MIB objects were retrieved from the device but they are not in any of the MIBs known to MIMIC. This [FAQ entry](#) has more details.

Action

By default, the objects are not placed in the simulation. To get a complete simulation, compile the missing MIBs and rerun either a live recording or re-record the walk file generated by the recorder.

At MIMIC 5.20, you can specify the `--unknown` command line option to the MIMIC Recorder to attempt to simulate the unknown objects.

3. WARN **DATE** - restart non-tablewalk at **OBJECT**
DATE - GETNEXT violation: got **OID**
DATE - GETNEXT violation: sent **OID**

Cause

This is caused by a faulty agent implementation on the device. The MIMIC Recorder ensures that lexicographical order of retrieved variables is preserved according to the SNMP standard. (Otherwise the retrieval could go on forever.)

Action

The MIMIC Recorder uses efficient, multi-variable, per-PDU queries (to minimize network traffic and to traverse dynamic tables). Some agent implementations do not work correctly with this type of query. When this happens, the recorder automatically changes mode to a simple, single-variable, per-PDU GETNEXT walk for this table.

4. WARN **DATE** - skip object **OBJECT**
DATE - too many errors
DATE - GETNEXT violation: got **OID**
DATE - GETNEXT violation: sent **OID**

Cause

This is caused by a faulty agent implementation on the device. The MIMIC Recorder ensures that lexicographical order of retrieved variables is preserved according to the SNMP standard. (Otherwise the retrieval could go on forever.)

The recorder allows some number of errors, then skips the object to continue the walk.

Action

If you know that the agent does not correctly export a table (e.g. all entries are in reverse lexicographic order), and you still want to record this table, then you can set the MIMIC_REC_NO_GETNEXT environment variable before invoking mimirc. This will disable the GETNEXT violation checking in the recorder.

In the C shell, do:

```
% setenv MIMIC_REC_NO_GETNEXT 1
```

In the Bourne shell, do:

```
# MIMIC_REC_NO_GETNEXT=1; export MIMIC_REC_NO_GETNEXT
```

On Windows, use the System Control Panel to set this environment variable. For more details, see [Microsoft's instructions](#).

5. ERROR **DATE** - walk done for target **TARGET**
DATE - PDU timed out

Cause

This means that the device did not respond to a query from the MIMIC Recorder. This could be due to a congested network or to a busy device. You can use the `-rexmits` and `--timeout` options to cause the recorder to wait longer for responses, although the default settings should already wait long enough.

Action

Use the `--rexmits` and `--timeout` options to wait longer for responses from the device.

6. ERROR **DATE** - discovery phase failed
DATE - No variables retrieved, aborting..

Cause

This means that the device did not respond to any query from the MIMIC Recorder. This could be due to several reasons:

- The device is not accessible
- The community string is not the correct one
- The SNMP agent on the device is using a non-standard port
- The SNMP agent uses some non-standard access control mechanism.

Action

1. First make sure that the device is accessible to the ping command on any operating system.
2. Then make sure you are using the correct read community string on the device. If you don't know the community string, talk to your network administrator.
3. Also, some devices use non-standard port numbers for their SNMP agents.
4. Some devices have a configurable setting for which management stations are allowed to query them. If so, make sure that the system running MIMIC Recorder is in that list.
5. Finally, try any other MIB browser to talk to your device. If that is possible, then the MIMIC Recorder should be able to access it.

7. WARN **DATE** - restart non-tablewalk at **OBJECT**
DATE - PDU timeout, could be due to faulty agent

Cause

This means that the device did not respond to a query from the MIMIC Recorder. This could be due to a congested network or to a busy device. You can use the `--rextmits` and `--timeout` options to cause the recorder to wait longer for responses, although the default settings should already wait long enough.

Another cause could be a faulty agent implementation on the device. The MIMIC Recorder uses efficient, multi-variable, per-PDU queries (to minimize network traffic and to traverse dynamic tables). Some agent implementations do not work correctly with this type of query.

The MIMIC Recorder automatically switches to single-variable GETNEXT traversal to recover from this condition. Most agents will recover, but some crash).

Action

As of MIMIC version 13.00 you can use the `--tablewalk` command line option. For earlier versions of MIMIC you can set the `MIMIC_REC_NO_TABLEWALK` environment variable before invoking `mimicrec` to disable the multi-variable table-walk of the recorder.

In the C shell, do:

```
% setenv MIMIC_REC_NO_TABLEWALK 1
```

In the Bourne shell, do:

```
# MIMIC_REC_NO_TABLEWALK=1; export MIMIC_REC_NO_TABLEWALK
```

On Windows, use the System Control Panel to set this environment variable. If you are using the recorder from the DOS prompt (COMMAND.COM), you can also use

```
DOS> set MIMIC_REC_NO_TABLEWALK=1
```

You will know that this is set correctly if you do not see any messages in the log containing "starting table walk".

8. WARN **DATE** - setting counter rate **OBJECT** to 0
DATE - counter before sysUptime is known

Cause

Counter rates are interpolated from the current time, as indicated by the current value of `sysUptime`. If `sysUptime` is not known, the interpolation cannot be done and the rate is set to 0. Normally, the Recorder expects the value of `sysUptime` somewhere at the beginning of the recording (ie. in the walkfile), so that all following counters can be simulated accurately. The lack of `sysUptime` can be produced by:

- live recording of MIB subtrees (`--root` or `--start`) without the MIB-2 system group
- omitting the MIB-2 (ie. RFC1213-MIB) in the Simulation Wizard
- omitting RFC1213-MIB with `mib2walk` utility

Action

To calculate counter rates correctly, include the MIB-2 system group in your recording (ie. walkfile).

9. WARN **DATE** - retry walk at **OBJECT**
DATE - Got NOSUCHNAME, could be faulty agent

Cause

Some faulty agents do not transition between MIBs, instead returning a NOSUCHNAME error. To detect such a condition, the MIMIC Recorder retries a couple of times with successively lexicographically higher OIDs.

Action

You can set the `MIMIC_REC_NO_ENDMIBRETRY` environment variable before invoking `mimicrec` to disable the end-of-MIB detection heuristics of the recorder.

In the C shell, do:

```
% setenv MIMIC_REC_NO_ENDMIBRETRY 1
```

In the Bourne shell, do:

```
# MIMIC_REC_NO_ENDMIBRETRY=1; export MIMIC_REC_NO_ENDMIBRETRY
```

On Windows, use the System Control Panel to set this environment variable. If you are using the recorder from the DOS prompt (COMMAND.COM), you can also use

```
DOS> set MIMIC_REC_NO_ENDMIBRETRY=1
```

You will know that this is set correctly if you see a message in the log containing `option: no end of MIB retry`.

10. INFO **DATE** - received SNMPv3 REPORT. synchronizing engine params...

Cause

This is an informational message that tells the user that SNMPv3 discovery and time synchronization is happening. This is a normal process of any SNMPv3 session.

Action

Nothing to do.

11. ERROR **DATE** - failed line **LINE** object **OID**
DATE - Oid out of order, previous **OID**, complete **OID**

ERROR **DATE** - errors from line **LINE** to **LINE**

Cause

Some third-party walkfile programs do not store the retrieved messages in lexicographic order, as is the case with GETNEXT or GETBULK traversal. After conversion in this case with the [walkfile converter](#) the retrieved MIB objects and values will be in the correct format, but will still be out of order. Since there are too many pathological cases in this condition, by default the MIMIC Recorder will skip out-of-order objects, resulting in an incomplete simulation.

Action

The first solution is to sort the converted walkfile with any text-processing utility, such as the Unix `sort`.

Else, you can set the `MIMIC_REC_OID_ORDER` environment variable before invoking `mimicrec` to disable the end-of-MIB detection heuristics of the recorder.

In the C shell, do:

```
% setenv MIMIC_REC_OID_ORDER 0
```

In the Bourne shell, do:

```
# MIMIC_REC_OID_ORDER=0; export MIMIC_REC_OID_ORDER
```

On Windows, use the System Control Panel to set this environment variable. If you are using the recorder from the DOS prompt (COMMAND.COM), you can also use

```
DOS> set MIMIC_REC_OID_ORDER=0
```

You will know that this is set correctly if you see a message in the log containing option: `don't skip out of order OIDs`.

This will still not guarantee a complete simulation, since you are forcing the MIMIC Recorder to handle the pathological condition. Depending on how bad the order is, the simulation may still be not as good as it would be in sorted order.

Compiler

1. WARN **DATE** - No import statement for **OBJECT**. Assuming **MIB**

Cause

The MIB file uses an object that has not been imported from a known MIB.

Action

The compiler checks some well-known MIB files to declare this object.

2. WARN **DATE** - Cannot handle type **TYPE** for **OBJECT**

Cause

Even though the type is imported, the compiler could not use it.

Action

The object that uses this type cannot be simulated. You can bypass this limitation by using the `config/mimiccom.types` file.

3. WARN **DATE** - Ignoring subtype data
DATE - Too many (2) alternates in subtype for **OBJECT**

Cause

The compiler cannot handle more than one alternate range or size specification for the object.

Action

The range or size specification is ignored.

4. WARN **DATE** - Ignoring import statement
DATE - Imported type **TYPE** not found in **TYPEFILE**

WARN **DATE** - Ignoring import statement

DATE - Unable to get path of **MIB** for type **TYPE**

Cause

The specified type could not be imported from the specified MIB. If it is one of the well-known types, this condition is ignored and the compile continues.

Action

The basic SNMP types Counter64, Counter32, etc are built-in. You can ignore this warning.

5. WARNING: Table **TABLE** does NOT have an INDEX clause

Cause

The specified table does not have an INDEX clause.

Action

The table needs an INDEX clause to correctly simulate indices.

MIMICView

1. Application initialization failed: no display name and no \$DISPLAY environment variable

Cause

MIMICView on Linux and Solaris is a X Window System application, and as such needs to have a `DISPLAY` environment variable set to specify the display location.

Action

Set the `DISPLAY` environment variable to specify where to display. Eg. to display on the local display:

In the C shell, do:

```
% setenv DISPLAY :0.0
```

In the Bourne shell, do:

```
# DISPLAY=:0.0; export DISPLAY
```

If you are getting this error from the MIMICShell, and want to run the shell without a GUI, then specify the `--nogui` command line option.

2. Application initialization failed: couldn't connect to display "HOST:0"

Cause

MIMICView on Linux and Solaris is a X Window System application. The display specified by the `DISPLAY` environment variable is not running an X Window Server.

Action

Make sure to set the `DISPLAY` environment variable to an X Server display.

3. Xlib: connection to "HOST:0.0" refused by server

Xlib: No protocol specified

Application initialization failed: couldn't connect to display "HOST:0"

Cause

MIMICView on Linux and Solaris is a X Window System application. The `DISPLAY` environment variable is set to specify an X Server display that is not accepting connections from your host.

Action

Use the `xhost` application on the X Window Server to accept connections from your host.

4. ERROR **DATE** - initialization failed

DATE - Cannot get serial number

Cause

The serial number of the system cannot be inferred. This is likely because the network interface is down.

Action

MIMIC will not run without a working network interface. Contact support@gambitcomm.com to solve this problem.

[<< Previous Section](#) [Next Section >>](#)



MIMIC Frequently Asked Questions

Last updated Tue Mar 24 10:29:12 EDT 2020

Save the latest [FAQ page](#) into your MIMIC installation help/ directory to make it an integral part of your online documentation.

Table of Contents

General

1. Q. What is a "gambit"?
2. Q. How can I use MIMIC for regression testing?
3. Q. What are the hardware considerations for running MIMIC?
4. Q. Why does MIMIC run slow sometimes?
5. Q. Can I run MIMIC on Windows 95 / 98 / Me / NT?
6. Q. Do you have any SNMP tools for Windows?
7. Q. How do I upgrade to a newer release of MIMIC?
8. Q. After upgrading to a newer version of MIMIC we get "mimic session open: MIMIC daemon protocol is not x.xx.". Why?
9. Q. How do you customize and program MIMIC?
10. Q. What books on Tcl/Tk can you recommend? How about sample scripts?
11. Q. What is the fastest API to control MIMIC?
12. Q. How can I trace PDU exchanges between my management application and MIMIC?
13. Q. What is the best way of reporting problems with MIMIC?
14. Q. If I want to simulate 100 routers with multiple interfaces, what size of MIMIC do I need?
15. Q. How do I simulate routers with multiple interfaces?
16. Q. Is there a way to modify the configuration of MIMIC to use another drive for its use?
17. Q. How do I share simulations in my private area with other users?
18. Q. How do I export a MIMIC device?
19. Q. How do I clean the MIMIC database?
20. Q. How do I backup the MIMIC database?
21. Q. What do I need to do to run MIMIC on Red Hat Enterprise Linux?
22. Q. How do I get MIMIC to work with SuSE Linux ?
23. Q. Will there be a port conflict if I install the MIMIC software on a machine with HPOV Network Node Manager 5.01 installed?
24. Q. Why does HPOV Network Node Manager not discover the MIMIC agents?
25. Q. How can I use MIMIC in a Client/Server configuration ?
26. Q. Can you run MIMIC and a Web Server on the same machine?
27. Q. How do I simulate a web server for each of my simulated agents?
28. Q. How do I apply new license keys to the installed software?
29. Q. What environment variables are available?
30. Q. How can I use Microsoft Operations Manager to collect SYSLOG messages generated by MIMIC?
31. Q. Where do I download your recommended versions of Linux?
32. Q. How does MIMIC Update Wizard work through a SOCKS/SSH proxy?
33. Q. What is the best way to create a new simulation of a device?

Recorder

1. Q. How do I efficiently record and simulate an exact copy of a large network?
2. Q. How do I discover and record all devices in a subnet?
3. Q. When I record a device, I get a "PDU timed out" error message after it has retrieved many variables. Why?
4. Q. When I record a device, I get a "GETNEXT violation" error message after it has retrieved many variables. Why?
5. Q. How do I re-record a device when it is no longer there?
6. Q. When recordings are taken of MIB values where is this data stored?
7. Q. Is there any way to change the name of the agent?
8. Q. Why are my enterprise MIBs not simulated by the MIMIC Recorder?
9. Q. Why does the MIMIC Recorder not handle unknown objects?
10. Q. In general, how many errors would be too many as a percentage?
11. Q. How do I remove unwanted simulations?

12. Q. Can I create a simulation from an external third-party SNMP walk utility?

• Simulator

1. Q. I cannot start agents in MIMIC. I get errors in the log when starting an agent instance. Why?
2. Q. I have started agents in MIMIC, but I cannot ping them from my management station. Why?
3. Q. How do I access my IPv6 agents from a remote system?
4. Q. I can ping my agents, but they don't respond to requests. Why?
5. Q. What is the fastest simulation I can run? Why is it not the default?
6. Q. When I run a simulation, I see some diagnostic messages in the log window. What do they mean?
7. Q. How can we truncate the log when it gets too large?
8. Q. Even though a counter simulation rate is greater than 0, I am not seeing the counters incrementing. Why?
9. Q. How can I force a Counter wrap for a MIB object?
10. Q. How can I reset a counter?
11. Q. I am seeing the message "buffer full from ADDRESS to ADDRESS" in the error log. What does it mean?
12. Q. Isn't it true that accessing instance 0 on a table is an error? MIMIC executes it without an error. Why?
13. Q. How do I verify whether a trap is generated?
14. Q. Does MIMIC support dynamic row creation?
15. Q. Does MIMIC support RowStatus semantics?
16. Q. Does MIMIC support multiple communities per IP address?
17. Q. What is source-address indexing?
18. Q. How can I create many interfaces on a host?
19. Q. When I set the multicast, broadcast ethernet counters, this won't affect the traffic, right?
20. Q. How do I add a MIB to a simulation?
21. Q. Why can't I generate a trap for an object?
22. Q. How do I send the current value of sysUpTime with a trap?
23. Q. How does MIMIC determine what version of the TRAP PDU is generated?
24. Q. Is there any way of setting up more than one agent to send traps at any one time?
25. Q. How can I send traps from IP aliases of my agent?
26. Q. How can I send traps from different source ports of my agent?
27. Q. How do I generate a series of traps upon an SNMP SET?
28. Q. How can I interleave agents among multiple network interfaces?
29. Q. Why is my throughput intermittently slow on a dual NIC Linux system?
30. Q. What SNMPv3 features are supported by MIMIC?
31. Q. Why does it take so long to start SNMPv3 agents?
32. Q. How can I get MIMICShell scripts to terminate at the end of execution?
33. Q. How can I iterate through configured agents in MIMICShell?
34. Q. What is the difference between an action script and a MIMICShell script?
35. Q. How can I debug my action script?
36. Q. How do I maintain state in my action script?
37. Q. How can I assign action scripts via another script?
38. Q. Can I interact with the user in my action script?
39. Q. Is there a way to start up a timer script for each agent?
40. Q. How can I control the order of starting agents?
41. Q. Why do GET statistics not increase when I use the Get Value dialog?
42. Q. Why can I not see the value of an index object in the Get Value dialog?
43. Q. How do I simulate a binary OCTET STRING value?
44. Q. How do I simulate an object of type BITS?
45. Q. Do you have any Tcl scripts to manipulate BITSTRING values?
46. Q. How do I simulate a table with an OCTET STRING INDEX?
47. Q. How do I simulate a sparse table?
48. Q. How do I simulate a numeric object with a random value?
49. Q. How do I simulate a step function for a Counter object?
50. Q. How can I get MIMIC to accept SETs on an object with different type?
51. Q. Is it better to access a MIB object by name or OID?
52. Q. What is the most efficient way to access many variables in the MIMIC value space?
53. Q. Why do I see multiple "mimicd" processes on Linux?
54. Q. How can I do dynamic DNS updates for my MIMIC agents?
55. Q. How do I start MIMIC upon system boot?
56. Q. Can MIMIC simulate SNMP looping?

GUI

1. Q. How do I change the fonts or colors of the MIMICView user interface?
2. Q. How do I turn off the tooltip popups in MIMICView?
3. Q. When I start mimicview, the log window does not appear. Why?
4. Q. Why do I sometimes get "Cannot connect to the MIMIC daemon." when I start mimicview?
5. Q. Even after Terminating Mimic, I see a couple wish80 processes in the system. They just don't quit !
6. Q. How do I see MIB object details in the Value Browser?
7. Q. How do I list the SNMPv1 traps in the MIB browser?
8. Q. Why are you using different syntax for trap-OIDs and variable-OIDs ?
9. Q. Is there a way to assign icons to my devices?
10. Q. MIMIC ran pretty slowly after 3 scripts and about 20 devices. Do you have any idea?
11. Q. How does one turn off scripts running against an agent instance?
12. Q. Why is the "Device" button greyed out in the "Agent Configuration" dialog?
13. Q. Why do I get "Font specified in font.properties not found" messages when I run the Java-based tools?
14. Q. How do I access from Windows the MIMICview GUI running on Linux?

Compiler

1. Q. How do I change my simulation to add missing tables/objects?
2. Q. Can I redefine a trap to send extra variables?
3. Q. How to remove unwanted MIBs?
4. Q. Why does the compiler emit errors when others don't?
5. Q. Where can I get the Extreme Networks MIBs?
6. Q. Where can I get the Foundry Networks MIBs?
7. Q. Where can I get the Alteon MIBs?
8. Q. Where can I get the Juniper MIBs?
9. Q. Where can I get the Avaya MIBs?
10. Q. Where can I get the Siemens MIBs?
11. Q. Where can I get the Brocade MIBs?
12. Q. Where can I get the Netopia MIBs?
13. Q. Where can I get the WIMAX MIBs?

General

Q. What is a "gambit"?

A. A "gambit" is a term from the game of chess, a risky opening move with high potential return. Also see [Merriam-Webster Dictionary](#):

Main Entry: gambit
Pronunciation: 'gam-b&t
Function: noun
Etymology: Italian gambetto, literally, act of tripping someone, from gamba leg, from Late Latin gamba, camba, from Greek kampE bend; probably akin to Gothic hamfs maimed, Lithuanian kampas corner Date: 1656
1 : a chess opening in which a player risks one or more minor pieces to gain an advantage in position
2 a (1) : a remark intended to start a conversation or make a telling point
(2) : TOPIC
b : a calculated move : STRATAGEM

Q. How can I use MIMIC for regression testing?

A. MIMIC is particularly well suited for regression testing of network management applications. Here are some of the things being done with MIMIC:

With a constant simulation, you can predict the values of scalars as well as counters. Since counter simulations in MIMIC are rate based, you know that if an object has a rate of 10/sec, at time = 30 sec the value will be 300.

But, since time advances, it is normally impossible to guarantee that a query is responded to at a precise time. With MIMIC, you can use the [Agent ->Pause](#) command to stop time for an agent. This way you can guarantee your values at any point in time you want.

You can use the [Agent ->Pause](#) functionality further to simulate behaviour instantaneously at any time, whether 15 seconds, 15 minutes or 15 days after booting. This is useful to test counter value wraps.

To test service-level-management data collection over long periods of time, you can combine the pause functionality with [mimicsh](#) scripting to advance time at a rapid pace (at least 10 times faster). This way you can get months' worth of data collection done in days.

Since counter rates can be changed at any point during the simulation, you can create pathological scenarios at will. For example, you can change the traffic counters with a high rate to represent peak usage scenarios, or increase the rate of error counters to see how fast and reliably an alarming threshold is detected.

Q. What are the hardware considerations for running MIMIC?

> In your support page, you give the minimum hardware configuration and
> the preferred hardware configuration, but the number of agents simulated
> doesn't seem to be taken into account. I mean is the hardware configuration
> the same for all the versions of MIMIC (MIMIC Single, MIMIC Lite, ...,
> MIMIC Global) which simulate different numbers of agents?

A. The preferred configuration listed on the web page is for a typical case. Obviously, the more demanding the simulation, the more hardware you have to throw at it. The main thing to realize is that the hardware requirements for MIMIC can be viewed as satisfying an equation, eg.

performance (management side) <= performance (agent side)

meaning that the performance of the agent side (MIMIC) has to be at least as good as that needed by the management side. The performance requirements are thus driven in large part by the management application. The second point is that the equation is not controlled by a single variable (eg. "requests per second"). There are many variables which determine the exact demands on the simulation:

- the number of agents
- the set of MIBs for each agent
- the complexity of the simulations for the different MIB objects
- the trap generation rates
- the number of management threads (eg. pollers)
- the poll rates (average, sustained, peak)
- the make-up of the requests (single-variable vs. multi-variable vs. bulk)

This is just a partial list, but gives you an idea of the considerations. Ultimately, there is no generic answer and each customer has unique performance requirements. We can help you to determine these requirements through empirical evaluation. Your requirements may change over time, so your hardware solution should accommodate this change (more CPUs, more memory, more network cards). MIMIC is designed to take advantage of all the hardware you throw at it.

MIMIC supports up to 100,000 agents on one workstation. The main concern is the performance for a fully loaded workstation. You want at least hundreds of PDUs per second to make a simulation viable.

MIMIC performs significantly better on some operating systems (OS) than others. Our standard performance test reports should help you select an OS whose functions are aligned for the requirements that MIMIC imposes.

For MIMIC, performance is primarily governed by the amount of physical memory (RAM). The memory requirements depend on the simulations you are going to run. Obviously, a high-end router simulation with hundreds of interfaces, RMON tables, etc. is going to take more memory than the simulation of an end system.

As a ball-park estimate, we like to see at least 1MB of physical RAM per simulated agent, e.g., a 100 agent scenario should run fine on a 128MB system (depending on how much memory is used by the OS and other processes). For better performance (less swapping), 2MB per simulation is recommended.

After version 4.30, MIMIC has a new feature - memory optimization. That means more agents' MIB data needs less memory than before. Agents with identical simulation will only require one copy of data in memory. For example, in the common case if 100,000 agents are identical, only a couple of MB of RAM is needed. However, if 5000 agents are running the same simulation, and 5,000 agents are each different, then 5GB will be recommended.

You can more accurately measure this by running a simulation configuration, and checking on memory usage before and after starting the desired agent simulations. Notice that MIMIC uses memory on demand, so you should measure the memory after doing a walk of the desired tables (or a complete MIB walk). Eg. on Windows use the Windows Task Manager to check "Memory Usage", and on Unix use the "top" utility.

The memory usage by MIMIC is approximately the same for all platforms.

After version 8.10, MIMIC can run on 64-bit systems in 64-bit mode, overcoming the 4GB virtual memory limitation of 32-bit mode. If you foresee a large simulation going beyond 4GB, you will want to run in 64-bit mode.

The CPU is of secondary importance. Most modern processors (e.g., Intel or AMD) are adequate. MIMIC works with multi-processor systems, since it is a multi-threaded, distributed application. Agent thread processing will be distributed across multiple CPUs. From our internal experience, we have run 20,000 agents on dual and quad-CPU Intel or AMD systems, and 100,000 on 16-CPU systems.

The final bottleneck would be the network pipe to your agents. 100Mb Ethernet is adequate for low-volume traffic, 1Gb or faster is better for more demanding applications. MIMIC works with multiple network adapters on your system, so you can talk to the simulations over separate network pipes. MIMIC works with the OS-native protocol stacks, so that all network interface cards that your OS supports can be used. You can even run MIMIC on the Internet (eg. to a public cloud) over VPNs.

Q. Why does MIMIC run slow sometimes?

When we encounter this question, often the answer is "because other software is hogging the system". All bets are off if MIMIC has to compete on the system with other software.

If you want to run MIMIC with predictable performance, then you have to cut down on interference from other software running on the system. This not only includes the bundled services we have identified for the supported OS platforms, ie. [Windows Services](#) and [Linux daemons](#), but also applies to any other third-party programs that run concurrently, including Virus checkers, Automatic Software Updates such as [Windows Automatic Updates](#), or any other background processes.

The first step to check for interference is always a process listing, such as that displayed by [Windows Task Manager](#), and [the top utility on Unix](#).

The tricky part is that interference can be sporadic, short-lived and quite hidden, thus sometimes hard to track down. For that reason, if you care about MIMIC performance, we highly recommend running MIMIC on a dedicated system with minimal other extraneous software.

Q. Can I run MIMIC on Windows 95 / 98 / Me / NT?

A. These old versions of Windows are not supported any more.

We recommend running MIMIC on the latest supported Windows version, or Linux platforms as detailed on our "Supported Platforms" [page](#).

Q. Do you have any SNMP tools for Windows?

A. You can download an unsupported binary distribution of the [net-snmp \(was UCD SNMP\) toolset](#) from [our website](#).

Q. How do I upgrade to a newer release of MIMIC?

A. The MIMIC installation instructions contain upgrade information:

1. make sure you terminate MIMIC (use File->Terminate) before running the installation program.
2. Do NOT uninstall the older release (this way you can always fallback in case of problems).
3. Install this release in a different directory from the older release. All your MIMIC data files will be fetched out of your existing private area.
4. For minor release upgrades (eg. from 5.00 to 5.10) you only need to supply your existing license keys to the install script and start the new MIMIC.
5. For major release upgrades (eg. from 4.xx to 5.xx) you may need to request new license keys. Send your HOSTID to support@gambitcomm.com.

Q. After upgrading to a newer version of MIMIC we get "mimic session open: MIMIC daemon protocol is not x.xx.". Why?

A. MIMIC is a distributed/client-server application, and you still have the old daemon process running. You need to terminate the old version of the MIMIC Simulator before you run the new version. You have a couple of choices:

1) kill the MIMIC daemon. On Linux do something similar to

```
# ps ax | grep mimicd
```

and kill that process. On Windows, kill the mimicd.exe from the task manager.

OR

2) run the old mimicview. Do "File --> Terminate".

When you restart the new mimicview, a log window should appear with the log output of the new MIMIC simulator daemon.

Q. How do you customize and program MIMIC?

> Also, how is MIMIC programmed? What programming language? Tcl?
> Scripts? What type of development environment? How customizable is it?

MIMIC is highly customizable. In the basic simulation, you can export any MIB data you want, by adding new MIBs to the already compiled 2500+ MIBs (with [MIB Wizard](#)), recording real device data (with [Simulation -> Record](#) or [Snapshot Wizard](#) or [Discovery Wizard](#)), or defining simulations (with [Simulation Wizard](#)).

To create advanced simulations, you can use several levels of programming:

1. write MIMICshell scripts, which run externally to the MIMIC Simulator in a MIMICshell client and control the simulation (eg. start/stop agents, change values, generate traps, etc). These scripts are written in a superset of Tcl (with MIMIC command extensions, as documented in [Simulator Guide](#), [MIMICShell reference section](#)).
2. write [action scripts](#) which are executed inside the MIMIC daemon on particular events, eg. when an SNMP request is received for a MIB object. The action script is written in a superset of Tcl (with the MIMIC command extensions, as documented in [Simulator Guide](#), [Agent -> Actions section](#)).
3. write timer action scripts, which are executed periodically by the Simulator. Again, timer scripts are written in the same language as 1) and 2). Online help is in the [Simulator Guide](#), [Run Menu section](#).
4. write trap action scripts, which are executed when a trap is generated. Again, trap scripts are written in the same language as 1), 2) and 3). Online help is in the [Simulator Guide](#), [Agent -> Generate Traps](#) section.
5. as of MIMIC 4.40, you can use a Java API to control MIMIC in place of/ in addition to 1). This is documented in the [Java API Guide](#).
6. as of MIMIC 5.10, you can use a Perl API to control MIMIC in place of/ in addition to 1). This is documented in the [Perl API Guide](#).
7. as of MIMIC 5.40, you can use a C++ API to control MIMIC in place of/ in addition to 1), 2), 3) and 4). This is documented in the [C++ API Guide](#). In particular, actions can now be written in C++ for best performance.
8. as of MIMIC 9.00, you can use a Python API to control MIMIC in place of/ in addition to 1). This is documented in the [Python API Guide](#). The Python API is an option package installable with the [Update Wizard](#).
9. as of MIMIC 16.10, you can use a PHP API to control MIMIC in place of/ in addition to 1). This is documented in the [PHP API Guide](#). The PHP API is an option package installable with the [Update Wizard](#).
10. as of MIMIC 18.10, you can use a Javascript API to control MIMIC in place of/ in addition to 1). This is documented in the [Javascript API Guide](#).
11. there is a proprietary simulation language (as documented in [Compiler Guide](#), [MIMIC Simulation Language Reference](#) section), which you can use to control the simulation. This is needed only in rare cases, where you want the most speed and control.

A commercial TCL development environment is available from [ActiveState](#), called Tcl Dev Kit. It gives you a symbolic debugger environment, etc. You can use it to debug scripts written for 1).

Q. What books on Tcl/Tk can you recommend? How about sample scripts?

A. There are plenty of book lists and recommendations on Tcl/Tk. See the References section on our [Support page](#).

MIMIC provides a set of simple, illustrative Tcl/Tk-based MIMICshell and action scripts in the `scripts/` subdirectory.

Q. What is the fastest API to control MIMIC?

A. As detailed [above](#) MIMIC can be controlled from various programming languages. Each of them has different performance characteristics (eg. interpreted vs. compiled). This section attempts to show how they stack up to each other.

Oftentimes your choice of language will be determined by factors other than just speed, such as

- knowledge of the language
- efficiency and convenience of other services, eg. graphics, sorting, etc
- policy/politics

but if you have a choice, it could be influenced by speed, specially if controlling MIMIC means issuing many commands (eg. frequently configuring thousands of agents, or millions of MIB values) within time constraints.

The control of MIMIC is a `remote procedure call (RPC)` mechanism into the MIMIC server. MIMIC commands are performed via RPC calls. The performance of an RPC call is impacted by:

```
time(RPC call) = time(language overhead to package and send request) +
                time (transport of request) +
                time (MIMIC server dispatch) +
                time (transport of response) +
                time (language overhead to receive and parse response)
```

The first thing to realize is that the MIMIC server dispatch time will be the same for all language bindings. For this test, this can be considered a constant for each distinct RPC call.

Second, the transport mechanism for the MIMIC RPC calls is TCP for remote calls (MIMIC client and MIMIC server on distinct machines), and a faster local transport mechanism for local calls (within a machine), such as Unix domain sockets for Unix, and pipes for Windows. Some languages, such as Java on Windows, do not support the local transport mechanism, thus local transport will be slower than on languages that do.

Third, the difference in performance between the language bindings will be reflected primarily in the `time(language overhead to package request)` and `time (language overhead to parse response)` numbers, thus our performance tests did focus on calls where the other 3 numbers do not dominate the equation (such as large requests, for which the difference will be much smaller).

Fourth, within a language you may find small fluctuations for different versions of compilers or interpreters. We tested with a recent version for each language.

This matrix contains numbers (calls per second) for some representative RPC calls for the different languages in local and remote mode for a representative Fedora 19 Linux (quad-core Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz) and Windows 8.1 (quad-core Intel(R) Core(TM) i5-2400 @ 3.10GHz) system:

Linux						Windows					
Mode	C++	Tcl	Java (1)	Perl	Python	Mode	C++	Tcl	Java (2)	Perl (2) (3)	Python (2)
mimic get max						mimic get max					
local	69013	19790	55866	15202	18650	local	62735	21815	35753	12850	16548
remote (4)	40453	14888	33036	12492	14769	remote	46041	11111	30618	12982	16548
mimic agent get host						mimic agent get host					
local	91408	19670	64725	24777	21964	local	73584	21515	39262	18443	15939
remote	50865	16639	40064	18758	16488	remote	52029	10441	39746	18129	15939
mimic value get						mimic value get					
local	62933	16958	53022	21959	22173	local	52882	17373	30479	16038	11305
remote	38168	13569	31636	16787	16920	remote	37864	9916	30331	16160	11305

NOTES:

1. Java has Unix domain transport support on Unix.
2. Java, Perl and Python have no local mode transport on Windows.
3. On Windows, the numbers are for ActivePerl. The numbers for Cygwin Perl were consistently worse.
4. All remote mode calls were made within each system.

As expected, the clear winner is C++, in local mode (which is the most common form of control of MIMIC). The other compiled language, Java, is about 20% slower than C++. The interpreted languages are significantly slower than C++ in both modes, less than half that of the compiled languages.

Q. How can I trace PDU exchanges between my management application and MIMIC?

A. Tcpdump is a free public domain protocol analyzer. Most Linux distributions include this valuable diagnostic tool.

Wireshark (formerly ethereal) is a great free public domain protocol analyzer. It decodes SNMPv3. Download it for your favorite platform from [their website http://www.wireshark.org/](http://www.wireshark.org/).

MIMIC allows you to trace all received PDU requests for any agent instance with the `mimic agent set trace` command in [the MIMICShell](#). The logged PDUs will be displayed in the MIMIC [log window](#). This way you can see which MIB objects are being accessed. This feature is designed to complement the protocol analyzer output (OIDs, values) by displaying the MIB Object name.

As of MIMIC 9.00, all objects and values are displayed in the log, to enable diagnosis of encrypted SNMPv3 requests.

NOTE: tracing slows down the simulator considerably, and should only be used temporarily to track down a specific problem. To do long-term protocol analysis, one of the external protocol analyzers is recommended, preferably running on a different computer than the MIMIC system.

Q. What is the best way of reporting problems in MIMIC?

A. After MIMIC 5.10 the [Diagnostic Wizard](#) presents a convenient interface to report problems.

Otherwise, the fastest way of resolving problems is by sending e-mail to support@gambitcomm.com with a brief description of the problem, and supporting information, such as excerpts from the log window that show the problem.

If there is a workaround, we will let you know as soon as possible. If the problem requires a fix, we will open a trouble ticket and schedule it for an upcoming release. All customers are notified of new releases as soon as they become available through the [MIMIC User Group mailing list](#).

If you have a large supporting file from one of the tools (core file, log file, walk file), please don't email it yet since our mail server has limited resources (bandwidth and space). Tell us about the problem first, and we will ask you for the core file.

Q. If I want to simulate 100 routers with multiple interfaces, what size of MIMIC do I need?

```
If I wanted a 100 node network, I need a 100 agent license. Is this correct? How does a router fit in? For example, if I have a router with 10 interfaces -- is that one agent or 10? I suspect one, but I want to ensure my understanding so we know what were buying.
```

A. Strictly speaking, MIMIC was until 3.33 licensed by number of IP addresses you want to talk to via SNMP (regardless of how many agents the real device implements). Thus, if you want to simulate a device with multiple management interfaces, then you needed to count those towards your license requirements. If the interfaces were not management interfaces (ie. your management application uses a single management interface, read, IP address, to manage the device), then you only needed 1.

After MIMIC 4.00, agents can have IP Aliases, ie. alternate IP addresses that the management application can talk to the agent. These IP aliases do not count toward the number of agents in your license. Thus, you can have many more agents configured with aliases than in earlier releases. There are OS platform limitations of how many IP addresses you can configure. On Windows Vista and later the limit is 20,000; on Linux this is currently 100,000.

After MIMIC 4.00, the answer is: "a 100 agent license supports 100 agents, no matter how many interfaces each has (upto a limit on the OS platform you are running MIMIC on)".

Q. How do I simulate routers with multiple interfaces?

```
> If I have defined routers with Mib IP interfaces which are not shown
> in the MIMIC map, how would I go about make these IPs available? That is
> having a simulated router with the IP 192.9.200.180 and the router's MIB
> contains 120 interfaces with addresses ranging from 155.133.1.1 to
> 155.133.1.24 with a mask of 255.255.255.254 each - shouldn't I use
> ifDiag.exe in order to create these IP addresses? or there another way to
> make these IPs available through mimic TCL programmatically?
```

A. You can use the IP aliases feature (available after MIMIC 4.00) when you need to associate additional IP Addresses with a simulation that can be pinged and also answer SNMP requests.

The IP aliases can be manipulated using the 'mimic agent ipalias' commands. The list of supported commands are :

```
Usage:
mimic agent ipalias list
mimic agent ipalias add ipaddress, port, [mask], [interface]
mimic agent ipalias delete ipaddress, port
mimic agent ipalias start ipaddress, port
mimic agent ipalias stop ipaddress, port
```

You can use the 'add' subcommand to add new IP aliases and 'delete' subcommand to remove any existing IP aliases. Also use 'start'/'stop' to actually activate or deactivate the same.

Q. Is there a way to modify the configuration of MIMIC to use another drive for its use?

```
The problem: the drive that MIMIC is using has become full due to MIMIC usage. Is there a way to modify the configuration of MIMIC to use another drive for its use?
This is running on an NT Server with a drive C and D. Mimic has been using drive C:
```

A. You can set the environment variable MIMIC_PRIV_DIR to point to a location on another drive to do this. Determine what directory is being used currently by MIMIC for storing your [private data](#). Copy this over to a different drive and then point MIMIC_PRIV_DIR to this location. Restart MIMIC fresh and all subsequent

data should be stored on the new drive.

First, terminate MIMIC (in MIMICView, use File -> Terminate).

- On Unix, in the Cshell do this before invoking MIMICView:
`% setenv MIMIC_PRIV_DIR new-path-to-your-private-area`
- On Unix, in the Bourne shell do this before invoking MIMICView:
`# MIMIC_PRIV_DIR=new-path-to-your-private-area; export MIMIC_PRIV_DIR`
- On Windows, use the System Control Panel to set MIMIC_PRIV_DIR to the new path.

NOTE: You may have to modify your saved lab configurations to remove knowledge of your old private directory. You will know this if your simulations don't start any more and the agents are greyed out in the MIMICView front panel.

If this is the case, you can fix it in MIMICView, by re-configuring each agent again (just use [Edit->Configure->All...](#), click Apply). Or you can edit the config/*.cfg files in your private directory with your favorite text editor, search for lines beginning with `priv_dir`, and remove them.

Q. How do I share simulations in my private area with other users?

```
I have recorded agents and have placed the results in my
"private file space". How do I move those agents to the
"shared file space"
```

A. Any private area can be "shared" with other users of MIMIC. In effect, you can have multiple private areas, and choose among them when using MIMIC. By default, MIMIC looks at the `mimic/` subdirectory in your home directory. You can override this with the `MIMIC_PRIV_DIR` environment variable.

Thus if you setup some simulations in your private area, you can move it anywhere, and tell other users to set their `MIMIC_PRIV_DIR` to that directory path.

Moving the private data into the MIMIC shared area has these downsides compared to the above solution:

- 1) you can have only one shared area: the MIMIC installation directory.
- 2) it's tricky - you need to copy the `data/sim/*` directories, any lab configuration files (`config/agents/*.cfg`) and update `config/dev.cfg`.
- 3) it's not upgradeable out of the box - when you upgrade to your next version of MIMIC, you'll have to perform the same manual steps.

Thus I recommend the first solution, especially if the other users only want read access to the simulations.

Q. How do I export a MIMIC device?

```
> I have recorded an agent and created a new device. I now want to
> "export" this device to the other Mimics
> They are on different PCs and locations in the company.
> Is it possible?
```

A. Yes this is possible. From MIMIC version 10.30 on you can use the [Update Wizard](#) to install an optional patch to export/import simulations. For earlier versions of MIMIC, read on.

Since a device has been designed to be as self-contained as possible, usually consisting only of a simulation directory and some MIBs, you will be required to copy the following directories from the source computer's MIMIC private area to your destination MIMIC private area. (The private area of your system will be displayed on top title bar of the MIMICView GUI application.)

- 1) `/data/sim/<sim-name>/` (each simulation will have one directory, so copy the desired simulation);
- 2) if the simulation also requires some additional MIBs that you imported/compiled in your private area, then copy `/data/mibs/*` ;
- 3) rarely (and only if you use the most advanced features), the device simulation invokes action scripts located outside of the simulation directory. If so, those also need to be copied.

After successful transfer, use the MIMICView Simulation->Devices dialog on your destination system and verify that the new simulation shows up in the list.

NOTE: if you are copying from Unix to Windows, you need to do CRLF translation for all ASCII files. This can be done in a variety of ways, eg.:

- **FTP in ASCII mode**
- **use the `unix2dos` utility**
`find . -type f -exec unix2dos {} \;`

Q. How do I clean the MIMIC database?

A. MIMIC uses two different areas to maintain its data.

The first is the shared install area i.e. the directory where the installation is done. This is fresh for each version installed. The only MIMIC component that will change this area after installation is the [Update Wizard](#) to install optional components.

The second is the [private data area](#), where the additional simulations and MIBs are stored on a per user basis. You can find the private area by using `File->Open`, and then hitting the `Browse` button to bring up the dialog where it is listed above the buttons. So to start fresh, use `File->Terminate`, rename the private area directory and restart MIMIC.

Q. How do I backup/source-code-control the MIMIC database?

```
Is there a recommended method to save off some of the simulations
in some kind of source control? Like which files/etc?
```

A. As the [previous topic](#) explains, the MIMIC shared area does not need to be backed up, because all changes are normally saved in the [private data area](#) for each MIMIC user. This is normally in a subdirectory called `mimic` in the user's home directory. If your backup schedule backs up user home directories, your MIMIC data is already backed up. Otherwise, we recommend to just tar/gzip or Winzip the entire private data area of each MIMIC user.

As for source-code control, all data files in MIMIC are ASCII files, thus can easily be source-code controlled (you cannot easily diff binary files). The level of source-code control depends on the amount of changes to the data. For example, if your data rarely changes, you can use a backup schedule to maintain the changes. But, if you are developing and changing a lot of scripts, you want to source-code control those. If you are changing simulation data, the easiest process is to modify (and source-code control) the walkfiles used to generate the simulations (in effect, the walkfile becomes the source-code for the simulations).

Q. What do I need to do to run MIMIC on Red Hat Enterprise Linux?

A. This version of Linux uses newer libraries by default. If you run MIMIC on the default system, you'll encounter this error:

```
error while loading shared libraries:
libstdc++-libc6.2-2.so.3: cannot open shared object file: No such
file or directory
```

To fix this, you need to install several optional RPMs to fulfill the dependency on specific versions of the system libraries.

The solution is to select "Legacy Software Development" during install, or run the RPM package manager after installation and select the same option. The individual RPMs that are selected are:

- compat-glibc
- compat-libstdc++
- compat-gcc
- compat-libstdc++-devel
- compat-gcc-c++

Q. How do I get MIMIC to work with SuSE Linux ?

A. Try running MIMICview first as is.

If you get the following error:

```
./mimicview: error in loading shared libraries
libtclx.so: cannot open shared object file: No such file or directory
```

then you will need to get the latest libtclx (8.0.4) library. Gambit can provide this for you upon request. Once you get the library you will have to do the following:

1. copy it to `/usr/lib`
2. `ln -s libtcl8.0.4.so /usr/lib/libtclx.so`

You will also need to have the directory `/usr/lib/tclX8.0.4` present with all the necessary files present. This can also be gotten from Gambit.

You could also get the following error once MIMICview is started:

```
./mimicd: error in loading shared libraries
libstdc++-libc6.1-1.so.2: cannot open shared object file: No such
file or directory
```

Check and make sure if you have `/usr/lib/libstdc++.so.2.9.0`. If you do not have it, you will need to install `gppshare-990315-9` and then do the following:

1. `ln -s libstdc++.so.2.9.0 libstdc++-libc6.1-1.so.2`

If you have `gppshare-990315-9` already installed and `libstdc++.so.2.9.0` is not in `/usr/lib` then you can get it from Gambit by request.

Q. Will there be a port conflict if I install the MIMIC software on a machine with HPOV Network Node Manager 5.01 installed?

A. MIMIC should allow you to do what you want. The snag is that HP/Openview requires an SNMP agent to run on the management station. This agent conflicts with agent instances running on MIMIC. You have 2 choices:

a) run MIMIC on one machine, HP/Openview on another. This would require 2 laptops to do your demos. We have found this to be a better solution than b), because HP/Openview and other management applications put a lot of burden on the machine (memory, CPU utilization).

b) run MIMIC and HP/Openview on same machine, but this only works if you use non-standard port numbers for the MIMIC SNMP agent instances. You will have to configure HP/Openview to probe these non-standard ports.

Q. Why does HPOV Network Node Manager not discover the MIMIC agents?

A. HPOV does not discover foreign networks automatically. I quote from the NNM Runtime manual, section "Maps" --> "Customizing your Network Map View" --> "Expanding Your IP Network Map" --> "Adding a Network":

```
>For security purposes, Network Node Manager does not discover networks
>in your internet, beyond your local gateways. You can add an object for
>an network that NNM has not discovered to an Internet submap, by placing
>a network symbol on that submap. If you are adding an object for an IP
>network, NNM will eventually discover it. For network objects that NNM
>cannot discover, the network symbol remains on the user plane.
```

For example, if you want to discover the 192.9.201.0 network, you'll have to create a "IP Network" object in the "Internet" map.

Q. How can I use MIMIC in a Client/Server configuration ?

A. MIMIC is a distributed client/server application, which means you can access the MIMIC Simulator daemon both locally (from same machine) and remotely (over the network) from multiple MIMIC clients (MIMICview, MIMICShell, etc). Each MIMIC client normally opens one session and upto 20 sessions are allowed. To use MIMIC remotely, all you have to do is this:

1) install MIMIC on the client system. Use empty license keys while installing.

2) invoke MIMICView from a command prompt with the --host option. Change to the bin/ folder of the MIMIC install area and type

```
mimicview --host THE-HOST-WHERE-MIMIC-DAEMON-IS-RUNNING
```

The assumption for the Client/Server function to work 100% is that the private data areas are accessible at the same file path on both client and server machines for the agents in the user's view (if access control is used). This is accomplished on Unix systems with NFS, and on Windows with shared drives. Eg. on Windows, mount a drive containing the private data area on both systems with the same drive name.

All MIMIC APIs (Tcl, Perl, Java, C++) can also work in remote mode by specifying the hostname where the MIMIC simulator daemon is running while opening a MIMIC session.

Even though the MIMIC clients are not licensed on a node-locked basis, the MIMIC Compiler and MIMIC Recorder are, and as such can only run on the machine that they are licensed for.

Q. Can you run MIMIC and a Web Server on the same machine?

```
> Do you see any issues or know of things to watch out if
> we want to
>   o Run Both MIMIC and Apache web server on a linux machine
>   o Run both MIMIC and IIS web server on an NT machine?
```

A. MIMIC does not care what other servers run on your system, as long as they are not making use of the same ports as MIMIC (eg. UDP SNMP port 161). Since standard web servers use TCP ports 80 or 8080, there is minimal likelihood of a clash.

Q. How do I simulate a web server for each of my simulated agents?

A. Before MIMIC 14.00, you had to use a web server with virtual hosting functionality, such as [Apache](#). For Apache, a different virtual web server can easily be setup for each of your simulated agents with the VirtualHost primitive. Each virtual web server can export a customized web site.

With MIMIC 14.00, we added the [WEB Services Simulator](#) that simulates complex web services.

Q. How do I apply new license keys to the installed software?

A. If you want to apply new license keys to an already installed version of MIMIC (eg. if you want to change the evaluation keys to permanent keys, or upgrade in size), all you have to do is edit the license key files in config/*.lic . There is one file per licensed component (Simulator, Compiler, Recorder). Open each file with your favorite text editor, and copy/paste the corresponding key.

Q. What environment variables are available?

A. In general, there is no need to use environment variables with MIMIC. MIMIC is highly configurable in several ways (configuration files, command line options, etc), and environment variables are a convenient way to override default behavior for highly specialized circumstances. The references below contains details for the user-configurable environment variables:

- [MIMIC_DISABLE_ACTION](#)
- [MIMIC_ADAPTIVE_VSPACE](#)
- [MIMIC_DEFAULT_NETDEV](#)
- [MIMIC_DISCARD_RESENDS](#)
- [MIMIC_REC_NO_GETNEXT](#)
- [MIMIC_REC_NO_TABLEWALK](#)
- [MIMIC_EDITOR](#)
- [MIMIC_PRIV_DIR](#)

Q. How can I use Microsoft Operations Manager to collect SYSLOG messages generated by MIMIC?

A. For a HOWTO on configuring MOM to receive SYSLOG messages see this [article](#) at the Microsoft support site.

For details on how to generate SYSLOG messages from MIMIC see the [SYSLOG Protocol Module Guide](#) in the MIMIC documentation.

Q. Where do I download your recommended versions of Linux?

A. Older versions of Fedora are downloadable from [the Fedora download site](#)

Q. How does MIMIC Update Wizard work through a SOCKS/SSH proxy?

A. MIMIC Update Wizard needs access to the MIMIC update server(s) over the internet, either directly or through a proxy server.

MIMIC Update Wizard will work through a [SOCKS proxy server](#) with the help of the [socat relay utility](#) which is an optional installable on newer versions of Fedora Linux. You can get it running in 3 easy steps:

- First, install socat on a Linux system with

```
# yum install socat
```

- Second, run socat to forward through your proxy server. Assuming that your proxy server is running on the machine `proxy-server`, invoke socat with something like

```
socat TCP4-LISTEN:9799,fork SOCKS4:proxy-server:mirror2.gambitcommunications.com:80
```

- Third, set the proxy server and port in the [Configuration Wizard](#) to `HOST-IPADDR` port 9799, ie. `HOST-IPADDR` is the host IP address where the socat is running. If the socat is running on the same system as MIMIC, then use `localhost`

We have also verified Update Wizard to work through a SSH proxy like [MobaXterm](#) or [PuTTY](#) .

- First, install either proxy tool from the URL above on a Windows system.
- In the SSH tool, enable a SSH tunnel from local port 9799 to our update server `mirror2.gambitcommunications.com` port 80.
- Third, set the proxy server and port in the [Configuration Wizard](#) to `HOST-IPADDR` port 9799, ie. `HOST-IPADDR` is the host IP address where the SSH tool is running. If the SSH tool is running on the same system as MIMIC, then use `localhost`

After that, Update Wizard should work.

Q. What is the best way to create a new simulation of a device?

A. For [MIMIC SNMP Simulator](#) the process is detailed in the [Process Overview](#) section of the [QuickStart](#) Guide, specially this [section](#). It depends on what you have access to. This matrix tries to quickly present the different cases:

How to create a simulation if you have...			
MIB	Live device	Walkfile	Action
Yes	Yes	Yes No	<p>Wizard -> MIB to compile MIBs if not already in MIMIC. Even if you have a walkfile, it's best to record the live device. Then</p> <p>File -> Record Live for single device or a small number; see also QuickStart</p> <p>or</p> <p>Wizard -> Discovery for many devices; see also this FAQ entry and this one</p> <p>or</p> <p>Wizard -> Snapshot to capture changes over time</p>
Yes	No	Yes	<p>Wizard -> MIB to compile MIBs if not already in MIMIC. MIMIC handles many third-party walkfiles as detailed here. Then</p> <p>Simulation -> Record File for single walkfile or a small number;</p> <p>or</p> <p>Wizard -> Discovery for large number of walkfiles;</p> <p>or</p> <p>Wizard -> Simulation if you want to change the simulation</p>
Yes	No	No	<p>Wizard -> MIB to compile MIBs if not already in MIMIC. Then</p> <p>Wizard -> Simulation and input values for the MIB objects.</p>

The [MIMIC CLI Simulator](#) already supports extensive Cisco IOS and Juniper JUNOS simulations. To create additional command-line simulations you need to record an existing CLI (command line interface) with [Wizard -> CLI](#) . The wizard can record from a live device, from a transcript, or interactively.

For [MIMIC NetFlow Simulator](#) you need to use [Wizard -> NetFlow](#) to record either from a packet capture (PCAP) or in live mode from a real exporter.

Web Service simulations for [MIMIC WEB Simulator](#) can be created with [Wizard -> Web](#) from either a PCAP or in live mode. They require the management application to access the real web server so that we can record the requests and responses.

For [MIMIC MQTT Simulator](#) you need to use [mqttrec](#) and [mqttconv](#) tools to record either from a packet capture (PCAP), or in live mode from a real sensor.

Recorder

Q. How do I efficiently record and simulate an exact copy of a large network?

A. After MIMIC 4.00, use the [MIMIC Discovery Wizard](#) .

For versions earlier than MIMIC 4.00, the most efficient way of recording a large network is with shell scripts. Here is one that records a set of devices at known IP addresses (assuming you run it from the MIMIC shared area top directory):

```
bin/mimicrec --target [ADDR1] --simulation faq-[ADDR1] --scenario 1
```

```
bin/mimicrec --target [ADDR2] --simulation faq-[ADDR2] --scenario 1
...
bin/mimicrec --target [ADDRn] --simulation faq-[ADDRn] --scenario 1
```

Substitute [ADDR1], etc with the addresses of your devices. This script will work in most scripting languages such as Unix shells and Windows .BAT files. Notice that we are creating a simulation for each device with a name that contains its IP address (you can choose any unique prefix aside from "faq-"; do not use "sim-" since MIMICView uses this by default).

Once you have recorded your devices, you need to create an agent configuration that uses the simulations. Rather than doing this manually in the graphical user interface, you can write another script that does this. Here is an example using Unix text processing tools:

1) Extract your set of simulations from the device configuration file(s) in your private data area:

```
# MIMIC version 4.10 and earlier
% cd ~/mimic
% grep ^faq- config/dev.cfg > /tmp/devices.lst
or
# MIMIC version 4.20 and later
% cd ~/mimic
% cat data/sim/faq-*/dev.cfg > /tmp/dev.cfg
% grep ^triplets /tmp/dev.cfg | sed 's/triplets = //' > /tmp/devices.lst
```

You can further sort this file by IP address with the sort utility.

2) Create configuration files with text-processing tools:

```
% cat > /tmp/awk.prog
BEGIN {print "version = 2.1"}
{print "agent = {"}
{print "hostaddr = FOOBAR" $0 "FOOBAR" }
{print "portno = 161"}
{print "read_str = public"}
{print "write_str = netman"}
{print "delay = 0"}
{print "start_time = 0"}
{print "pdusize = 4500"}
{print "drop_rate = 0"}
{print "triplet = " $0 }
{print "}"}
```

The ^D means to press CTL and D simultaneously (to generate the EOF character). On Windows, you need to use CTL-Z followed by Return, instead of CTL-D.

```
% awk -f /tmp/awk.prog /tmp/devices.lst > /tmp/almost.cfg
```

```
% cat /tmp/almost.cfg | sed 's/FOOBARfaq-/' | sed 's/,.*FOOBAR/' > \
/tmp/copy.cfg
```

The file /tmp/copy.cfg will contain the lab configuration necessary to run your simulations. Copy it into the config/ directory with your desired name.

On Windows, you can use the free GNU implementation of the Unix shell and text-processing tools downloadable from <http://www.cygnum.com/misc/gnu-win32/> .

Q. How do I discover and record all devices in a subnet?

A. After MIMIC 4.00, use the [MIMIC Discovery Wizard](#).

For versions earlier than MIMIC 4.00, you can run the following Bourne shell script to discover and record all devices in a subnet (scripts can be easily integrated with MIMIC command-line tools in any of the popular languages, including Perl, Tcl, shells, Windows .BAT files etc.). This script is supplied in common/discover.sh.

```
# discover.sh network-address
# this little script will discover and record all devices in a subnet
# (class C) that is specified in the first argument
# It needs to be customized to your site, eg.
# the path and syntax of a "SNMPGET" utility needs to be changed.
```

```
if [ a$1 = a ]
then
    echo "Usage: discover.sh network-address"
    echo "eg. discover.sh 192.9.200"
    exit 1
fi
```

```
net=$1
```

```
# loop from 1 to 254
host=0
while [ a = a ]
do
    host=`expr $host + 1`
    if [ $host -gt 254 ]
    then
        break
    fi

    addr=$net.$host
    echo "Querying $addr"
```



```

# this determines if there is an agent at the address
# any publicly available SNMP GET utility will do. I am using
# the HP OpenView implementation at my site.
# could do this with sysDescr and filter out certain device types
response=`e:/apps/openview/bin/snmpget -p 161 -t 1 -r 0 $addr \
sysObjectID.0 2>NUL`

if [ "$response" = "" ]
then
    echo "$addr is down"
    continue
fi

echo "$addr is $response"

# we could do a filter here to only look at certain devices
# or exclude certain devices

# now do what you want with discovered node
./mimicrec --target $addr --sim faq-$addr --scen 1
done

```

Q. When I record a device, I get a "PDU timed out" error message after it has retrieved many variables. Why?

A. This means that the device did not respond to a query from the MIMIC Recorder. This could be due to congested network, or a busy device. You can use the `--rextmits` and `--timeout` options to cause the Recorder to wait longer for responses, although the default settings should already wait long enough.

Another cause could be a faulty agent implementation on the device. The MIMIC Recorder uses efficient, multi-variable per PDU queries (as do most good management applications) to minimize network traffic and to traverse dynamic tables. Some agent implementations do not work correctly with this type of query. You can use the `--tablewalk` command line option, or set the `MIMIC_REC_NO_TABLEWALK` environment variable before invoking `mimicrec` to disable the default behavior of the Recorder.

- In the Cshell do:
% `setenv MIMIC_REC_NO_TABLEWALK 1`
- In the Bourne shell do:
`MIMIC_REC_NO_TABLEWALK=1; export MIMIC_REC_NO_TABLEWALK`
- On Windows, use the system Control Panel to set this environment variable.

You will know that this is set correctly if you do not see any messages containing "starting table walk".

Q. When I record a device, I get a "GETNEXT violation" error message after it has retrieved many variables. Why?

A. This is caused by a faulty agent implementation on the device. The MIMIC Recorder ensures that the lexicographical order of retrieved variables is preserved according to the SNMP standard. Otherwise the retrieval could go on forever.

The MIMIC Recorder uses efficient, multi-variable per PDU queries (as do most good management applications) to minimize network traffic and to traverse dynamic tables. Some agent implementations do not work correctly with this type of query. You can use the `--tablewalk` command line option, or set the `MIMIC_REC_NO_TABLEWALK` environment variable before invoking `mimicrec` to disable the default behavior of the Recorder.

- In the C shell do:
% `setenv MIMIC_REC_NO_TABLEWALK 1`
- In the Bourne shell do:
`MIMIC_REC_NO_TABLEWALK=1; export MIMIC_REC_NO_TABLEWALK`
- On Windows, use the system Control Panel to set this environment variable.

You will know that this is set correctly if you do not see any messages containing "starting table walk". If you still see the GETNEXT violation message, then you can use the `MIMIC_REC_NO_GETNEXT` environment variable to disable GETNEXT checking. This will only work if the agent (e.g., a Windows NT agent) incorrectly orders entries in a table, but otherwise proceeds with the walk.

If you are now hung in an infinite loop (if it takes forever and a tail of the walkfile shows repeating objects), kill the recording, and use the `--start` or `--root` options to bypass the faulty objects.

Q. How do I re-record a device when it is no longer there?

A. With MIMIC, you don't need the live device to be able to re-record it. When you record it the first time, the MIMIC Recorder creates a walkfile in the walkfile directory (before MIMIC version 8.10 this was by default `/tmp`, after that the `walks/` subdirectory in your private data area). You can use the Recorder's `--file` command to instruct it to get its data from this walkfile, or you can use [Simulation->Record File...](#) in MIMICView.

Q. When recordings are taken of MIB values where is this data stored?

```

> Would it be possible to edit the retrieved data to include
> our missing interface, etc info?

```

A. While recording a live device, the Recorder creates a walkfile in the walkfile directory. Before MIMIC 8.10 this was a file named `walkfile-[IP-ADDRESS].wlk`, stored on Unix in the `/tmp` directory, or on Windows in a folder `DRIVE:\tmp\` (where `DRIVE` is the drive letter on which you installed MIMIC), where `IP-ADDRESS` is the IP-address of the device. In MIMIC 8.10 or later, the walkfiles are stored in the `walks/` subdirectory in your private

data area.

You can edit the walkfile, and rerecord from this file with [Simulation->Record File...](#) What I would do first, though, is to recompile the MIBs, then rerecord the live device. The MIMIC Recorder then should pick up the missing variables. The [Simulation Wizard](#) presents a user-friendly interface for doing this.

Q. Is there any way to change the name of the agent?

```
> Is there any way to change the name of the agent. It now appears
> as unknown - first - first where first-first is the name of simulation
> and scenario. How can I change the unknown?
```

A. As a convenience to the user, the Recorder names the agent by default with the contents of the sysDescr.0 object. Otherwise it calls it "Unknown...". You can add sysDescr.0 to your walkfile, and it will pick up that name in subsequent runs of the Recorder (recommended), or you can change the name of any simulated device with [Simulation->Devices...](#) dialog in MIMICView. The third option is to edit the config/dev.cfg file.

Note: the device name is only used to display a user-friendly string in MIMICView. It is not actually used for simulation. The simulation and scenario name of a device are used to drive the simulation.

Q. Why are my enterprise MIBs not simulated by the MIMIC Recorder?

```
> I performed Record Live from Simulation menu on a real switch. The
> process ended with 0 errors, but when I looked into list of simulated
> devices it shows that my simulation includes only 3 standard (I mean not
> enterprise) mibs. So I added the enterprise mibs from the list of mibs but
> now I had to init all the mib variables by myself, which is very long and
> arduous process. My question is: How to add the enterprise mibs at the
> beginning of the recording process, so that I should receive starting
> complete simulation of the real device as a result of recording process.
```

A. After compiling the enterprise MIBs, you can re-record from the walkfile generated by the Live Record. This should simulate the earlier missing MIBs.

Otherwise, it is likely that you did not traverse the enterprise MIBs when you recorded the switch. You can verify this by looking at the walkfile generated by the MIMIC Recorder.

The file is called walkfile-[switch-address].wlk and resides in the walkfile directory (see previous [article above](#)).

Look at the end of the walkfile, if there are OIDs starting with 1.3.6.1.4.1, that would be in the enterprise MIB (likely your enterprise MIB). If so, please e-mail us the walkfile for us to diagnose any further problems.

If not, you did not record the enterprise MIB. This can be due to a number of reasons, as we have encountered in the past with a variety of equipment from other manufacturers:

- 1) your enterprise MIB is accessible only with a different community string. You need to record your switch with the different community string.
- 2) your enterprise MIB is accessible only at a different port number. You need to record your switch with the different port number.
- 3) the agent on your switch has a problem traversing from the standard MIBs to the enterprise MIB. You need to start recording your switch with -start 1.3.6.1.4.1 or somesuch.

For all these cases, you will end up with 2 walkfiles, the first containing the standard MIBs, the second containing the enterprise MIB. You can combine the 2 walkfiles into one (append the enterprise walk to the standard walk). Then record the combined file with "Simulation->Record File". The [Simulation Wizard](#) presents a user-friendly interface for doing this.

Q. Why does the MIMIC Recorder not handle unknown objects?

```
> Why am I seeing error messages with "unknown objects" in the recorder log?
```

A. The MIMIC Recorder is used to create a simulation by taking a snapshot of a real device. By default, it accomplishes this with a "snmp walk": a complete traversal of the MIB. It relies on the MIB definitions to create a default [basic simulation](#), one that is useful to most users.

One side effect of every live MIMIC Recorder session is a "walkfile" with the contents of that "snmp walk". The second phase is to create the simulation from that walkfile, where it uses the MIB definitions that are compiled into MIMIC to create a default simulation.

MIMIC does this by default because if you are interested in a MIB simulation, you will have the MIB definition, since your management app uses this MIB definition as a contract between it and the agent.

Of course, there are a myriad of options to override the defaults, and the best combination depends on your exact requirements. For example, if you don't have the MIB definitions, but still want to create a simulation, the recorder can record "unknown" MIBs. The accuracy is decreased, because guesses will only go so far, maybe 90%. But if they go wrong, they really go wrong. That is why that is not the default.

So, your recording generated a walkfile with a lot of unknown objects, and that is what the log is telling you. You can now choose to either import the MIB into MIMIC, then re-record from the walkfile to create the best simulation, or to create an approximation with the --unknown option to mimirc.

Q. In general, how many errors would be too many as a percentage?

The manual states that if there are too many errors, you might have to obtain the MIB from the equipment vendor and compile it. In general, how many errors would be too many as a percentage?

A. Generally, less than 10% is ok, unless you are interested in the MIB objects in the missing 10%. The simulation only really needs to export the MIB objects that your management application retrieves. Even if you record the entire device, and you are only interested in MIB-II, then you can instruct MIMIC to only simulate MIB-II (eg. to conserve memory if you are running lots of agents).

Q. How do I remove unwanted simulations?

```
> 2. When I do simulation, sometimes I also want to remove some
> unnecessary simulation and scenario. Besides modify the simulation file
> in /mimic/data/sim, is there a GUI function allow me to remove these
> unwanted simulations ?
```

A. For removing unwanted simulations use the [Simulation->Devices...](#) menu option. Select the simulations that you want to get rid of and press the Delete button. After you have completed cleanup press save to commit these changes to the disk.

Once you have done the above you can delete the simulation directory from under `data/sim` to complete the cleanup.

Q. Can I create a simulation from an external third-party SNMP walk utility?

```
> Can you tell me whether it's possible to use an external SNMP walk tool
> to get information from a device, and then import that data into MIMIC ?
>
> If so, what format would the data need to be in ?
>
> One of our clients has devices that respond to SNMP requests only from
> given IP addresses, and we would like to try and avoid having to get
> them to reconfigure the network just so we can run a MIMIC trace.
```

A. Yes, you can create a simulation from any walkfile format, but you need to convert it to MIMIC walkfile format first. This walkfile can then be fed to the MIMIC Recorder, eg. in MIMICView use Simulation->Record File.

There are problems with walkfile conversion, specially from formats with ambiguous and incomplete output. Eg. if the output is

```
someObject.0: 00:11:22:33:44:55
```

is that an OCTET STRING with hexadecimal or ASCII values? 99% of the time it is hexadecimal, so the conversion filters will blindly convert them to hex. But, for the 1% of the cases where it is indeed ASCII, the conversion will return incorrect values.

If you must resort to third-party tool to record walkfiles, we recommend the [net-snmp toolset](#) with the `-one` command line options.

After MIMIC 5.10, anytime you manipulate walkfiles in MIMICView, the MIMIC walkfile converter is automatically invoked.

Otherwise, MIMIC ships with conversion scripts for these common walkfile formats:

- HP/Openview
- NetMetrix
- UCD snmpwalk
- Concord Network Health

See the [walkfile converters](#) section in the Utilities Guide for more details.

Simulator

Q. I cannot start agents in MIMIC. I get errors in the log window when starting an agent instance. Why?

A. There can be many reasons for this problem, but it is most likely caused by the existence of another SNMP agent running on this system. The solution is explained in detail in [Appendix C](#), for [Windows](#).

Q. I have started agents in MIMIC, but I cannot ping them from my management station. Why?

A. When you start agent instances with IP (IPv4) addresses on a subnet different from the one that your management station is on, you need to tell the management station how to get to the subnet.

This can be done in most operating systems via a static route with the `route` command. Assuming that your management station and MIMIC system are on the same LAN, your agent instances are on the 192.9.200.0 subnet and that the address of your management station machine is IPADDR, here are the route commands for some common operating systems:

- [Windows 7/8/10](#)

From the DOS command prompt:

```
C> route add 192.9.200.0 mask 255.255.255.0 IPADDR
```

- [Solaris 2.6, 7](#)

From any shell as root:

To add a route:

```
# route add -net 192.9.200.0 IPADDR 0
```

To delete a route:

```
# route delete -net 192.9.200.0 IPADDR
```

- [Solaris 2.5](#)

From any shell as root:

```
# route add 192.9.200.0 IPADDR 0
```

- [Red Hat Linux 5.x](#)

From any shell as root:

```
# route add -net 192.9.200.0 gw IPADDR
```

- [Red Hat Linux 6.x and later](#)

From any shell as root:

To add a route:

```
# route add -net 192.9.200.0 netmask 255.255.255.0 INTERFACE
```

To delete a route:

```
# route del -net 192.9.200.0 netmask 255.255.255.0
```

- [HP/UX](#)

From any shell as root:

```
# route add 192.9.200.0 IPADDR 0
```

If there is a gateway between the 2 LANs, then you change the above commands to supply IPADDR with the gateway IP address. You'll also need to configure routes on the gateway to route between the subnets, just as you would on any regular network.

If you have setup the routes, and you still cannot ping the agents, then a firewall could be the culprit. See also [this section](#).

Q. How do I access my IPv6 agents from a remote system?

IPv6 addresses of agents can be reachable from remote IPv6 management systems on the same LAN by adding a default route, if it doesn't already exist, via the main network adapter of the management system as below:

- [Windows](#)

```
netsh interface ipv6 add route ::/0 INTERFACE-NAME-OR-INDEX
```

INTERFACE-NAME-OR-INDEX can be found using:

```
netsh interface ipv6 show interface
```

Example:

```
C:\> netsh interface ipv6 show interface
Querying active state...
```

Idx	Met	MTU	State	Name
6	2	1280	Disconnected	Teredo Tunneling Pseudo-Interface
5	0	1500	Connected	main-192.9.200.64
4	0	1500	Connected	second-192.9.200.66
3	1	1280	Connected	6to4 Tunneling Pseudo-Interface
2	1	1280	Connected	Automatic Tunneling Pseudo-Interface
1	0	1500	Connected	Loopback Pseudo-Interface

```
C:\> netsh interface ipv6 add route ::/0 5
Ok.
```

- [Linux](#)

Check whether the default route exists:

```
$ip -6 route show | grep ^default
```

Add route if does not exist:

```
route -A inet6 add ::/0 dev eth0
```

- [Solaris:](#)

Check whether the default route exists:

```
netstat -f inet6 -rn | grep ^default
```

Add route as root:


```
route -f inet6 add ::/0 IPv6-ADDRESS-OF-MAIN-ADAPTER 0
```

Example:

```
route -f inet6 add ::/0 fe80::209:3dff:fe00:c69e 0
```

Q. I can ping my agents, but they don't respond to requests. Why?

A. Once you have determined reachability via ping, an agent can fail to respond (your application times out even after a number of retransmits) due to numerous reasons. You can determine each of them as follows:

- community string mismatch (you are sending the wrong community string or other authentication information): an error appears in the [Error Log](#). See also [this video](#)  .
- protocol mismatch (the agent does not support the SNMP version you are using): an error appears in the [Error Log](#).
- port number mismatch: a protocol analyzer trace would show an ICMP response with "port unreachable".
- IP address conflict: this will ALWAYS result in indeterminate behavior. If you are re-using an IP address of another system which does not run an SNMP agent, the symptoms will be the same as "port number mismatch". If the other system does run an SNMP agent, it may respond instead of the MIMIC agent.
- incorrect routing entries on MIMIC system: if there are routing entries setup on the system that is running MIMIC, they can interfere with the proper behavior of MIMIC. This is easily determined by examining the routing table as returned with the `netstat -r -n` or `route print` command.
- firewalls: a firewall (eg. BlackICE) may limit interactions with MIMIC, specially if it determines that a "denial of service" attack on SNMP agents is occurring. The first thing to determine is whether firewall software is running on the MIMIC PC. If so, the only solution is to re-configure or even disable the firewall software. Problems with external firewalls are tough to diagnose, but a protocol analyzer running on the MIMIC PC and on the NMS station should determine where packets are being dropped.
- retransmits: MIMIC discards retransmits for efficiency reasons (servicing unnecessary retransmits uses up CPU bandwidth). To disable this behavior, set the environment variable `MIMIC_DISCARD_RESENDS` to 0 as follows:
 - In the C shell do:
`% setenv MIMIC_DISCARD_RESENDS 0`
 - In the Bourne shell do:
`# MIMIC_DISCARD_RESENDS=0; export MIMIC_DISCARD_RESENDS`
 - On Windows, use the `System Control Panel` to set this environment variable.

In any of the cases, the first action would be to consult the [Error Log](#) and the agent statistics with the [Agent->Statistics](#) dialog. Use of a [protocol analyzer](#) such as `tcpdump`, `wireshark` or `etherfind` can reveal further causes of the problem.

Q. What is the fastest simulation I can run? Why is it not the default?

A. By default, MIMIC runs the most useful basic simulation clause for a wide variety of applications: a close snapshot of a device SNMP agent, with static objects taking on the values that were observed, and Counter objects taking on a rate that was interpolated from observed values, both entirely configurable at run-time. For details, see [the QuickStart tutorial](#).

This default basic simulation is more expensive than faster, simpler simulations. The simplest simulation is to just return a constant value for all object instances. If the management application does not care about values retrieved, then you can use this simulation.

The sample simulation clauses that were compared in a performance experiment were (for more details, see the [SIMULATE clause](#) section in the [Compiler Guide](#)):

- INDEX
- SIMULATE{ 1 }
- SIMULATE{ constant(1) }
- SIMULATE{ uniform(1) }
- SIMULATE{ constant_per_tu(1, 60) }
- SIMULATE{ uniform_per_tu(1, 60) }
- SIMULATE{ constant(lookup("r")) }
- SIMULATE{ uniform(lookup("r")) }
- SIMULATE{ constant_per_tu(lookup("r"), lookup("tu")) }
- SIMULATE{ uniform_per_tu(lookup("r"), lookup("tu")) }
- SIMULATE{ uniform_per_tu ((lookup ("r") == 0 ? 1 : lookup ("r")), lookup ("tu")) }

Here is a chart comparing the performance of different simulation clauses for objects. When requests are performed with only one variable per PDU, the difference in performance between the fastest and default simulations is approximately 50%. But, the more variables that are packed into a PDU, the larger the computation part becomes (compared to network overhead, etc), and for 100 variables per PDU, the difference between the fastest and default simulations is a factor of more than 5.

Performance comparison of SIMULATE clause

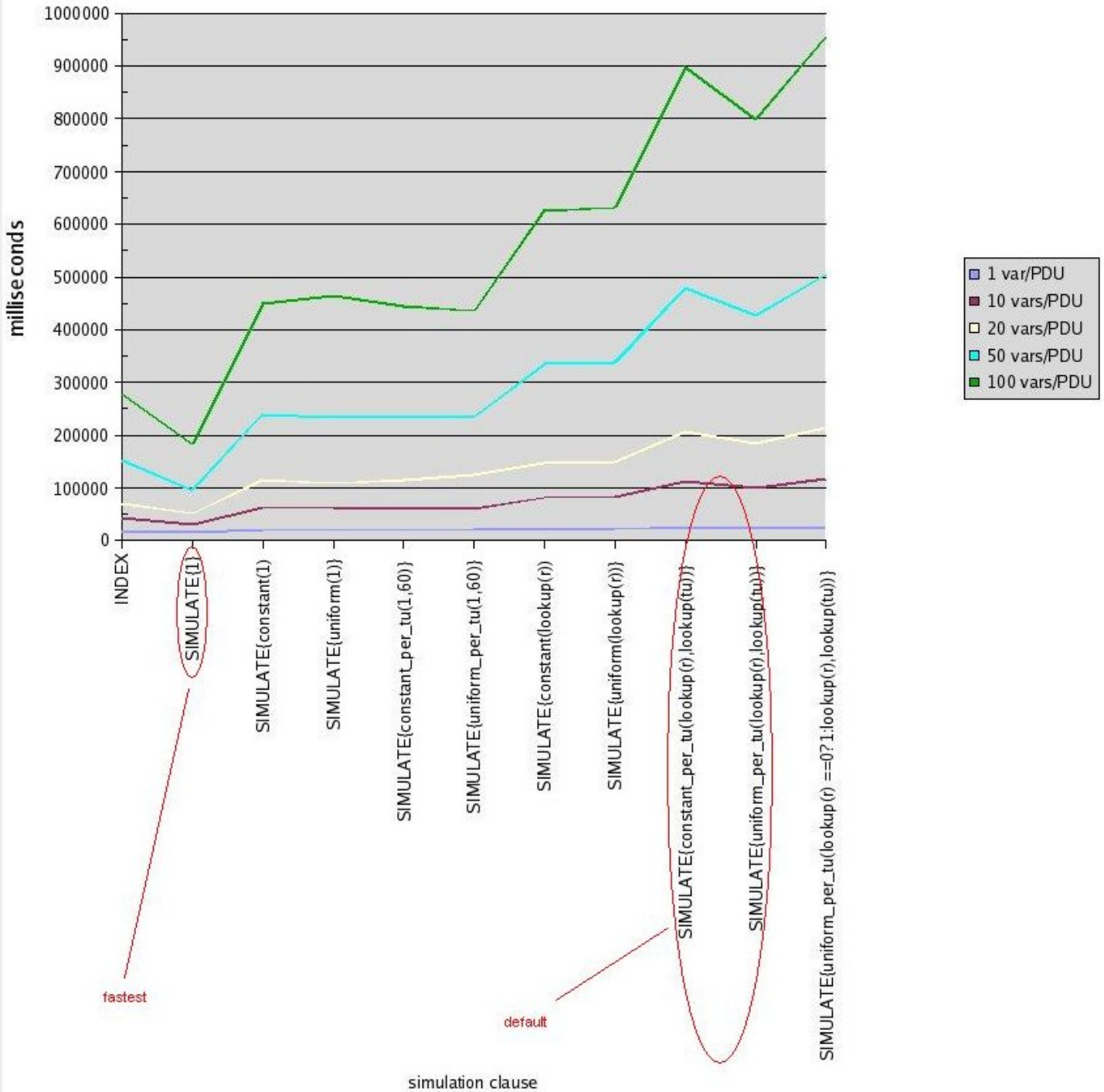


Figure 1: SIMULATE comparison

You can easily create the fastest simulation with the `--fast` command line option to the MIMIC Recorder.

Q. When I run a simulation, I see some diagnostic messages in the log window. What do they mean?

A. MIMIC does extensive error logging to justify its actions. If something is not going the way you want it, you can find out why from the error log. The error log is normally displayed in a **log window**, as well as dumped into a file `mimiclog.date.time` in your temporary directory (`/tmp` in Unix, `\TMP` in Windows).

The most common error messages are described in [Appendix C](#) of the online documentation.

Q. How can we truncate the log when it gets too large?

A. The `MIMICview` GUI application automatically changes the log every night at midnight, if so configured (for details see [Configuration Wizard](#)). You can also change the log manually with the **File->Log->Save** menu item.

If you are not running MIMICView, or want to have more control over the log size, then you can truncate it with the `timer+truncateLog.mtcl` timer script. See the [Run->Timer Scripts](#) menu item in MIMICView for more details.

This script runs periodically (by default every minute), and changes the active log when it exceeds a maximum size (by default 10 MB).

Old inactive log files can automatically be removed from the system with periodic batch files (eg. through crontab).

Q. Even though a counter simulation rate is greater than 0, I am not seeing the counters incrementing. Why?

A. The rate-based simulation of counter objects in MIMIC is governed by 2 variables: rate ("r") and time-unit ("tu"). Both together determine the actual rate of the counter, eg. `r=10, tu=60` would increment a counter at a rate of 10 per minute.

When you configure a counter object rate, make sure you set the desired time-unit ("tu") variable for that object.

Q. How can I force a Counter wrap for a MIB object?

```
> Using MIMIC, I want to test how my management station handles 64-bit
> counter wrap. I want to be able to send SNMP queries to the agent and have
> the agent return the data that follows
>
> Query          ifHCInOctets Value
> -----
> 1              18446744073707551614
> 2              18446744073708551614
> 3              100
>
> This sequence would enable me to test the handling of huge 64-bit values
> AND handling of the 64-bit counter wrap. How can I get MIMIC to behave
> this way?
```

A. After MIMIC version 12.00, you can use the [Agent->Change Simulation](#) dialog. Else, follow these instructions.

You have a couple of choices, depending on your exact requirements. All of the choices have the basic rate-based simulation of Counter* objects as a starting point, and here is probably the quickest way to achieve what you want:

1) this assumes an agent instance with IF-MIB configured. Start the agent, set the rate of `ifHCInOctets` to something large, eg. 1,000,000,000 (1 billion) / sec. You can do this with the `Agent->Value Space...` dialog by adding a variable `r` with value 1000000000 for the instance you are interested in.

2) your object is now advancing at a rate of 1 billion per second, so you should be getting values larger than 2^{32} after 4 seconds of running.

3) if you need larger values, you can pause the agent at a later time, eg. use `Agent->Pause`, set to 5 days, then use `Agent->Resume`.

Here are the values returned when I tried it:

```
% snmpget 192.9.201.133 public .1.3.6.1.2.1.1.3.0 .1.3.6.1.2.1.31.1.1.1.6.1
system.sysUpTime.0 = Timeticks: (43202330) 5 days, 0:00:23.30
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInOctets.1 = Counter64: 431983340054944
```

```
% snmpget 192.9.201.133 public .1.3.6.1.2.1.1.3.0 .1.3.6.1.2.1.31.1.1.1.6.1
system.sysUpTime.0 = Timeticks: (43202698) 5 days, 0:00:26.98
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInOctets.1 = Counter64: 431987020076191
```

4) using only this method, it's going to be hard to make it wrap on demand, ie. you would have to calculate the time to set for the agent, then pause it at exactly the right time.

One alternative to force the wrap would be to advance the time on the agent (`Agent->Pause`), but this only works if your management app is not looking at `sysUpTime`.

So, we are going to use action scripts to force the value down to 100 after 2 accesses. Use the [Agent -> Actions -> On GET/SET...](#) dialog, type `ifHCInOctets` into the Object text field, press the `Browse...` button for the `Get Scripts` field, click `New...`, call the action script `ifHCInOctets.mtcl`. Select the script in the list, press `Edit...`, and paste this into the editor window:

```
# this action returns a special value after 2 GET* accesses to this object
# instance
set count [mimic value get ifHCInOctets $gCurrentInstance count]
if { $count > 1 } {
    puts stderr "ifHCInOctets: special case"
    set gCurrentValue 100
}
```

```
incr count
mimic value set ifHCInOctets $gCurrentInstance count $count
Press Save, then Exit in the editor, press OK in the File Browser window, press OK in the Action dialog, and you have set the action script for the ifHCInOctets object.
```

Now when you access the object, you will get 2 large values, then the small value thereafter until you restart the agent.

```
% snmpget 192.9.201.133 public .1.3.6.1.2.1.1.3.0 .1.3.6.1.2.1.31.1.1.1.6.1
system.sysUpTime.0 = Timeticks: (43204776) 5 days, 0:00:47.76
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInOctets.1 = Counter64: 432007800089978
```

```
% snmpget 192.9.201.133 public .1.3.6.1.2.1.1.3.0 .1.3.6.1.2.1.31.1.1.1.6.1
system.sysUpTime.0 = Timeticks: (43204919) 5 days, 0:00:49.19
```

```
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInOctets.1 = Counter64: 432009230047969
```

```
% snmpget 192.9.201.133 public .1.3.6.1.2.1.1.3.0 .1.3.6.1.2.1.31.1.1.1.6.1
system.sysUpTime.0 = Timeticks: (43205110) 5 days, 0:00:51.10
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInOctets.1 = Counter64: 100
```

Another alternative for exactly predictable values is to use a [GET Action Script](#) for the desired object that returns hard-coded values in the script itself or retrieved from the [Agent Store](#).

Q. How can I reset a counter?

A. After MIMIC version 12.00, you can use the [Agent->Change Simulation](#) dialog. Else, follow these instructions.

In SNMP, the definition of a Counter object implies a non-monotonically increasing value, and the default simulation in MIMIC implements this. But, as a side effect of resetting state on a device (eg. an interface), some counters may have to be reset. This sequence of MIMICShell commands will reset a counter object instance (eg. for instance 1 of ifInOctets) and apply a new rate of increase NEWRATE:

```
mimicsh> mimic value set ifInOctets 1 r 0
mimicsh> mimic value set ifInOctets 1 _o 0
mimicsh> mimic value set ifInOctets 1 r $NEWRATE
```

Q. I am seeing the message "buffer full from ADDRESS to ADDRESS" in the error log. What does it mean?

A. The details for this error message are described in [Appendix C](#) of the online documentation.

Q. Isn't it true that accessing instance 0 on a table is an error? Mimic executes it without an error. Why?

```
mimicsh>mimic value set saSysNvCfgCntrl 0 v 2
0
mimicsh>mimic value get saSysNvCfgCntrl 0 v
2
whereas it fails, as expected when tried through snmpget/set
snmpget 152.148.26.56 saSysNvCfgCntrlEntry.1.0
snmpget: Agent reported error with variable #1.
```

A. Through the MIMIC interface, you can do whatever you want. Remember, all you are doing is setting variables in the "value space". Whether those variables are used or not is another matter, and is governed by the simulation expression. Hypothetically, we could have a simulation expression which uses variables of illegal instances.

The SNMP semantics are enforced in the SNMP PDU processing code, and in the simulation expression engine.

BTW, having 0-valued subids in table instances is allowed, eg. in MAC or IP address instances, you can have a 0, eg. 192.9.0.50 . RFC1212 says that the index cannot be 0 for INTEGER-valued subids. For error-testing, MIMIC allows you to have 0-valued indices, ie. in the example above the snmpget would succeed if you did

```
mimicsh> mimic value add saSys...Entry 0
```

Q. How do I verify whether a trap is generated?

```
Generating a trap from a simulated Cisco router arrives to
Netview. But from a simulated IBM-2210 it does not arrive
there. Is there a tool or a Unix command I can use to see
what is wrong. The destination address for the trap is correct
(same as the Cisco one) and I can ping the Netview from
the Sun station too. What should I do to investigate this
problem ?
```

A. You can look at the statistics for that agent instance, in MIMICView with [Agent->Statistics](#) . If "Trap PDUs" shows a positive number, then the traps are sent out. If not, you may have to define any variables that are to be sent with the trap in the "Advanced" tab in the "Generate Traps" dialog.

Q. Does MIMIC support dynamic row creation?

```
1. About dynamic rows. Let me clarify my question. I want
to add entries in the tables thru snmp-set's from the manager
and not by running scripts. Taking ur eg : I would like to
add ifTable.101 when an snmp-set for this table is sent from
the manager.
```

A. Yes. MIMIC does correctly simulate row creation. Below is a sample exchange with a MIMIC simulated agent. Notice that the ifTable only has 2 entries (1 and 2). Then we do a SET on row 5. Then row 5 is created. You need to make sure that you use the correct write-community string.

```
% snmpwalk 192.9.201.150 hp_admin interfaces.ifTable.ifEntry.ifType
interfaces.ifTable.ifEntry.ifType.1 = ethernet-csmacd(6)
interfaces.ifTable.ifEntry.ifType.2 = 28
% snmpwalk 192.9.201.150 hp_admin interfaces.ifTable.ifEntry.ifAdminStatus
interfaces.ifTable.ifEntry.ifAdminStatus.1 = up(1)
interfaces.ifTable.ifEntry.ifAdminStatus.2 = up(1)
% snmpset 192.9.201.150 netman interfaces.ifTable.ifEntry.ifAdminStatus.5 i 2
interfaces.ifTable.ifEntry.ifAdminStatus.5 = down(2)
interfaces.ifTable.ifEntry.ifAdminStatus.5 = down(2)
```



```
% snmpwalk 192.9.201.150 hp_admin interfaces.ifTable.ifEntry.ifType
interfaces.ifTable.ifEntry.ifType.1 = ethernet-csmacd(6)
interfaces.ifTable.ifEntry.ifType.2 = 28
interfaces.ifTable.ifEntry.ifType.5 = 0
% snmpwalk 192.9.201.150 hp_admin interfaces.ifTable.ifEntry.ifAdminStatus
interfaces.ifTable.ifEntry.ifAdminStatus.1 = up(1)
interfaces.ifTable.ifEntry.ifAdminStatus.2 = up(1)
interfaces.ifTable.ifEntry.ifAdminStatus.5 = down(2)
```

Q. Does MIMIC support RowStatus semantics?

```
> I checked it for creation and deletion of rows with the RowStatus oid
> without any success.
> What I did was compiling a mib that has a mibtable and then I tried
> setting the RowStatus oid of that table to 4 (createAndGo) and
> then to 6 (destroy).
> What I found out is that every initial value including 6 creates a new
> row full with default values,
> and the value 6 (destroy) doesn't destroy the row.
```

A. You can simulate a RowStatus object (as defined in RFC 2579) with a SET action script. For a discussion on action scripts see the [QuickStart Tutorial](#).

As of MIMIC 12.00, you can use the [Agent->Change Simulation...](#) dialog to deploy this simulation for RowStatus objects.

To set this up manually, copy the `scripts/rowstatus.mtcl` to your simulation directory and associate to the RowStatus object in your table as a SET action with the [Agent -> Actions -> On GET/SET...](#) dialog. Now when you send the SNMP SET request with RowStatus values it will trigger the action script and will create or destroy the row as per the value specified. Other custom side-effects will have to be programmed into the action script.

You can use the supplied `scripts/setup-rowstatus.mtcl` script to assign the default rowStatus action script to all Rowstatus typed objects for an agent. This script will start the agent, run through its MIB tree looking for RowStatus objects, and assign to each the default action script. Furthermore, this script assigns the fast **C++-based** action scripts, rather than the slower Tcl-based scripts.

Q. Does MIMIC support multiple communities per IP address?

```
> We are trying to simulate a Core Builder 9000 from 3com but we have
> some difficulty.
> Indeed the CB9000 is a chassis which have several communication cards.
> Only the chassis have an IP address.
> The chassis have a community name : public, but all the other card
> have different SNMP name like public@slot-1.1 for the first card,
> public@slot-2.2 for the second card...
> How can we simulate this agent ?
```

A. This is called "community-string-indexing", and is supported by MIMIC 3.30 and later. You need to create an agent instance for each community string. They all have the same IP address and port, but different community string.

Then you need to setup the correct simulation for each agent instance. For example, you can record an existing CB9000 at the different community strings (eg. public, public@slot-1.1) and assign the generated simulations to different agent instances.

At MIMIC v7.10 we have introduced the same concept for SNMPv3 with "engine-id indexing". Multiple agents with the same IP address but different engine-id can be configured. Thus, you can simulate the same IP address with different user and context names, and return different information. In the following example the system description is different for different context names to the same IP address:

```
% snmpget -v 3 -e 800000020109840302 -u user11 -n context11 10.0.0.1 sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco Internetwork Operating System Software ..
IOS (tm) 3000 Software (IGS-D-L), Version 11.0(14a), RELEASE SOFTWARE (fc1)...

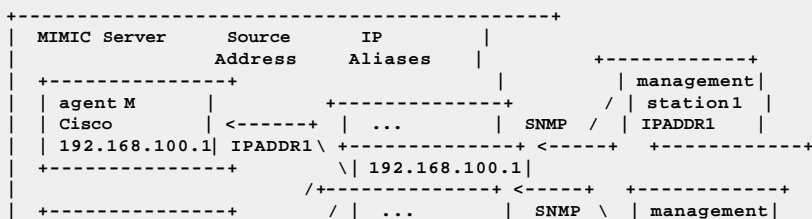
% snmpget -v 3 -e 800000020109840301 -u user11 -n context21 10.0.0.1 sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: HP ETHERNET MULTI-ENVIRONMENT,ROM A.03.00,
JETDIRECT,JD24,EEPROM A.03.06
```

Q. What is source-address indexing?

A. Source-address indexing is an extension of MIMIC's multi-user capabilities. Not only does MIMIC allow multiple users to create their own simulations, but it also allows these users to have their own virtual networks, even with overlapping IP addresses.

Suppose one user needs IP address 192.168.100.1 for simulating a Cisco device, but so does another user to simulate a Juniper. Traditionally, the only way to resolve this issue has been to each run their own simulator, adding to cost. In MIMIC, you can configure 2 different agents, with all the same addressing parameters (IP address, port, community string), but with an additional discriminator of source-address, ie. the address of the management station of each of the users.

This diagram illustrates the capability:



```

| | agent N | IPADDR2/ +-----+ | | station2 |
| | Juniper | <-----+ | | IPADDR2 |
| | 192.168.100.1 | | | +-----+
| +-----+ |
+-----+

```

Q. How can I create many interfaces on a host?

> Is it possible to quickly create many interfaces on a host?

A. As of MIMIC 12.00, you can use the [Agent->Change Simulation...](#) dialog to invoke the `Populate` table advanced simulation for the `ifEntry` object.

Otherwise, you can use MIMICShell scripting to populate the desired table. For example

```

for {set i 1} {$i <= 100} {incr i} {
  mimic value add ifEntry $i
  # set the values for the interface, eg. set to Ethernet type
  mimic value set ifType $i v 6
}

```

Q. When I set the multicast, broadcast ethernet counters, this won't affect the traffic, right?

```

When I set the multicast, broadcast ethernet counters, this
won't affect the traffic, right?
Yesterday our site got a broadcast storm. I hope it wasn't
because of the broadcast counters I set on a mimic device.

```

A. MIMIC manipulates MIB data only. It does not generate any traffic other than trap PDUs only (that too only if the user has programmed it to do that). Setting the counter rates to any value on the simulation will not cause any of the side effects you mentioned.

Q. How do I add a MIB to a simulation?

```

> I want to generate the portLinkDownEvent9 trap for Xylan switch, as defined
> in the XYLANTRAP-9-MIB; but this mib is not associated (if that's the right
> word) with the agent I have running.
> My question, what do I have to do so that I can begin generating this trap.

```

TRAP objects are non-accessible, and thus cannot be recorded with MIMIC Recorder. If those objects are in a MIB together with accessible MIB objects that are recorded, they end up in the resulting simulation. But, if they are in a separate MIB, it is not added from a recording session of a real device. The MIB needs to be added manually to the simulation.

You can add a MIB to the simulation in the [Simulation --> Devices](#) dialog. Remember to click `OK` or `Apply` to commit the changes. After 4.00, it also creates the simulation files. You need to reconfigure the agent instance with this new simulation, with [Edit->Configure->All...](#). Restart the agent, and the trap MIB should be loaded, allowing you to send the desired trap.

Q. Why can't I generate a trap for an object?

```

One thing I just noticed while reviewing the wftraps.asn file is that
wfFakeEventString is defined as OBJECT-TYPE instead of TRAP-TYPE. Could
this be the reason we can't seem to generate this trap from within MIMIC?

```

A. The object that you pointed out is a `DisplayString` and hence cannot be used as a Trap object name in MIMIC. Only objects that are defined using the macro `TRAP-TYPE` can be used.

Q. How do I send the current value of `sysUpTime` with a trap?

```

> The question I have for the sysUpTime is that: for every notification
> being generated, there is an unique sysUpTime attached to the
> notification. How do I get the sysUpTime for the specific notification
> that I generate? I tried agent=>get value, it keeps return default
> value 0.

```

A. The `sysUpTime` object is simulated in real-time by default. It does not have a variable in the MIMIC value space, because its simulation does not need one.

In order to send the value of `sysUpTime` in a trap you can do this in the MIMICShell:

```

# this evaluates sysUpTime to the current value
mimicsh> set cur_uptime [mimic value eval sysUpTime 0]
# this tells MIMIC to generate a trap once with the sysUpTime value
# change the value of myTrap to your trap object
mimicsh> set mytrap "coldStart"
mimicsh> mimic value set $mytrap 0 r 1
mimicsh> mimic value set $mytrap 0 tu 1
mimicsh> mimic value set $mytrap 0 sysUpTime $cur_uptime
mimicsh> mimic value set $mytrap 0 c 1

```

After MIMIC 4.10, you can have an action script associated with a trap, which will be executed every time the trap is generated. The dynamic assignment of the `sysUpTime` variable binding can be accomplished there. See the [QuickStart Tutorial](#) for more details.

Q. How does MIMIC determine what version of the TRAP PDU is generated?

```
> I am wondering how mimic determines which version to use when sending a
> trap. I noticed that there is no way to specify the version (v1, v2c, etc)
> when in the Generate Traps dialog. I know you can set the versions support
> by a simulated agent when you Add or Configure an agent. Is it determined by
> the trap definition in the mib?
```

A. This depends on the trap definition primarily. If the trap is defined using the SMI-V1 TRAP-TYPE definition, then a v1 trap will be generated. If the trap is defined as a NOTIFICATION-TYPE object, and if the agent is configured to support v3 only, then a v3 notification is generated. If it is configured as v2p but not v2c, then a v2p notification is generated, otherwise a v2c trap is generated.

Q. Is there any way of setting up more than one agent to send traps at any one time?

Is there any way of setting up more than one agent to send traps at any one time, i.e. something similar to selecting 10 agents and Agent -->Generate Traps, then putting the 10 agent ID's in and each one will generate the trap.?

A. Here is a MIMICShell script that does what you would like :

```
# Sample MIMICShell script
# This assumes that agents 1-10 are configured and running
# Generate linkUp traps from 10 agents 10/sec for 100 secs)
for { set i 1 } { $i <= 10 } { incr i } {
    mimic agent assign $i                ;# select agent
    mimic value set linkUp 0 ifIndex 1    ;# var-bind ifIndex=1
    mimic value set linkUp 0 r 10         ;# rate
    mimic value set linkUp 0 tu 1         ;# timeunit (sec)
    mimic value set linkUp 0 c 100       ;# duration (sec)
}
```

This should do what you want. You might have to customize for different traps/agents etc but the methodology should be clear.

Q. How can I send traps from IP aliases of my agent?

A. If you would like to generate this trap from MIMICView then select the Agent->Generate Traps menu item, select the desired trap object in the MIB browser, and click the IP Aliases tab where you will find a list of all IP aliases including the main agent IP address. Select the IP alias for which you would like to generate trap and press OK. This will generate the trap for the selected IP alias.

If you would like to automate the process, then you can use a MIMICShell script such as:

```
# this procedure generates a trap on all aliases of specified agents
# numbered start to end (inclusive)
# this uses synchronous trap generation since we cannot have more than one
# outstanding asynchronous trap of the same name per agent
proc syncr_trap_all_ipaliases {start end trapname} {
    for {set agent $start} {$agent <= $end} {incr agent} {
        mimic agent assign $agent
        mimic value set $trapname 0 r 1
        mimic value set $trapname 0 tu 1

        # generate a trap for each of the IP aliases (the 0-th IP
        # alias being the main address of the agent)
        set max [llength [mimic agent ipalias list]]
        for {set i 0} {$i <= $max} {incr i} {
            mimic value set $trapname 0 ip $i
            # synchronous trap
            mimic value set $trapname 0 c "s"

            # wait until the trap is actually sent
            while { 1 } {
                if { [mimic value get $trapname 0 c] == "0" } {
                    break
                }
            }
            after 100
        }
    }
}

# for example generate the coldStart trap for the first agent
syncr_trap_all_ipaliases 1 1 coldStart
exit
```

Q. How can I send traps from different source ports of my agent?

```
> Many agents in the field source every trap from a
> different UDP port, which means these traps are treated differently by
> firewalls evaluating whether every trap is part of the same session or
> whether every trap opens a new session. The firewall considers whether
> all 4 fields match in the sourceIP:port -> destinationIP:port
```

```
> connection. We would like to use MIMIC to simulate both types of agent:
> those that source all traps from the same UDP port and those that source
> each trap from a different UDP port.
```

A. You can use the IP Alias feature [detailed in the previous section](#) to send traps from different source ports for your agent. For each desired port, add an alias, then send desired traps for that alias.

Q. How do I generate a series of traps upon an SNMP SET?

A. Generating an immediate single trap upon processing a SET PDU is simple in MIMIC: you create an action script which contains the `mimic` value commands necessary to generate the single trap.

The next complication is to delay the trap: you achieve this by scheduling a timer script in your SET action script, which when executed in the future will issue the same `mimic` value commands.

But what if you want to generate a sequence of traps, possibly at varying time intervals?

The solution is to combine the output of the [Trap Wizard](#) with the mechanisms explained above. The output of the Trap Wizard is a sequence of trap generation scripts, which schedule the generation of different traps successively at the intervals they were observed when they were captured by the wizard. This sequence is by default started for the configured simulation when the agent is started up. Applying this to a SET action script involves these modifications:

1. turn off the automatic generation of the traps at agent startup in the [Edit->Configure->All...](#) dialog.
2. associate a SET action script with the desired MIB object. This script will merely schedule the initial trap generation script, usually with the statement:

```
catch { mimic agent timer script add sequence.1/begin.mtc1 0 } msg
```

This will invoke the first sequence of traps that were captured.
3. any sub-sequence of traps can be invoked separately by changing which trap generation script is scheduled in the SET action script.

Q. How can I interleave agents among multiple network interfaces?

```
> Do you happen to know how SUN Solaris assigns IP addresses across the
> network interface cards that it has available? That is, in our case we have
> one quad NIC and one single NIC. Will Solaris assign IP Address #1 to NIC
> #1, IP Address #2 to NIC #2, etc., or will it assign the first 8000 IP
> addresses to NIC #1 and then move on the the second card?
>
> If Solaris handles the IP address assignment in the second way, is there any
> way we can configure it to assign addresses across all NICs available?
```

A. By default, MIMIC does not interleave IP addresses among network interface cards on any of the OS platforms. The assignment of IP addresses to NICs has to be done explicitly (otherwise the address is configured on the default network device, usually the first available NIC), either in the Interface field of the Advanced tab of the agent configuration dialog in MIMICView, or with the `mimic agent set interface` command in MIMICShell.

You can do this by first configuring all agents with the default interface, eg. in MIMICView with the [Edit->Add->Agent...](#) menu item, and leave the Interface field blank. Then run a MIMICShell script that interleaves each agent on the desired NIC. After that, save the configuration ([File->Save As](#) menu item in MIMICView, or `mimic saves` in MIMICShell) to make this persistent.

Here is a script fragment which does strict interleaving of agents on all configured network interfaces:

```
set ifs [mimic get interfaces]
set curif 0
for {set i 1} {$i <= [mimic get last]} {incr i} {
    mimic agent assign $i
    # this assumes that all agents are configured, otherwise
    # check for state == 5
    mimic agent set interface [lindex $ifs $curif]
    incr curif
    if { $curif >= [llength $ifs]} {
        set curif 0
    }
}
```

Q. Why is my throughput intermittently slow on a dual NIC Linux system?

A. If your Linux MIMIC system has multiple active network interfaces connected to the same LAN, and you are experiencing drops in PDU throughput while polling MIMIC agents, it may be due to one network interface running slower than the other. Linux will route traffic through the NIC according to the arp tables regardless of the interface an agent IP address is assigned to. The command `ethtool` run as root can be used to discover the current speed and duplex settings of each NIC.

```
# ethtool eth0
Settings for eth0:
    Supported ports: [ TP ]
    Supported link modes:   10baseT/Half 10baseT/Full
                          100baseT/Half 100baseT/Full
    Supports auto-negotiation: Yes
    Advertised link modes:  10baseT/Half 10baseT/Full
                          100baseT/Half 100baseT/Full
    Advertised auto-negotiation: Yes
    Speed: 100Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 1
    Transceiver: internal
    Auto-negotiation: on
```

```
Supports Wake-on: umbg
Wake-on: g
Current message level: 0x00000001 (1)
Link detected: yes
```

Any [protocol analyzer](#) can be used to capture packets between the walkhost and the agent to determine the MAC address of the NIC handling the traffic.

Q. What SNMPv3 features are supported by MIMIC?

A. MIMIC version 16.20 and later adds SHA-2 Authentication as specified in RFC 7860.

MIMIC version 9.20 and later adds AES-192 and AES-256 Privacy.

MIMIC version 7.30 and later adds AES Privacy as specified in RFC 3826.

MIMIC version 7.20 and later adds user management support.

MIMIC version 7.10 and later adds engine-id indexing.

MIMIC version 5.00 and later adds INFORM PDUs, and SNMPv3 in Recorder (see [Release Notes](#)).

MIMIC version 4.40 and later adds AUGMENTed tables, and ContextName based indexing.

MIMIC version 4.20 and later adds DES Privacy.

MIMIC version 4.10 and later adds SHA Authentication.

MIMIC version 4.00 and later supports full SNMPv3 with USM, VACM and MD5 Authentication.

MIMIC 3.30 onwards supports a partial SNMPv3 implementation. This includes the noAuthNoPriv, AuthNoPriv implementation where the authentication is done using the MD5 protocol. USM is supported with the user database allowed to be configured as required. All the relevant report PDUs are supported. VACM, SHA and DES support will be added in later releases.

For example, all of the following [net-snmp](#) commands using different combinations of authentication and privacy protocols will work against an SNMPv3 agent with default MIMIC SNMPv3 configuration files at address 10.0.201.1:

```
% snmpget -v3 -uuser1 -lnoAuthNoPriv 10.0.201.1 sysDescr.0
% snmpget -v3 -uuser2 -lauthNoPriv -amd5 -A1234567890abcdef 10.0.201.1 sysDescr.0
% snmpget -v3 -uuser3 -lauthNoPriv -aSHA -A1234567890abcdef 10.0.201.1 sysDescr.0
% snmpget -v3 -uuser4 -lauthPriv -amd5 -A1234567890abcdef -xDES \
-X 1234567890abcdef 10.0.201.1 sysDescr.0
% snmpget -v3 -uuser5 -lauthPriv -aSHA -A1234567890abcdef -xDES \
-X 1234567890abcdef 10.0.201.1 sysDescr.0
% snmpget -v3 -uuser6 -lauthPriv -amd5 -A1234567890abcdef -xAES \
-X 1234567890abcdef 10.0.201.1 sysDescr.0
% snmpget -v3 -uuser7 -lauthPriv -aSHA -A1234567890abcdef -xAES \
-X 1234567890abcdef 10.0.201.1 sysDescr.0
% snmpget -v3 -ncontext11 -uuser11 -lnoAuthNoPriv 10.0.201.1 sysDescr.0
% snmpget -v3 -ncontext12 -uuser12 -lauthNoPriv -amd5 -A1234567890abcdef \
10.0.201.1 sysDescr.0
% snmpget -v3 -ncontext13 -uuser13 -lauthNoPriv -aSHA -A1234567890abcdef \
10.0.201.1 sysDescr.0
% snmpget -v3 -ncontext14 -uuser14 -lauthPriv -amd5 -A1234567890abcdef -xDES \
-X 1234567890abcdef 10.0.201.1 sysDescr.0
% snmpget -v3 -ncontext15 -uuser15 -lauthPriv -aSHA -A1234567890abcdef -xDES \
-X 1234567890abcdef 10.0.201.1 sysDescr.0
```

Using [Extended Security](#) requires extra configuration in MIMIC. You need to uncomment the appropriate users from the `v3usm.conf` configuration file. [NET-SNMP](#) prior to version 5.8 did not support these privacy options, but with [NET-SNMP 5.8](#) we interoperate with all supported authentication/privacy combinations, and we have tested interoperability of AES-192 and AES-256 privacy with at least these packages:

- SNMP++
- SNMP4J
- SnmpSoft
- PySNMP

SHA-2 interoperability has been tested with at least these

- NET-SNMP
- SNMP Research

This is a recommendation from [SNMP Research](#):

```
"When using AES-256 with authentication protocols with key lengths less than 256 bits (such as md5 and sha1), there needs to be a standard mechanism to produce the localized keys. Since SNMPv3 does not currently provide such a standard, there needs to be an agreed upon way to do key localization. The most common approach is the one used by Cisco which is based on a variation of the Reeder 3DES draft. This is not an issue when using SHA-2 with 256 or more bits for authentication; therefore, it is recommended to use SHA-2 for authentication when using AES-256 for encryption because no key extension is needed.
"
```

so we support the Reeder key extension by default.

For example, with NET-SNMP 5.8 these work:

```
% snmpgetnext -v 3 -l authPriv -u user31 -a SHA-224 -A 1234567890abcdef -x AES -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user32 -a SHA-256 -A 1234567890abcdef -x AES -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user33 -a SHA-384 -A 1234567890abcdef -x AES -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user34 -a SHA-512 -A 1234567890abcdef -x AES -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user35 -a SHA-224 -A 1234567890abcdef -x AES-192 -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user36 -a SHA-224 -A 1234567890abcdef -x AES-256 -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user37 -a SHA-256 -A 1234567890abcdef -x AES-192 -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user38 -a SHA-256 -A 1234567890abcdef -x AES-256 -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user39 -a SHA-384 -A 1234567890abcdef -x AES-192 -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user40 -a SHA-384 -A 1234567890abcdef -x AES-256 -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user41 -a SHA-512 -A 1234567890abcdef -x AES-192 -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3  
% snmpgetnext -v 3 -l authPriv -u user42 -a SHA-512 -A 1234567890abcdef -x AES-256 -X 1234567890abcdef \  
-m "" -One 10.48.0.1 .1.3
```

Q. Why does it take so long to start SNMPv3 agents?

A. Prior to version 8.31, starting SNMPv3 agents took a lot longer than SNMPv1 and/or SNMPv2c agents, because SNMPv3 authentication and security info needed to be loaded for each SNMPv3 agent. After 8.31, for most cases, starting SNMPv3 agents takes approximately the same amount of time as agents without SNMPv3. This is because the authentication info is the same for all agents in most cases, and in those cases we use a cache.

The following information is only relevant if you cannot upgrade MIMIC to version 8.31 or later.

In a simple experiment, on a representative Linux system, starting 1000 SNMPv3 agents took 30 times longer than non-SNMPv3 agents (450 seconds vs. 13 seconds), whereas on Windows it took 10 times longer (180 seconds vs. 18 seconds).

Thus, if you do not need SNMPv3 simulation, and start many agents, it is better to disable the SNMPv3 protocol for all relevant agents.

Q. How can I get MIMICShell scripts to terminate at the end of execution?

```
> But is there a way for mimicsh to exit automatically after having  
> executed the script ?
```

A. You need to explicitly end your script with "exit".

Q. How can I iterate through configured agents in MIMICShell?

```
> Do you know how to find out if an agent exists or not? I'm trying to build a  
> loop that manipulates each agent without failing if the agent numbering is  
> not contiguous from 1.
```

A. Unconfigured agents have state 5 (see [Simulator Guide](#)). So, to manipulate all configured agents, you can use a for-loop like

```
for {set i 1} {$i <= [mimic get last]} {incr i} {  
    mimic agent assign $i  
    set state [mimic agent get state]  
  
    # ignore unconfigured agents  
    if { $state == 5 } {  
        continue  
    }  
    # do something interesting with configured agents...  
}
```

An equivalent, more efficient way of achieving this loop is

```
set list_of_agents [mimic get configured_list]  
foreach i $list_of_agents {  
    mimic agent assign $i  
    # do something interesting with configured agents...  
}
```

Q. What is the difference between an action script and a MIMICShell script?

```
> I was trying to run a script by right clicking on the agent icon and  
> selecting Script... and I get this error:  
>  
> Error connecting to MIMIC server  
> localhost on port 9797 as limit  
> of total connections exceeded.
```

```

>
> The scripts (which are basically just puts statements) work fine when I use
> the Action... method and a remote MIB browser to invoke them.
>
> Is this due to the fact that I am not logged in with Administrator access on
> my WinNT machine or could it be something else?
> (I know in the manual it said that you need to have Administrator
> privileges, but since everything seemed to install and work fine without it,
> I never bothered!)
>
> How does this method of running scripts differ from using the Action...
> method?

```

A. The reason for the error you are seeing is that there is a limit of 5 client connections to the MIMIC daemon at a time (after version 5.00 the limit is 20 connections).

The reason that one method works and the other does not is due to the fact that Action method is run within the mimicd process and so does not count as a new client connection, whereas the Script method essentially forks of a mimicsh which has to make a connection with the MIMIC daemon to run and so counts as one more client connection.

Q. How can I debug my action script?

A. You can use "puts stderr" to print debugging info to the [Log Window](#).

Q. How do I maintain state in my action script?

```

> I have a small problem to obtain a value increase at each polling of the
> value.

```

A. To maintain state between 2 invocations of the action script, you need to keep another global variable which you increment. Also, you need to return the value from the script. For example, this fragment returns increasing values until 90, after which it starts from 0 again:

```

global myglobal
if { ![info exists myglobal] } {
    set myglobal 0
}

if {$myglobal > 90} then {
    set myglobal 0
} else {
    incr myglobal 10
}

return $myglobal

```

The global Tcl variable (defined above with the global statement) is accessible only by the same Tcl interpreter. All PDU action scripts for the same agent will run in the same Tcl interpreter, but other action scripts (startup, timer, etc) will run in different interpreters. Thus if you want to share variables the safest method is to use the [MIMIC Variable Store](#). This will let you share global variables across action scripts, agents, even across different MIMIC daemons and API languages.

Q. How can I assign action scripts via another script?

```

>Is there a way to assign action scripts via another script? I have some
>action scripts written, but I will want to apply them to a large number of
>agents. Assigning action scripts in the GUI for a large number of agents
>is very tedious.

```

A. Anything that can be done in the GUI can be done in the script. You can assign action scripts using the `mimic value set` command. The following command sequence sets the action script `foo-get.mtcl` for the GET action and `foo-set.mtcl` for the SET action on the object `ifStatus`.

This assumes that the files `foo-get.mtcl` and `foo-set.mtcl` exist in the simulation directory for the agent.

```

mimic value set ifStatus 0 g foo-get.mtcl
mimic value set ifStatus 0 s foo-set.mtcl

```

NOTE: The instance used is always 0 for assigning the action scripts even for tables.

Q. Can I interact with the user in my action script?

```

> If I want do I/O in this "GET action" script what should I do?
>
> e.g.
> #this is GET action script,
> puts "please input 0:right return 1: error return  "
> gets stdin line
> set gError $line
> return

```

A. Direct user interaction is not possible, since an action script is invoked in response to a SNMP request, and if it were to wait for user input then this would cause timeouts and retransmits from the manager. Also, the agent would be blocked and would not be able to process any other request.

The way to allow user-controllable decision points in action scripts is to use a variable in the Value Space, which you control from the MIMICShell or MIMICView. Eg. if you wanted a decision point for the `sysDescr` object, use a variable, eg. "decision", so your action script could contain:

```

set decision [mimic value get sysDescr 0 decision]
switch $decision {
  choice1 {
    # do first choice
    break
  }
  # other choices...
}

```

You can then cause the action to behave differently by interactively doing (or the equivalent in the Value Browser of MIMICView)

```
mimic value set sysDescr 0 decision some-value
```

Q. Is there a way to start up a timer script for each agent?

```

>I have a question about timer scripts. It mentions in the documentation
>how to set up timer scripts, and further mentions that they are "global"
>timer scripts. Is there any way to start up a timer script for each
>agent?

```

A. Timer scripts are designed to be executed globally. However from a given script you can address any particular agent by using the 'mimic agent assign' command to select that agent. Typically you can write a loop that runs through the list of agents, selecting them one after the other and doing whatever stuff is required.

Example script

```

set num_agents [mimic get last]
for { set i 1 } { $i <= $num_agents } { incr i } {

  # select an agent
  mimic agent assign $i

  # Do stuff here

}

```

Another approach would be to schedule an agent timer script for each of the agents with [mimic agent timer add](#). Each script invocation will select a different agent right at the start. The problem with this approach is that performance will be affected by the large number of timer scripts.

In general, you should use global timer scripts if you need to iterate over a large number of agents (less "context switching", ie. less overhead of switching between different timer scripts). Use agent timer scripts only to iterate over a small number of agents or as a side effect from action scripts.

Q. How can I control the order of starting agents?

```
> It looks like MIMIC doesn't start the agents in any specific order,
```

A. By default, MIMIC does an efficient "asynchronous" start of agents, eg. if you do:

```

foreach agent $list_of_agents {
  mimic agent assign $agent
  mimic agent start
}

```

or, if all agents are supposed to start, equivalently

```
mimic start
```

the MIMIC daemon parallelizes tasks in its multi-threaded architecture. The "mimic agent start" command just tells the MIMIC daemon to start an agent, which then happens asynchronously.

If you require the starting of agents to proceed in a particular order, you can script it through a "synchronous loop", by waiting for each agent to actually be in a "running" state, before continuing to the next agent. For example, this loop

```

foreach agent $list_of_agents {
  mimic agent assign $agent
  mimic agent start
  # wait for the agent to be running
  while { 1 } {
    if { [mimic agent get state] == "1" } {
      break
    }
    after 1000
  }
}

```

is guaranteed to start the agents in the order listed in the array "list_of_agents". Notice that this loop will be much slower than the asynchronous start, because only one task is happening simultaneously.

Q. Why do GET statistics not increase when I use the Get Value dialog?

A. The MIMICView Get Value and Set Value dialogs (as well as the MIMICShell `mimic value get` and `mimic value set` commands) do not use SNMP to interact with the agent. They use a proprietary command protocol to accomplish their function.

Since the statistics only increment on SNMP activity, you will not see any change. Also, PDU actions are only executed upon receiving SNMP requests.

Q. Why can I not see the value of an index object in the Get Value dialog?

```

> The message I got, when I tried to get the value with a "v" , which I know
> you said wouldn't work, is
>

```



```

> ERROR 01/28.14:04:26 - management operation failed
> 01/28.14:04:26 - command: value get
> 01/28.14:04:26 - 10 36 ospfNbrAddressLessIndex 1 32.81.243.85.0 v
> 01/28.14:04:26 - value space lookup failed
> 01/28.14:04:26 - no such index

```

A. The simulation for INDEX objects does not look at the value space, since their value is implicit in the index part of the OID. Thus it is normal that you cannot find anything for this MIB object with Value Get.

Instead of a "mimic value get" you can use "mimic value eval", such as

```

mimicsh> mimic value eval ospfNbrAddressLessIndex 32.81.243.85.0
32.81.243.85.0

```

Q. How do I simulate a binary OCTET STRING value?

> We are trying to set hrSystemDate to the correct value. How do we do this?

A. The object you are trying to set (hrSystemDate from HOST-RESOURCE-MIB) is defined as type DateAndTime with size 8 OR 11, as follows

```

DateAndTime ::= OCTET STRING (SIZE (8 | 11))
--      A date-time specification for the local time of day.
--      This data type is intended to provide a consistent
--      method of reporting date information.
--
--      field  octets  contents                range
--      ----  -
--      1      1-2    year                    0..65536
--                        (in network byte order)
--      2      3      month                    1..12
--      3      4      day                      1..31
--      4      5      hour                     0..23
--      5      6      minutes                   0..59
--      6      7      seconds                   0..60
--                        (use 60 for leap-second)
--      7      8      deci-seconds              0..9
--      8      9      direction from UTC      "+" / "-"
--                        (in ascii notation)
--      9      10     hours from UTC             0..11
--      10     11     minutes from UTC        0..59
--
--      Note that if only local time is known, then
--      timezone information (fields 8-10) is not present.

```

To set its value in MIMIC, you will need to set it as a hexadecimal value, eg. to convert 2003-12-22 18:25:40:05

```

Year 2003 = 07 d3
Month 12 = 0c
Day 22 = 16
Hour 18 = 12
Min. 25 = 19
Sec. 40 = 28
Deci-sec. 05 = 05

```

In MIMICshell you would do

```

mimicsh> mimic value set hrSystemDate 0 v "\\x07 d3 0c 16 12 19 28 05"

```

(the back-slash is an escape character, so needs to be repeated to escape itself), or from MIMICView Value Space Browser or Simulation Wizard then give the value as

```

\x07 d3 0c 16 12 19 28 05.

```

This is the canonical representation of a binary octet string, but MIMIC is flexible to accept other formats. Eg. these are all equivalent:

```

\xaa bb cc
\xAA BB CC
\xAA  BB  CC
\xAABBCC
\xAA:BB:CC
\xAA-BB-CC

```

Q. How do I simulate an object of type BITS?

A. According to RFC 3417 "Transport Mappings for the Simple Network Management Protocol (SNMP)", section 8. 3):

- (3) When encoding an object whose syntax is described using the BITS construct, the value is encoded as an OCTET STRING, in which all the named bits in (the definition of) the bitstring, commencing with the first bit and proceeding to the last bit, are placed in bits 8 (high order bit) to 1 (low order bit) of the first octet, followed by bits 8 to 1 of each subsequent octet in turn, followed by as many bits as are needed of the final subsequent octet, commencing with bit 8. Remaining bits, if any, of the final octet are set to zero on generation and ignored on receipt.

For example, in RMON2-MIB there is a BITS type object called probeCapabilities with this definition

```
probeCapabilities OBJECT-TYPE
    SYNTAX BITS {
        etherStats(0),
        historyControl(1),
        etherHistory(2),
        alarm(3),
        hosts(4),
        hostTopN(5),
        matrix(6),
        filter(7),
        capture(8),
        event(9),
        tokenRingMLStats(10),
        tokenRingPStats(11),
        tokenRingMLHistory(12),
        tokenRingPHistory(13),
        ringStation(14),
        ringStationOrder(15),
        ringStationConfig(16),
        sourceRouting(17),
        protocolDirectory(18),
        protocolDistribution(19),
        addressMapping(20),
        nlHost(21),
        nlMatrix(22),
        alHost(23),
        alMatrix(24),
        usrHistory(25),
        probeConfig(26)
    }
```

To set the etherStats(0) bit from the MIMICView Value Space browser, you will set the value in hexadecimal format as \x8000 (from the MIMICShell set the value as \\x8000).

To set the historyControl(1) bit, you will set the value as \x4000 (\\x4000 from MIMICShell).

Q. Do you have any Tcl scripts to manipulate BITSTRING values?

A. These scripts will let you set/clear bits in a BITSTRING object. Save the following in scripts/set-bit.tcl under your private MIMIC directory, and invoke as documented, then customize to your need:

```
# Copyright (c) 2001-2014 by Gambit Communications, Inc.
#
# script to calculate BIT string in hex by setting desired bits
# This is my sample run trying scenario of setting bits 3,7 and 11:
#
# ./mimicsh --nosession --script ~/mimic/scripts/set-bit.tcl
# org hexbit 00 00 00 00 00 00 00 00
# org bitstring 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
# set bits 3, 7, 11
# new hexbit 11 10 00 00 00 00 00 00
# new bitstring 00010001 00010000 00000000 00000000 00000000 00000000 00000000 00000000
# clear bits 3, 7, 11
# new hexbit 00 00 00 00 00 00 00 00
# new bitstring 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
#
#
#
#
proc bit {varName pos {bitval {}}} {
    upvar 1 $varName var
    if {[info exist var]} {set var 0}
    set element [expr {$pos/8}]
    while {$element >= [llength $var]} {lappend var 0}
    set bitpos [expr {1 << $pos%8}]
    set word [lindex $var $element]
    if {$bitval != ""} {
        if {$bitval} {
            set word [expr {$word | $bitpos}]
        } else {
            set word [expr {$word & ~$bitpos}]
        }
    }
    lset var $element $word
}
expr {($word & $bitpos) != 0}
}

proc bits bitvec {
    set res {}
    set pos 0
    foreach word $bitvec {
        for {set i 0} {$i<8} {incr i} {
            if {$word & 1<<$i} {lappend res $pos}
            incr pos
        }
    }
}
```

```

}
set res
}

proc hex2bin hex {
    set t [list 0 0000 1 0001 2 0010 3 0011 4 0100 \
        5 0101 6 0110 7 0111 8 1000 9 1001 \
        a 1010 b 1011 c 1100 d 1101 e 1110 f 1111 \
        A 1010 B 1011 C 1100 D 1101 E 1110 F 1111]
    regsub {^[0[xX]]} $hex {} hex
    return [string map -nocase $t $hex]
}

proc set_rbit {varName nth bit_val} {
    upvar 1 $varName var
    bit var [expr 7 - $nth % 8] $bit_val
}

# this procedure sets the "nth" bit in the BITSTRING "hex_val"
# to the bit value "bval"
proc set_bit {hex_val nth bval} {
    upvar 1 $hex_val hex
    set idx [expr $nth/8]
    set rem [expr $nth%8]
    set curr_byte [lindex $hex $idx]
    #puts "curr_byte $curr_byte"
    scan $curr_byte %x val
    set_rbit val $rem $bval
    set hex [lreplace $hex $idx $idx [format %02X $val]]
    #puts "int val $val"
    #puts "hex val [format %02X $val]"
}

# Example
# setting bit 3,7,11
set hexbit "00 00 00 00 00 00 00"
set org_hexbit $hexbit

puts "org hexbit $org_hexbit"
puts "org bitstring [hex2bin [join [split $org_hexbit]]]"

set_bit hexbit 3 1
set_bit hexbit 7 1
set_bit hexbit 11 1

puts "set bits 3, 7, 11"
puts "new hexbit $hexbit"
puts "new bitstring [hex2bin [join [split $hexbit]]]"

# clear bit 3,7,11 to get back to original
puts "clear bits 3, 7, 11"
set_bit hexbit 3 0
set_bit hexbit 7 0
set_bit hexbit 11 0

puts "new hexbit $hexbit"
puts "new bitstring [hex2bin [join [split $hexbit]]]"
puts ""
exit

```

Q. How do I simulate a table with an OCTET STRING INDEX?

```

> My script is filling up the vacmAccessTable (in the vacmMIBObjects)
> with values (that works fine). But when the script deletes the
> table rows, an error occurs.
>
> Here is the error that is printed as output from the script:
> {Failed INDEX simulation for table}
> while executing
> "mimic value instances 1.3.6.1.6.3.16.1.4.1.9"
>...

```

A. In your script, adding a new instance to the vacmAccessTable is not working correctly i.e. in the "mimic value add" command, the length of the OCTET STRING is missing. If you look in the Log window, you will see errors of the type:

```

ERROR DATE - management operation failed
DATE - command: value add
DATE - 26 6 1.3.6.1.6.3.16.1.4.1 1 99.99.0.3.1 v
DATE - table add entry failed
DATE - cannot add entry 99.99.0.3.1
DATE - index length (5) less than required (102)

```

According to the vacmAccessTable table definition it has four INDEX objects viz. vacmGroupName, vacmAccessContextPrefix, vacmAccessSecurityModel and vacmAccessSecurityLevel. The first object vacmGroupName is defined as SnmpAdminString(SIZE(1..32)), i.e. a variable length OCTET STRING. To instantiate an OCTET STRING object in the INDEX clause, the first octet should be preceded by the size of the string and then the actual octets.

This is explained in RFC2578 "Structure of Management Information Version2 SMIV2)":

(3) string-valued, variable-length strings (not preceded by the IMPLIED keyword): `n+1' sub-identifiers, where `n' is the length of the string (the first sub-identifier is `n' itself, following this, each octet of the string is encoded in a separate sub-identifier);

The following code snippet accomplishes this

```
#
# this proc converts an ASCII string to an OID.
# This is useful for OCTET STRING indices.
#
proc convert_ASCII_to_OID {str} {
    set i 0
    set tmp ""
    foreach c [split $str {}] {
        if {$i > 0} {
            append tmp .
        }
        append tmp [scan $c %c]
        incr i
    }
    return $tmp
}

# here is a sample usage
set controlGroup [convert_ASCII_to_OID "controlGroup"]
set strlen [llength [split $controlGroup .]]
mimic value add 1.3.6.1.6.3.16.1.4.1 $strlen.$controlGroup.0.3.1
```

Q. How do I simulate a sparse table?

```
> Is there a way to have Mimic skip over missing entries in a sparse table?
> By sparse table I mean that for some rows of a column in a table, there was
> no value supplied by the recorded device and so there is a blank value in
> the MIB walk file.
```

A. There is a difference between the case where an empty value is returned, or no value is returned at all. In the former, an empty value is a normal occurrence for OCTET STRING objects. The normal behavior for tables is that a row in the table exists or it does not exist, ie. either all columnar object instances are returned for a particular row index or none at all. There are some device agents which will have gaps in tables, ie. not all columnar objects for a row will be returned in a table traversal. This is highly irregular and frowned upon, since it unnecessarily complicates data retrieval for management applications, which is the reason why most standard MIBs explicitly prohibit table gaps. Nevertheless there are exceptions, and MIMIC can simulate this behavior by turning off the adaptive nature of the [MIMIC Value Space](#).

By default, the MIMIC Value Space is forgiving about missing information for MIMIC simulations. Eg. if the simulation of an object requires the value of a variable in the Value Space, if that value is missing, the Value Space will return some default value, rather than failing the simulation. This event is logged with a warning in the error log. This causes MIMIC to function even when there is missing information. You can turn off the adaptive Value Space, by setting the environment variable `MIMIC_ADAPTIVE_VSPACE` to 0 prior to running MIMIC. Doing so will not return object instances for which there is missing information in the Value Space.

- In the C shell do:
% setenv MIMIC_ADAPTIVE_VSPACE 0
- In the Bourne shell do:
MIMIC_ADAPTIVE_VSPACE=0; export MIMIC_ADAPTIVE_VSPACE
- On Windows, use the system Control Panel to set this environment variable.

The Simulator will show a confirmation in its initial startup messages in the log

```
adaptive vspace = 0
```

Q. How do I simulate a numeric object with a random value?

A. By default, the [basic simulation](#) of numeric objects in MIMIC returns "static" values, by using the value space lookup("`v`") [MIMIC simulation language function](#). This function will return the same value for successive queries of a MIB object instance. You can change this value by changing the variable `v` in the [MIMIC Value Space](#) at any point in time.

One way of making this default simulation "random" would be to periodically change the MIMIC Value Space, either through [the MIMIC scripting interface](#), [action scripts](#) or [timer scripts](#).

A simpler way would be to use the `random (low, high)` [MIMIC simulation language function](#). This function will return successively random values between a low and high limit. To apply this function to an object, use `simulation->Edit` [menu item](#) to edit the simulation expression for the desired object and change it to

```
SIMULATE { random ( lookup ("lo"), lookup ("hi")) }
```

which will make it return random values between the `lo` and `hi` variables for that object instance in the MIMIC Value Space. Compile this expression with `simulation->Compile` and the expression will take effect the next time you start the agent. You control the range of the random values through the MIMIC Value Space, just like for the default static expression.

As a short-cut, you can copy the `random.sdb` file from the `template/` directory of your installation to the `data/sim/YOUR-SIMULATION/MIB-OF-OBJECT/OBJECT.sdb` file. For example, if your simulation is called `mysim`, and the object is `ifOutQLen` in the `IF-MIB`, you would copy `template/random.sdb` to `data/sim/mysim/IF-MIB/ifOutQLen.sdb`.

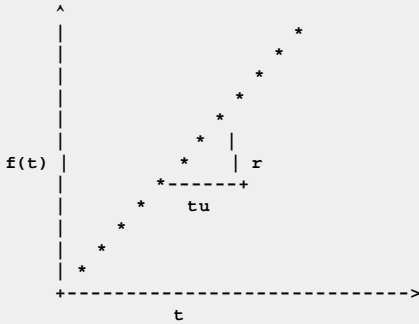
To control the random values for this object, create variables `lo` and `hi` in the Value Space and assign required values. Once you evaluate this object, or retrieve it's values via `SNMP GET*` requests, the values should be randomly returned in the specified range.

Q. How do I simulate a step function for a Counter object?

A. Given a Counter value which increases according to the linear function

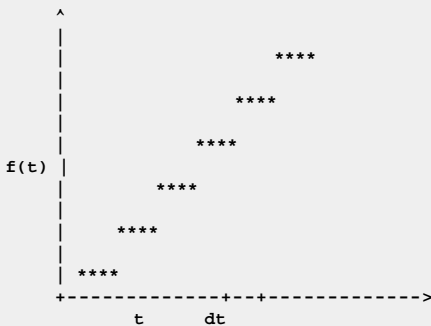
$$f(t) = t * r / tu$$

where r is the rate and tu is the timeunit, eg. with $r = 10$, $tu = 1$ the value increases 10 every second.
The graph of this function looks like this:



This constant linear function is simulated in MIMIC with the basic [constant_per_tu](#) simulation expression using the r and tu variables.

We can define a step function based on this linear function, so that every dt timeunits the value increases, like this:



This function is simulated in the MIMIC Simulation language with the simulation expression using the "r", "tu" and "dt" variables:

```
SIMULATE { ( lookup ("dt") == 0 ) ? 0 : constant_per_tu (lookup ("r"),
lookup ("tu")) - ( ( ( sysuptime() ) - ( ( sysuptime() ) / lookup ("dt") ) ) *
lookup ("dt") ) ) * lookup ("r") ) / ( 100 * ( ( lookup ("tu") == 0 ) ? 1 :
lookup ("tu") ) ) ) }
```

where "dt" is in TimeTicks (10 milliseconds). Eg. if you wanted the step to take every 5 seconds, dt would be 500. With this simulation expression, you can change any of the variables at any time with correct results.

As a short-cut, you can copy the `step.sdb` file from the `template/` directory of your installation to the `data/sim/YOUR-SIMULATION/MIB-OF-OBJECT/OBJECT.sdb` file. For example, if your simulation is called `mysim`, and the object is `ifInMulticastPkts` in the `IF-MIB`, you would copy `template/step.sdb` to `data/sim/mysim/IF-MIB/ifInMulticastPkts.sdb`.

Q. How can I get MIMIC to accept SETs on an object with different type?

```
> In my MIB I have the following definition:
>
> myAddressEntry ::= SEQUENCE {
>     myIndex Integer32,
>     myAddress SnpUDPAddress
> }
>
> and in my work scenario, my manager need set myAddress to "". But when I
> get this SET request, MIMIC warns me that
>
> WARN 03/21.09:01:19 snmpv2c.cc:841 - agent 73 SNMP SET failed.
>      03/21.09:01:19 mimic.cc:1314 - object myAddress : illegal size.
>
> I don't know how to set this object to empty.
```

A. By default MIMIC will do syntax validation check on a SET request for the set value. If it does not match according to the object definition (assuming `SnpUDPAddress` is a fixed-size OCTET STRING), the MIMIC agent will return an error. You can disable this default behavior of validation using the command `mimic agent set validate` as detailed in the [Simulator Guide](#), [MIMICShell commands section](#). The validation policy is a bitmask in which with the following bits (from LSB) check for

- type
- length (size)
- range
- access

A default value of 65535 does all validation checking.

For example, if you would like to disable size and range validation, you would unset bits 1 and 2, i.e. hex FFF9, or decimal 65529 with

```
mimicsh> mimic agent set validate 65529
```

If you would like to disable all syntax validation, use

```
mimicsh> mimic agent set validate 0
```

Q. Is it better to access a MIB object by name or OID?

A. Accessing a MIB object by name through the MIMIC APIs or GUI is a convenience (it is easier to remember a name, than a sequence of numbers). But, this convenience has a price in terms of efficiency: whenever a name is used, that name has to be converted to an OID using some sort of search.

MIMIC has optimized this search to take advantage of localization of reference: when referring to names, one usually refers to localized names, eg. within a table. For this reason, when you use a name, MIMIC will search first among the siblings of the last referenced MIB object. Only if the name is not found then will the search be from the beginning of the MIB.

Even with this optimization, the linear searches involved are much less efficient than referring to a MIB object by OID. This access is extremely fast in MIMIC.

Thus, while accessing MIB objects by name is a handy convenience specially for beginners, if you want to write efficient MIMIC scripts, we recommend to use OIDs. Even for this you can use mnemonics, eg. in Tcl

```
set sysName 1.3.6.1.2.1.1.5
mimic value eval $sysName 0
```

Q. What is the most efficient way to access many variables in the MIMIC value space?

A. The first performance improvement to implement is the access by OID instead of name, as detailed in the [previous section](#).

The MIMIC API features "multiple-variable" get and set operations on the MIMIC value space, which get and set multiple variables in one operation. The Tcl [mimic value mget](#) (and analogous calls in the other supported languages) is the equivalent to [mimic value get](#) for multiple variables. The same applies for the [mimic value mset](#) operation. These calls are specially efficient if you need to get/set many variables in your simulation.

We ran a performance test to access a large number of variables (on the order of 100,000), first with `mimic value get`, then with `mimic value mget` with 1, then 10, and finally 100 variables per call. We repeated the experiment for `mimic value mset`.

This matrix compares the performance of the operations (in variables per second):

Linux (1)						Windows (2)					
Command	Tcl	C++	Java (3)	Perl	Python	Command	Tcl	C++	Java (4)	Perl (4) (5)	Python (4)
mimic value get	17596	64641	51975	23196	21978	mimic value get	18345	54230	30479	15094	11106
mimic value mget 1	13992	45228	37879	14116	16075	mimic value mget 1	14461	43253	24618	10158	7854
mimic value mget 10	65790	228310	120480	56950	49800	mimic value mget 10	70180	188320	92760	26670	13140
mimic value mget 100	117600	400000	123300	80000	64500	mimic value mget 100	111500	266700	116400	34200	15400
Command	Tcl	C++	Java (3)	Perl	Python	Command	Tcl	C++	Java (4)	Perl (4) (5)	Python (4)
mimic value set	18106	59916	43554	24820	23223	mimic value set	16656	34602	22936	14447	13523
mimic value mset 1	13753	44287	38447	17715	19474	mimic value mset 1	13635	29360	18879	12121	12666
mimic value mset 10	56430	208330	125160	76750	67930	mimic value mset 10	35110	66670	49630	29360	29340
mimic value mset 100	94300	344800	111400	113800	91200	mimic value mset 100	50300	72700	52900	41800	41300

As can be seen, packing more variables per operation can result in orders of magnitude faster access to the MIMIC value space.

Notes:

1. Linux was tested on Fedora Core 19 running on a quad-core Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz

2. Windows was tested on Windows 8.1 on a quad-core Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz
3. Java has Unix domain transport support on Unix.
4. Java, Perl and Python have no local mode transport on Windows. See also [above](#).
5. On Windows, the numbers are for ActivePerl. See also [above](#).

Q. Why do I see multiple "mimicd" processes on Linux?

A. Since MIMIC is multi-threaded, and since on older versions of Linux each thread shows up in process listings by the `ps` and `top` utilities, you will see multiple `mimicd` entries. This is normal behaviour on older versions of Linux. With newer versions of `top`, use the `-H` command line option to see CPU usage by thread.

In combination with the MIMIC thread information that the [Diagnostic Wizard](#) prints, you can examine MIMIC thread CPU usage.

Q. How can I do dynamic DNS updates for my MIMIC agents?

A. You can use the `nsupdate` dynamic DNS client program to send dynamic DNS updates to your DNS server. The `nsupdate` program is provided with Solaris and Linux

Before you start, make sure your DNS server allows dynamic updates. For the [ISC BIND server](#) which is shipped on Solaris and Linux, you need to use the `allow-update` primitive in `named.conf`, eg. to allow all updates:

```
zone "yourzone.com" {
    // other configs here...
    allow-update {
        any;
    };
};
```

Test that `nsupdate` will work, eg. if a MIMIC agent with address 10.1.120.142 is running, from the shell do

```
% nsupdate -d
> local 10.1.120.142
> update add yoursystem.yourzone.com 86400 A 10.1.120.142
> send
Reply from SOA query:
;; ->HEADER<- opcode: QUERY, status: NXDOMAIN, id: 51587
;; flags: qr aa rd ra ; QUESTION: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION:
yoursystem.yourzone.com.      IN SOA

;; AUTHORITY SECTION:
yourzone.com.                86400 IN SOA ns.yourzone.com. hostmaster.yourzone.com.
2002092502 28800 7200 3600000 86400
```

```
Found zone name: yourzone.com
The master is: ns.yourzone.com
```

```
Reply from update query:
;; ->HEADER<- opcode: UPDATE, status: NOERROR, id: 55048
;; flags: qr ra ; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
```

```
> Destroy DST lib
Detach from entropy
```

```
% dig yoursystem.yourzone.com

;<<>> DiG 9.1.3 <<>> yoursystem.yourzone.com
;; global options: printcmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 26804
;; flags: qr aa rd ra ; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
yoursystem.yourzone.com.      IN      A

;; ANSWER SECTION:
yoursystem.yourzone.com.      86400  IN      A      10.1.120.142

;; AUTHORITY SECTION:
yourzone.com.                86400  IN      NS      nameserver.yourzone.com.

;; ADDITIONAL SECTION:
nameserver.yourzone.com.      86400  IN      A      192.9.200.70

;; Query time: 41 msec
;; SERVER: 192.9.200.70#53(192.9.200.70)
;; WHEN: Thu May 22 12:59:06 2003
;; MSG SIZE rcvd: 107
```

Then you can use `nsupdate` in an agent startup action script.

Q. How do I start MIMIC upon system boot?

... We will be having a power work done on our building this weekend. I want to setup our Linux MIMIC machine to automatically startup upon power up. Currently we have to start it manually with the mimicadmin id. What is the preferred method? Would the following entry in /etc/inittab suffice?

```
mimicadmin:3:::/mimic/7.11/bin/mimicd
```

A. Here are the answers for the different supported platforms:

On Linux

We recommend to start mimicd from /etc/rc.local. Anything in this file is executed after all other sysinit. For example, add the following lines towards the end of /etc/rc.local:

```
# start mimicd
# this code fragment assumes that MIMIC is installed in /usr/local/mimic
MIMIC_DIR=/usr/local/mimic
export $MIMIC_DIR
mimicd_args=""
mimicd_log=/tmp/mimic.`date '+%y%m%d%H%M%S'`.log
prog="./mimicd"
echo "Starting $prog >& $mimicd_log &"
cd $MIMIC_DIR/bin
$prog >& $mimicd_log &
Please make sure to replace the MIMIC_DIR variable with your MIMIC installation area absolute path.
```

Instructions to start MimicD.exe as a user-defined Windows service on Windows Server 2008 and Windows 7

1. Create a batch file in the MIMIC\bin folder with this content:

```
@echo off

REM MIMIC daemon startup script
REM useful when running as a service

REM *****CUSTOMIZATION BEGIN*****
REM *****PLEASE DO YOUR CUSTOMIZATIONS HERE*****

REM set proper MIMIC installation directory path
set MIMIC_DIR=C:\Apps\Mimic.1130

REM set proper MIMIC user's private directory
set MIMIC_PRIV_DIR=C:\mimic

REM set proper MIMIC LOG path
REM timestamp for the log
set hh=%time:~0,2%
if "%time:~0,1%"==" " set hh=0%hh:~1,1%
set yymmddhhmmss=%date:~12,2%%date:~4,2%%date:~7,2%%hh%%time:~3,2%%time:~6,2%

set MIMIC_LOG=C:\tmp\mimic.%yymmddhhmmss%.log
REM *****CUSTOMIZATION END*****

cd %~dp0

set MIMIC_ARGV=
:loop0
if %1!==! goto done0
set MIMIC_ARGV=%MIMIC_ARGV% %1
shift
goto loop0
:done0

REM execute MIMIC daemon
.\MimicD.exe %MIMIC_ARGV%
```

2. Run below command in COMMAND prompt to register MIMIC as a service:

```
C:>sc create MIMIC binPath="c:\Apps\Mimic.11.30\bin\MimicStart.bat -f agent.cfg -s"
DisplayName="MIMIC Simulator" start=auto obj=YOURMACHINE\YOURACCOUNT password=YOURPASSWORD
[SC] CreateService SUCCESS
```

NOTE: you may have to change these in the command line above:

- the path to the MimicStart.bat may be different in your installation.
- there are optional mimicd.exe arguments to the batch file above, e.g., -f agent.cfg -s. If you don't want to pass any arguments then remove those extra options.
- YOURMACHINE, YOURACCOUNT and YOURPASSWORD will be unique to your site

3. Optionally, run below command in COMMAND prompt to add a description to the MIMIC service:

```
C:>sc description MIMIC "Simulates IP networks"
[SC] ChangeServiceConfig2 SUCCESS
```


4. Now if you reboot the system, MIMIC will start automatically. There will be `MimicD.exe` in the process list. Just invoke MIMICView to connect to it. You can also run the MIMIC service manually going into Services window. It will complain that the service start fails with

```
Error 1053: The service did not respond to the start or control request in a timely fashion
```

but `MimicD.exe` starts successfully and can be seen in the process list. To solve this problem you need to use `INSTSRV` and `SRVANY` as documented below (contact us for detailed instructions - mention bug 4604).

5. If you want to remove the service, run this command:

```
C:\>sc delete MIMIC
[SC] DeleteService SUCCESS
```

NOTE:

- The `mimiclog` will be appended to the file provided in `MIMIC_LOG` for every `mimicd` session.
- Please be aware that this procedure requires changes in Windows Registry and we are not responsible for any mis-configuration that might occur on your system after the changes.

Q. Can MIMIC simulate SNMP looping?

```
>> Got a question that how to let MIMIC simulate an infinite loop when
>> walking snmp mib ?
>>
>> For example, I snmp recorded a cisco device, then I want to make an
>> infinite loop on one of its oid, in order that when my application to
>> snmpwalk this oid, MIMIC simulator will return data in an infinite loop.
```

A. We can interpret this problem in 2 different ways:

1. lexicographic ordering

By default, MIMIC simulates correct lexicographic ordering as most real-world agents do.. In order to simulate incorrect lexicographic ordering of OIDs, you can use the `mimic action jump` command in a GET action script.

For example, if you want incorrect ordering in the `ifTable` eg. by looping on the `ifIndex` column, you can assign this GET action script

```
mimic action jump 1.3.6.1.2.1.2.2.1.1
set gError -2
return
```

to the `ifDescr` object, and you will get a traversal such as:

```
$snmpwalk -v1 -cpublic 10.11.1.1 ifEntry
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
...
IF-MIB::ifIndex.35 = INTEGER: 35
IF-MIB::ifIndex.36 = INTEGER: 36
IF-MIB::ifIndex.1 = INTEGER: 1
Error: OID not increasing: IF-MIB::ifIndex.36 >= IF-MIB::ifIndex.1
```

Notice that the SNMP tool `snmpwalk` detects such OIDs out of lexicographic order and stops. Other buggy/faulty applications do not detect this and will loop indefinitely.

2. dynamic tables

One problem with some applications is that they inefficiently traverse dynamic tables, ie. tables whose indices change all the time. This condition can be simulated by growing the simulated table in a timer script that is invoked frequently. For example, invoking this timer script every millisecond:

```
catch {mimic value add ifEntry [clock clicks]}
```

will make the `ifTable` grow rapidly (which it normally does not in the real world). You need to worry about memory usage by the simulator with growing tables, so you should also purge old entries, as happens with many dynamic tables in the real world, eg. `RMON-MIB` tables. This sample script does that:

```
set instances [mimic value instances ifDescr]
set first [lindex $instances 0]
set next [expr [lindex $instances end] + 1]
catch {mimic value add ifEntry $next}
# assign desired values to object instances in the new row
catch {mimic value remove ifEntry $first}
```

GUI

Q. How do I change the fonts or colors of the MIMICView user interface?

A. Many parameters of the graphical user interface are configurable through the [Configuration Wizard](#) or with the `MimicView` resource file in the `bin/` directory. The file uses the format for the X Window System Resource Configuration.

For example, this line configures the font used to display the information for each of the agent instances:

```
*subnet_fr*Canvas.font: -adobe-courier-medium-r-normal--10-*-*-*--iso 8859-1
```

and these define the foreground and background colors of dialog fields:

```
*Dialog*Entry.background: yellow
*Dialog*Entry.foreground: red
```

If you want to change any predefined parameter, just edit the `MimicView` resource file and restart `mimicview`. We recommend preserving the old setting by commenting it out with a `!` at the beginning of the line.

If you need assistance with any of the parameters, contact Gambit Communications Technical Support (support@gambitcomm.com).

Q. How do I turn off the tooltip popups in MIMICView?

A. At MIMIC 7.20, the MIMICView GUI provides tooltip balloon popups when you mouse over buttons and dialog fields. You can disable these balloon popups with the `Help` tab in the Configuration Wizard. Uncheck the `Tooltips Balloon` checkbox and press `Apply`.

Check out [this video on YouTube](#).

Q. When I start `mimicview`, the log window does not appear. Why?

A. MIMIC is a client/server application, i.e., it consists of two complementary sets of programs. The server program `mimicd` needs to run before the client programs `mimicview` and `mimicsh` can do their work.

When you start `mimicview`, the GUI client, it starts the server program `mimicd` automatically, if it is not already running. If so, it also starts a logging window to dump the diagnostic output of the server program.

If `mimicd` is already running, `mimicview` just attaches to it. This could be the case if you stopped `mimicview` before and closed the log window before, without terminating the server program with `File->Terminate`.

You can look at the diagnostic output of the running `mimicd` by selecting `File->View Log` from `mimicview`.

Q. Why do I sometimes get "Cannot connect to the MIMIC daemon." when I start `mimicview`?

A. As explained in the previous section, MIMIC is a client/server application. The server program `mimicd` needs to run before the client programs `mimicview` and `mimicsh` can do their work.

When you start `mimicview`, the GUI client, it starts the server program `mimicd` automatically, if it is not already running. If so, it also starts a logging window to dump the diagnostic output of the server program.

After starting the `mimicd` daemon, `mimicview` waits for a little while for the daemon to initialize. Occasionally the daemon takes too long, and `mimicview` times out with the error message:

```
Cannot connect to the
MIMIC daemon.
Please check the log window.
Retry connecting?
Inspect the log window, and if there are no errors, simply choose Yes. It will reconnect to the already-started daemon.
```

Q. Even after Terminating Mimic, I see a couple `wish80` processes in the system. They just don't quit !

A. This is a known issue. On Windows, if you kill the MIMICShell using the 'x' the `wish80` still hangs on ... but if you used 'File->Exit' or just type 'exit' on the command prompt then you won't find the `wish80` in the process list.

You can just "End Process" any lingering `wish80` processes from the Windows Task Manager.

Q. How do I see MIB object details in the Value Browser?

A.

```
I am looking at a MIB object in the Value Browser, and see
their type information in the icon. Can I see more details
somehow?
```

The MIMICView [MIB Browser](#) that is displayed in the dialogs invoked by the `Agent->Value Space...`, `Agent->Evaluate...`, and `Agent->Browse MIB...` menu items shows the type, access, and enumeration values for the selected MIB object. In addition, when you click the `Details...` button, it will display the context sensitive portion of the MIB source code for the selected MIB object. The requirement for the `Details...` button to be enabled, is that the MIB source be imported and compiled into MIMIC, eg. for the MIBs that ship with MIMIC, by downloading the "MIB sources" update package from `Update Wizard`, or for new MIBs with the `MIB Wizard`.

Q. How do I list the SNMPv1 traps in the MIB Browser?

A.

```
When I generate traps the mib which is listed shows only
generic traps and no specific traps either for a Cisco router or a
IBM one.
```

The SNMPv1 traps are listed in the Generate Traps dialog in a (initially hidden) tree starting at the root, named "traps". You can open it by clicking on the '+' in the top left corner, then clicking again. A 'traps' tree will show, from which you can browse the SNMPv1 traps.

At SNMPv2, NOTIFICATIONS have been correctly defined in the OID name-space, and will appear in the MIB Browser under their defined branch in the `.iso.org` tree. (Also see [next entry](#) in this FAQ.)

Q. Why are you using different syntax for trap-OIDs and variable-OIDs ?

Why are you using different syntax for trap-OIDs and variable-OIDs ?
like:
.traps._iso._org._dod._internet._mgmt._mib-2._snmp
compared to:
.iso.org.dod.internet.mgmt.mib-2.snmp
That is not an SMI syntax standard, is it?

A. SNMPv1 traps do not fit into the OID hierarchy. Eg. there is no OID for the linkDown trap (it is sent with an Enterprise OID and a trap number). They fixed this for traps defined with the SNMPv2 NOTIFICATION construct.

In order to display SNMPv1 traps in the MIB Browser OID hierarchy, we artificially give an OID for the SNMPv1 traps that is the concatenation of 0, their Enterprise OID, and the trap number. We also decided to show SNMPv1 traps in their own OID tree and artificially labelled that tree with "traps".

All of this is only visible in the MIB Browser. Everything else behaves strictly to SNMPv1 and SNMPv2 standards, eg. the MIMIC daemon translates OIDs to correctly send SNMPv1 traps.

Q. Is there a way to assign icons to my devices?

Is there a way to assign icons to my devices?
I have created 4 device types which MIMIC view assigns
the icon of a terminal, and I would like to distinguish
each type of device with an icon.

A. Icons bitmaps are picked up from the bitmaps/ directory in the MIMIC installation or private area. The bitmap is associated with an agent by matching the first word in the string describing the device (as seen in the icon legend). So if your device is described on the screen as "Foo device blah blah ..." then you need a file called "Foo" in the bitmaps/ directory.

We use the XBM format for defining the bitmaps. The size needs to be 40x40.

Q. Mimic ran pretty slowly after 3 scripts and about 20 devices. Do you have any idea?

A. As for performance ... multiple scripts running in a loop will lead to sluggishness. The way to optimize this can be using a single script to do the job of 3 if possible. Also use as large an interval with the 'after' command as possible so that CPU is not hogged by a particular script.

Q. How does one turn off scripts running against an agent instance?

A. The scripts run in a separate Tcl shell (shows up on Windows as wish80.exe in the task list). So just terminating MIMIC still does not stop the script execution. You can instead use the Task Manager on Windows to select the 'Console' task and terminate it to get rid of the script.

Another way of stopping a MIMICshell script is to select "File->Exit" from the "Tk Console" window menu bar.

Q. Why is the "Device" button greyed out in the "Agent Configuration" dialog?

A. You need to stop the agent and then click configure. The device button is disabled if the agent is already running.

Q. Why do I get "Font specified in font.properties not found" messages when I run the Java-based tools?

A. This can be fixed by simply commenting out those 'adobe' related font in jre_install_path/lib/font.properties:

```
vi /home/shared/jre1.3.1_01/lib/font.properties

# Serif font definition
#
serif.0=-b&h-lucidabright-medium-r-normal--*-%d-*-%p*-iso8859-1
#serif.1=-symbol-medium-r-normal--*-%d-*-%p*-adobe-fontspecific

serif.italic.0=-b&h-lucidabright-medium-i-normal--*-%d-*-%p*-iso8859-1
#serif.italic.1=-symbol-medium-r-normal--*-%d-*-%p*-adobe-fontspecific

serif.bold.0=-b&h-lucidabright-demibold-r-normal--*-%d-*-%p*-iso8859-1
#serif.bold.1=-symbol-medium-r-normal--*-%d-*-%p*-adobe-fontspecific

serif.bolditalic.0=-b&h-lucidabright-demibold-i-normal--*-%d-*-%p*-iso8859-1
#serif.bolditalic.1=-symbol-medium-r-normal--*-%d-*-%p*-adobe-fontspecific

# SansSerif font definition
#
sansserif.0=-b&h-lucidasans-medium-r-normal-sans-*-%d-*-%p*-iso8859-1
#sansserif.1=-symbol-medium-r-normal--*-%d-*-%p*-adobe-fontspecific

sansserif.italic.0=-b&h-lucidasans-medium-i-normal-sans-*-%d-*-%p*-iso8859-1
#sansserif.italic.1=-symbol-medium-r-normal--*-%d-*-%p*-adobe-fontspecific
```

Q. How do I access from Windows the MIMICview GUI running on Linux?

A. You have 2 alternatives to display a MIMICview running on Linux on a display monitor on a Windows system.

The MIMICview GUI running on Linux can display locally on a [X Window System](#) server (henceforth called X server) to display on a monitor connected to the system, or remotely to a X server running on another system.

For alternative 1, you would run the MIMICview GUI X client on Linux and display on a X server running on the Windows system. There are many X server implementations for Windows, but we prefer [MobaXterm](#) or [Cygwin/X](#) . Your DISPLAY environment variable on Linux would have to be set to the remote system X display, eg. in bash `export DISPLAY=IPADDR:0.0` where IPADDR is the IP address of the remote system.

The second alternative is to access the Linux system with the Windows [Remote Desktop Protocol](#), which requires installing an RDP server on the Linux system. We have tried it according to these instructions for [xrdp](#) and it works well.

Compiler

Q. How do I change my simulation to add missing tables/objects?

A. When the Recorder creates a simulation, it only deals with the actual objects and tables it finds on the device. For any missing objects, an error message such as:

```
WARN 03/03.10:50:05 oiddb.cc:1119 - index simulation load failed ifStackEntry, continuing...
03/03.10:50:05 simdb_load.cc:305 - cannot find index file for ifStackEntry.
03/03.10:50:05 searchpath.cc:348 - data/sim/hp-lanprobe.random/IF-MIB/ifStackEntry.idb not in search path
```

will be generated, and no values will be returned by MIMIC.

If you have a walkfile, you can use the [Simulation Wizard](#) to add entries to a table.

If you don't have the walkfile, to add a table, the first thing to do is to change the simulation file (e.g., in MIMICView with `simulation->Edit`) to add [simulation expressions](#) for the table objects.

For the `ifStackEntry` table, you would cut and paste the following objects from the MIB file `rfc2863.mib` (you can get the MIB source file name from [Appendix B: Pre-compiled MIBs](#)), and then reduce to the minimal format with simulation expressions:

```
ifStackEntry OBJECT-TYPE
SIMULATE { table.static }

ifStackHigherLayer OBJECT-TYPE
SIMULATE { table.index }

ifStackLowerLayer OBJECT-TYPE
SIMULATE { table.index }

ifStackStatus OBJECT-TYPE
SIMULATE { lookup ("v") }
```

Once you compile this simulation (eg. in MIMICView with `simulation->Compile`), you are able to simulate this table.

The next step is to add entries to the table (eg. in MIMICView with `Agent->Value space...`) and assign values to objects.

Q. Can I redefine a trap to send extra variables?

```
> We have a problem with a linkDown trap. We have managed to successfully
> send a standard linkDown trap (i.e. a generic value of 2) to our
> management station. This trap is sent with the variable ifIndex.
> However, our problem is that our management station is configured to
> accept the linkDown trap with extra variables - namely ifName and
> ifDescr. As a result, our management station returns an error stating
> that variables are missing when it receives the standard linkDown trap.
> We were wondering if it would be possible to send the linkDown trap with
> extra variables? We tried to re-define the trap with extra variables but
> MIMIC seemed to persist with only sending ifIndex. Is MIMIC "hard-coded"
> to only send ifIndex for traps of generic value 2 (ie. linkDown trap)?
> Is it possible to add an Enterprise ID (similar to a specific trap) with
> linkDown.
> Does MIMIC store information on the traps that an agent instance can
> send? If so, where does it store this information?
```

A. The syntax information about traps comes from the MIB definition, in particular the `TRAP-TYPE` and `NOTIFICATION-TYPE` clauses. These clauses define the version of the trap to be sent (for details see [above](#)) and the variables to be sent with the trap.

If you wanted to temporarily send extra variable bindings with the trap, you could define them at trap generation time. In the MIMICView [Generate Traps](#) dialog use the `Advanced` tab to add extra variables. In MIMICShell, use [mimic value set](#) to control trap generation with the `o` variable in the Value Space.

If you want to change a trap definition permanently, you would edit the MIB source file, compile it into MIMIC, from then on the new trap syntax is used.

Eg. if you wanted to change the trap definition of `linkDown`, you would need to change the `IF-MIB` definition as follows:

- 1) Use [MIB --> Import](#) to import the `mibs/rfc1573.mib` file from the MIMIC distribution. Please save it as the same filename and enterprise (no name). This file is saved in your private area and will effectively override the MIB supplied with MIMIC.
- 2) Use [MIB --> Edit](#) to edit the file, or edit with your favorite editor.
- 3) Use [MIB --> Compile](#) to compile this new MIB file.

You cannot redefine this trap in a different MIB, because your device configuration specifies a list of MIBs, and the definition in the original `IF-MIB` will have precedence (since it likely comes earlier in the device configuration as shown in the [Simulation --> Devices](#) dialog). Only by redefining it

in the same MIB will you override the original definition.

Q. How to remove unwanted MIBs?

```
> 1. When I import MIB files, it will create a MIB directory for me in
> the MIB browser. Question 1: How to remove unwanted MIB directories?
> I tried to modify the /mimic/data/mimic.dir file. Because I dont know
> the format of the file, with the modified mimic.dir actually caused the
> MIMIC can't be started. Is there a easy way to remove the unwanted MIB
> directory through GUI?
```

A. If your MIMICView has a MIB->Manage MIBs... menu item, you can use it. Else use the following instructions:

MIMIC accesses the compiled MIBs using the file `mimic.dir` as the directory listing of the MIBs which are compiled into MIMIC. The general format of `data/mibs/mimic.dir` is as follows :

```
<version>
"<name-of-mib-1>
"<data/mibs/<db-file-path>
<number-of-top-oids>
<top-oid-1>
<top-oid-2>
...
<top-oid-n>
"<name-of-mib-2>
"<data/mibs/<db-file-path>
<number-of-top-oids>
<top-oid-1>
<top-oid-2>
...
<top-oid-n>
and so on ...
```

To remove the entries you don't want you need to remove the block corresponding to those MIBs and also decrement the topmost number by the same amount. PLEASE backup `mimic.dir` before starting any changes.

Once you have done the above you can delete the `.db` files in the `data/mibs` directory to complete the cleanup.

Q. Why does the compiler emit errors when others don't?

The MIMIC compiler will alert the user to problems with SMI syntax in their MIB, rather than silently ignoring them. Wouldn't you rather fix your MIB to follow proper SMI syntax, rather than finding errors later, which are invariably more expensive to fix?

For a list of frequent problems in MIBs, see [this white-paper at InterWorking Labs](#) or [this Muonics page](#).

Q. Where can I get the Extreme Networks MIBs?

A. After MIMIC 5.10, you can use Update Wizard to install the Extreme MIB update. Otherwise, you can download the MIBs from <http://www.extremenetworks.com/support/documentation.asp> Then use the MIB Wizard to import them into MIMIC.

Q. Where can I get the Foundry Networks MIBs?

A. The Foundry Networks MIBs are available as an update to MIMIC which you can install with the [Update Wizard](#).

Q. Where can I get the Alteon MIBs?

A. The Alteon MIBs are available as an update to MIMIC which you can install with the [Update Wizard](#).

Q. Where can I get the Juniper MIBs?

A. After MIMIC 5.10, you can use Update Wizard to install the Juniper MIB update. Otherwise, you can download the latest MIBs from http://www.juniper.net/techpubs/software/index_mibs.html Then use the MIB Wizard to import them into MIMIC.

The MIBs containing TRAP definitions (`v1_traps*.txt`) need minor modifications in order to compile with MIMIC:

- `v1_traps_bgp.txt`
Add this to the IMPORTS clause (eg. before TRAP-TYPE line):

```
-- GAMBIT: added imports for compile purposes
bgpPeerLastError, bgpPeerState      FROM RFC1269-MIB
```

- `v1_traps_chassis.txt`
Add this to the IMPORTS clause (eg. before TRAP-TYPE line):

```
-- GAMBIT: added import for compile purposes
jnxContentsContainerIndex, jnxContentsL1Index,
jnxContentsL2Index, jnxContentsL3Index, jnxContentsDescr
FROM JUNIPER-MIB
```

- `v1_traps_mpls.txt`
Add this to the IMPORTS clause (eg. before TRAP-TYPE line):

```
-- GAMBIT: added imports for compile purposes
```

```
mplsLspName, mplsPathName FROM MPLS-MIB
```

- v1_traps_ospf.txt
Add this to the IMPORTS clause (eg. before TRAP-TYPE line):

```
-- GAMBIT: added imports for compile purposes
ospfRouterId, ospfExtLsdbLimit,
ospfLsdbAreaId, ospfLsdbType,
ospfLsdbLsid, ospfLsdbRouterId,
ospfIfIpAddress,ospfAddressLessIf,
ospfIfState, ospfVirtIfAreaId,
ospfVirtIfNeighbor, ospfVirtIfState,
ospfNbrIpAddr, ospfNbrAddressLessIndex,
ospfNbrRtrId, ospfNbrState,
ospfVirtNbrArea, ospfVirtNbrRtrId,
ospfVirtNbrState, ospfConfigErrorType,
ospfPacketType, ospfPacketSrc
FROM OSPF-MIB
```

Q. Where can I get the Avaya MIBs?

A. After MIMIC 5.30, you can use Update Wizard to install the Avaya MIB update. Otherwise, you can search for the latest MIBs on Google. Then use the MIB Wizard to import them into MIMIC.

Q. Where can I get the Siemens MIBs?

A. After MIMIC 5.30, you can use Update Wizard to install the Siemens MIB update. Otherwise, you can search for the latest MIBs on Google. Then use the MIB Wizard to import them into MIMIC.

Q. Where can I get the Brocade MIBs?

A. After MIMIC 5.30, you can use Update Wizard to install the Brocade MIB update. Otherwise, you can search for the latest MIBs on Google. Then use the MIB Wizard to import them into MIMIC.

Q. Where can I get the Netopia MIBs?

A. After MIMIC 5.30, you can use Update Wizard to install the Netopia MIB update. Otherwise, you can search for the latest MIBs on Google. Then use the MIB Wizard to import them into MIMIC. You'll have to do a minor fix to change "DdpAddress" to "DdpNodeAddress", as standardized in the AppleTalk MIB (RFC 1742).

Q. Where can I get the WiMAX MIBs?

A. You can download the [802.16f](#) and [802.16i](#) MIBs from IEEE directly. We cannot distribute them due to copyright restrictions.



[<< Previous Section](#) [Next Section >>](#)

Table of Contents

[Table of Contents](#)

[MIMIC Release Notes](#)

- [New Functionality in this release](#)
- [New Functionality in Previous Releases](#)

[Windows Installation Instructions](#)

- [Overview](#)
- [Account Privileges](#)
- [Firewalls](#)
- [Disk Space](#)
- [Assigning IP Addresses](#)
- [Duplicate IP Address](#)
- [Media Sense on Windows 2000](#)
- [Crashes](#)
- [Known Problems](#)

[Quick Start Guide](#)

- [Chapter 1: Overview](#)
- [Chapter 2: Using MIMIC Virtual Lab](#)
 - [Starting the lab](#)
 - [Accessing a device](#)
 - [Running an exercise](#)
- [Chapter 3: Troubleshooting](#)
 - [Online Help](#)
 - [Inspect the Log](#)
 - [Common Errors](#)
 - [Common Questions](#)
 - [Crashes](#)
- [Chapter 4: Background](#)
 - [Important Concepts](#)

[User Guide](#)

- [Overview](#)
- [User Reference](#)
 - [Startup](#)
 - [File Menu](#)
 - [Lab Menu](#)
 - [Device Menu](#)
 - [Connection Menu](#)
 - [Help Menu](#)

[JRE](#)

[Appendix A: IOS Commands](#)

[Appendix C: Common Error Messages](#)

[Appendix D: Frequently Asked Questions](#)

[<< Previous Section](#) [Next Section >>](#)



[<< Previous Section](#) [Next Section >>](#)

Release Notes

New functionality in this release

MIMIC Virtual Lab CCNA v5.30

Based on [MIMIC Simulator 10.30](#)

New functionality in previous releases

MIMIC Virtual Lab CCNA v5.20

Based on [MIMIC Simulator 10.20](#)

MIMIC Virtual Lab CCNA v5.10

Based on [MIMIC Simulator 10.10](#)

MIMIC Virtual Lab CCNA v4.40

MIMIC Virtual Lab CCNA Plus v4.40

MIMIC Virtual Lab Enterprise v5.40

Based on [MIMIC Simulator 9.40](#)

MIMIC Virtual Lab CCNA v4.30

MIMIC Virtual Lab CCNA Plus v4.30

MIMIC Virtual Lab Enterprise v5.30

Based on [MIMIC Simulator 9.30](#)

MIMIC Virtual Lab CCNA Plus v4.20

MIMIC Virtual Lab Enterprise v5.20

Based on [MIMIC Simulator 9.20](#)

MIMIC Virtual Lab CCNA v4.10

MIMIC Virtual Lab BSCI v5.10

Based on [MIMIC Simulator 9.10](#)

MIMIC Virtual Lab CCNA v3.42

Tech2000 Virtual Lab CCNA v3.41

Based on new ICND2 lab guide.

Based on [MIMIC Simulator 8.41](#)

1. [MIMIC Virtual Lab CCNA Plus v2.31](#)

[MIMIC Virtual Lab Enterprise v4.31](#)

Based on [MIMIC Simulator 8.31](#)

1. [MIMIC Virtual Lab CCNA v2.30](#)

[MIMIC Virtual Lab BSCI v4.30](#)

Based on [MIMIC Simulator 8.30](#)

1. [MIMIC Virtual Lab CCNA v2.20](#)

[MIMIC Virtual Lab CCNA Plus v2.20](#)

[MIMIC Virtual Lab BSCI v4.20](#)

[MIMIC Virtual Lab Enterprise v4.20](#)

Based on [MIMIC Simulator 8.20](#)

1. [MIMIC Virtual Lab CCNA v2.00](#)

[MIMIC Virtual Lab CCNA Plus v2.00](#)

[MIMIC Virtual Lab BSCI v4.00](#)

[MIMIC Virtual Lab Enterprise v4.00](#)

Based on [MIMIC Simulator 8.00](#)

1. [MIMIC Virtual Lab CCNA v1.60](#)

[MIMIC Virtual Lab BSCI v3.30](#)

[MIMIC Virtual Lab Enterprise v3.30](#)

[OEM Versions of MIMIC Virtual Lab](#)

Based on [MIMIC Simulator 7.31](#)

add/remove links between devices

1. [MIMIC Virtual Lab CCNA v1.50](#)

[MIMIC Virtual Lab Cisco v4.20](#)

[MIMIC Virtual Lab BSCI v3.20](#)

[MIMIC Virtual Lab Enterprise v3.20](#)

OEM Versions of MIMIC Virtual Lab

Based on [MIMIC Simulator 7.20](#)

disconnect/reconnect links between devices

SNMPv1 configuration via "snmp-server" command

dynamic command aliases via "alias" command

h. MIMIC Virtual Lab CCNA v1.11

MIMIC Virtual Lab Cisco v4.10

MIMIC Virtual Lab BSCI v3.10

MIMIC Virtual Lab Enterprise v3.10

Based on [MIMIC Simulator 7.00](#)

telnet client simulation via "telnet" command

remote logging through SYSLOG via "logging" command

Cisco TRAP generation via "snmp-server" command

Console vs. telnet simulation

i. MIMIC Virtual Lab Cisco v3.10

MIMIC Virtual Lab BSCI v2.10

MIMIC Virtual Lab Enterprise v2.10

Based on [MIMIC Simulator 6.30](#)

Downloadable MPLS lab.

j. MIMIC Virtual Lab Cisco v3.00

MIMIC Virtual Lab BSCI v2.00

MIMIC Virtual Lab Enterprise v2.00

Based on [MIMIC Simulator 6.20](#)

Support the Linux and Solaris platforms.

Allow to change the network address for the lab.

k. MIMIC Virtual Lab Cisco v2.10

MIMIC Virtual Lab BSCI v1.10

MIMIC Virtual Lab Enterprise v1.00

Based on [MIMIC Simulator 6.10](#)

l. Cisco Lab I v2.00

Based on [MIMIC Simulator 6.00](#)

Exercises to dynamically alter the lab.

MIB Browser.

Load multiple lab configurations.

Update labs over the Internet.

1. **Cisco Lab I v1.10**

Based on MIMIC Simulator 5.99

1. **Cisco Lab I v1.00**

First virtual lab containing 4 routers, 2 switches, 5 end systems

Based on MIMIC Simulator 5.46

[<< Previous Section](#) [Next Section >>](#)



Windows Installation Instructions

Table of Contents

- [Overview](#)
- [Account Privileges](#)
- [Firewalls](#)
- [Disk Space](#)
- [Assigning IP Addresses](#)
- [Duplicate IP Address](#)
- [Media Sense on Windows 2000 and newer](#)
- [Windows Vista](#)
- [Crashes](#)
- [Known Problems](#)

1. Overview

MIMIC Virtual Lab runs on

- Windows NT 4.0 Service Pack 4 or newer
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista Business Edition
- Windows 7

with at least [Java Runtime Environment \(JRE\) 1.6](#).

Notice that although MIMIC Simulator supports the other versions of Windows (95, 98, Me, NT SP 3), the limitations are too severe to run MIMIC Virtual Lab on them.

The following are some of the most common problems encountered on Windows, and their fixes:

1. Account Privileges

On Windows NT, 2000, XP or Server 2003, you need to install MIMIC from a user account with Administrator privileges, since the install script needs write access to restricted parts of the Registry.

You also need to run MIMIC from an account with Administrator rights, since it uses special privileges to add additional IP addresses to the machine you are running on.

We recommend establishing a separate user account with Administrator privileges for running MIMIC. This strictly controls who has access to this functionality.

1. Firewalls

Due to pervasive security attacks against Windows systems connected to the Internet, it has become common to run a software firewall on recent versions of Windows.

MIMIC will coexist with a software firewall, provided that the firewall is configured to recognize MIMIC as a program allowed to access the network. MIMIC will, due to its very nature of simulating networked components, open network sockets and communicate with external applications (eg. network management applications, telnet clients, etc).

There are certain components of MIMIC that will access the Internet (eg. specific web sites to determine software updates, etc).

1. Disk Space

MIMIC creates a lot of small files for its simulations. This is handled efficiently on modern filesystems, such as NTFS and Unix file systems. On the other hand, the FAT (or FAT16) filesystem on all Windows 95 (and some Windows 98 or NT) systems is notoriously inefficient, and you will use 10 to 100 times more disk space on a FAT file system. (The FAT32 file system on Windows 98 is also inefficient, but it is acceptable.) To find out what file system you have installed on your Windows system, select the `Properties` dialog on your partition from the Windows Explorer.

1. Assigning IP Addresses

MIMIC requires at least one operational network interface card (NIC). On Windows NT, 2000, XP or Server 2003, as on the Unix platforms, MIMIC dynamically assigns IP addresses when starting each agent instance.

Duplicate IP Address

If Windows detects that an IP address on one of its Network Interface Cards (NICs) conflicts with another system (duplicate IP address), then it tries to resolve this problem by shutting down the NIC and displays a message such as:

```
The System has detected an IP address conflict with another system on the network. The local interface has been disabled. More details are available in the system event log. Consult your network administrator to resolve the conflict.
You must not have duplicate IP addresses on a connected network, neither with MIMIC or otherwise.
```

NOTE: on Windows 2000, XP or Server 2003, the agent will not start and will print an error message in the [Log](#).

Media Sense on Windows 2000 and newer

Newer versions of Windows (Windows 2000 onwards) have a TCP/IP feature whereby it can sense if a NIC is actually connected to the network. By default, a NIC is disabled if it is not found to be on the network, which prevents agents from starting in MIMIC. There is a way to disable this behaviour so that you can work on standalone Windows machines. Attached is the [Microsoft KB](#) article on this topic... Please remember to make a copy of your registry before making any changes just to be on the safe side.

NOTE Windows Vista and newer have no way to disable this feature. You can only run MIMIC on Windows Vista and newer with the system connected to a network.

How to Disable Media Sense for TCP/IP in Windows 2000

The information in this article applies to:

- a.. Microsoft Windows 2000 Advanced Server
- b.. Microsoft Windows 2000 Datacenter Server
- c.. Microsoft Windows 2000 Professional
- d.. Microsoft Windows 2000 Server

SUMMARY

Windows 2000 contains the "Media Sensing" feature. You may use this feature on a Windows 2000-based computer using Transmission Control Protocol/Internet Protocol (TCP/IP) to detect whether or not your network media is in a "link state". A "link state" is defined as the physical media connecting or inserting itself on the network. For example, assuming a 10bt or 100bt physical media, Ethernet network adapters and hubs typically have a "link" light to indicate the current connection status. This is the same condition in which Windows 2000 can detect a link. Whenever Windows 2000 detects a "down" state on the media, it removes the bound protocols from that adapter until it is detected as "up" again. There may be situations where you may not want your network adapter to detect this state, and you can configure this by editing the registry.

NOTE: 10b2 or coaxial (RG-58) Ethernet cable is not a connection-based media. Because of this, Windows 2000 does not attempt to detect a "connect" state if this type of cabling is used.

MORE INFORMATION

WARNING: Using Registry Editor incorrectly can cause serious problems that may require you to reinstall your operating system. Microsoft cannot guarantee that problems resulting from the incorrect use of Registry Editor can be solved. Use Registry Editor at your own risk.

For information about how to edit the registry, view the "Changing Keys and Values" Help topic in Registry Editor (Regedit.exe) or the "Add and Delete Information in the Registry" and "Edit Registry Data" Help topics in Regedt32.exe. Note that you should back up the registry before you edit it. If you are running Windows NT or Windows 2000, you should also update your Emergency Repair Disk (ERD).

To prevent your network adapter from detecting the link state:

NOTE: NetBEUI and IPX do not recognize Media Sense.

1.. Use Registry Editor (Regedt32.exe) to view the following key in the registry:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters

Add the following registry value:

Value Name: DisabledDHCPMediaSense

Data Type: REG_DWORD -Boolean

Value Data Range: 0, 1 (False, True) Default: 0 (False)

Description: This parameter controls DHCP Media Sense behavior. If you set this value data to 1, DHCP, and even non-DHCP, clients ignore Media Sense events from the interface. By default, Media Sense events trigger the DHCP client to take an action, such as attempting to obtain a lease (when a connect event occurs), or invalidating the interface and routes (when a disconnect event occurs).

2.. Restart your computer.

NOTE: There are some side effects of disabling the "Media Sensing" feature. For example, if you have a machine with two network adapters, and you have the "Media Sensing" feature enabled, if one network adapter does not work, it is unbound, and associated routes are removed so that all traffic goes through the other network adapter (assuming a default gateway is there). Also, if you are a roaming (portable) user, the "Media Sensing" feature is what provides the ability to connect to any network and have everything work, without restarting, release and renewing, and so on. After disabling Media Sense and restarting, Windows 2000 still shows the "Network Disconnected" icon on the TaskBar and the 'ipconfig' command still shows a "Media State : Cable Disconnected" message when the cable is disconnected. However, the Network Interface is bound to TCP/IP and you can verify this by looking at the route table --you can use the "route print" command-- which shows the interface IP address (you are also able to ping the IP address assigned to the NIC).

END

1. Windows Vista

1. User Account Control

Windows Vista has the new User Account Control feature, which impacts the running of MIMIC. For details, consult this [Analysis of the Windows Vista Security Model](#) from Symantec. In order to enable to run MIMIC on Vista, you have 2 options:

- Disable User Account Control
This turns UAC off globally. NOTE: do this only if you are aware of the implications of this action.
 - Open User Accounts Via Start->Control Panel->User Accounts->User Accounts.
 - Click on Turn User Account Control on or off
 - Clear the checkbox for Use User Account Control (UAC) to help protect your computer.
 - Click Ok
 - A dialog will popup prompting you to Restart Now or Restart Later. Choose appropriately. User Account Control will be disabled once the system reboots.
- Run MIMIC with User Account Control enabled
This involves changing the access control level of the MIMIC programs.
 - Change the privilege level of MIMIC Virtual Lab application in the MIMIC Start Program group using the following steps:
 - Click Start->All Programs->MIMIC Virtual Lab ... x.xx
 - Move the cursor to the MIMIC Virtual Lab ... entry
 - Right click and select Properties.
 - In the Compatibility tab, check Run this program as an administrator
 - Once this is done, MIMIC Virtual Lab can be started as above or by running the VLABx.bat script in the bin folder of the MIMIC installation.
 - If MimicD.exe will be run directly, set the privilege level of it using the following steps:
 - In Windows Explorer, select it.
 - Right click and select Properties.
 - In the Compatibility tab, check Run this program as an administrator

1. Duplicate Address Detection

On Windows Vista, the new TCP/IP stack tries to do "duplicate address detection" by default. This prevents MIMIC from starting agents, because IP aliasing is delayed, and even with a workaround in our software would unacceptably slow down the starting of agents. To correctly workaround the problem, you need to disable "duplicate address detection" for the network interface using the Windows netsh utility:

```
netsh interface ipv4 set interface "name or index" dadtransmits=0
```

The interface name and index info can be obtained by

```
netsh interface ipv4 show interfaces
```

For example:

```
H:\>netsh interface ipv4 show interfaces
```

Idx	Met	MTU	State	Name
1	50	4294967295	connected	Loopback Pseudo-Interface 1
7	20	1500	connected	Local Area Connection

```
H:\>netsh interface ipv4 set interface "7" dadtransmits=0
```

i. Vista Power Management

The default power options will put the Windows Vista system to sleep after 1 hour of inactivity. To disable this, perform the following:

Open Power Options using Control Panel->System and Maintenance->Power Options.

Change Preferred Plan from Balanced to High Performance.

Verify by clicking on Change Plan Settings for High Performance. Ensure that Put the computer to sleep setting is Never.

l. Program Compatibility Assistant

After the install is completed or aborted, the Program Compatibility Assistant may prompt with the message

This program might not have installed correctly.

Please select This program installed correctly if the install completed. Else, select Cancel.

j. Crashes

Prior to Windows Vista, crashes can be analysed post-mortem using the [Dr. Watson Tool](#) crash dumps. This requires that Dr. Watson be enabled to handle any application exceptions on the system.

To install Dr. Watson as the default exception handler :

- Click Start->Run.
- Type `drwtsn32 -i`
- Click Ok.

A subsequent crash should popup the Dr. Watson dialog. Search for the following files in the Windows directory (this location can be changed using the Dr. Watson GUI) : `drwtsn32.log` and `user.dmp` . Send these to Gambit Technical Support (support@gambitcomm.com).

On Windows Vista, by default the Problem Reports and Solutions feature handles program crashes. Crash information, including minidumps when available, is automatically sent to Microsoft.

You can check if a MIMIC program crashed and if minidumps are available. Please use the following steps to extract any available MIMIC program crash data and forward it to Gambit Technical Support (support@gambitcomm.com).

- Go to Problem Reports and Solutions using Control Panel->System and Maintenance->Problem Reports and Solutions
- Click on View problem history
- Find MIMIC programs in the list
- Double click on an entry to view the problem details
- If there is a Files that help describe the problem section, click on the View a temporary copy of these files link below that section
- The files will be extracted into a temporary directory and an Explorer window will be opened to view them
- Forward these to Gambit Technical Support

If the Problem Reports and Solutions settings are changed to check with the user before sending the crash information to Microsoft, Windows Vista will prompt the user when a program crash occurs. If you choose Close the program, no additional details are generated. If you choose Check online for a solution and close the program, crash data may be saved.

Microsoft's [Debug Diagnostic Tool](#) version 1.1 onwards may be used on Windows Vista to handle program crashes. If this is installed, please use the following steps to generate crash data.

- Open Debug Diagnostic Tool
- Click on the Rules tab
- Click on Add Rule button
- In the Select Rule Type dialog, choose Crash and click on Next
- In the Select Target Type dialog, choose A specific program and click on Next
- In the Select Target dialog, browse the process list, select `MimicD.exe` and click on Next
- In the Advanced Configurations (Optional) dialog, change the number of userdumps as needed and click on Next
- In the Select Dump Location and Rule Name (Optional) dialog, change the path and name as needed and click on Next
- In the Rule Completed dialog, choose Activate the rule now and click on Finish

When a crash occurs, the Rules tab in Debug Diagnostic Tool will show the userdump count. Forward the available files from the configured dump location to Gambit Technical Support.

. Known Problems

We are constantly working to remove limitations, but currently we know of the following:

- On Windows Vista and newer you cannot run MIMIC on a standalone PC (not connected to a network). This is due to the OS having removed the ability to disable media sense. We have found no workaround for this limitation.
- MIMIC Virtual Lab relies on the native telnet client program on the platform it runs on to connect to the IOS simulations. On Windows NT, there are bugs in the telnet client that prevent interrupting commands such as ping with CTL-Shift-6. This works fine on later Windows versions and all Unix versions.

On Windows Vista and later, the telnet client is not enabled by default. You need to enable it with Control Panel->Programs->Programs and Features->Turn Windows features on or off.

On 64-bit versions of Windows, the telnet client is not launched correctly. The workaround for this problem is to copy the Windows telnet executable from c:\windows\system32\telnet.exe to the bin/ folder of the MIMIC Virtual Lab install area.

- On Windows NT, if the host is using DHCP to obtain its address, no agent instance can use that same address to export a MIB. The problem is that on stopping the agent this address is deleted which shuts off the TCP/IP services (ftp/telnet/internet). To restore working you either need to REBOOT or start the agent on the DHCP assigned address again (keep it running).
- On Windows, certain network interface cards have limitations supporting multiple IP addresses. In particular, some adapters and/or drivers from 3com have been giving us trouble (eg. 3C905-TX or 3Com 3C90x Ethernet Adapter). The symptom is that a small number of agent instances can be started and polled correctly, but connectivity is lost to the box when starting more.
- On Windows, certain software is incompatible with MIMIC. In particular, if Novell Client is running, your machine may hang after starting and stopping a small number of agents (the System task will use 99+% of CPU). Just unchecking the box in Local Area Connection Properties is not sufficient - you have to uninstall it.

This problem is unrelated to MIMIC. Any test program (eg. ifdiag shipped with MIMIC) which uses the Windows API to register/unregister network addresses will reproduce the problem.

- MIMIC will not run inside virtual machine software such as VMWare.

In case of difficulties, please contact Gambit Technical Support (support@gambitcomm.com).

Obviously, Windows is a trademark of Microsoft. All other product and brand names are trademarks or registered trademarks of their respective holders.

[<< Previous Section](#) [Next Section >>](#)



Quick Start Guide

Preface

This guide is a quick overview to using MIMIC Virtual Lab. It assumes that you are familiar with networking and network management concepts, particularly Simple Network Management Protocol (SNMP), Management Information Base (MIB), and telnet.

Organization

[Using MIMIC Virtual Lab](#) is recommended for use with an installed and running MIMIC to demonstrate the overall functionality of the product.

[Troubleshooting](#) guides you through solving problems with MIMIC Virtual Lab.

[Important Concepts](#) contains useful introductory definitions.

Typography Conventions

Normal	Text
Typewriter	Computer output; names of functions and data types
Typewriter	Interface components; menus, buttons and entry fields
<i>Italics</i>	Values you can input; variable names, numbers, strings
Bold Normal	What you have to type correctly, for example, filenames, Unix commands, function names, command-line entries

Table of Contents

- [Chapter 1: Overview](#)
- [Chapter 2: Using MIMIC Virtual Lab](#)
 - [Starting the lab](#)
 - [Using the GUI](#)
 - [Shortcuts](#)
 - [Accessing a device](#)
 - [Running an exercise](#)
- [Chapter 3: Troubleshooting](#)
 - [Online Help](#)
 - [Known Problems](#)
 - [Inspect the Log](#)
 - [Common Errors](#)
 - [Common Questions](#)
 - [Crashes](#)
- [Chapter 4: Background](#)
 - [Important Concepts](#)
 - [What Is a Device Instance?](#)
 - [What Is a Simulation?](#)
 - [What Is an Agent Instance?](#)
 - [What Is The Lab?](#)
 - [References for Further Reading](#)

Chapter 1: Overview

MIMIC Virtual Lab is simulation software that creates a user-friendly virtual lab environment for training purposes. It is part of the MIMIC suite of network simulation tools:

The [MIMIC SNMP Agent Simulator](#) lets you simulate up to 50,000 SNMP-manageable devices on one Intel-based PC or Sun Sparc. Your network management application can send SNMP (v1, v2, v2c, v3) requests to the simulated agent, which can return SNMP responses or traps. Any SNMP-based device is supported. You can run a variety of device configurations and customizable them at runtime. Because MIMIC responds to SNMP queries on any of its configured IP addresses, it looks to the application as though it were communicating to actual devices.

The [MIMIC Cable Modem Simulator](#) extends the MIMIC SNMP Agent Simulator with the protocols necessary for simulating cable modems from an Operations Support System (OSS) perspective. The additional protocols are DHCP, TFTP, TOD and MGCP.

The [MIMIC IOS Simulator](#) adds the capability to respond to Cisco IOS commands over Telnet. It gives Network Engineers an ability to practice for certifications instead of just reading from the instructions.

There are 2 types of MIMIC Virtual Labs:

- "Networked" labs, which allow remote access to the lab, either via telnet or SNMP. One such lab is [MIMIC Virtual Lab CCNA Plus](#).
- "Lite" labs, which do not allow remote access to the lab. The only way to access the devices in the lab is through the Device->Console or Device->Telnet menu items in the Virtual Lab user interface. One such lab is [MIMIC Virtual Lab CCNA](#).

Since this documentation covers both product lines, it may talk about remote functionality that does not apply to the "Lite" product you have installed.

4. Chapter 2: Using MIMIC Virtual Lab

Chapter Contents

- Starting the lab
- Using the GUI
- Shortcuts
- Accessing a device
- Running an exercise

Starting the lab

To start the MIMIC Virtual Lab on Microsoft Windows, invoke MIMIC Virtual Lab in the MIMIC Program Group (on Windows) or from the command line, which brings up the main front panel.

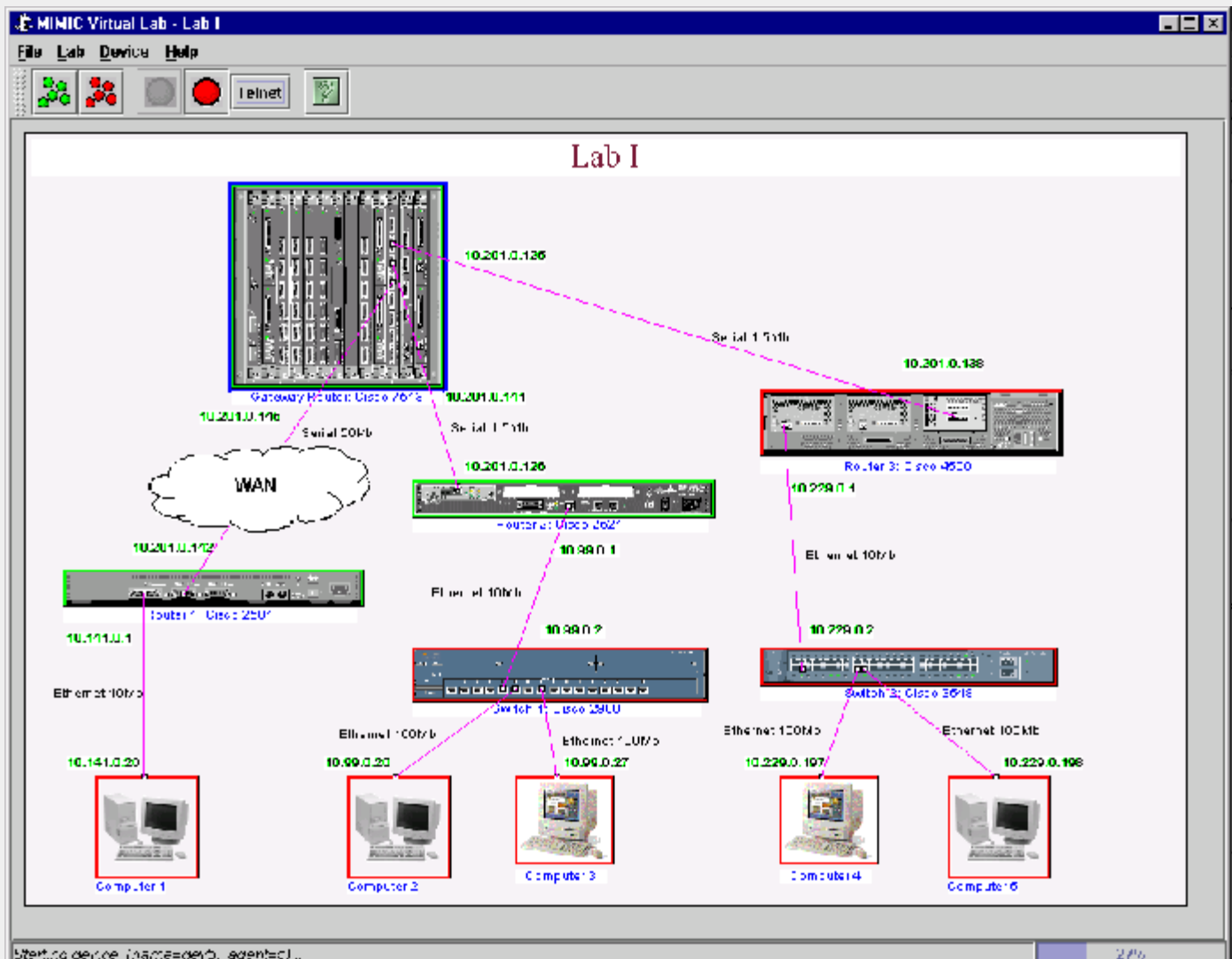


Figure - MIMIC Virtual Lab Front Panel

When you start the lab, the background MIMIC Simulator daemon will be invoked automatically if it is not already running. Each device agent in the lab is shown with an icon in the front panel, and color coded with red when it is stopped, and green when it is running. Initially the lab will be stopped, which is the same as if the real devices were not powered up. You need to start the lab with **Lab->Start** to access the devices.

Using the GUI

Although the various labs may have slight look-and-feel differences, the MIMIC Virtual Lab GUI always contains the following components (from the top):

- the title bar;
- the menu bar;
- the speed bar;
- the main canvas; and
- the status bar;

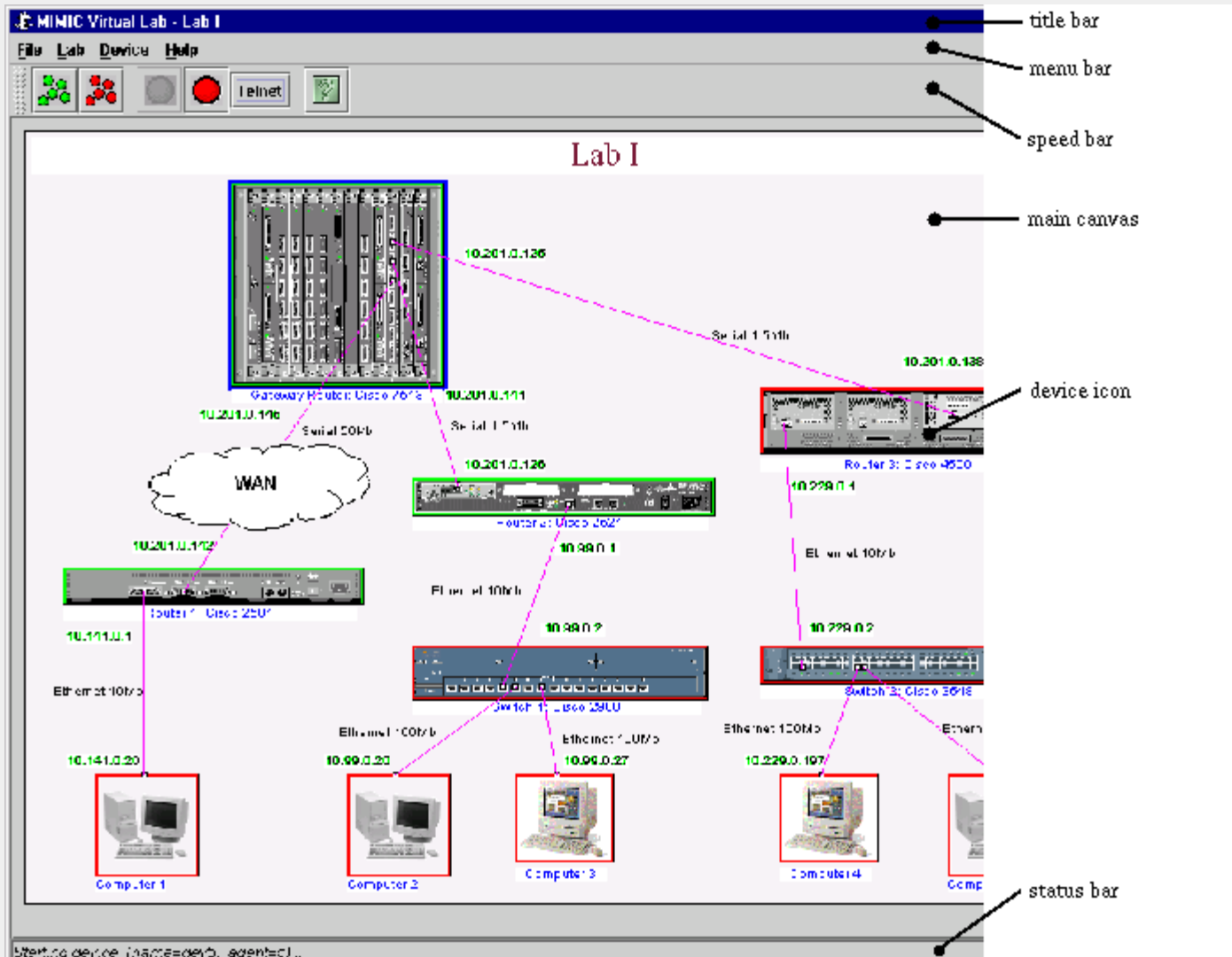


Figure - MIMIC Virtual Lab components

In general, if you want to control a device, you select its device icons in the main canvas and perform actions with the Device menu items or speed bar buttons. The sections below introduce some of the tasks you can accomplish.

Shortcuts

Besides the ALT+letter keyboard shortcuts for menu entries, Virtual Lab also accepts the Tab key as a shortcut to the most common actions, which are shown in the speedbar below the top menu bar.



In addition, you can right-click on a device icon to select the device, and pop up a copy of the Device menu. In this tutorial, we will continue to use the menu entries for clarity. We suggest you use them until you get familiar, then start using the shortcuts.

Accessing a device

Once a device is started, you can access it just like a real device, for example with a telnet client through [Device->Console...](#), [Device->Telnet...](#) or any SNMP application (if you are running a "networked" lab). You can log into the devices with username lab and password lab123. The [Device->Info...](#) command will give you more information about the device, including IOS login, other passwords and SNMP community strings.

The list of supported commands is in [Appendix A](#).

You can look at the device MIB with [Device->MIB](#).

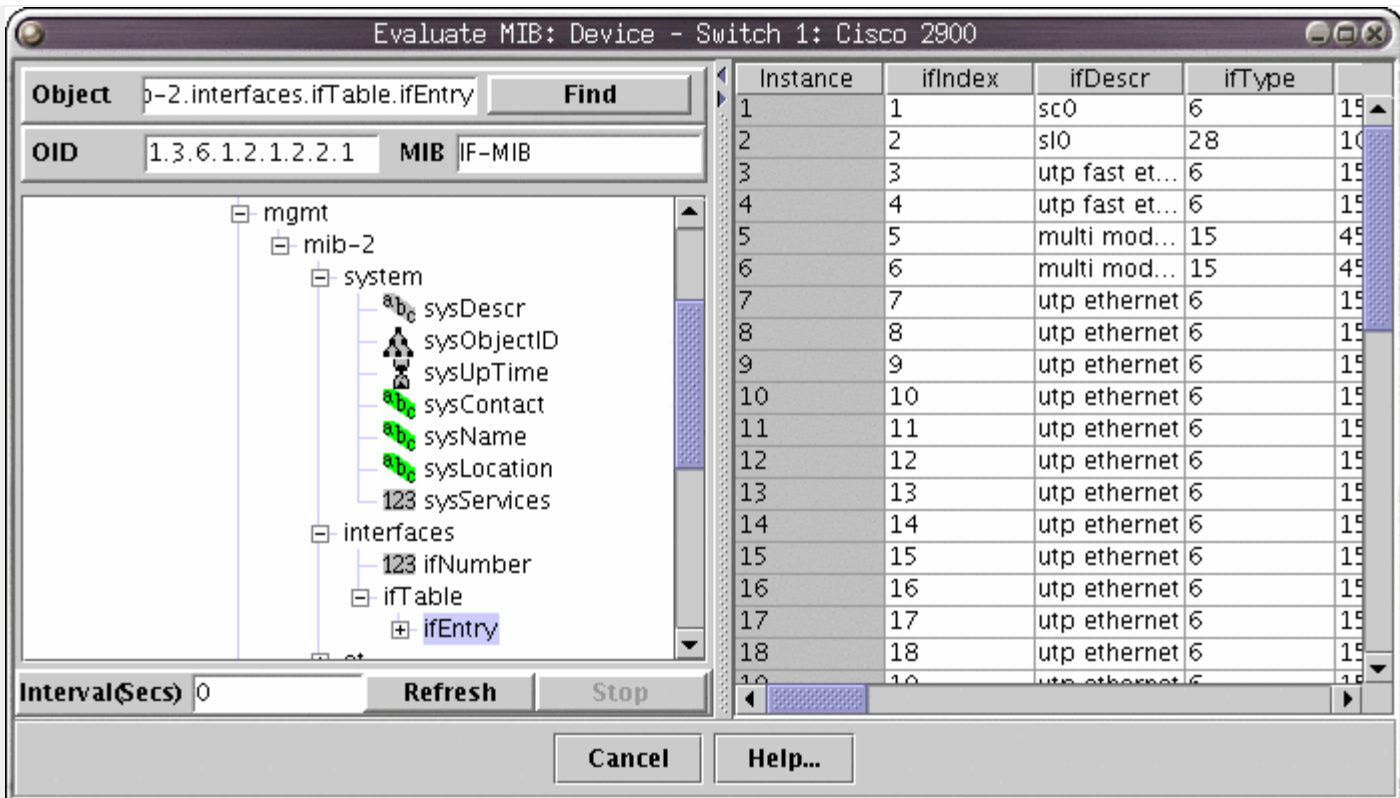


Figure - Device MIB Dialog

Running an exercise

As such, the lab is fairly static. You can now run exercises which change the lab in desired ways. Use [Lab -> Exercise](#) to invoke the Lab Exercise dialog.

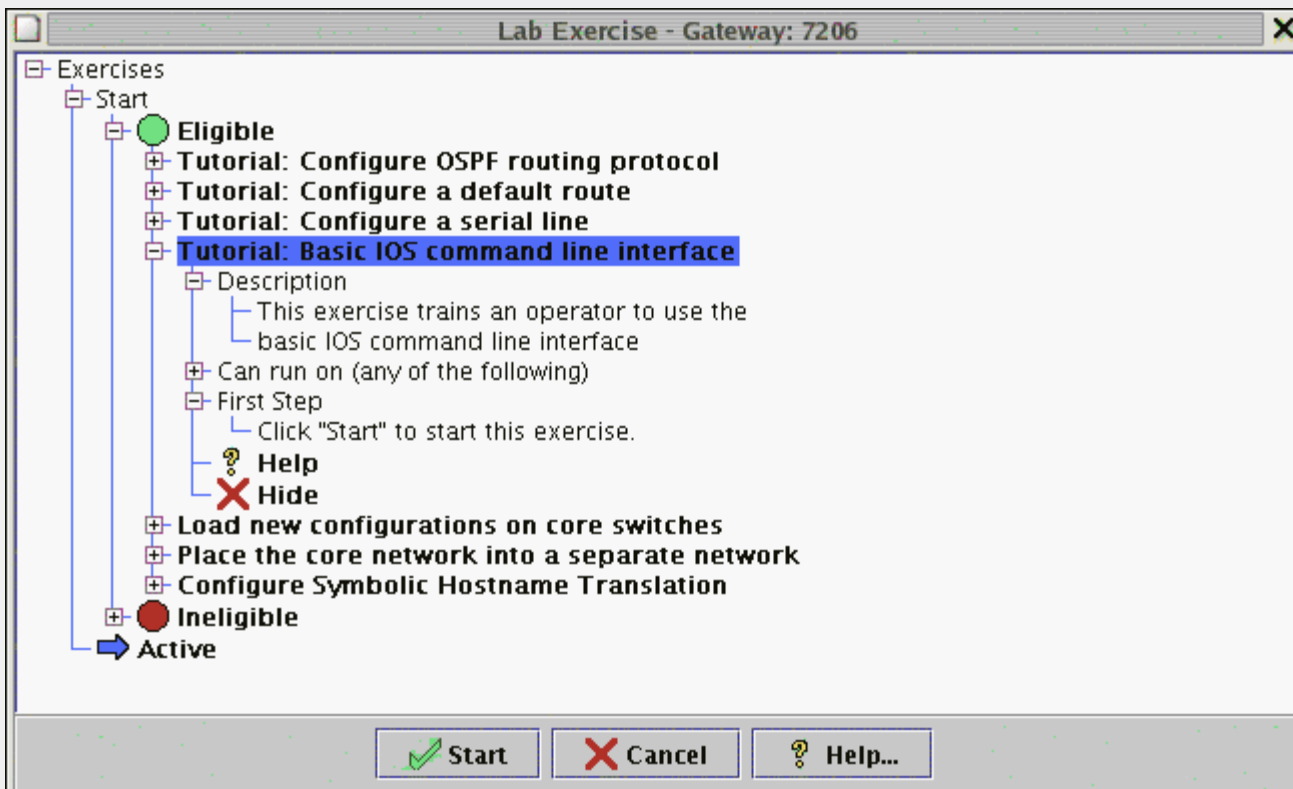


Figure - Lab Exercise Dialog

To run an exercise, select if from the Eligible list. If an exercise is listed under Ineligible you need to select the correct device to run it on. You can expand each exercise node with the + icon to reveal more informational nodes. Once you have selected an eligible exercise, click Apply to start it. Once activated, the exercise will appear under the Active node, and cycle through each step, prompting you to click Apply to continue at your own pace.

We highly recommend to run at least the Tutorial: Basic IOS command line interface exercise once. It will give you brief introduction to how things work in

the MIMIC Virtual Lab.

Chapter 3: Troubleshooting

Chapter Contents

- [Online Help](#)
- [Known Problems](#)
- [Inspect the Log](#)
- [Common Errors](#)
- [Common Questions](#)
- [Crashes](#)

This chapter lists the recommended troubleshooting procedures for quickest resolution of your problem.

Online Help

All MIMIC Virtual Lab dialogs have a context-sensitive online help section, which you can invoke with the Help button. The complete online documentation is accessible with `Help->Contents`.

Known Problems

Each of the supported platforms has known problems. Check there first to see if yours is one of them:

- [Windows](#)

Inspect the Log

MIMIC logs all abnormal events in a log viewable with the [Lab ->Troubleshoot](#) menu item. In case anything goes wrong, inspect it first.

Common Errors

Common errors in the log are detailed in [Appendix C - Common Error Messages](#). Consult this section for details on your particular error.

Common Questions

Common questions and their answers are detailed in [Appendix D - Frequently Asked Questions](#).

Crashes

MIMIC, as any other complex software, occasionally terminates abnormally (crashes). In order to help us diagnose and fix the problem, we will request you to provide some additional information about the problem such as

- how did the crash occur?
- what simulation was running?
- how long had MIMIC been running?
- can you reproduce the crash?

In addition, we will request you to enable dumping of process memory on the crash. Details for Windows are in the [Windows Installation](#) sections.

Chapter 4: Background

Chapter Contents

- [Important Concepts](#)
 - [What Is a Device Instance?](#)
 - [What Is a Simulation?](#)
 - [What Is an Agent Instance?](#)
 - [What Is The Lab?](#)
 - [References for Further Reading](#)

Important Concepts

What Is a Device Instance?

In MIMIC terms, a device is a real-world entity on a network managed primarily via the Simple Network Management Protocol (SNMP) or telnet-based command-line interfaces such as Cisco IOS. The command-line interface is accessible with any telnet client. To be manageable via SNMP, the device exports a Management Information Base (MIB) with embedded software called an SNMP agent. The MIB is usually composed of a collection of standard and enterprise-specific MIB fragments, for example, MIB-2, IF-MIB, and SNMP-REPEATER-MIB, which we just call MIBs. Each MIB is defined in a syntax called "Structure of Management Information" (SMI).

An SNMP-capable network management application interacts with one or more SNMP agents by manipulating MIB objects.

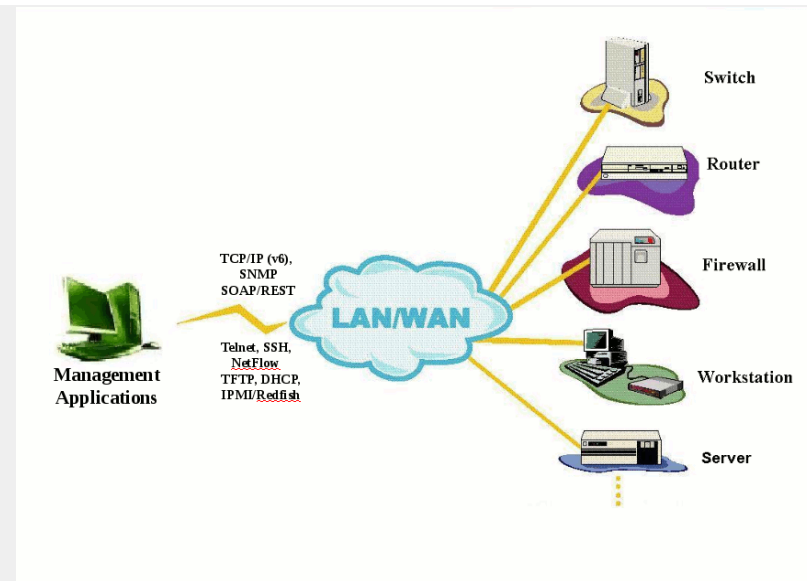


Figure - Network Management Topology

You use MIMIC to simulate one or more instances of a device from the network management perspective, ie. you simulate the SNMP agent or telnet server. There are many different classes of devices, from data communications equipment to end systems, from tele-communications equipment to databases.

What Is a Simulation?

A protocol simulation is the act of allowing protocol interaction with standard applications just as with a real-world device, but without the actual physical device. For SNMP that means exporting MIB object instances and values, generating TRAPS. For command-line interfaces that means exporting a command set such as Cisco IOS via telnet. The network management applications interact with the simulations within MIMIC just as it would with real-world devices.

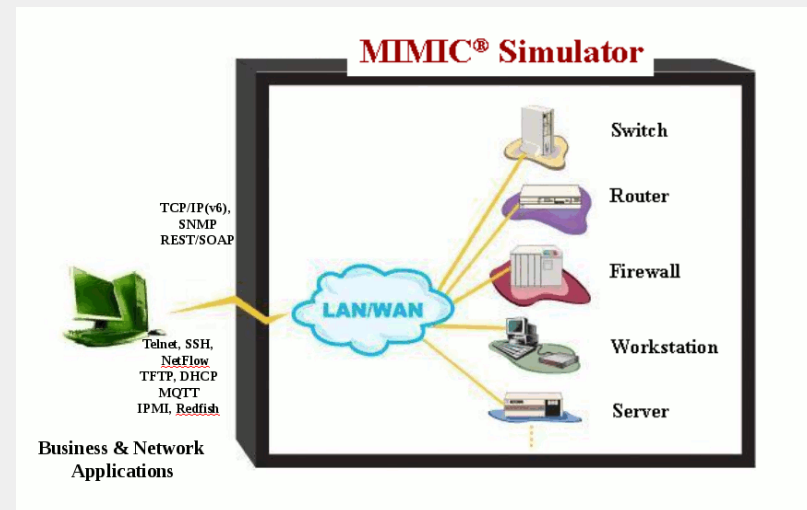


Figure - Simulations with MIMIC

What Is an Agent Instance?

An agent instance is a simulation of a device instance within MIMIC. There can be more than one agent instance of the same device, such as 2 routers or computers of a particular type. The main thing to realize is that each agent instance is independent of the others.

What Is The Lab?

The lab is just a collection of these agent instances in a realistic scenario. This implies that:

- the agents will have been running for a while
- the devices will be pre-configured with interesting data
- there may be hidden devices that interact with the shown devices

References for Further Reading

For more information on Network Management and SNMP, we recommend these books:

- Marshall Rose, *The Simple Book: An Introduction to Networking Management*, Prentice Hall, 1994
- David T. Perkins and Evan McGinnis, *Understanding SNMP MIBs*, Prentice Hall, 1996
- David T. Perkins, *RMON: Remote Monitoring of SNMP-Managed LANs*, Prentice Hall, 1999
- William Stallings, *SNMP, SNMPv2, and RMON : Practical Network Management*, Addison-Wesley, 1996
- William Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Addison-Wesley, 1999.

[<< Previous Section](#) [Next Section >>](#)



[<< Previous Section](#) [Next Section >>](#)

User Guide

Table of Contents

- [Overview](#)
- [User Reference](#)
 - [Startup](#)
 - [File Menu](#)
 - [Lab Menu](#)
 - [Device Menu](#)
 - [Connection Menu](#)
 - [Help Menu](#)
 - [Speed Bar](#)

1. Overview

MIMIC Virtual Lab is an easy-to-use, data-driven interface to MIMIC Simulator. It presents a simulated lab environment consisting of multiple simulated devices, which you can manage just like real devices.

Where other components of the MIMIC suite emphasize power and flexibility, MIMIC Virtual Lab strives to hide MIMIC from the user. This interface is thus ideal for users who need a transparent, ready-made virtual lab. The interface allows to start/stop the lab, or individual devices, and a handful of other useful functions to manipulate the lab.

2. User Reference

Startup

Invoke vlab from a shell command prompt with `vlab*` or in Windows from the MIMIC program group in the taskbar, or by double-clicking on the `vlab*.bat` icon in Windows Explorer.

(*) the vlab will usually have a number, eg. `vlab1`.

The lab configuration file is read and the main panel displays the loaded topology.

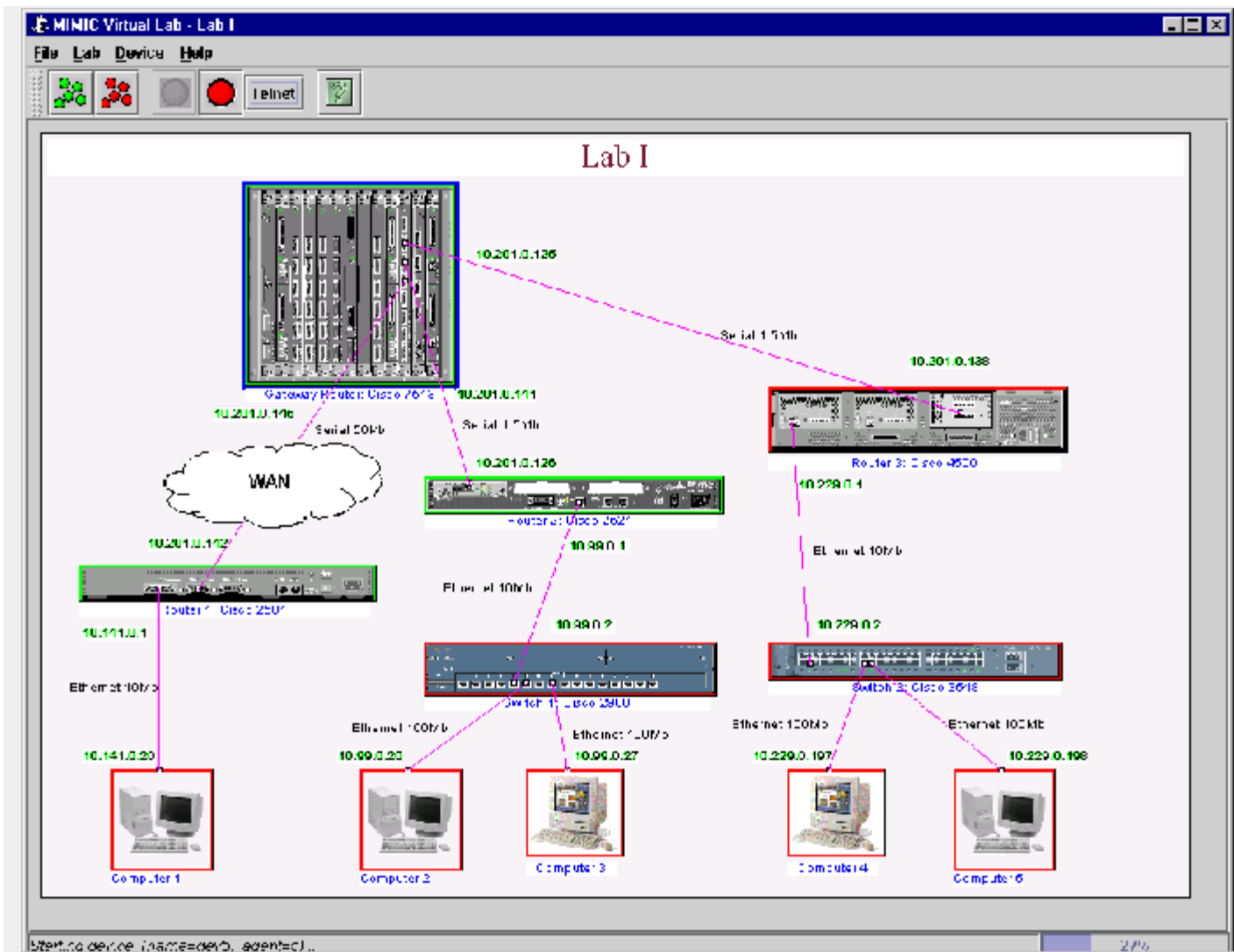
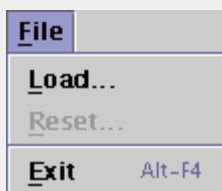


Figure 1 - MIMIC Virtual Lab Front Panel

File Menu

The File menu lets you load or save lab configurations and exit from the program.



File->Load...

Load a different lab configuration. This is only enabled for products that allow to load different lab configurations (such as Virtual Lab Enterprise).

File->Reset...

Reset any changes and cause a restart of all the devices, so that you can go back to the installed defaults. Notice that this does NOT cause a clearing of the device configuration, but the simulation data reverts to the state it was when first installed.

File->Exit

Exit the lab. You will be prompted to save any changed state for the next restart of the lab.

Lab Menu

The Lab menu lets you manipulate the loaded lab.

Lab	Device	Conne
Refresh		F5
Start		
Stop		
Exercise...		
Info...		
Troubleshoot...		
Addresses...		

Lab->Refresh

Sometimes you need to refresh the panel.

Lab->Start

This menu item is equivalent to turning power on. Each device is powered up and ready to access. Notice that each device will take on an initial state, which may be different from just powering it on. For example:


- the device may have been running for a while;
- the device may have loaded a running configuration with the `copy IOS` command;
- the device state may have been changed if you saved it from a previous run.





Lab->Stop

This menu item is equivalent to turning power off, but may do more. Each device is powered down so it is no longer accessible. If its state has changed, you will have the option of saving this state so that on the next start you can continue as if it had never been stopped. Notice:





- on the next start, devices do not reconfigure with saved device configuration files, such as IOS startup-config files.
- to reload a device with a saved configuration file, you need to use the device's reloading procedure, such as running IOS `reload` command.


Lab->Exercise...




This dialog lets you run exercises in the virtual lab. An exercise consists of a number of steps which accomplish desired dynamic changes to the lab. The dialog shows an Exercises tree, with 2 top-level nodes `start` and  **Active**. In general, if a node has an icon in front of it or is in bold text, you can find out more about it by placing your cursor on it. A hint popup gives you more information.

Selecting one of the  **Eligible** exercises under `start` followed by a click of the  `start` button will activate that exercise for the selected device. Clicking the  `Continue` button a halted exercise under  **Active** will continue to the next step in that exercise.

You can check details for an exercise by opening its node (click the + icon in front of it). The `Description` node describes the exercise in general terms, the `First Step`, `Next Step`, `Status` and `Completed` items describe the state of the exercise.

You can double-click on the  **Help** node in an exercise to display its online documentation. When you finish with an exercise (you cannot click  `Continue` anymore), you have to double-click on  **Cleanup** to dismiss the exercise. You can double-click on the  **Abort** node to prematurely end an exercise.

Double-clicking on  **Diagnostic Log** is only necessary in case of trouble-shooting, and displays a [Log window](#).

Once an eligible exercise is no longer interesting, you can hide it by double-clicking the  **Hide** node. If hidden, it will continue to appear under the  **Ineligible** node, where you can restore it to be eligible with the  **Unhide** node. Exercises can also be ineligible for other reasons. To determine why an exercise is ineligible, open its **Reason** node. One reason could be that you need to select a device to run the exercise on.

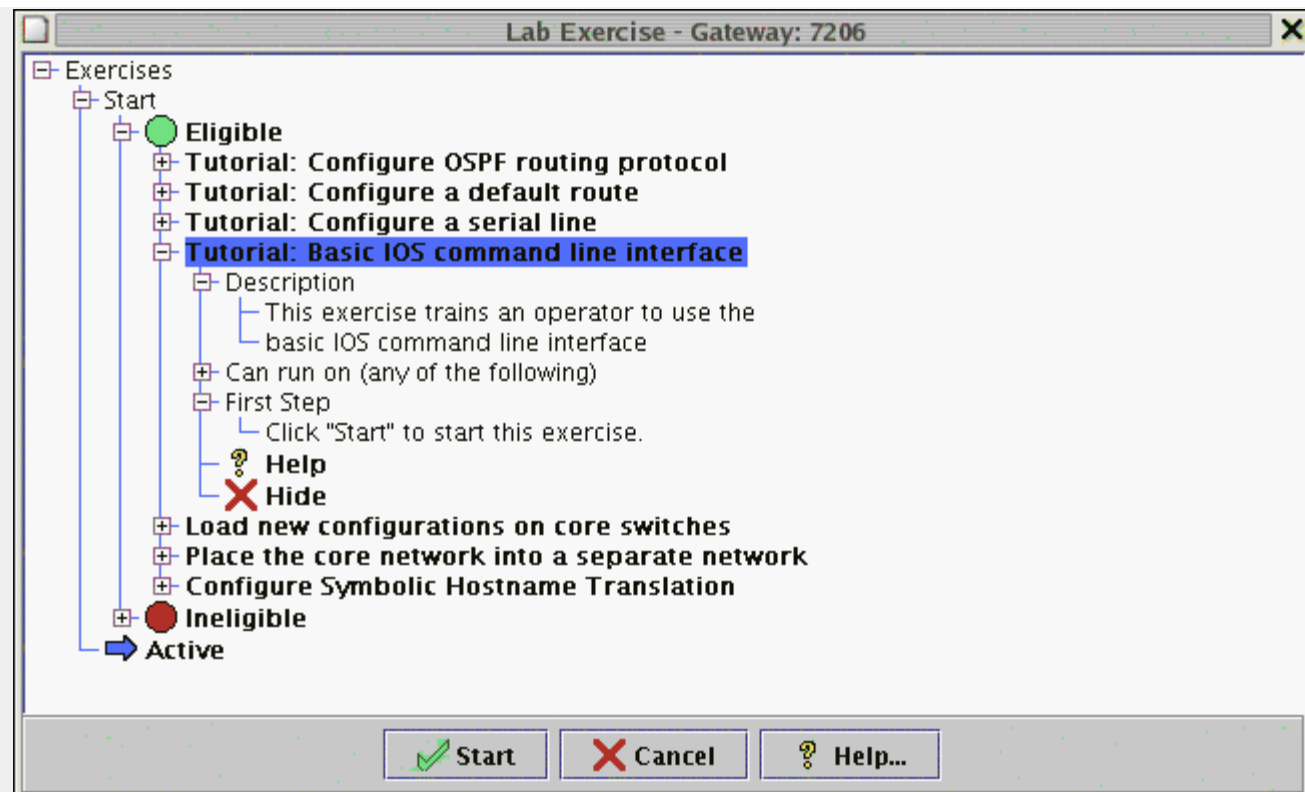


Figure 2 - Lab Exercise Dialog

Lab->Info...

The `General` tab displays detailed information about all devices. The `Traps` tab allows to enter trap destinations. The `Statistics` tab displays detailed statistics for the devices present in the lab.

Lab->Troubleshoot...

To aid in troubleshooting, the log window captures the diagnostics output for MIMIC Virtual Lab. All informational and [error messages](#) are displayed and output to a file given in the title of the window. This log window or output file should be referred to whenever you have problems with MIMIC.

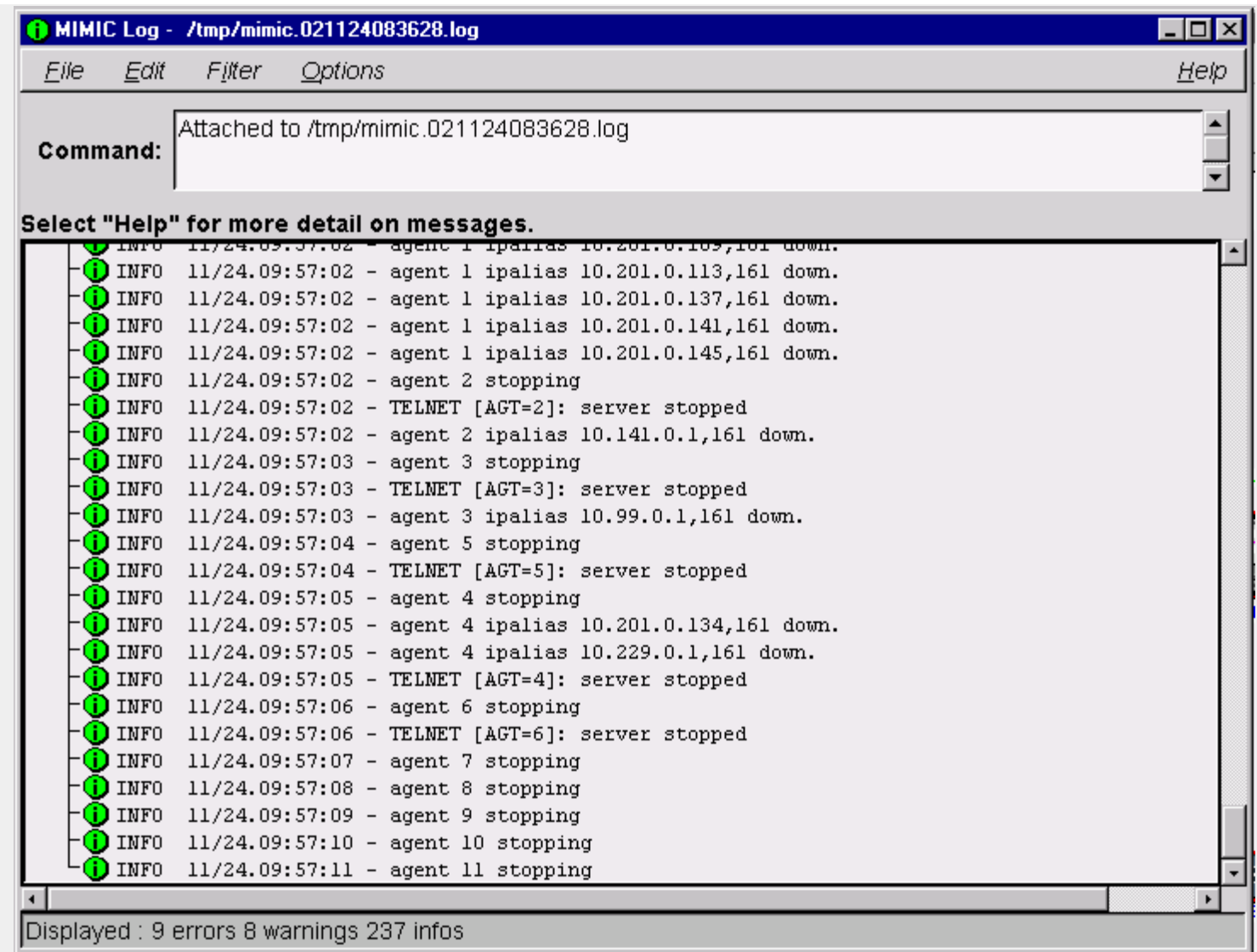





Figure 3 - MIMIC Log Window

All log files are placed in /tmp/ and are ordered by date of invocation. The log file contains additional message details, which are filtered out in the log window for legibility.

In the log window, each diagnostic message is listed with the following severities:

1.  **INFO** - informational message (shows progress, no impact on correctness of results)
2.  **WARN** - warning message (some impact on correctness of results)
3.  **ERROR** - error message (more impact on correctness of results; possibly fatal)

NOTE: Severities are subjective under most circumstances. In general, you can ignore warnings, unless your management application relies on correctness of the impacted part of the simulation. Error messages merit closer inspection.

Certain messages are multi-line because the underlying cause of the event is also logged. This leads to more accurate diagnosis of the problem. Multi-line messages are shown with a leading + sign. For legibility the log window shows only the first line of the diagnostic message. The causes of any error message can be expanded by clicking on the + or double-clicking on the message itself.

The log window will be displayed until you select **File->Close**.

A single selected message or a set of messages can be copied using **Edit->Copy** and then pasted into other applications. This makes it easy to e-mail related error message information to help Gambit support personnel diagnose your problem.

You can filter out existing messages from being displayed:

- **Filter->Message** filters out the individual selected message(s).
- **Filter->Type** filters out all messages of the same type as the selected message(s).
- **Filter->Severity** filters out all messages of the same severity as the selected message(s).
- **Filter->Off** turns off filtering for selected messages and (re)displays them.

The `Options` menu lets you change the way the log window behaves. You can turn off the tracking of the end of the log (the "tail") by deselecting the `Tail` item, and you can turn on the expansion of nested messages with the `Nested` option.

The most common error messages are listed in [Appendix C](#).

Lab->Addresses...

This menu item will invoke the dialogs to change IP addresses in your lab. This procedure is invoked the first time you run your lab, and should only be performed after that if the selected addresses clash with some part of your network.

Device Menu

The Device menu lets you manipulate the selected device.



Device->Start

This menu item starts the selected device simulation. One would think this is equivalent to powering it up, but it can vary from that:

- the device may have been running for a while;
- the device may have loaded a running configuration with the `copy IOS` command;
- the device state may have been changed if you saved it from a previous run.

First the device icon border will turn cyan, indicating that the device simulation is starting. When the device icon shows green, the device is fully started and ready to be accessed.

Device->Stop

This menu item stops the selected device, which is equivalent to powering it down, but may do more. When the device icon shows red, the device is fully stopped and cannot be accessed. If its state has changed, you will have the option of saving this state so that on the next start you can continue as if it had never been stopped.

Device->Console...

You can use this menu item to conveniently log into the device. The console client window pops up and you can login to the device. The console client attempts to simulate as closely as possible the real-world behavior of the console port.

Device->Telnet...

You can use this menu item to conveniently telnet into the device. The telnet client window pops up and you can login to the device. Of course you can use any other telnet client to access the device.

Device->Info...

This menu item displays more information for the selected device, such as its interfaces in the `General` tab, the SNMP attributes (port and community strings) in the `SNMP` tab, and telnet attributes (user and password for the different IOS modes) in the `Telnet` tab.

Device->MIB...

This dialog visualizes the MIB of the selected device. It shows a MIB Browser on the left, which displays a tree diagram of the MIB object hierarchy. Each node in the tree is either a subtree or a leaf MIB object.

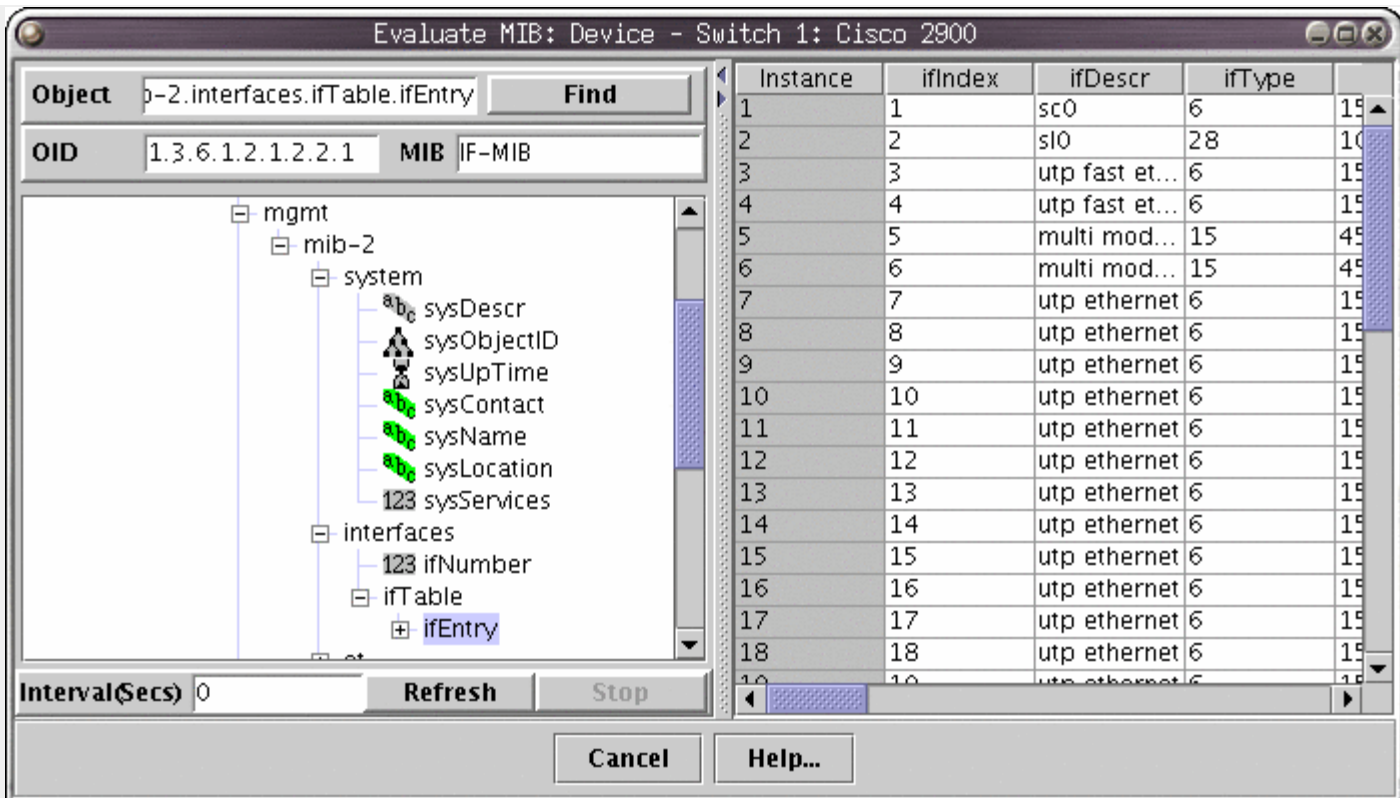


Figure 4 - Device MIB Dialog

You can open subtrees in the hierarchy by double-clicking on branch nodes that are preceded by a plus (+) box, or by single-clicking on the plus box itself. You can close subtrees by single-clicking on the minus (-) box before a branch node.

MIB object leaf nodes contain information for the object. A symbol denotes the type of the MIB object. These are the currently displayed object types:

- 123 - Integer
- abc - OctetString
- [Counter Icon] - Counter, Counter64
- [Gauge Icon] - Gauge
- [IP Icon] - IpAddress
- [Tree Icon] - OBJECT IDENTIFIER
- [Clock Icon] - TimeTicks
- [SNMPv1 Icon] - SNMPv1 Trap
- [SNMPv2 Icon] - SNMPv2 Trap
- [D Icon] - Address
- [0110 Icon] - BITSTRING
- [NetAddress Icon] - NetAddress
- [H Icon] - Opaque

In addition, the color indicates the access to the object:

- gray - read-only, not-accessible, accessible-for-notify
- green - read-write, write-only
- yellow - read-create

You select a MIB object by clicking on the leaf node in the tree.

You can type a object name in the Object field, and click the Find button to directly select it.

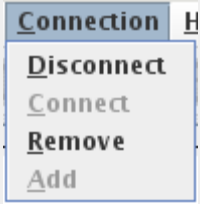
For example, for the outgoing octets counter of a network interface you would use ifOutOctets.

The right side contains a matrix, which displays all columns in a MIB table, or scalar objects underneath a branch. Clicking (Shift-left mouse button) on the top row expands or contracts the width of a row between 3 possible states: wide enough to display all values in the column, 10 characters wide, contracted. Clicking on a value has no effect - the edit mode is allowed to inspect the entire value.

To redisplay the value (eg. to monitor increasing Counter objects), just click on the object and the values will be updated.

Connection Menu

The Connection menu lets you manipulate the selected connection.



Connection->Disconnect

This menu item disconnects the link. This is analogous to unplugging a cable at either end, or physically cut it. The MIMIC Virtual Lab will attempt to simulate this condition just like in the real world. The link will be shown with a dashed line.

Connection->Connect

This menu item reconnects the disconnected link. This is analogous to plugging the cable in at both ends.

Connection->Remove

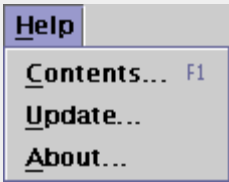
This menu item removes the selected link. This is analogous to removing the cable entirely. The difference between disconnecting and removing the cable is that the latter allows connecting a different cable to a different port to either end.

Connection->Add

Selecting an unused port (ie. no cable connected to it) and clicking this menu item highlights all the ports that this port can be connected to. Once you select the second end-point, a cable will be drawn to connect them. Initially the cable will be unplugged, but you can [reconnect](#) it to gain connectivity between the ports.

Help Menu

The Help menu displays online documentation and revision information.



Help->Contents...

The HTML documentation is displayed in your HTML browser of choice.

Help->Update...

This menu item invokes the Update Wizard.

• Introduction

The Update Wizard makes it easier to update to newer versions of MIMIC over the Internet, or to install optional software from CD-ROM. It automatically notifies you of new software updates, and presents a friendly front-end to install them.

MIMIC Virtual Lab will notify you of the availability of new updates of MIMIC. This occurs whenever you start MIMIC Virtual Lab, but at most once a day. MIMIC Virtual Lab will download the MIMIC update database, and check for applicable updates to your software.

If there are new updates, you will be notified with a dialog. This lets you either:

- update immediately, by invoking the Update Wizard.
- update later. You have to manually invoke the Update Wizard from the `Help->Update...` menu.
- don't notify. This turns off the automatic update notification feature.

First you need to select the source of your update, either from the Gambit Web site, or from CD-ROM. Since the CD-ROM can be at a different directory path on each system, you will need to supply it's location in the `path` field.

1. Select Update

A log window similar to the one displayed by [Lab->Troubleshoot](#) is displayed which logs all steps taken in the update.

In this dialog, select the update you want to install for your version of MIMIC. If there are no applicable updates, you are done.

Pressing `Next` downloads the update.

i. Download Images

The update is downloaded from the Gambit Communications Web site. The download progress is displayed on the Status and Progress bars at the bottom.

If the update has already been downloaded, you are asked whether to download again.

If the extraction tools (gzip, tar) are not executable, they are also downloaded.

j. Prepare For Installation

Certain updates need extra preparation to complete the update. This is done here. You will be prompted for any extra steps.

k. Extract Files

Files are extracted from the update. The list of files is displayed for your information.

Pressing **Next** extracts the contents of the update, and moves files to the shared area. Any existing files are backed up.

l. Move Files

First, the contents of the update are extracted into a temporary directory. Then files are moved from the temporary directory to the shared area.

While files are being moved, the progress is displayed on the Status and Progress bars.








Help->About...

This displays the revision and contact information.

Speed Bar



The Speed Bar provides quick access to the most common commands. The action performed by each button will be displayed in a pop-up window under the Speed Bar when the cursor is positioned over the button.

-  Start the entire lab. This menu item is equivalent to turning on the power for all the devices in the lab.
-  Stop the entire lab. This menu item is equivalent to turning off the power for all the devices in the lab.
-  Start the selected device. This menu item is equivalent to turning on the power for the selected device.
-  Stop the selected device. This menu item is equivalent to turning off the power for the selected device.
-  Telnet to the selected device. This menu item will open a telnet client window in order to login to the device.
-  Display values in MIB. This menu item will display a window with a MIB Browser on the left and a matrix that displays all columns in the MIB table on the right.
-  Display help contents. This menu item displays the online help and revision information.

Java Configuration

MIMIC needs to know about the Java Runtime Environment (JRE) installed on your machine. It attempts to detect it in the default paths, but you may have to point it to the correct location (bin/ subdirectory of the JRE install area).



[<< Previous Section](#) [Next Section >>](#)

MIMIC Virtual Lab and JRE

- Requirements

MIMIC Virtual Lab requires the use of the [Java Platform, Standard Edition \(Java SE\)](#) 6 or later. If you don't have it, download it from [Sun Microsystems](#).

[<< Previous Section](#) [Next Section >>](#)



Appendix A: CLI Commands

IOS Commands

This is a partial list of supported commands:

Router IOS Commands	Mode
?	um, en, ct, ci, cm, pm
< command > ?	um, en, ct, ci, cm, pm
access-list	ct, ci, cl
auto-summary	ro
bandwidth	ci
banner	ct
banner login #	ct
begin	um, en
cdp [enable holdtime run timer]	ct
class	pm
class-map	ct
clear arp-cache	en
clock rate	ci
configure terminal	en
connect	um,en
copy	en
copy running-config startup-config	en
copy startup-config running-config	en
copy tftp flash	en
copy tftp running-config	en
copy tftp startup-config	en
debug all	en
debug dialer [events packets]	en
debug eigrp [neighbors packets]	en
debug frame-relay lmi	en
debug ip igrp transactions	en
debug ip ospf [database events neighbor]	en
debug ip rip	en
debug isdn q931	en
deny	ipa
description	ci
dialer [fast-idle idle-timeout]	ci

dialer map	ci
dialer pool-member	ci
dialer remote-name	ci
dialer string	ci
dialer-group	ci
dialer-list	ct
dir	en
disable	en
disconnect	um,en
duplex	ci
enable	um,ct
enable [password secret]	en,ct
encapsulation dot1q	ci
encapsulation frame-relay [cisco ietf]	ci
encapsulation [isl ppp]	ci
end	ct, ci, cl, cm, ipa, ro
exclude	um, en
exec-timeout	cl
exit	en,ci, ct, ipa
exponential-weighting-constant	ct
fair-queue	ct
frame-relay	ct
frame-relay adaptive-shaping {becn foresight}	mc
frame-relay bc [in out] bits	mc
frame-relay be [in out] bits	mc
frame-relay cir [in out] bps	mc
frame-relay idle-timer [in out] seconds	mc
frame-relay intf-type	ci
frame-relay lmi-n392dce threshold	ci
frame-relay lmi-n393dce events	ci
frame-relay lmi-t393dce seconds	ci
frame-relay lmi-type	ci
frame-relay route	ci
frame-relay switching	ct
help	um, en, ct, ci, cm, pm
hostname	ct, ci, cl, cm, ipa, ro
interface	ct, ci
ip access-group	ci
ip access-list	ct
ip address	ci

ip classless	ct
ip default-gateway	ct,cm
ip host	ct
ip mtu	ci
ip nat inside source	ci
ip nat outside source	ci
ip nat pool	ct
ip nat translation	ct
ip ospf cost	ci
ip route	ct
ip router isis	ci
ip routing	ct
ip summary-address eigrp	ci
ipv6	ct, ci
ipv6 address	ci
ipv6 rip enable	ci
ipv6 router	ct
ipv6 unicast-routing	ct
isdn switch-type	ct, ci
keepalive (f/r lmi command)	ci
line	ct
line aux 0	ct
line console 0	ct
line vty 0 4	ct
logging	ct
login	cl
logout	um,en
mac-address	ci
map-class	ct
match	cm
neighbor remote-as	ro
net (is-is)	ro
network (bgp)	ro
network (eigrp)	ro
network (rip)	ro
no access-list	ct
no banner	ct
no banner login	ct
no cdp enable	ci
no cdp holdtime	ct
no cdp run	ct

no cdp timer	ct
no clock rate	ci
no debug eigrp packets	en
no description	ci
no dialer-list	ct
no duplex	ci
no enable password	ct
no encapsulation	ci
no frame-relay switching	ct
no ip access-group	ci
no ip address	ci
no ip classless	ct
no ip default-gateway	cm
no ip host	ct
no ip nat inside source	ci
no ip nat outside source	ci
no ip nat pool	ct
no ip nat translation	ct
no ip ospf cost	ci
no ip route	ct
no ip router isis	ci
no ip summary-address eigrp	ci
no ipv6	ct, ci
no ipv6 address	ci
no ipv6 rip enable	ci
no ipv6 router	ct
no ipv6 unicast-routing	ct
no isdn switch-type	ct, ci
no keepalive (f/r lmi command)	ci
no logging	ct
no login	cl
no mac-address	ci
no neighbor remote-as	ro
no net (is-is)	ro
no network (bgp)	ro
no rate-limit	ci
no router bgp	ct
no router eigrp	ct
no router igrp	ct
no router isis	ct

no router ospf	ct
no router rip	ct
no shutdown	ci
no snmp-server	ct
no snmp trap	ci
no snmp trap link-status	ci
no trunk group	ci
permit	ipa
ping	um,en
police	pm
policy-map	ct
ppp authentication	ci
priority	cl
queue-limit	ct
random-detect	ci
rate-limit	ci
reload	en
remark	ipa
resume	um,en
router bgp	ct
router eigrp	ct
router igrp	ct
router ospf	ct
router rip	ct
service-policy	ci, cl, cm, mc, pm
show access-lists	um,en
show adjacency	um,en
show arp	um,en
show backup	um,en
show buffers	um,en
show cdp	um,en
show cdp entry *	um,en
show cdp entry name	um,en
show cdp interface	um,en
show cdp neighbors	um,en
show clock	um,en
show compress	um,en
show configuration	en
show controllers	um,en
show diag	um,en
show dialer	um,en

show flash all	um,en
show flash chips	um,en
show flash detailed	um,en
show flash err	um,en
show flash summary	um,en
show frame-relay lmi	um,en
show frame-relay map	um,en
show frame-relay pvc	um,en
show frame-relay route	um,en
show history	um,en
show hosts	um,en
show interface [type type n/n]	um,en
show inventory	um,en
show inventory raw	um,en
show ip access-lists	en
show ip aliases	um,en
show ip arp	um,en
show ip bgp	um,en
show ip bgp neighbors	um,en
show ip bgp summary	um,en
show ip eigrp interfaces	um,en
show ip eigrp neighbors	um,en
show ip eigrp topology	um,en
show ip interface	um,en
show ip nat	en
show ip ospf	um,en
show ip ospf database	um,en
show ip ospf neighbor	um,en
show ip ospf interface	um,en
show ip ospf interface brief	um,en
show ip protocols	um,en
show ip rip	um,en
show ip route	um,en
show ip route bgp	um,en
show ip route eigrp	um,en
show ip route ospf	um,en
show ip route rip	um,en
show ipv6	um,en
show ipv6 interface	um,en
show ipv6 rip	um,en

show ipv6 route	um,en
show isis	um,en
show key	en
show key chain	en
show location	um,en
show logging	um,en
show memory	um,en
show modemcap	um,en
show processes	um,en
show processes cpu	um,en
show processes memory	um,en
show protocols	um,en
show running-config	en
show running-config interface	en
show snmp	um,en
show startup-config	um,en
show tcp statistics	um,en
show terminal	um,en
show users	um,en
show users all	um,en
show users wide	um,en
show version	um,en
shutdown	ci
snmp trap	ci
snmp trap link-status	ci
systat	um, en
systat all	um,en
terminal history size	um, en
terminal length	um, en
terminal width	um, en
traceroute	en
trunk group	ci
undebug all	en
where	um
write erase	en
write memory	en
Switch IOS Commands	Mode
access-list	ct
banner	ct
cdp	ct, ci
clear arp-cache	en

clear port-security	en
configure	en
connect	en
copy	en
debug	en
description	ci
dir	en
disable	en
disconnect	en
duplex	ci
enable	en
enable password	ct
enable secret	ct
end	ct, ci, cl, ipa
exit	en, ct, ci, ipa
help	um, en, ct, ci, ipa
hostname	ct, ci, cl, ipa, vl
interface	ct
ip	ct
ip access-list	ct
ip default-gateway	ct
ip host	ct
ipv6	ct, ci
ipv6 address	ci
ipv6 rip enable	ci
ipv6 router	ct
ipv6 unicast-routing	ct
line	ct
logging	ct
logout	um, en
mac-address aging-time	ct
mac-address static	ct
mac-address-table	ct
no	en, ct
ping	um, en
random-detect	ci
reload	en
resume	um,en
show access-lists	en
show adjacency	um,en

show arp	en
show buffers	en
show cdp	en
show clock	en
show configuration	en
show flash:	en
show history	en
show hosts	en
show interfaces	en
show interfaces switchport	en
show interfaces vlan	en
show inventory	um,en
show inventory raw	um,en
show ip	en
show ipv6	en
show location	en
show logging	en
show mac-address-table	en
show memory	um,en
show port-security	en
show processes	um,en
show processes cpu	um,en
show processes memory	um,en
show running-config	en
show snmp	en
show spanning-tree	en
show startup-config	en
show tcp statistics	um,en
show terminal	en
show users	en
show version	en
show vlan	en
show vtp status	en
shutdown	ci
snmp-server	ct
switchport access vlan	ci
systat	en
telnet	um, en
terminal	en
traceroute	en
trunk	ci

vlan	ct
vlan database	en
vtp	vl
where	um, en
write erase	en
write memory	en
Catalyst Switch Commands	Mode
show trunk	en
clear trunk	en
set interface	en
set trunk	en

Modes

um = User EXEC mode
 en = Privileged EXEC mode (enable command)
 ct = Global Configuration mode (configure terminal command)
 ci = config-if mode (interface command)
 cl = config-line mode (line command)
 cm = config-cmap mode (class-map command)
 ipa = ip access-list configuration (named access-lists)
 mc = map-class
 pm = policy-map
 ro = router config
 vl = VLAN
 For details see, eg. [Cisco IOS 12.4 Documentation](#).

JUNOS Commands

This is a partial list of supported commands:

JUNOS Commands	Mode
?	om, cm
< command > ?	om, cm
connect	om
configure	om
ping	om
show arp	om
show cli	om
show interfaces	om
show route	om
show system uptime	om
show version	om
telnet	om
quit	om, cm
set apply-groups	cm
set interfaces	cm
set routing-options static	cm
set system	cm
set system dcmname	cm

set system host-name	cm
set system host-name	cm
top	cm

Modes

om = Operational mode

cm = Configuration mode

For details see, eg. [Junos OS Documentation, Release 12.3](#).

<< Previous Section Next Section >>



Appendix C: Common Error Messages

Since MIMIC is an extremely complex tool, it was designed to contain extensive diagnostic information. All events of interest are logged and displayed in the log window viewable through the [Lab->Troubleshoot](#) menu item. Since we cannot display a detailed explanation with every message at runtime, they are listed here.

Information that can change from message to message is shown in **BOLD**.

To use the list, search it by some unique words in the message (this should stay the same across releases).

If you don't see a message explained here, please cut and paste it into an e-mail message and send it to support@gambitcomm.com. We will include it in the next rev of the documentation.

Agent Simulator

1. ERROR **DATE** - agent **NUMBER** cannot read PDU from **ADDRESS**
DATE - snmp_agent_parse failed
DATE - unknown community: **COMMUNITY**

Cause
An SNMP request with a community string was received which differs from the configured community string for the agent instance.

Action
The SNMP request is ignored. To accept requests at the community, change the agent instance configuration.
2. ERROR **DATE** - agent **NUMBER** cannot read PDU
DATE - snmp_agent_parse failed
DATE - Bad Version: 1

Cause
An unknown SNMP protocol is being used. MIMIC currently only supports SNMPv1. Some management applications (like HP/OpenView) may want to talk a different protocol, such as the newer SNMP Security Protocol, which has not been standardized), and usually first try it, then fall back to SNMPv1.

Action
The SNMP request is ignored.
3. ERROR **DATE** - no receiver at **ADDRESS**

Cause
This message means an SNMP request was received for an address that has no agent instance running. This request could be for the host's real IP address, in which case this message happens occasionally.

Action
The SNMP request is ignored.
4. WARN **DATE** - cannot receive from **ADDRESS**. continuing...
DATE - rcvfrom: Connection refused
WARN **DATE** - cannot receive from **ADDRESS**. continuing...
DATE - rcvfrom: Resource temporarily unavailable

WARN **DATE** - cannot receive from **ADDRESS**. continuing...
DATE - rcv: No error

ERROR **DATE** - agent **NUMBER** send failed from **ADDRESS** to **ADDRESS**
DATE - type=**NUMBER** size=**NUMBER**
DATE - sendto: Connection refused

WARN **DATE** - cannot accept connection from **ADDRESS**. continuing...
DATE - accept: Resource temporarily unavailable

Cause
Any of these messages means a protocol request (SNMP, Telnet, TFTP, etc) or connection was aborted by the management application. This happens occasionally, and could be an indication of a faulty management application.

Action
The request is ignored.
5. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - Please refer to Appendix C for more details ERROR **DATE** - cannot bind receive IP address
DATE - bind: Permission denied

Cause
This message means that you are running without sufficient privileges to bind to the selected SNMP socket.

Action
You need to run the MIMIC daemon mimicd with sufficient privileges.

On Unix, this means running mimid as root, or with setuid-root. The installation by default installs mimid as setuid-root.

On Windows NT, you need to run MIMIC as a user with Administrator privileges.

6. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - Please refer to Appendix C for more details
DATE - cannot bind receive IP address **ADDRESS** port **PORT**
DATE - bind: no error

Cause

This message on Windows NT means there is already another process running that uses the selected SNMP port. This is very likely an already-running instance of an SNMP agent.

On Windows 2000 and XP it could also mean that you are trying to use an IP address that is already assigned to another node running on the network. See [Windows Installation Guide](#) for more details.

Action

Only one process can simultaneously use the selected SNMP port on a host.

To verify if there is such a program, stop MIMIC with File->Terminate and use the netstat utility from the DOS command line prompt, for example:

```
C> netstat -a -n | find "161"
TCP 0.0.0.0:161 0.0.0.0:0 LISTENING
UDP 0.0.0.0:161 *:*
```

If these lines show, then start the Windows task manager and see if there is an SNMP agent process running, e.g. snmp.exe, that you need to kill.

Otherwise, contact Technical Support on how to find any other programs using this port.

The Windows NT SNMP service can only be killed from the **Services** control panel.

7. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address
DATE - cannot add address **ADDRESS**
DATE - Cannot add IPAddress **ADDRESS**
DATE - Failed DhcpNotifyConfigChange.

Cause

You can only run MIMIC from an account with Administrator privileges. Consult the [Windows Installation Instructions](#) section for details.

Action

Login to an account with Administrator privileges and run MIMIC.

8. ERROR **DATE** - AGNT[**x**]: cannot start(2) agent **NUMBER**
DATE - cannot set address
DATE - cannot add address **ADDRESS**
DATE - AddIPAddress failed. Reason=A device attached to the system is not functioning.

```
ERROR DATE - AGNT[x]: cannot start(2) agent NUMBER
DATE - cannot set address
DATE - cannot add address ADDRESS
DATE - Cannot add IPAddress ADDRESS
DATE - DeleteIPAddress failed. Reason=The specified network resource or device is no longer available.
```

Cause

This error means that the network interface configured for the agent is not available, and usually happens on laptop computers, where you can pull out network cards (PCMCIA, docking stations), or on newer versions of Windows, when the network is unplugged. Consult the [Windows Installation Instructions](#) section for details.

Action

Configure a different network interface for the agent.

9. ERROR **DATE** - cannot listen on remote management socket
DATE - listen: No error

Cause

On Windows, this likely means you are running a software firewall which is preventing MIMIC from running properly.

Action

Configure your software firewall to allow MIMIC to access the network. See also the [Windows Installation Instructions](#).

10. ERROR **DATE** - initialization failed
DATE - Cannot read license

```
ERROR DATE - initialization failed
DATE - License expired
```

```
ERROR DATE - initialization failed
DATE - cannot get license
DATE - Invalid license key
```

```
ERROR DATE - initialization failed
DATE - cannot get license
DATE - License corrupt
```

Cause

The licensing information in the license file mimid.lic is incorrect.

Action

MIMIC will not run without correct license keys which you can obtain from support@gambitcomm.com.

You can copy/paste the keys when prompted by the installation program. Or, if you have already installed MIMIC, then edit the config/*.lic files to paste the correct key (also see [FAQ](#)).

11. ERROR: oid.cc:47: assertion failed - constructor failed

Cause

Any assertion failure is fatal, and should be reported to support@gambitcomm.com. Any message with "constructor failed" is likely a lack of virtual memory. You need to increase your swap space as detailed in the OS-specific installation instructions.

Action

Send email to support@gambitcomm.com.

12. ERROR **DATE** - buffer full from ADDRESS to ADDRESS

Cause

The "buffer full" message is displayed when the management application sends too many requests at once. MIMIC cannot service them all, and buffers them (as all real SNMP agents do). The message alerts you when the buffer overflows, and messages are discarded (as all real SNMP agents do, except they do it silently).

Action

This condition can be caused due to 2 reasons:

a) MIMIC is too slow (running on a underpowered machine). If this message occurs occasionally, you can overcome this problem by either putting MIMIC in overdrive by disabling action scripts, and/or increasing the buffer size for each agent.

If you are not using [action scripts](#), you can disable the extra processing on every received request, resulting in significant performance gain. To disable actions, set the MIMIC_DISABLE_ACTION environment variable to any value prior to running MIMIC (if it is running, you must terminate it with File->Terminate from MIMICView).

In the C shell, do:

```
% setenv MIMIC_DISABLE_ACTION 1
```

In the Bourne shell, do:

```
# MIMIC_DISABLE_ACTION=1; export MIMIC_DISABLE_ACTION
```

On Windows NT, use the System Control Panel to set this environment variable.

To increase the buffer size for each agent, edit the config/mimicd.cfg file and add a line

```
agent_qsize = value
```

where "value" is a number larger than 10 (the default). Try 20.

If this problem persists, you may want to run MIMIC on a more powerful machine.

b) the management application has a performance bug, ie. it sends too many requests simultaneously (as we have seen). A real agent will never alert you to this condition, except that performance suffers, since the app will retransmit the discarded requests.

A common bug in management applications is the issuance of too many simultaneous requests in a "burst". Performance bugs are violations of the performance requirements for managing devices. These are subtle bugs, since their only symptoms are degraded performance, which is hard to measure. MIMIC helps you detect these violations.

An example of this condition is "aggressive retransmission policy", which could trigger this effect: the app is sending a request, which may be delayed. The app times out, and retransmits. If this happens more than a certain number of times consecutively, the buffer overflows. This is independent of the overall rate.

A short protocol analyzer session would verify this: Gambit ships a free (unsupported) protocol analyzer called tcpdump downloadable from <http://www.gambitcomm.com/unsupported>. If you run tcpdump as follows from root:

```
# tcpdump -s 256 -n host agent-IP-address and port 161
```

it will dump all SNMP packets to/from that agent IP address. Run this analyzer until the buffer full error happens. Then send us the output.

An aggressive retransmit policy could be a bug in the application, ie. it will have performance problems interacting with any agent, whether MIMIC or anybody else's.

13. WARN **DATE** - USM Error: sending report PDU

```
DATE - unknown engine id: ""
```

Cause

This warning is an indication that the management application did a discovery of the SNMPv3 engine ID from the agent: in order to detect the engine ID it sends an illegal engine ID, causing the agent to respond with an unknownEngineID REPORT PDU.

Action

None necessary, if your SNMPv3 management application works correctly. Otherwise, you may have to do manual configuration of the engine ID.

14. WARN **DATE** - agent **NUMBER** cannot remove primary alias

```
DATE - no receiver thread for socket NUMBER
```

```
WARN DATE - cannot clear pollfd NUMBER
```

```
DATE - T[x], socket NUMBER not registered
```

```
WARN DATE - cannot clear pollfd NUMBER
```

```
DATE - T[x], socket NUMBER not found in map
```

```
WARN DATE - NR[x]: continuing poll on agents...
```

DATE - socket **NUMBER** not registered

WARN **DATE** - cannot switch socket **NUMBER** to inactive thread, continuing...

DATE - command buffer overflow

Cause

Any of these messages indicates temporary problems in the protocol dispatcher in the simulator. These messages happen occasionally, under extreme stress of the simulator.

Action

The messages indicate a recovery action by the simulator, and may result in dropped messages to the agents. No action is necessary, unless the messages happen frequently. If they do, please contact support@gambitcomm.com to remedy the problem.

15. ERROR **DATE** - **PROTOCOL** [AGT=**NUMBER**]: cannot start server

DATE - cannot enable IP address for

DATE - cannot start ipalias

DATE - cannot open socket

DATE - Please refer to Appendix C for more details

DATE - cannot bind receive IP address **ADDRESS** port **PORT**

DATE - bind: Address already in use.

Cause

This error means that there is already a service running on the indicated port, eg. there is already a Telnet service running on port 23. Only one service can be bound to a port at a time. On Linux systems, the Telnet service is managed with xinetd.

Action

You have 2 options:

1. start the MIMIC protocol server on a different port. Eg. start the telnet server on port 2423.
2. disable the platform-native Telnet service. On Linux systems, this can be accomplished with `/usr/sbin/setup`, in the System Services menu, or via `chkconfig`.

[<< Previous Section](#) [Next Section >>](#)



MIMIC Virtual Lab Online Documentation

MIMIC Virtual Lab Frequently Asked Questions

Last updated Fri May 14 08:29:02 EDT 2010

Save the latest [FAQ page](#) into your MIMIC installation help/ directory to make it an integral part of your online documentation.

Table of Contents

General

1. [Q. What is a "gambit"?](#)
2. [Q. What are the hardware considerations for running MIMIC?](#)
3. [Q. Can I run MIMIC Virtual Lab on Windows 7?](#)
4. [Q. Can I run MIMIC Virtual Lab on Windows 95 / 98 / Me?](#)
5. [Q. Do you have any SNMP tools for Windows?](#)
6. [Q. How do you customize and program MIMIC?](#)
7. [Q. How can I trace PDU exchanges between my management application and MIMIC?](#)
8. [Q. What is the best way of reporting problems with MIMIC?](#)
9. [Q. Is there a way to modify the configuration of MIMIC to use another drive for its use?](#)
10. [Q. Will there be a port conflict if I install the MIMIC software on a machine with HPOV Network Node Manager 5.01 installed?](#)
11. [Q. Why does HPOV Network Node Manager not discover the MIMIC agents?](#)
12. [Q. Can you run MIMIC and a Web Server on the same machine?](#)
13. [Q. How do I apply new license keys to the installed software?](#)
14. [Q. Why is my firewall warning me about access to the Internet?](#)

Simulator

1. [Q. I cannot start agents in MIMIC. I get errors in the log when starting an agent instance. Why?](#)
2. [Q. I have started agents in MIMIC, but I cannot ping them from my management station. Why?](#)
3. [Q. When I run a simulation, I see some diagnostic messages in the log window. What do they mean?](#)
4. [Q. I am seeing the message "buffer full from ADDRESS to ADDRESS" in the error log. What does it mean?](#)
5. [Q. How do I change the look and feel of the user interface?](#)

General

Q. What is a "gambit"?

A. A "gambit" is a term from the game of chess, a risky opening move with high potential return. Also see [Merriam-Webster Dictionary](#):

Main Entry: gambit
Pronunciation: 'gam-b&t
Function: noun
Etymology: Italian gambetto, literally, act of tripping someone, from gamba leg, from Late Latin gamba, camba, from Greek kampE bend; probably akin to Gothic hamfs maimed, Lithuanian kampas corner Date: 1656
1 : a chess opening in which a player risks one or more minor pieces to gain an advantage in position
2 a (1) : a remark intended to start a conversation or make a telling point
(2) : TOPIC
b : a calculated move : STRATAGEM

Q. What are the hardware considerations for running MIMIC?

> In your support page, you give the minimum hardware configuration and
> the preferred hardware configuration, but the number of agents simulated
> doesn't seem to be taken into account. I mean is the hardware configuration
> the same for all the versions of MIMIC (MIMIC Single, MIMIC Lite, ...,
> MIMIC Global) which simulate different numbers of agents?

A. The preferred configuration listed on the web page is for a typical case. MIMIC Virtual Lab will run in any run-of-the-mill PC better than the minimum configuration listed. For MIMIC Simulator, read on.

Obviously, the more demanding the simulation, the more hardware you have to throw at it. The main thing to realize is that the hardware requirements for MIMIC can be viewed as satisfying an equation, eg.

performance (management side) <= performance (agent side)

meaning that the performance of the agent side (MIMIC) has to be at least as good as that needed by the management side. The performance requirements are thus driven in large part by the management application. The second point is that the equation is not controlled by a single variable (eg. "requests per second"). There are many variables which determine the exact demands on the simulation:

- the number of agents
- the set of MIBs for each agent
- the complexity of the simulations for the different MIB objects
- the trap generation rates
- the number of management threads (eg. pollers)
- the poll rates (average, sustained, peak)
- the make-up of the requests (single-variable vs. multi-variable vs. bulk)

This is just a partial list, but gives you an idea of the considerations. Ultimately, there is no generic answer and each customer has unique performance requirements. We can help you to determine these requirements through empirical evaluation. Your requirements may change over time, so your hardware solution should accommodate this change (more CPUs, more memory, more network cards). MIMIC is designed to take advantage of all the hardware you throw at it.

MIMIC supports up to 50,000 agents on one workstation. The main concern is the performance for a fully loaded workstation. You want at least hundreds of PDUs per second to make a simulation viable.

For MIMIC, performance is primarily governed by the amount of physical memory (RAM). The memory requirements depend on the simulations you are going to run. Obviously, a high-end router simulation with hundreds of interfaces, RMON tables, etc. is going to take more memory than the simulation of an end system.

As a ball-park estimate, we like to see at least 1MB of physical RAM per simulated agent, e.g., a 100 agent scenario should run fine on a 128MB system (depending on how much memory is used by the OS and other processes). For better performance (less swapping), 2MB per simulation is recommended.

After version 4.30, MIMIC has a new feature - memory optimization. That means more agents' MIB data needs less memory than before. Agents with identical simulation will only require one copy of data in memory. For example, in the common case if 10,000 agents are identical, only a couple of MB of RAM is needed. However, if 5000 agents are running the same simulation, and 5,000 agents are each different, then 5GB will be recommended.

You can more accurately measure this by running a simulation configuration, and checking on memory usage before and after starting the desired agent simulations. Notice that MIMIC uses memory on demand, so you should measure the memory after doing a walk of the desired tables (or a complete MIB walk). Eg. on Windows NT use the Windows Task Manager to check "Memory Usage", and on Unix use the "top" utility.

The memory usage by MIMIC is approximately the same for all platforms.

The CPU is of secondary importance. Most modern processors (e.g., Intel Pentium 800MHz or faster, and Ultra Sparc) are adequate. MIMIC works with multi-processor systems, since it is a multi-threaded, distributed application. Agent thread processing will be distributed across multiple CPUs. From our internal experience, we have run 10,000 agents on dual and quad-CPU Pentiums, and Ultra 10 to E4500 Sparcs.

The final bottleneck would be the network pipe to your agents. 10Mb Ethernet is adequate for low-volume traffic, 100Mb or faster is better for more demanding applications. MIMIC works with multiple network adapters on your system, so you can talk to the simulations over separate network pipes. MIMIC works with the OS-native protocol stacks, so that all network interface cards that your OS supports can be used. You can even run MIMIC over PPP.

Q. Can I run MIMIC Virtual Lab on Windows 7?

A. MIMIC Virtual Lab based on MIMIC Simulator 10.20 or later supports Windows 7.

Q. Can I run MIMIC Virtual Lab on Windows 95 / 98 / Me?

A. There is no support for MIMIC Virtual Lab on Windows 95 / 98 / Me as documented in the online documentation [Windows Installation Instructions](#).

Q. Do you have any SNMP tools for Windows?

A. You can download an unsupported binary distribution of the [Net-Snmp \(was UCD SNMP\) toolset](#) from [our website](#).

Q. How do you customize and program MIMIC?

> Also, how is MIMIC programmed? What programming language? Tcl?
> Scripts? What type of development environment? How customizable is it?
MIMIC Virtual Lab is a static environment, but MIMIC Simulator is highly customizable. Check for details at [our web site](#).

Q. How can I trace PDU exchanges between my management application and MIMIC?

A. Tcpdump is a free public domain protocol analyzer. Most Linux distributions include this valuable diagnostic tool. You can download an unsupported binary executable with SNMPv1 and SNMPv2 support from our website [for Solaris](#).

Ethereal is a great free public domain protocol analyzer. It decodes SNMPv3. Download it for your favorite platform from [their website http://www.ethereal.com/](#).

Q. What is the best way of reporting problems in MIMIC?

A. The fastest way of resolving problems is by sending e-mail to support@gambitcomm.com with a brief description of the problem, and supporting information, such as excerpts from the log window that show the problem.

If there is a workaround, we will let you know as soon as possible. If the problem requires a fix, we will open a trouble ticket and schedule it for an upcoming

release. All customers are notified of new releases as soon as they become available.

If you have a large supporting file from one of the tools (core file, log file, walk file), please don't email it yet since our mail server has limited resources (bandwidth and space). Tell us about the problem first, and we will ask you for the core file.

Q. Is there a way to modify the configuration of MIMIC to use another drive for its use?

The problem: the drive that MIMIC is using has become full due to MIMIC usage. Is there a way to modify the configuration of MIMIC to use another drive for its use?

This is running on an NT Server with a drive C and D. Mimic has been using drive C:

A. You can set the environment variable MIMIC_PRIV_DIR to point to a location on another drive to do this. Determine what directory is being used currently by MIMIC for storing your [private data](#). Copy this over to a different drive and then point MIMIC_PRIV_DIR to this location. Restart MIMIC fresh and all subsequent data should be stored on the new drive.

First, terminate MIMIC Virtual Lab (use File -> Exit).

- On Windows NT, use the System Control Panel to set MIMIC_PRIV_DIR to the new path.

Q. Will there be a port conflict if I install the MIMIC software on a machine with HPOV Network Node Manager 5.01 installed?

A. MIMIC should allow you to do what you want. The snag is that HP/Openview requires an SNMP agent to run on the management station. This agent conflicts with agent instances running on MIMIC. You have 2 choices:

a) run MIMIC on one machine, HP/Openview on another. This would require 2 laptops to do your demos. We have found this to be a better solution than b), because HP/Openview and other management applications put a lot of burden on the machine (memory, CPU utilization).

b) run MIMIC and HP/Openview on same machine, but this only works if you use non-standard port numbers for the MIMIC SNMP agent instances. You will have to configure HP/Openview to probe these non-standard ports.

Q. Why does HPOV Network Node Manager not discover the MIMIC agents?

A. HPOV does not discover foreign networks automatically. I quote from the NNM Runtime manual, section "Maps" --> "Customizing your Network Map View" --> "Expanding Your IP Network Map" --> "Adding a Network":

```
>For security purposes, Network Node Manager does not discover networks
>in your internet, beyond your local gateways. You can add an object for
>an network that NNM has not discovered to an Internet submap, by placing
>a network symbol on that submap. If you are adding an object for an IP
>network, NNM will eventually discover it. For network objects that NNM
>cannot discover, the network symbol remains on the user plane.
```

For example, if you want to discover the 192.9.201.0 network, you'll have to create a "IP Network" object in the "Internet" map.

Q. Can you run MIMIC and a Web Server on the same machine?

```
> Do you see any issues or know of things to watch out if
> we want to
>   o Run Both MIMIC and Apache web server on a linux machine
>   o Run both MIMIC and IIS web server on an NT machine?
```

A. MIMIC does not care what other servers run on your system, as long as they are not making use of the same ports as MIMIC (eg. UDP SNMP port 161). Since standard web servers use UDP ports 80 or 8080, there is minimal likelihood of a clash.

Q. How do I apply new license keys to the installed software?

A. If you want to apply new license keys to an already installed version of MIMIC (eg. if you want to change the evaluation keys to permanent keys, or upgrade in size), all you have to do is edit the license key files in config/*.lic . There is one file per licensed component (Simulator, Compiler, Recorder). Open each file with your favorite text editor, and copy/paste the corresponding key.

Q. Why is my firewall warning me about access to the Internet?

```
> Why when I try to telnet my firewall tells me that the prog is trying
> to access the internet?
```

A. When you use the Device->Telnet... menu item, MIMIC Virtual Lab just invokes the native Telnet application of the OS that MIMIC is running on. Just like in the real world, the Telnet is connecting to the IP address of the simulated device, and likely it is trying to do DNS name resolution on that IP address.

Another reason for Internet access is the Update Wizard trying to download updates for your software. The Update Wizard runs in a program called wish, which will be reported in the firewall popup. If you allow access to our download site, you will be able to install updates and optional software.

Simulator

Q. I cannot start agents in MIMIC. I get errors in the log window when starting an agent instance. Why?

A. This problem is likely caused by the existence of another SNMP agent running on this system. The solution is explained in detail in [Appendix C](#), for [Solaris](#) or [Windows NT](#).

Q. I have started agents in MIMIC, but I cannot ping them from my management station. Why?

A. When you start agent instances with IP addresses on a subnet different from the one that your management station is on, you need to tell the management station how to get to the subnet.

This can be done in most operating systems via a static route with the `route` command. Assuming that your agent instances are on the 192.9.200.0 subnet and that the address of your management station machine is **IPADDR**, here are the route commands for some common operating systems:

- Windows NT
From the DOS command prompt:
`C> route add 192.9.200.0 mask 255.255.255.0 IPADDR`
- Solaris 2.6, 7
From any shell as root:

To add a route:
`# route add -net 192.9.200.0 IPADDR 0`

To delete a route:
`# route delete -net 192.9.200.0 IPADDR`
- Solaris 2.5
From any shell as root:
`# route add 192.9.200.0 IPADDR 0`
- Red Hat Linux 5.x
From any shell as root:
`# route add -net 192.9.200.0 gw IPADDR`
- Red Hat Linux 6.x
From any shell as root:

To add a route:
`# route add -net 192.9.200.0 netmask 255.255.255.0 INTERFACE`

To delete a route:
`# route del -net 192.9.200.0 netmask 255.255.255.0`
- HP/UX
From any shell as root:
`# route add 192.9.200.0 IPADDR 0`

Q. When I run a simulation, I see some diagnostic messages in the log window. What do they mean?

A. MIMIC does extensive error logging to justify its actions. If something is not going the way you want it, you can find out why from the error log. The error log is normally displayed in a log window with the [Lab->Troubleshoot](#) menu item, as well as dumped into a file `mimiclog.date.time` in your temporary directory (`/tmp` in Unix, `\TMP` in Windows).

The most common error messages are described in [Appendix C](#) of the online documentation.

Q. I am seeing the message "buffer full from ADDRESS to ADDRESS" in the error log. What does it mean?

A. The details for this error message are described in [Appendix C](#) of the online documentation.

Q. How do I change the look and feel of the user interface?

A. The MIMIC Virtual Lab Java-based user interface is highly configurable as documented by the [Sun Java documentation](#). By default, the user interface runs with the Java Metal look and feel. The easiest way to change the look and feel is to specify it in the batch file which starts the application, eg. on Windows to get the Windows interface look and feel (including inheriting its properties as set in the Display Properties dialog) in `vlab*.bat`, use the additional

```
-Dswing.defaultlaf=com.sun.java.swing.plaf.windows.WindowsLookAndFeel  
on the java invocation.
```

- Agent
- Client
- Cloud
- Debug
- debugSession
- ErrorInfo
- Mgmt
- Oid
- Pod
- SearchPath
- Session
- StringParser
- Utils
- Valuespace

Package Mimic

Class Summary

Class	Description
Agent	
Client	
Cloud	
Debug	
debugSession	
ErrorInfo	
Mgmt	
Oid	
Pod	
SearchPath	
Session	
StringParser	
Utils	
Valuespace	

Mimic

Class Agent

java.lang.Object
Mimic.Agent

```
public class Agent  
extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
static int	AGENT_HALTED agent's state : halted
static int	AGENT_NOT_CONFIG agent's state : not configured
static int	AGENT_PAUSED agent's state : paused
static int	AGENT_RUNNING agent's state : running
static int	AGENT_STOPPED agent's state : stopped
static int	AGENT_STOPPING agent's state : stopping (or starting)

Constructor Summary

Constructors

Constructor and Description
Agent (int agent_no, Session session) Creates an agent object for a given session

Method Summary

All Methods	Static Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description		
void	add_ipalias (java.lang.String address, int port, java.lang.String mask, java.lang.String iface) Add a new ipalias (See mimic agent ipalias add)		
void	add_timer_script (java.lang.String script, int interval, java.lang.String args) Add (and start) timerscript with given interval and arguments (See mimic timer script add)		
void	agent_store_append (java.lang.String var, java.lang.String val, int persist) Append (and start) storage with a given variable and value (See mimic agent store append)		
void	agent_store_copy (int other) Copy variable store from other agent (See mimic agent store copy)		
int	agent_store_exists (java.lang.String var) Examine existance of a store variable (See mimic agent store exists)		
java.lang.String	agent_store_get (java.lang.String var) Return variable value (See mimic agent store get)		
java.util.LinkedList	agent_store_list () List variables (See mimic agent store list)		
void	agent_store_lreplace (java.lang.String var, int index, java.lang.String val) Treat variable as a list and modify entry for given index with given value (See mimic agent store lreplace)		
java.util.LinkedList	agent_store_mget (java.util.LinkedList vars) Get multiple variable values (See mimic agent store mget)		
void	agent_store_mlreplace (java.util.LinkedList varlist, java.util.LinkedList indexlist, java.util.LinkedList vallist) Treat each variable as a list and modify entry for given index with given value (See mimic agent store mlreplace)		
int	agent_store_persists (java.lang.String var) Examine persistence of a store variable (See mimic agent store persists)		
void	agent_store_set (java.lang.String var, java.lang.String val, int persist) Set (and start) storage with a given variable and value (See mimic agent		

store set)

void	<code>agent_store_unset</code> (java.lang.String var) Unset a variable-value pair (See mimic agent store unset)
void	<code>configure</code> (java.lang.String address, java.util.LinkedList mibs) Configure agent
void	<code>del_ipalias</code> (java.lang.String address, int port) Delete a configured ipalias (See mimic agent ipalias delete)
void	<code>del_timer_script</code> (java.lang.String script) Delete a running script with specified name (See mimic timer script delete)
void	<code>from_add</code> (java.lang.String host, int port) Add a new 'from' entry (See mimic agent from add)
void	<code>from_del</code> (java.lang.String host, int port) Delete a configured 'from' entry (See mimic agent from delete)
java.util.LinkedList	<code>from_list</code> () List the 'from' entry list for the agent (See mimic agent from list)
int	<code>get_agent_no</code> () Get agent's number
int	<code>get_changed</code> () Get the changed flag (See mimic agent get)
int	<code>get_config_changed</code> () Get the config changed flag (See mimic agent get)
int	<code>get_delay</code> () Get the delay (See mimic agent get)
int	<code>get_drops</code> () Get the drop rate (See mimic agent get)
java.lang.String	<code>get_host</code> () Get the primary ip address (See mimic agent get)
int	<code>get_inform_retries</code> () Get the inform retries (See mimic agent get)
int	<code>get_inform_timeout</code> () Get the inform timeout (See mimic agent get)
java.lang.String	<code>get_interface</code> () Get the interface for the primary ipalias (See mimic agent get)
java.lang.String	<code>get_mask</code> () Get the primary subnet mask (See mimic agent get)

java.util.LinkedList	<code>get_mibs()</code> Get the MIB triplets (See mimic agent get)
java.lang.String	<code>get_oiddir()</code> Get the OidDir filename (See mimic agent get)
java.lang.String	<code>get_owner()</code> Get the owner (See mimic agent get)
int	<code>get_pdu_size()</code> Get the maximum PDU size (See mimic agent get)
int	<code>get_port()</code> Get the port number (See mimic agent get)
java.lang.String	<code>get_privdir()</code> Get the private directory (See mimic agent get)
java.util.LinkedList	<code>get_protocols()</code> Get the list of protocols (See mimic agent get)
java.lang.String	<code>get_read_community()</code> Get the read community (See mimic agent get)
java.lang.String	<code>get_scen()</code> Get the simulation scen (See mimic agent get)
Session	<code>get_session()</code> Get agent's session
java.lang.String	<code>get_sim()</code> Get the sim name (See mimic agent get)
int	<code>get_starttime()</code> Get the start time (See mimic agent get)
int	<code>get_state_changed()</code> Get the state changed flag (See mimic agent get)
static java.lang.String	<code>get_state_descr(int state)</code> Get the description of the specified agent state (See mimic agent get)
int	<code>get_state()</code> Get the agent state (See mimic agent get)
java.util.LinkedList	<code>get_statistics_Long()</code> Get the list of statistics (Long) (See mimic agent get)
java.util.LinkedList	<code>get_statistics_String()</code> Get the list of statistics (String) (See mimic agent get)
java.util.LinkedList	<code>get_statistics()</code> Get the list of statistics (See mimic agent get)

int	<code>get_trace()</code> Get the trace flag (See mimic agent get)
int	<code>get_validate()</code> Get the validate flag (See mimic agent get)
Valuespace	<code>get_valuespace()</code> Get the valuespace object for this agent.
java.lang.String	<code>get_write_community()</code> Get the write community (See mimic agent get)
void	<code>halt()</code> Halt agent (from running) (See mimic agent halt)
java.util.LinkedList	<code>list_ipaliases()</code> List the ipaliases for the agent (See mimic agent ipalias list)
java.util.LinkedList	<code>list_timer_scripts()</code> List the currently running timer scripts (See mimicsh_timer_script_list)
void	<code>pause_at(int time)</code> Pause agent at given time (from running) (See mimic agent pause)
void	<code>pause_now()</code> Pause agent now (from running) (See mimic agent pause)
java.lang.String	<code>protocol_msg(java.lang.String protocol, java.lang.String msg)</code> Pass agent's message string to the protocol
void	<code>remove()</code> Delete a configured agent (See mimic agent remove)
void	<code>resume()</code> Resume agent (from paused/halted) (See mimic agent resume)
void	<code>save()</code> Save the agent's valuespace changes (See mimic agent save)
void	<code>set_delay(int delay)</code> Set the delay (See mimic agent set)
void	<code>set_drops(int drops)</code> Set the drop rate (See mimic agent set)
void	<code>set_host(java.lang.String host)</code> Set the primary ip address (See mimic agent set)
void	<code>set_inform_retries(int retries)</code> Set the inform retries (See mimic agent set)
void	<code>set_inform_timeout(int timeout)</code>

	Set the inform timeout (See mimic agent set)
void	<code>set_interface</code> (java.lang.String iface) Set the interface for primary ipalias (See mimic agent set)
void	<code>set_mask</code> (java.lang.String mask) Set the primary subnet mask (See mimic agent set)
void	<code>set_mibs</code> (java.util.LinkedList mibs) Set the MIB triplets (See mimic agent set)
void	<code>set_oiddir</code> (java.lang.String oiddir) Set the OidDir filename (See mimic agent set)
void	<code>set_pdu_size</code> (int pdu_size) Set the maximum PDU size (See mimic agent set)
void	<code>set_port</code> (int port) Set the port number (See mimic agent set)
void	<code>set_protocols</code> (java.util.LinkedList protocols) Set the list of protocols supported (See mimic agent set)
void	<code>set_read_community</code> (java.lang.String readcomm) Set the read community (See mimic agent set)
void	<code>set_starttime</code> (int starttime) Set the starttime (See mimic agent set)
void	<code>set_trace</code> (int flag) Set the trace flag (See mimic agent set)
void	<code>set_validate</code> (int flag) Set the validate flag (See mimic agent set)
void	<code>set_write_community</code> (java.lang.String writecomm) Set the write community (See mimic agent set)
void	<code>start_ipalias</code> (java.lang.String address, int port) Start the configured ipalias (See mimic agent ipalias start)
void	<code>start</code> () Start agent (from stopped) (See mimic agent start)
int	<code>status_ipalias</code> (java.lang.String address, int port) Status of the configured ipalias for the agent (See mimic agent ipalias status)
void	<code>stop_ipalias</code> (java.lang.String address, int port) Stop the configured ipalias (See mimic agent ipalias stop)
void	<code>stop</code> () Stop agent (from running, paused, halted) (See mimic agent stop)

void	<code>trap_config_add</code> (java.lang.String address, int port) Set the trap destination (See mimic agent trap config add)
void	<code>trap_config_del</code> (java.lang.String address, int port) Remove the trap destination from list (See mimic agent trap config delete)
java.util.LinkedList	<code>trap_config_list</code> () Get the trap destination list (See mimic agent trap config list)
java.util.LinkedList	<code>trap_list</code> () Get the trap list

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

AGENT_RUNNING

```
public static final int AGENT_RUNNING
```

agent's state : running

See Also:

[Constant Field Values](#)

AGENT_STOPPED

```
public static final int AGENT_STOPPED
```

agent's state : stopped

See Also:

[Constant Field Values](#)

AGENT_HALTED

```
public static final int AGENT_HALTED
```

agent's state : halted

See Also:

[Constant Field Values](#)

AGENT_PAUSED

```
public static final int AGENT_PAUSED
```

agent's state : paused

See Also:

[Constant Field Values](#)

AGENT_NOT_CONFIG

```
public static final int AGENT_NOT_CONFIG
```

agent's state : not configured

See Also:

[Constant Field Values](#)

AGENT_STOPPING

```
public static final int AGENT_STOPPING
```

agent's state : stopping (or starting)

See Also:

[Constant Field Values](#)

Constructor Detail

Agent

```
public Agent(int agent_no,  
             Session session)
```

Creates an agent object for a given session

Parameters:

agent_no - (IN) agent number

session - (IN) session object

Method Detail

get_agent_no

```
public int get_agent_no()
```

Get agent's number

Returns:

int - agent's number

get_session

```
public Session get_session()
```

Get agent's session

Returns:

Session - agent's session

start

```
public void start()  
        throws ErrorInfo
```

Start agent (from stopped) (See [mimic agent start](#))

Throws:

[ErrorInfo](#) - on error

stop

```
public void stop()  
        throws ErrorInfo
```

Stop agent (from running, paused, halted) (See [mimic agent stop](#))

Throws:

[ErrorInfo](#) - on error

pause_now

```
public void pause_now()  
        throws ErrorInfo
```

Pause agent now (from running) (See [mimic agent pause](#))

Throws:

[ErrorInfo](#) - on error

pause_at

```
public void pause_at(int time)
    throws ErrorInfo
```

Pause agent at given time (from running) (See [mimic agent pause](#))

Parameters:

time - int

Throws:

`ErrorInfo` - on error

halt

```
public void halt()
    throws ErrorInfo
```

Halt agent (from running) (See [mimic agent halt](#))

Throws:

`ErrorInfo` - on error

resume

```
public void resume()
    throws ErrorInfo
```

Resume agent (from paused/halted) (See [mimic agent resume](#))

Throws:

`ErrorInfo` - on error

configure

```
public void configure(java.lang.String address,
    java.util.LinkedList mibs)
    throws ErrorInfo
```

Configure agent

Parameters:

address - (IN) ip-address

mibs - (IN) MIB triplets

Throws:

`ErrorInfo` - on error

remove

```
public void remove()  
    throws ErrorInfo
```

Delete a configured agent (See `mimic agent remove`)

Throws:

`ErrorInfo` - on error

get_interface

```
public java.lang.String get_interface()  
    throws ErrorInfo
```

Get the interface for the primary ipalias (See `mimic agent get`)

Returns:

String - interface

Throws:

`ErrorInfo` - on error

get_host

```
public java.lang.String get_host()  
    throws ErrorInfo
```

Get the primary ip address (See `mimic agent get`)

Returns:

String - ip-address

Throws:

`ErrorInfo` - on error

get_mask

```
public java.lang.String get_mask()  
    throws ErrorInfo
```

Get the primary subnet mask (See `mimic agent get`)

Returns:

String - subnet mask

Throws:

`ErrorInfo` - on error

get_port

```
public int get_port()  
    throws ErrorInfo
```

Get the port number (See [mimic agent get](#))

Returns:

int - port number

Throws:

`ErrorInfo` - on error

get_read_community

```
public java.lang.String get_read_community()  
    throws ErrorInfo
```

Get the read community (See [mimic agent get](#))

Returns:

String - read community

Throws:

`ErrorInfo` - on error

get_write_community

```
public java.lang.String get_write_community()  
    throws ErrorInfo
```

Get the write community (See [mimic agent get](#))

Returns:

String - write community

Throws:

`ErrorInfo` - on error

get_mibs

```
public java.util.LinkedList get_mibs()  
    throws ErrorInfo
```

Get the MIB triplets (See [mimic agent get](#))

Returns:

LinkedList - list of MIB triplets

Throws:

`ErrorInfo` - on error

get_sim

```
public java.lang.String get_sim()  
    throws ErrorInfo
```

Get the sim name (See [mimic agent get](#))

Returns:

`String` - sim name

Throws:

`ErrorInfo` - on error

get_scen

```
public java.lang.String get_scen()  
    throws ErrorInfo
```

Get the simulation scen (See [mimic agent get](#))

Returns:

`String` - scenario

Throws:

`ErrorInfo` - on error

get_pdusize

```
public int get_pdusize()  
    throws ErrorInfo
```

Get the maximum PDU size (See [mimic agent get](#))

Returns:

`int` - PDU size

Throws:

`ErrorInfo` - on error

get_protocols

```
public java.util.LinkedList get_protocols()  
    throws ErrorInfo
```

Get the list of protocols (See [mimic agent get](#))

Returns:

`LinkedList` - list of protocols

Throws:

`ErrorInfo` - on error

get_delay

```
public int get_delay()
    throws ErrorInfo
```

Get the delay (See [mimic agent get](#))

Returns:

`int` - delay

Throws:

`ErrorInfo` - on error

get_starttime

```
public int get_starttime()
    throws ErrorInfo
```

Get the start time (See [mimic agent get](#))

Returns:

`int` - start time

Throws:

`ErrorInfo` - on error

get_state

```
public int get_state()
    throws ErrorInfo
```

Get the agent state (See [mimic agent get](#))

Returns:

`int` - state

Throws:

`ErrorInfo` - on error

get_statistics

```
public java.util.LinkedList get_statistics()
```

throws `ErrorInfo`

Get the list of statistics (See [mimic agent get](#))

Returns:

`LinkedList` - SNMP statistics (total discarded error GET GETNEXT SET BULK TRAP GETVAR GETNEXTVAR SETVAR BULKVAR)

Throws:

`ErrorInfo` - on error

`get_statistics_Long`

```
public java.util.LinkedList get_statistics_Long()  
                               throws ErrorInfo
```

Get the list of statistics (Long) (See [mimic agent get](#))

Returns:

`LinkedList(Long)` - SNMP statistics (total discarded error GET GETNEXT SET BULK TRAP GETVAR GETNEXTVAR SETVAR BULKVAR)

Throws:

`ErrorInfo` - on error

`get_statistics_String`

```
public java.util.LinkedList get_statistics_String()  
                               throws ErrorInfo
```

Get the list of statistics (String) (See [mimic agent get](#))

Returns:

`LinkedList(String)` - SNMP statistics (total discarded error GET GETNEXT SET BULK TRAP GETVAR GETNEXTVAR SETVAR BULKVAR)

Throws:

`ErrorInfo` - on error

`get_drops`

```
public int get_drops()  
           throws ErrorInfo
```

Get the drop rate (See [mimic agent get](#))

Returns:

`int` - drop rate

Throws:

`ErrorInfo` - on error

get_changed

```
public int get_changed()  
    throws ErrorInfo
```

Get the changed flag (See [mimic agent get](#))

Returns:

int - changed flag

Throws:

`ErrorInfo` - on error

get_config_changed

```
public int get_config_changed()  
    throws ErrorInfo
```

Get the config changed flag (See [mimic agent get](#))

Returns:

int - config changed flag

Throws:

`ErrorInfo` - on error

get_state_changed

```
public int get_state_changed()  
    throws ErrorInfo
```

Get the state changed flag (See [mimic agent get](#))

Returns:

int - state changed flag

Throws:

`ErrorInfo` - on error

get_trace

```
public int get_trace()  
    throws ErrorInfo
```

Get the trace flag (See [mimic agent get](#))

Returns:

int - trace flag

Throws:

`ErrorInfo` - on error

get_validate

```
public int get_validate()  
        throws ErrorInfo
```

Get the validate flag (See [mimic agent get](#))

Returns:

int - validate flag

Throws:

`ErrorInfo` - on error

get_owner

```
public java.lang.String get_owner()  
        throws ErrorInfo
```

Get the owner (See [mimic agent get](#))

Returns:

String - owner name

Throws:

`ErrorInfo` - on error

get_privdir

```
public java.lang.String get_privdir()  
        throws ErrorInfo
```

Get the private directory (See [mimic agent get](#))

Returns:

String - private directory

Throws:

`ErrorInfo` - on error

get_oiddir

```
public java.lang.String get_oiddir()  
        throws ErrorInfo
```


Get the OidDir filename (See mimic agent get)

Returns:

String - OidDir filename

Throws:

`ErrorInfo` - on error

get_inform_timeout

```
public int get_inform_timeout()  
           throws ErrorInfo
```

Get the inform timeout (See mimic agent get)

Returns:

int - timeout

Throws:

`ErrorInfo` - on error

get_inform_retries

```
public int get_inform_retries()  
           throws ErrorInfo
```

Get the inform retries (See mimic agent get)

Returns:

int - retries

Throws:

`ErrorInfo` - on error

trap_list

```
public java.util.LinkedList trap_list()  
           throws ErrorInfo
```

Get the trap list

Returns:

Linked list of trap OIDs

Throws:

`ErrorInfo` - on error

trap_config_list

```
public java.util.LinkedList trap_config_list()  
                                throws ErrorInfo
```

Get the trap destination list (See [mimic agent trap config list](#))

Returns:

Linked list of trap destination,port

Throws:

`ErrorInfo` - on error

trap_config_add

```
public void trap_config_add(java.lang.String address,  
                            int port)  
                            throws ErrorInfo
```

Set the trap destination (See [mimic agent trap config add](#))

Parameters:

address - target ip address

port - target port

Throws:

`ErrorInfo` - on error

trap_config_del

```
public void trap_config_del(java.lang.String address,  
                            int port)  
                            throws ErrorInfo
```

Remove the trap destination from list (See [mimic agent trap config delete](#))

Parameters:

address - target ip address

port - target port

Throws:

`ErrorInfo` - on error

set_interface

```
public void set_interface(java.lang.String iface)  
                            throws ErrorInfo
```

Set the interface for primary ipalias (See [mimic agent set](#))

Parameters:

iface - (IN) interface

Throws:

`ErrorInfo` - on error

set_host

```
public void set_host(java.lang.String host)
                throws ErrorInfo
```

Set the primary ip address (See [mimic agent set](#))

Parameters:

host - (IN) ip-address

Throws:

`ErrorInfo` - on error

set_mask

```
public void set_mask(java.lang.String mask)
                throws ErrorInfo
```

Set the primary subnet mask (See [mimic agent set](#))

Parameters:

mask - (IN) subnet mask

Throws:

`ErrorInfo` - on error

set_port

```
public void set_port(int port)
                throws ErrorInfo
```

Set the port number (See [mimic agent set](#))

Parameters:

port - (IN) port number

Throws:

`ErrorInfo` - on error

set_read_community

```
public void set_read_community(java.lang.String readcomm)
```

throws `ErrorInfo`

Set the read community (See [mimic agent set](#))

Parameters:

`readcomm` - (IN) read community

Throws:

`ErrorInfo` - on error

set_write_community

```
public void set_write_community(java.lang.String writecomm)
    throws ErrorInfo
```

Set the write community (See [mimic agent set](#))

Parameters:

`writecomm` - (IN) write community

Throws:

`ErrorInfo` - on error

set_protocols

```
public void set_protocols(java.util.LinkedList protocols)
    throws ErrorInfo
```

Set the list of protocols supported (See [mimic agent set](#))

Parameters:

`protocols` - (IN) list of protocol strings

Throws:

`ErrorInfo` - on error

set_delay

```
public void set_delay(int delay)
    throws ErrorInfo
```

Set the delay (See [mimic agent set](#))

Parameters:

`delay` - (IN) delay

Throws:

`ErrorInfo` - on error

set_starttime

```
public void set_starttime(int starttime)
    throws ErrorInfo
```

Set the starttime (See [mimic agent set](#))

Parameters:

starttime - (IN) start time

Throws:

[ErrorInfo](#) - on error

set_mibs

```
public void set_mibs(java.util.LinkedList mibs)
    throws ErrorInfo
```

Set the MIB triplets (See [mimic agent set](#))

Parameters:

mibs - (IN) list of MIB triplet strings

Throws:

[ErrorInfo](#) - on error

set_trace

```
public void set_trace(int flag)
    throws ErrorInfo
```

Set the trace flag (See [mimic agent set](#))

Parameters:

flag - (IN) trace flag

Throws:

[ErrorInfo](#) - on error

set_pdu_size

```
public void set_pdu_size(int pdu_size)
    throws ErrorInfo
```

Set the maximum PDU size (See [mimic agent set](#))

Parameters:

pdu_size - (IN) PDU size

Throws:

`ErrorInfo` - on error

set_drops

```
public void set_drops(int drops)
    throws ErrorInfo
```

Set the drop rate (See [mimic agent set](#))

Parameters:

drops - (IN) drop rate

Throws:

`ErrorInfo` - on error

set_validate

```
public void set_validate(int flag)
    throws ErrorInfo
```

Set the validate flag (See [mimic agent set](#))

Parameters:

flag - (IN) validate flag

Throws:

`ErrorInfo` - on error

set_oiddir

```
public void set_oiddir(java.lang.String oiddir)
    throws ErrorInfo
```

Set the OidDir filename (See [mimic agent set](#))

Parameters:

oiddir - (IN) OidDir filename

Throws:

`ErrorInfo` - on error

set_inform_timeout

```
public void set_inform_timeout(int timeout)
    throws ErrorInfo
```

Set the inform timeout (See [mimic agent set](#))

Parameters:

`timeout - (IN) timeout secs`

Throws:

`ErrorInfo - on error`

set_inform_retries

```
public void set_inform_retries(int retries)
    throws ErrorInfo
```

Set the inform retries (See [mimic agent set](#))

Parameters:

`retries - (IN) no of retries`

Throws:

`ErrorInfo - on error`

save

```
public void save()
    throws ErrorInfo
```

Save the agent's valuespace changes (See [mimic agent save](#))

Throws:

`ErrorInfo - on error`

add_ipalias

```
public void add_ipalias(java.lang.String address,
    int port,
    java.lang.String mask,
    java.lang.String iface)
    throws ErrorInfo
```

Add a new ipalias (See [mimic agent ipalias add](#))

Parameters:

`address - (IN) ip-address`

`port - (IN) port number`

`mask - (IN) subnet mask`

`iface - (IN) interface`

Throws:

`ErrorInfo - on error`

del_ipalias

```
public void del_ipalias(java.lang.String address,  
                        int port)  
    throws ErrorInfo
```

Delete a configured ipalias (See [mimic agent ipalias delete](#))

Parameters:

address - (IN) ip-address

port - (IN) port number

Throws:

[ErrorInfo](#) - on error

start_ipalias

```
public void start_ipalias(java.lang.String address,  
                          int port)  
    throws ErrorInfo
```

Start the configured ipalias (See [mimic agent ipalias start](#))

Parameters:

address - (IN) ip-address

port - (IN) port number

Throws:

[ErrorInfo](#) - on error

stop_ipalias

```
public void stop_ipalias(java.lang.String address,  
                         int port)  
    throws ErrorInfo
```

Stop the configured ipalias (See [mimic agent ipalias stop](#))

Parameters:

address - (IN) ip-address

port - (IN) port number

Throws:

[ErrorInfo](#) - on error

status_ipalias


```
public int status_ipalias(java.lang.String address,
                          int port)
    throws ErrorInfo
```

Status of the configured ipalias for the agent (See [mimic agent ipalias status](#))

Parameters:

address - (IN) ip address

port - (IN)

Returns:

int - status of the ipalias

Throws:

[ErrorInfo](#) - on error

list_ipaliases

```
public java.util.LinkedList list_ipaliases()
    throws ErrorInfo
```

List the ipaliases for the agent (See [mimic agent ipalias list](#))

Returns:

LinkedList - list of ipalias detail strings (address,port,mask,interface)

Throws:

[ErrorInfo](#) - on error

add_timer_script

```
public void add_timer_script(java.lang.String script,
                              int interval,
                              java.lang.String args)
    throws ErrorInfo
```

Add (and start) timerscript with given interval and arguments (See [mimic timer script add](#))

Parameters:

script - (IN) script to add

interval - (IN) interval in milli-seconds

args - (IN) arguments passed to the script

Throws:

[ErrorInfo](#) - on error

del_timer_script

```
public void del_timer_script(java.lang.String script)
    throws ErrorInfo
```

Delete a running script with specified name (See [mimic timer script delete](#))

Parameters:

script - (IN) script to delete

Throws:

`ErrorInfo` - on error

list_timer_scripts

```
public java.util.LinkedList list_timer_scripts()
    throws ErrorInfo
```

List the currently running timer scripts (See [mimicsh_timer_script_list](#))

Returns:

LinkedList list of scripts configured (script,interval,args)

Throws:

`ErrorInfo` - on error

protocol_msg

```
public java.lang.String protocol_msg(java.lang.String protocol,
    java.lang.String msg)
    throws ErrorInfo
```

Pass agent's message string to the protocol

Parameters:

protocol - (IN) protocol to send message to

msg - (IN) message string to be sent

Returns:

String reply buffer

Throws:

`ErrorInfo` - on error

get_valuespace

```
public Valuespace get_valuespace()
```

Get the valuespace object for this agent.

Returns:

Valuespace - agent's valuespace object

get_state_descr

```
public static java.lang.String get_state_descr(int state)
```

Get the description of the specified agent state (See [mimic agent get](#))

Parameters:

state - (IN)

Returns:

String - Description of agent state

agent_store_set

```
public void agent_store_set(java.lang.String var,  
                           java.lang.String val,  
                           int persist)  
    throws ErrorInfo
```

Set (and start) storage with a given variable and value (See [mimic agent store set](#))

Parameters:

var - (IN) variable to set

val - (IN) value to set

persist - (IN) persistence

Throws:

[ErrorInfo](#) - on error

agent_store_append

```
public void agent_store_append(java.lang.String var,  
                              java.lang.String val,  
                              int persist)  
    throws ErrorInfo
```

Append (and start) storage with a given variable and value (See [mimic agent store append](#))

Parameters:

var - (IN) variable to set

val - (IN) value to set

persist - (IN) persistence

Throws:

[ErrorInfo](#) - on error

agent_store_lreplace

```
public void agent_store_lreplace(java.lang.String var,  
                                int index,  
                                java.lang.String val)  
    throws ErrorInfo
```

Treat variable as a list and modify entry for given index with given value (See [mimic agent store lreplace](#))

Parameters:

var - (IN) variable to set

index - (IN) variable to set

val - (IN) value to set

Throws:

`ErrorInfo` - on error

agent_store_mlreplace

```
public void agent_store_mlreplace(java.util.LinkedList varlist,  
                                  java.util.LinkedList indexlist,  
                                  java.util.LinkedList vallist)  
    throws ErrorInfo
```

Treat each variable as a list and modify entry for given index with given value (See [mimic agent store mlreplace](#))

Parameters:

varlist - (IN) variable to set

indexlist - (IN) variable to set

vallist - (IN) value to set

Throws:

`ErrorInfo` - on error

agent_store_unset

```
public void agent_store_unset(java.lang.String var)  
    throws ErrorInfo
```

Unset a variable-value pair (See [mimic agent store unset](#))

Parameters:

var - (IN) variable to unset

Throws:

`ErrorInfo` - on error

Return variable value (See mimic agent store get)

Parameters:

var - (IN)

Returns:

value

Throws:

`ErrorInfo` - on error

agent_store_mget

```
public java.util.LinkedList agent_store_mget(java.util.LinkedList vars)
                                     throws ErrorInfo
```

Get multiple variable values (See mimic agent store mget)

Parameters:

vars - (IN) `LinkedList`

Returns:

`LinkedList` list of variables

Throws:

`ErrorInfo` - on error

agent_store_copy

```
public void agent_store_copy(int other)
                               throws ErrorInfo
```

Copy variable store from other agent (See mimic agent store copy)

Parameters:

other - (IN)

Throws:

`ErrorInfo` - on error

from_list

```
public java.util.LinkedList from_list()
                                     throws ErrorInfo
```

List the 'from' entry list for the agent (See mimic agent from list)

Returns:

Linked list entries of from host,port

Throws:

`ErrorInfo` - on error

from_add

```
public void from_add(java.lang.String host,
                    int port)
    throws ErrorInfo
```

Add a new 'from' entry (See mimic agent from add)

Parameters:

host - (IN) ip-address

port - (IN) port

Throws:

`ErrorInfo` - on error

from_del

```
public void from_del(java.lang.String host,
                    int port)
    throws ErrorInfo
```

Delete a configured 'from' entry (See mimic agent from delete)

Parameters:

host - (IN) ip-address

port - (IN) port

Throws:

`ErrorInfo` - on error

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class Client

```
java.lang.Object
  Mimic.Client
```

```
public class Client
  extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

Client()

Creates an default MIMIC client object

Method Summary

All Methods

Static Methods

Instance Methods

Concrete Methods

Modifier and Type

Method and Description

void

close_session(**Session** session)Closes an already open session (See **mimic session close**)

void

dump()

diagnostic dump

static boolean

isDebugEnabled()

java.util.LinkedList

list_sessions()returns list of all the open sessions (See **mimic session list**)**Session****open_session**(java.lang.String host, int port)Opens a new session to the host (See **mimic session open**)

static void

setDebugEnabled(boolean debugEnable)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait,

wait, wait

Constructor Detail

Client

```
public Client()
```

Creates an default MIMIC client object

Method Detail

open_session

```
public Session open_session(java.lang.String host,  
                             int port)  
    throws ErrorInfo
```

Opens a new session to the host (See [mimic session open](#))

Parameters:

host - (IN) ip-address of the host

port - (IN) management port number

Returns:

Session - newly opened session

Throws:

[ErrorInfo](#)

close_session

```
public void close_session(Session session)  
    throws ErrorInfo
```

Closes an already open session (See [mimic session close](#))

Parameters:

session - (IN) session to be closed

Throws:

[ErrorInfo](#)

list_sessions

```
public java.util.LinkedList list_sessions()
```

returns list of all the open sessions (See [mimic session list](#))

Returns:

LinkedList - list of open session

dump

```
public void dump()
```

diagnostic dump

isDebugEnabled

```
public static boolean isDebugEnabled()
```

setDebugEnabled

```
public static void setDebugEnabled(boolean debugEnable)
```

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class Cloud

```
java.lang.Object
Mimic.Cloud
```

```
public class Cloud
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

Cloud()**Method Summary**

All Methods

Static Methods

Concrete Methods

Modifier and Type	Method and Description
static java.lang.String	<code>_get_cloud_help_params</code> (Session session, java.lang.String key)
static java.lang.String	<code>_get_console_params</code> (Session session, int agent, java.lang.String key)
static java.lang.String	<code>_get_external_console_params</code> (Session session, int agent, java.lang.String key)
static java.lang.String	<code>_get_external_telnet_params</code> (Session session, int agent, java.lang.String key)
static java.lang.String	<code>_get_ssh_params</code> (Session session, int agent, java.lang.String key)
static java.lang.String	<code>_get_telnet_params</code> (Session session, int agent, java.lang.String key)
static boolean	<code>is_cloud_var_set</code> (Session session, java.lang.String var)

Methods inherited from class java.lang.Object

`_get_external_console_params`

```
public static java.lang.String _get_external_console_params(Session session,  
                                                             int agent,  
                                                             java.lang.String key)
```

`_get_ssh_params`

```
public static java.lang.String _get_ssh_params(Session session,  
                                                int agent,  
                                                java.lang.String key)
```

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class Debug

java.lang.Object
Mimic.Debug

```
public class Debug
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

Debug ()

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
static void	disable () Disables the display of debugging messages (using Debug.print) for the whole package.
static void	enable () Enables the display of debugging messages (using Debug.print) for the whole package.
static void	print (java.lang.String msg) Prints the debug message if debugging is enabled.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Debug

```
public Debug()
```

Method Detail

enable

```
public static void enable()
```

Enables the display of debugging messages (using `Debug.print`) for the whole package.

disable

```
public static void disable()
```

Disables the display of debugging messages (using `Debug.print`) for the whole package.

print

```
public static void print(java.lang.String msg)
```

Prints the debug message if debugging is enabled.

Parameters:

`msg` - (IN) - message string to print

[PACKAGE](#) [CLASS](#) [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class debugSession

java.lang.Object

Mimic.Session

Mimic.debugSession

```
public class debugSession
extends Session
```

Field Summary

Fields

Modifier and Type	Field and Description
int	sessionID

Constructor Summary

Constructors

Constructor and Description
debugSession (java.lang.String host, int port)

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
void	connect () Connects to the MIMIC daemon at the configured host and port
void	disconnect () Disconnects from the MIMIC daemon
java.lang.String	dispatch_msg (int req_id, java.lang.String req_msg) Dispatches a request to the MIMIC daemon, retrieves and returns the reply from the same


```
java.lang.String      dumpMarkers()  
  
static java.lang.String  getExceptionStackTraceAsString(java.lang.Exception exception)  
  
void                  mark(java.lang.String key, java.lang.Object value)
```

Methods inherited from class Mimic.Session

```
add_access, add_timer_script, agent_mget_changed, agent_mget_config_changed,  
agent_mget_delay, agent_mget_drops, agent_mget_host, agent_mget_inform_retries,  
agent_mget_inform_timeout, agent_mget_interface, agent_mget_mask, agent_mget_mibs,  
agent_mget_oiddir, agent_mget_owner, agent_mget_pdu_size, agent_mget_port,  
agent_mget_privdir, agent_mget_protocols, agent_mget_read_community,  
agent_mget_scen, agent_mget_sim, agent_mget_starttime, agent_mget_state_changed,  
agent_mget_state, agent_mget_statistics, agent_mget_trace, agent_mget_validate,  
agent_mget_write_community, agent_mset_delay, agent_mset_drops, agent_mset_host,  
agent_mset_inform_retries, agent_mset_inform_timeout, agent_mset_interface,  
agent_mset_mask, agent_mset_mibs, agent_mset_oiddir, agent_mset_pdu_size,  
agent_mset_port, agent_mset_protocol, agent_mset_read, agent_mset_starttime,  
agent_mset_trace, agent_mset_validate, agent_mset_write, cfg_load, cfg_load,  
cfg_load, cfg_load, cfg_new, cfg_new, cfg_save, cfg_save, cfg_saveas, cfg_saveas,  
del_access, del_timer_script, diag, diag, dump, get_access_acldb,  
get_access_admindir, get_access_adminuser, get_access_enabled,  
get_active_data_list_Long, get_active_data_list, get_active_list, get_agent,  
get_cfg_file_changed, get_cfgfile, get_changed_config_list, get_changed_state_list,  
get_clients, get_configured_list, get_expiration, get_host, get_interfaces,  
get_last, get_licensing, get_log, get_max, get_netaddr, get_netdev, get_port,  
get_privdir, get_product, get_protocols, get_sim_privdir, get_user, get_version,  
list_access, list_timer_scripts, load_access, protocol_msg, save_access,  
set_access_acldb, set_access_admindir, set_access_adminuser, set_access_enabled,  
set_log, start_all_agents, stop_all_agents, store_append, store_exists, store_get,  
store_list, store_lreplace, store_mget, store_mlreplace, store_persists,  
store_save, store_set, store_unset, terminate, trap_config_add, trap_config_delete,  
trap_config_list
```

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait,  
wait, wait
```

Field Detail

sessionID

```
public int sessionID
```

Constructor Detail

debugSession

```
public debugSession(java.lang.String host,
                    int port)
    throws ErrorInfo
```

Throws:

[ErrorInfo](#)

Method Detail

mark

```
public void mark(java.lang.String key,
                 java.lang.Object value)
```

dumpMarkers

```
public java.lang.String dumpMarkers()
```

connect

```
public void connect()
    throws ErrorInfo
```

Description copied from class: [Session](#)

Connects to the MIMIC daemon at the configured host and port

Overrides:

`connect` in class [Session](#)

Throws:

[ErrorInfo](#)

disconnect

```
public void disconnect()
    throws ErrorInfo
```

Description copied from class: [Session](#)

Disconnects from the MIMIC daemon

Overrides:

`disconnect` in class `Session`

Throws:

`ErrorInfo`

`getExceptionStackTraceAsString`

public

```
static java.lang.String getExceptionStackTraceAsString(java.lang.Exception exception)
```

`dispatch_msg`

```
public java.lang.String dispatch_msg(int req_id,  
                                     java.lang.String req_msg)  
    throws ErrorInfo
```

Description copied from class: `Session`

Dispatches a request to the MIMIC daemon, retrieves and returns the reply from the same

Overrides:

`dispatch_msg` in class `Session`

Parameters:

`req_id` - (IN) identifier for the request

`req_msg` - (IN) message buffer

Returns:

String reply for the command

Throws:

`ErrorInfo`

[PACKAGE](#) [CLASS](#) [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class `ErrorInfo`

```
java.lang.Object
  java.lang.Throwable
    Mimic.ErrorInfo
```

All Implemented Interfaces:

```
java.io.Serializable
```

```
public class ErrorInfo
  extends java.lang.Throwable
```

See Also:

[Serialized Form](#)

Field Summary

Fields

Modifier and Type	Field and Description
static int	FAILURE returned by <code>get_error_code()</code> indicating that no error occurred
static int	SUCCESS returned by <code>get_error_code()</code> indicating that some error occurred

Constructor Summary

Constructors

Constructor and Description

```
ErrorInfo(int err_code, java.lang.String err_msg)
```

Creates the `ErrorInfo` exception object thrown by the Mimic package on error conditions.

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method and Description
int	<code>get_error_code()</code> Get the error code for this exception
java.lang.String	<code>get_error_msg()</code> Get the error message string for this exception
java.lang.String	<code>getLocalizedMessage()</code>
java.lang.String	<code>getMessage()</code>

Methods inherited from class java.lang.Throwable

`addSuppressed`, `fillInStackTrace`, `getCause`, `getStackTrace`, `getSuppressed`, `initCause`, `printStackTrace`, `printStackTrace`, `printStackTrace`, `setStackTrace`, `toString`

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

SUCCESS

```
public static final int SUCCESS
```

returned by `get_error_code()` indicating that some error occurred

See Also:

[Constant Field Values](#)

FAILURE

```
public static final int FAILURE
```

returned by `get_error_code()` indicating that no error occurred

See Also:

[Constant Field Values](#)

Constructor Detail

ErrorInfo

```
public ErrorInfo(int err_code,  
                java.lang.String err_msg)
```

Creates the ErrorInfo exception object thrown by the Mimic package on error conditions.

Parameters:

err_code - (IN) error code

err_msg - (IN) error message for details

Method Detail

get_error_code

```
public int get_error_code()
```

Get the error code for this exception

Returns:

int - error code

get_error_msg

```
public java.lang.String get_error_msg()
```

Get the error message string for this exception

Returns:

String - error message

getLocalizedMessage

```
public java.lang.String getLocalizedMessage()
```

Overrides:

getLocalizedMessage in class java.lang.Throwable

getMessage

```
public java.lang.String getMessage()
```

Overrides:

getMessage in class java.lang.Throwable

Mimic

Class Mgmt

java.lang.Object
Mimic.Mgmt

```
public class Mgmt
    extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
static int	MGMT_DIAG_DISABLE_ERRORS
static int	MGMT_DIAG_ENABLE_ERRORS
static int	MGMT_PORT default management port

Constructor Summary

Constructors

Constructor and Description
Mgmt()

Method Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

MGMT_DIAG_ENABLE_ERRORS

```
public static final int MGMT_DIAG_ENABLE_ERRORS
```

See Also:

[Constant Field Values](#)

MGMT_DIAG_DISABLE_ERRORS

```
public static final int MGMT_DIAG_DISABLE_ERRORS
```

See Also:

[Constant Field Values](#)

MGMT_PORT

```
public static final int MGMT_PORT
```

default management port

See Also:

[Constant Field Values](#)

Constructor Detail

Mgmt

```
public Mgmt()
```

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class Oid

java.lang.Object
Mimic.Oid

```
public class Oid
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

oid()

Creates an empty Oid object

oid(java.lang.String string)

Creates an Oid object based on passed string

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type

Method and Description

java.lang.String

toString()

returns a string representation of the object

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Oid

```
public Oid()
```

Creates an empty Oid object

Oid

```
public Oid(java.lang.String string)
    throws ErrorInfo
```

Creates an Oid object based on passed string

Parameters:

string - - dotted notation string (e.g. 1.3.6.1.2.1)

Throws:

[ErrorInfo](#) - on error

Method Detail

toString

```
public java.lang.String toString()
```

returns a string representation of the object

Overrides:

toString in class [java.lang.Object](#)

Returns:

String - dotted notation OID string

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class Pod

java.lang.Object
Mimic.Pod

```
public class Pod
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

Pod(int pod_no, **Session** session)
Creates an pod object for a given session

Method Summary

All Methods

Static Methods

Instance Methods

Concrete Methods

Modifier and Type

Method and Description

static java.util.LinkedList	_get_configured_pods (Session session)
static Pod	_get_nth_pod (Session session, int pod_no)
static java.lang.String	_get_pod_id (Session session, int pod_no)
static int	find_by_id (Session session, java.lang.String user_key)
int	get_agent_position (int agentNum)
int	get_nth_agent (int agent_no)
int	get_nth_agent (int pod_no, int agent_no)
static int	get_nth_agent (Session session, int pod_no, int agent_no)
java.util.LinkedList	get_pod_agents (Session session, int pod)
static int	get_pod_assigned (Session session, java.lang.String source_ip)

static int	<code>get_pod_number(Session session, int agent_no)</code>
static Pod	<code>get_pod(Session session, int agent_no)</code>
static java.lang.Long	<code>get_stop_time(Session session, int podNum)</code>
static java.lang.String	<code>getAccessStatus(Session session, int podNum)</code>
static java.lang.String	<code>getDemoKeyPrefix(Session session)</code>
static int	<code>getDemoThrottle(Session session)</code>
static java.util.LinkedList	<code>getDemoThrottleException(Session session)</code>
static int	<code>getDemoThrottleFlushPeriod(Session session)</code>
static java.util.LinkedList	<code>getExpiredKeys(Session session)</code>
static int	<code>getExpireFlushFrequency(Session session)</code>
static int	<code>getInUseStatus(Session session, int podNum)</code>
static java.util.LinkedList	<code>getMessage(Session session, int podNum, java.lang.String type)</code>
static java.lang.String	<code>getPodMessage(Session session, int podNum)</code>
static java.lang.String	<code>getPoolName(Session session, int podNum)</code>
static java.lang.String	<code>getProxyServer(Session session, int podNum)</code>
static java.lang.String	<code>getTerminalServer(Session session, int podNum)</code>
static java.lang.String	<code>getVlabType(Session session, int podNum)</code>
static boolean	<code>isDemoThrottleOn(Session session)</code>
static boolean	<code>isExpireTrackOn(Session session)</code>
int	<code>number_of_agents()</code>
int	<code>number_of_agents(int pod_no)</code>
static int	<code>number_of_agents(Session session, int pod)</code>
int	<code>number_of_pods()</code>
static int	<code>number_of_pods(Session session)</code>
static boolean	<code>pod_init_done(Session session)</code>
static void	<code>pod_reset(Session session)</code>
static void	<code>set_attr(Session session, java.lang.String attr, int index, java.lang.String value, int persist)</code>
static void	<code>setInUseStatus(Session session, int podNum, int value)</code>
static void	<code>setMessage(Session session, int podNum, java.lang.String type, java.lang.String value)</code>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Pod

```
public Pod(int pod_no,  
           Session session)
```

Creates an pod object for a given session

Parameters:

pod_no - (IN) pod number

session - (IN) session object

Method Detail

number_of_pods

```
public static int number_of_pods(Session session)
```

number_of_pods

```
public int number_of_pods()
```

_get_nth_pod

```
public static Pod _get_nth_pod(Session session,  
                               int pod_no)
```

number_of_agents

```
public static int number_of_agents(Session session,  
                                   int pod)
```

number_of_agents

```
public int number_of_agents(int pod_no)
```

number_of_agents

```
public int number_of_agents()
```

get_nth_agent

```
public static int get_nth_agent(Session session,  
                                int pod_no,  
                                int agent_no)
```

get_nth_agent

```
public int get_nth_agent(int pod_no,  
                          int agent_no)
```

get_nth_agent

```
public int get_nth_agent(int agent_no)
```

get_pod_number

```
public static int get_pod_number(Session session,  
                                  int agent_no)
```

get_pod

```
public static Pod get_pod(Session session,  
                           int agent_no)
```

pod_init_done

```
public static boolean pod_init_done(Session session)
```

pod_reset

```
public static void pod_reset(Session session)
```

_get_pod_id

```
public static java.lang.String _get_pod_id(Session session,  
                                         int pod_no)
```

_get_configured_pods

```
public static java.util.LinkedList _get_configured_pods(Session session)
```

find_by_id

```
public static int find_by_id(Session session,  
                             java.lang.String user_key)
```

get_pod_assigned

```
public static int get_pod_assigned(Session session,  
                                   java.lang.String source_ip)
```

getVlabType

```
public static java.lang.String getVlabType(Session session,  
                                           int podNum)
```

getTerminalServer

```
public static java.lang.String getTerminalServer(Session session,  
                                                 int podNum)
```

getProxyServer

```
public static java.lang.String getProxyServer(Session session,  
                                              int podNum)
```

get_pod_agents

```
public java.util.LinkedList get_pod_agents(Session session,  
                                           int pod)
```

get_agent_position

```
public int get_agent_position(int agentNum)
```


getPoolName

```
public static java.lang.String getPoolName(Session session,  
                                           int podNum)
```

getInUseStatus

```
public static int getInUseStatus(Session session,  
                                  int podNum)
```

setInUseStatus

```
public static void setInUseStatus(Session session,  
                                  int podNum,  
                                  int value)
```

get_stop_time

```
public static java.lang.Long get_stop_time(Session session,  
                                             int podNum)
```

getPodMessage

```
public static java.lang.String getPodMessage(Session session,  
                                              int podNum)
```

getAccessStatus

```
public static java.lang.String getAccessStatus(Session session,  
                                                int podNum)
```

getMessage

```
public static java.util.LinkedList getMessage(Session session,  
                                              int podNum,  
                                              java.lang.String type)
```

setMessage

```
public static void setMessage(Session session,
```

```
int podNum,  
java.lang.String type,  
java.lang.String value)
```

isExpireTrackOn

```
public static boolean isExpireTrackOn(Session session)
```

getExpiredKeys

```
public static java.util.LinkedList getExpiredKeys(Session session)
```

getExpireFlushFrequency

```
public static int getExpireFlushFrequency(Session session)
```

getDemoThrottleException

```
public static java.util.LinkedList getDemoThrottleException(Session session)
```

getDemoThrottle

```
public static int getDemoThrottle(Session session)
```

getDemoThrottleFlushPeriod

```
public static int getDemoThrottleFlushPeriod(Session session)
```

isDemoThrottleOn

```
public static boolean isDemoThrottleOn(Session session)
```

getDemoKeyPrefix

```
public static java.lang.String getDemoKeyPrefix(Session session)
```

set_attr

```
public static void set_attr(Session session,
```

```
java.lang.String attr,  
int index,  
java.lang.String value,  
int persist)
```

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class SearchPath

java.lang.Object

Mimic.SearchPath

```
public class SearchPath
extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
static int	BOTH Both shared and private areas
static int	PRIVATE Private area only
static int	SHARED Shared area only

Constructor Summary

Constructors

Constructor and Description
SearchPath ()

Method Summary

All Methods

Static Methods

Concrete Methods

Modifier and Type	Method and Description
static java.lang.String	find (java.lang.String path, int where) Finds a file relative to the directories in the searchpath

```
static java.lang.String privpath(java.lang.String path)
```

Get the private path for a given path

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SHARED

```
public static final int SHARED
```

Shared area only

See Also:

[Constant Field Values](#)

PRIVATE

```
public static final int PRIVATE
```

Private area only

See Also:

[Constant Field Values](#)

BOTH

```
public static final int BOTH
```

Both shared and private areas

See Also:

[Constant Field Values](#)

Constructor Detail

SearchPath

```
public SearchPath()
```

Method Detail

privpath

```
public static java.lang.String privpath(java.lang.String path)
```

Get the private path for a given path

Parameters:

path - (IN) to be appended to the private path

Returns:

String - private path

find

```
public static java.lang.String find(java.lang.String path,  
                                   int where)
```

Finds a file relative to the directories in the searchpath

Parameters:

path - (IN) path to be searched

where - (IN) SHARED | PRIVATE | BOTH

Returns:

String - path found

[PACKAGE](#) [CLASS](#) [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class Session

java.lang.Object
Mimic.Session

Direct Known Subclasses:

[debugSession](#)

```
public class Session
extends java.lang.Object
```

Constructor Summary

Constructors

Modifier	Constructor and Description
protected	Session (java.lang.String host, int port) Creates a session for given host and port

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
void	add_access (java.lang.String user, java.lang.String agents, java.lang.String mask) Add user's access privilege for agents (See mimic access add)
void	add_timer_script (java.lang.String script, int interval, java.lang.String args) Add (and start) timerscript with given interval and arguments (See mimic timer script add)
java.util.LinkedList	agent_mget_changed (java.util.LinkedList agents) Get changed flags for specified agents (See mimic agent mget)
java.util.LinkedList	agent_mget_config_changed (java.util.LinkedList agents) Get config_changed flags for specified agents (See mimic agent mget)
java.util.LinkedList	agent_mget_delay (java.util.LinkedList agents)

Get delays for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_drops**(java.util.LinkedList agents)

Get drops for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_host**(java.util.LinkedList agents)

Get primary addresses for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_inform_retries**(java.util.LinkedList agents)

Get inform retries for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_inform_timeout**(java.util.LinkedList agents)

Get inform timeout for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_interface**(java.util.LinkedList agents)

Get interfaces for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_mask**(java.util.LinkedList agents)

Get masks for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_mibs**(java.util.LinkedList agents)

Get MIB triplets for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_oiddir**(java.util.LinkedList agents)

Get OidDir filenames for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_owner**(java.util.LinkedList agents)

Get owners for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_pdu_size**(java.util.LinkedList agents)

Get pdu sizes for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_port**(java.util.LinkedList agents)

Get port numbers for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_privdir**(java.util.LinkedList agents)

Get privdirs for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_protocols**(java.util.LinkedList agents)

Get protocols for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_read_community**(java.util.LinkedList agents)

Get read communities for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_scen**(java.util.LinkedList agents)

Get scen for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_sim**(java.util.LinkedList agents)

Get sim name for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_starttime**(java.util.LinkedList agents)

Get starttimes for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_state_changed**(java.util.LinkedList agents)

Get state_changed flags for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_state**(java.util.LinkedList agents)

Get states for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_statistics**(java.util.LinkedList agents)

Get statistics for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_trace**(java.util.LinkedList agents)

Get trace flags for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_validate**(java.util.LinkedList agents)

Get validate flags for specified agents (See **mimic agent mget**)

java.util.LinkedList **agent_mget_write_community**(java.util.LinkedList agents)

Get write communities for specified agents (See **mimic agent mget**)

void **agent_mset_delay**(java.util.LinkedList agents,
java.util.LinkedList delays)

Set delay for specified agents

void **agent_mset_drops**(java.util.LinkedList agents,
java.util.LinkedList drops)

Set drops for specified agents

void **agent_mset_host**(java.util.LinkedList agents,
java.util.LinkedList hosts)

Set ip addresses for specified agents

void **agent_mset_inform_retries**(java.util.LinkedList agents,
java.util.LinkedList retries)

Set inform retries for specified agents

void **agent_mset_inform_timeout**(java.util.LinkedList agents,
java.util.LinkedList timeout)

Set inform timeout for specified agents

void **agent_mset_interface**(java.util.LinkedList agents,
java.util.LinkedList interfaces)

Set interfaces for specified agents

void **agent_mset_mask**(java.util.LinkedList agents,
java.util.LinkedList masks)

Set subnet masks for specified agents

void **agent_mset_mibs**(java.util.LinkedList agents,
java.util.LinkedList mibs)

Set MIBs for specified agents

void **agent_mset_oiddir**(java.util.LinkedList agents,
java.util.LinkedList oiddirs)

Set OidDir filename for specified agents

void	<code>agent_mset_pdu_size</code> (java.util.LinkedList agents, java.util.LinkedList pdu_sizes) Set PDU size for specified agents
void	<code>agent_mset_port</code> (java.util.LinkedList agents, java.util.LinkedList ports) Set port numbers for specified agents
void	<code>agent_mset_protocol</code> (java.util.LinkedList agents, java.util.LinkedList protocols) Set protocols for specified agents
void	<code>agent_mset_read</code> (java.util.LinkedList agents, java.util.LinkedList readcomms) Set read communities for specified agents
void	<code>agent_mset_starttime</code> (java.util.LinkedList agents, java.util.LinkedList starttimes) Set start time for specified agents
void	<code>agent_mset_trace</code> (java.util.LinkedList agents, java.util.LinkedList flags) Set trace flags for specified agents
void	<code>agent_mset_validate</code> (java.util.LinkedList agents, java.util.LinkedList flags) Set validate for specified agents
void	<code>agent_mset_write</code> (java.util.LinkedList agents, java.util.LinkedList writecomms) Set write communities for specified agents
void	<code>cfg_load</code> (java.lang.String file) Loads the specified configuration file
void	<code>cfg_load</code> (java.lang.String file, java.lang.String agents) Loads the specified configuration file
void	<code>cfg_load</code> (java.lang.String file, java.lang.String agents, int start) Loads the specified configuration file
void	<code>cfg_load</code> (java.lang.String file, java.lang.String agents, int start, int merge) /** Loads the specified configuration file
void	<code>cfg_new</code> () Load a new blank configuration
void	<code>cfg_new</code> (java.lang.String agents) clear configuration for specified agents
void	<code>cfg_save</code> ()

	Saves the current configuration
void	<code>cfg_save</code> (java.lang.String agents) Saves the current configuration
void	<code>cfg_saveas</code> (java.lang.String file) Saves the current configuration as specified filename
void	<code>cfg_saveas</code> (java.lang.String file, java.lang.String agents) Saves the current configuration as specified filename
void	<code>connect</code> () Connects to the MIMIC daemon at the configured host and port
void	<code>del_access</code> (java.lang.String user) Delete user's access privilege (See mimic access del)
void	<code>del_timer_script</code> (java.lang.String script) Delete a running script with specified name (See mimic timer script delete)
void	<code>diag</code> (int diagid) Execute specified diagnostic code (used only for debugging)
void	<code>diag</code> (java.lang.String diagstr) Execute specified diagnostic code (used only for debugging)
void	<code>disconnect</code> () Disconnects from the MIMIC daemon
java.lang.String	<code>dispatch_msg</code> (int req_id, java.lang.String req_msg) Dispatches a request to the MIMIC daemon, retrieves and returns the reply from the same
void	<code>dump</code> () Set start time for specified agents
java.lang.String	<code>get_access_acldb</code> () Get data base for access control (See mimic access get acldb)
java.lang.String	<code>get_access_adminidir</code> () Get access administrative directory (See mimic access get adminidir)
java.lang.String	<code>get_access_adminuser</code> () Get mimic access administrator (See mimic access get adminuser)
int	<code>get_access_enabled</code> () Get access control enabled value (See mimic access get enabled)
java.util.LinkedList	<code>get_active_data_list_Long</code> () Get list of running agents with their statistics (See mimic get)
java.util.LinkedList	<code>get_active_data_list</code> () Get list of running agents with their statistics (See mimic get)

java.util.LinkedList	<code>get_active_list()</code> Get the list of running agents (See mimic get)
Agent	<code>get_agent(int agentno)</code> Gets the agent object
int	<code>get_cfg_file_changed()</code> Get flag indicating if config file has changed (See mimic get)
java.lang.String	<code>get_cfgfile()</code> Get the currently loaded configuration file (See mimic get)
java.util.LinkedList	<code>get_changed_config_list()</code> Get list of agents whose configuration has changed (See mimic get)
java.util.LinkedList	<code>get_changed_state_list()</code> Get a list of agents whose state has changed (See mimic get)
int	<code>get_clients()</code> Get the number of connected clients (See mimic get)
java.util.LinkedList	<code>get_configured_list()</code> Get the list of configured agents (See mimic get)
java.lang.String	<code>get_expiration(int days)</code> Get expiration of license within specified days (See mimic get)
java.lang.String	<code>get_host()</code> Gets the host address of the MIMIC daemon
java.util.LinkedList	<code>get_interfaces()</code> Get the interfaces (See mimic get)
int	<code>get_last()</code> Get the last configured agent (See mimic get)
java.lang.String	<code>get_licensing()</code> Get licensing info where MIMIC is running.
java.lang.String	<code>get_log()</code> Get the current logfile (See mimic get)
int	<code>get_max()</code> Get maximum agents for session (See mimic get)
java.lang.String	<code>get_netaddr()</code> Get network address of host where MIMIC is running.
java.lang.String	<code>get_netdev()</code> Get default network device of host where MIMIC is running.
int	<code>get_port()</code> Gets the management port number of the MIMIC daemon

java.lang.String	<code>get_privdir()</code> Gets the private directory for the session The Simulator private directory can be retrieved with <code>get_sim_privdir()</code> .
int	<code>get_product()</code> Get licensed product number (See mimic get)
java.util.LinkedList	<code>get_protocols()</code> Get all the protocols supported by the simulator (See mimic get)
java.lang.String	<code>get_sim_privdir()</code> Get the simulator's private directory (See mimic get)
java.lang.String	<code>get_user()</code> Gets the username for the session
java.lang.String	<code>get_version()</code> Get the current MIMIC daemon version (See mimic get)
java.util.LinkedList	<code>list_access()</code> List accesses assigned to users (See mimic access list)
java.util.LinkedList	<code>list_timer_scripts()</code> List the currently running timer scripts (See mimic timer script list)
void	<code>load_access(java.lang.String file)</code> Load access control file (See mimic access load)
java.lang.String	<code>protocol_msg(java.lang.String protocol, java.lang.String msg)</code> Pass session's message string to the protocol
void	<code>save_access(java.lang.String file)</code> Save access control file (See mimic access save)
void	<code>set_access_acldb(java.lang.String acldb)</code> Set access control's data base (See mimic access set acldb)
void	<code>set_access_admindir(java.lang.String dir)</code> Set access administrative directory.
void	<code>set_access_adminuser(java.lang.String user)</code> Set access administrator.
void	<code>set_access_enabled(int arg)</code> Set access control's enabled value (See mimic access set enabled)
void	<code>set_log(java.lang.String logfile)</code> Set a new logfile (See mimic set)
void	<code>start_all_agents()</code> Starts all configured agents
void	<code>stop_all_agents()</code>

Stops all running agents

void	<code>store_append</code> (java.lang.String var, java.lang.String val, int persist) Append (and start) storage with a given variable and value (See mimic store append)
int	<code>store_exists</code> (java.lang.String var) Examine existence (See mimic store exists)
java.lang.String	<code>store_get</code> (java.lang.String var) Get variable value (See mimic store get)
java.util.LinkedList	<code>store_list</code> () List variables (See mimic store list)
void	<code>store_lreplace</code> (java.lang.String var, int index, java.lang.String val) Treat variable as a list and modify entry for given index with given value (See mimic store lreplace)
java.util.LinkedList	<code>store_mget</code> (java.util.LinkedList vars) Get multiple variable values (See mimic store mget)
void	<code>store_mlreplace</code> (java.util.LinkedList varlist, java.util.LinkedList indexlist, java.util.LinkedList vallist) Treat each variable as a list and modify entry for given index with given value (See mimic store mlreplace)
int	<code>store_persists</code> (java.lang.String var) Examine persistence (See mimic store persists)
void	<code>store_save</code> () Set store variables persistent saving to disk (See mimic set)
void	<code>store_set</code> (java.lang.String var, java.lang.String val, int persist) Set (and start) storage with a given variable and value (See mimic store set)
void	<code>store_unset</code> (java.lang.String var) Unset a variable-value pair (See mimic store unset)
void	<code>terminate</code> () Terminates the connected MIMIC daemon
void	<code>trap_config_add</code> (java.lang.String destination, int port) add a trap destination (See mimic trap)
void	<code>trap_config_delete</code> (java.lang.String destination, int port) delete a trap destination (See mimic trap)
java.util.LinkedList	<code>trap_config_list</code> () List variables of trap destination (See mimic trap)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Session

```
protected Session(java.lang.String host,
                  int port)
    throws ErrorInfo
```

Creates a session for given host and port

Parameters:

host - (IN) host ip address

port - (IN) host management port number

Throws:

[ErrorInfo](#)

Method Detail

get_host

```
public java.lang.String get_host()
```

Gets the host address of the MIMIC daemon

Returns:

String host ip address

get_port

```
public int get_port()
```

Gets the management port number of the MIMIC daemon

Returns:

String host management port number

get_user

```
public java.lang.String get_user()
```

Gets the username for the session

Returns:

String user name for the session

```
get_privdir
```

```
public java.lang.String get_privdir()
```

Gets the private directory for the session The Simulator private directory can be retrieved with `get_sim_privdir()`.

Returns:

String private directory for the session

```
dispatch_msg
```

```
public java.lang.String dispatch_msg(int req_id,  
                                     java.lang.String req_msg)  
    throws ErrorInfo
```

Dispatches a request to the MIMIC daemon, retrieves and returns the reply from the same

Parameters:

`req_id` - (IN) identifier for the request

`req_msg` - (IN) message buffer

Returns:

String reply for the command

Throws:

[ErrorInfo](#)

```
connect
```

```
public void connect()  
    throws ErrorInfo
```

Connects to the MIMIC daemon at the configured host and port

Throws:

[ErrorInfo](#)

```
disconnect
```



```
public void disconnect()  
    throws ErrorInfo
```

Disconnects from the MIMIC daemon

Throws:
`ErrorInfo`

cfg_new

```
public void cfg_new()  
    throws ErrorInfo
```

Load a new blank configuration

Throws:
`ErrorInfo`

cfg_new

```
public void cfg_new(java.lang.String agents)  
    throws ErrorInfo
```

clear configuration for specified agents

Parameters:
agents - (IN) range of agents (Ex. "2,3,9-27")

Throws:
`ErrorInfo`

cfg_load

```
public void cfg_load(java.lang.String file)  
    throws ErrorInfo
```

Loads the specified configuration file

Parameters:
file - (IN) filename to load

Throws:
`ErrorInfo`

cfg_load

```
public void cfg_load(java.lang.String file,  
                    java.lang.String agents)
```

throws `ErrorInfo`

Loads the specified configuration file

Parameters:

file - (IN) filename to load

agents - (IN) range of agents (Ex. "2,3,9-27")

Throws:

`ErrorInfo`

cfg_load

```
public void cfg_load(java.lang.String file,
                    java.lang.String agents,
                    int start)
    throws ErrorInfo
```

Loads the specified configuration file

Parameters:

file - (IN) filename to load

agents - (IN) range of agents (Ex. "2,3,9-27")

start - (IN) starting agent number

Throws:

`ErrorInfo`

cfg_load

```
public void cfg_load(java.lang.String file,
                    java.lang.String agents,
                    int start,
                    int merge)
    throws ErrorInfo
```

/** Loads the specified configuration file

Parameters:

file - (IN) filename to load

agents - (IN) range of agents (Ex. "2,3,9-27")

start - (IN) starting agent number

merge - (IN) merge indicator

Throws:

`ErrorInfo`

cfg_save

```
public void cfg_save()  
    throws ErrorInfo
```

Saves the current configuration

Throws:

[ErrorInfo](#)

cfg_save

```
public void cfg_save(java.lang.String agents)  
    throws ErrorInfo
```

Saves the current configuration

Parameters:

agents - (IN) range of agents (Ex. "2,3,9-27")

Throws:

[ErrorInfo](#)

cfg_saveas

```
public void cfg_saveas(java.lang.String file)  
    throws ErrorInfo
```

Saves the current configuration as specified filename

Parameters:

file - (IN) filename to save as

Throws:

[ErrorInfo](#)

cfg_saveas

```
public void cfg_saveas(java.lang.String file,  
                       java.lang.String agents)  
    throws ErrorInfo
```

Saves the current configuration as specified filename

Parameters:

file - (IN) filename to save as

agents - (IN) range of agents (Ex. "2,3,9-27")

Throws:

ErrorInfo

terminate

```
public void terminate()  
    throws ErrorInfo
```

Terminates the connected MIMIC daemon

Throws:

ErrorInfo

start_all_agents

```
public void start_all_agents()  
    throws ErrorInfo
```

Starts all configured agents

Throws:

ErrorInfo

stop_all_agents

```
public void stop_all_agents()  
    throws ErrorInfo
```

Stops all running agents

Throws:

ErrorInfo

get_agent

```
public Agent get_agent(int agentno)  
    throws ErrorInfo
```

Gets the agent object

Returns:

Agent agent object instance

Throws:

ErrorInfo

get_max

```
public int get_max()  
    throws ErrorInfo
```

Get maximum agents for session (See [mimic get](#))

Returns:

int maximum number of agents that can be configured

Throws:

[ErrorInfo](#)

get_last

```
public int get_last()  
    throws ErrorInfo
```

Get the last configured agent (See [mimic get](#))

Returns:

int the last configured agent's id

Throws:

[ErrorInfo](#)

get_version

```
public java.lang.String get_version()  
    throws ErrorInfo
```

Get the current MIMIC daemon version (See [mimic get](#))

Returns:

String MIMIC daemon version

Throws:

[ErrorInfo](#)

get_clients

```
public int get_clients()  
    throws ErrorInfo
```

Get the number of connected clients (See [mimic get](#))

Returns:

int number of connected clients

Throws:

[ErrorInfo](#)

get_cfgfile

```
public java.lang.String get_cfgfile()  
                        throws ErrorInfo
```

Get the currently loaded configuration file (See [mimic get](#))

Returns:

String config filename

Throws:

[ErrorInfo](#)

get_configured_list

```
public java.util.LinkedList get_configured_list()  
                        throws ErrorInfo
```

Get the list of configured agents (See [mimic get](#))

Returns:

LinkedList list of agent-ids

Throws:

[ErrorInfo](#)

get_active_list

```
public java.util.LinkedList get_active_list()  
                        throws ErrorInfo
```

Get the list of running agents (See [mimic get](#))

Returns:

LinkedList list of agent-ids

Throws:

[ErrorInfo](#)

get_active_data_list

```
public java.util.LinkedList get_active_data_list()  
                        throws ErrorInfo
```

Get list of running agents with their statistics (See [mimic get](#))

Returns:

LinkedList list of list of (agent-id,snmp-stats)

Throws:
`ErrorInfo`

`get_active_data_list_Long`

```
public java.util.LinkedList get_active_data_list_Long()  
                            throws ErrorInfo
```

Get list of running agents with their statistics (See [mimic get](#))

Returns:
`LinkedList` list of list of (agent-id,snmp-stats)

Throws:
`ErrorInfo`

`get_changed_config_list`

```
public java.util.LinkedList get_changed_config_list()  
                            throws ErrorInfo
```

Get list of agents whose configuration has changed (See [mimic get](#))

Returns:
`LinkedList` list of agent-ids

Throws:
`ErrorInfo`

`get_changed_state_list`

```
public java.util.LinkedList get_changed_state_list()  
                            throws ErrorInfo
```

Get a list of agents whose state has changed (See [mimic get](#))

Returns:
`LinkedList` list of list of (agent-id,state)

Throws:
`ErrorInfo`

`get_log`

```
public java.lang.String get_log()  
                        throws ErrorInfo
```

Get the current logfile (See [mimic get](#))

Returns:

String log filename

Throws:

`ErrorInfo`

get_interfaces

```
public java.util.LinkedList get_interfaces()  
                               throws ErrorInfo
```

Get the interfaces (See [mimic get](#))

Returns:

String interface name

Throws:

`ErrorInfo`

get_cfg_file_changed

```
public int get_cfg_file_changed()  
           throws ErrorInfo
```

Get flag indicating if config file has changed (See [mimic get](#))

Returns:

int flag (0|1)

Throws:

`ErrorInfo`

get_sim_privdir

```
public java.lang.String get_sim_privdir()  
                          throws ErrorInfo
```

Get the simulator's private directory (See [mimic get](#))

Returns:

String simulator's private directory path

Throws:

`ErrorInfo`

get_protocols

```
public java.util.LinkedList get_protocols()
```


throws `ErrorInfo`

Get all the protocols supported by the simulator (See `mimic get`)

Returns:

`LinkedList` list of protocol name strings

Throws:

`ErrorInfo`

`get_product`

```
public int get_product()  
           throws ErrorInfo
```

Get licensed product number (See `mimic get`)

Returns:

`int` product

Throws:

`ErrorInfo`

`get_netaddr`

```
public java.lang.String get_netaddr()  
                        throws ErrorInfo
```

Get network address of host where MIMIC is running. (See `mimic get`)

Returns:

`String` netaddr

Throws:

`ErrorInfo`

`get_netdev`

```
public java.lang.String get_netdev()  
                        throws ErrorInfo
```

Get default network device of host where MIMIC is running. (See `mimic get`)

Returns:

`string` netdev

Throws:

`ErrorInfo`

get_expiration

```
public java.lang.String get_expiration(int days)
    throws ErrorInfo
```

Get expiration of license within specified days (See `mimic get`)

Returns:

String expiration

Throws:

`ErrorInfo`

get_licensing

```
public java.lang.String get_licensing()
    throws ErrorInfo
```

Get licensing info where MIMIC is running. (See `mimic get`)

Returns:

string licensing

Throws:

`ErrorInfo`

set_log

```
public void set_log(java.lang.String logfile)
    throws ErrorInfo
```

Set a new logfile (See `mimic set`)

Parameters:

logfile - (IN) filename to use for logging

Throws:

`ErrorInfo`

store_save

```
public void store_save()
    throws ErrorInfo
```

Set store variables persistent saving to disk (See `mimic set`)

Throws:

`ErrorInfo`

add_timer_script

```
public void add_timer_script(java.lang.String script,
                             int interval,
                             java.lang.String args)
    throws ErrorInfo
```

Add (and start) timerscript with given interval and arguments (See [mimic timer script add](#))

Parameters:

script - (IN) script to add

interval - (IN) interval in milli-seconds

args - (IN) arguments passed to the script

Throws:

[ErrorInfo](#)

del_timer_script

```
public void del_timer_script(java.lang.String script)
    throws ErrorInfo
```

Delete a running script with specified name (See [mimic timer script delete](#))

Parameters:

script - (IN) script to delete

Throws:

[ErrorInfo](#)

list_timer_scripts

```
public java.util.LinkedList list_timer_scripts()
    throws ErrorInfo
```

List the currently running timer scripts (See [mimic timer script list](#))

Returns:

LinkedList list of scripts configured (script,interval,args)

Throws:

[ErrorInfo](#)

add_access

```
public void add_access(java.lang.String user,
                       java.lang.String agents,
                       java.lang.String mask)
```

throws `ErrorInfo`

Add user's access privilege for agents (See [mimic access add](#))

Parameters:

user - (IN) user to add

agents - (IN) agents for access

mask - (IN) mask applies to

Throws:

`ErrorInfo`

del_access

```
public void del_access(java.lang.String user)
                    throws ErrorInfo
```

Delete user's access privilege (See [mimic access del](#))

Parameters:

user - (IN) user to delete

Throws:

`ErrorInfo`

list_access

```
public java.util.LinkedList list_access()
                    throws ErrorInfo
```

List accesses assigned to users (See [mimic access list](#))

Returns:

LinkedList list of accesses (user agents mask)

Throws:

`ErrorInfo`

get_access_adminuser

```
public java.lang.String get_access_adminuser()
                    throws ErrorInfo
```

Get mimic access administrator (See [mimic access get adminuser](#))

Returns:

String the administrator

Throws:

`ErrorInfo`

`get_access_admindir`

```
public java.lang.String get_access_admindir()  
                        throws ErrorInfo
```

Get access administrative directory (See `mimic access get admin`)

Returns:

String the administrative directory

Throws:

`ErrorInfo`

`get_access_acldb`

```
public java.lang.String get_access_acldb()  
                        throws ErrorInfo
```

Get data base for access control (See `mimic access get acldb`)

Returns:

String acldb

Throws:

`ErrorInfo`

`get_access_enabled`

```
public int get_access_enabled()  
          throws ErrorInfo
```

Get access control enabled value (See `mimic access get enabled`)

Returns:

int enabled value

Throws:

`ErrorInfo`

`set_access_adminuser`

```
public void set_access_adminuser(java.lang.String user)  
                                throws ErrorInfo
```

Set access administrator.

Parameters:

user - (IN) user to be administrator

Throws:

`ErrorInfo`

set_access_admindir

```
public void set_access_admindir(java.lang.String dir)
        throws ErrorInfo
```

Set access administrative directory.

Parameters:

dir - (IN) directory to be admin directory

Throws:

`ErrorInfo`

set_access_acldb

```
public void set_access_acldb(java.lang.String acldb)
        throws ErrorInfo
```

Set access control's data base (See [mimic access set acldb](#))

Parameters:

acldb - (IN) data base for access control

Throws:

`ErrorInfo`

set_access_enabled

```
public void set_access_enabled(int arg)
        throws ErrorInfo
```

Set access control's enabled value (See [mimic access set enabled](#))

Parameters:

arg - (IN) value to pass to enabled

Throws:

`ErrorInfo`

save_access

```
public void save_access(java.lang.String file)
        throws ErrorInfo
```

Save access control file (See mimic access save)

Parameters:

file - (IN) filename to save as acldb.

Throws:

`ErrorInfo`

load_access

```
public void load_access(java.lang.String file)
    throws ErrorInfo
```

Load access control file (See mimic access load)

Parameters:

file - (IN) acldb to load in

Throws:

`ErrorInfo`

protocol_msg

```
public java.lang.String protocol_msg(java.lang.String protocol,
    java.lang.String msg)
    throws ErrorInfo
```

Pass session's message string to the protocol

Parameters:

protocol - (IN) protocol to send message to

msg - (IN) message string to be sent

Returns:

String reply buffer

Throws:

`ErrorInfo`

diag

```
public void diag(java.lang.String diagstr)
    throws ErrorInfo
```

Execute specified diagnostic code (used only for debugging)

Parameters:

diagstr - (IN) diagnostic function to execute

Throws:

ErrorInfo

diag

```
public void diag(int diagid)
    throws ErrorInfo
```

Execute specified diagnostic code (used only for debugging)

Parameters:

diagid - (IN) diagnostic function to execute

Throws:

ErrorInfo

agent_mget_interface

```
public java.util.LinkedList agent_mget_interface(java.util.LinkedList agents)
    throws ErrorInfo
```

Get interfaces for specified agents (See mimic agent mget)

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of interface strings

Throws:

ErrorInfo

agent_mget_host

```
public java.util.LinkedList agent_mget_host(java.util.LinkedList agents)
    throws ErrorInfo
```

Get primary addresses for specified agents (See mimic agent mget)

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of address strings

Throws:

ErrorInfo

agent_mget_mask


```
public java.util.LinkedList agent_mget_mask(java.util.LinkedList agents)
    throws ErrorInfo
```

Get masks for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of mask strings

Throws:

[ErrorInfo](#)

agent_mget_port

```
public java.util.LinkedList agent_mget_port(java.util.LinkedList agents)
    throws ErrorInfo
```

Get port numbers for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of port numbers

Throws:

[ErrorInfo](#)

agent_mget_read_community

```
public java.util.LinkedList agent_mget_read_community(java.util.LinkedList agents)
    throws ErrorInfo
```

Get read communities for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of community strings

Throws:

[ErrorInfo](#)

agent_mget_write_community

```
public java.util.LinkedList agent_mget_write_community(java.util.LinkedList agents)
    throws ErrorInfo
```

Get write communities for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of community strings

Throws:

[ErrorInfo](#)

agent_mget_mibs

```
public java.util.LinkedList agent_mget_mibs(java.util.LinkedList agents)
                                     throws ErrorInfo
```

Get MIB triplets for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of list of MIB triplet strings

Throws:

[ErrorInfo](#)

agent_mget_sim

```
public java.util.LinkedList agent_mget_sim(java.util.LinkedList agents)
                                     throws ErrorInfo
```

Get sim name for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of sim name

Throws:

[ErrorInfo](#)

agent_mget_scen

```
public java.util.LinkedList agent_mget_scen(java.util.LinkedList agents)
                                     throws ErrorInfo
```

Get scen for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of scen

Throws:

[ErrorInfo](#)

agent_mget_pdu_size

```
public java.util.LinkedList agent_mget_pdu_size(java.util.LinkedList agents)
    throws ErrorInfo
```

Get pdu sizes for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of PDU size numbers

Throws:

[ErrorInfo](#)

agent_mget_protocols

```
public java.util.LinkedList agent_mget_protocols(java.util.LinkedList agents)
    throws ErrorInfo
```

Get protocols for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of list of protocol strings

Throws:

[ErrorInfo](#)

agent_mget_delay

```
public java.util.LinkedList agent_mget_delay(java.util.LinkedList agents)
    throws ErrorInfo
```

Get delays for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of delay numbers

Throws:

[ErrorInfo](#)

agent_mget_starttime

```
public java.util.LinkedList agent_mget_starttime(java.util.LinkedList agents)
    throws ErrorInfo
```

Get starttimes for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of start time numbers

Throws:

[ErrorInfo](#)

agent_mget_state

```
public java.util.LinkedList agent_mget_state(java.util.LinkedList agents)
    throws ErrorInfo
```

Get states for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of state numbers

Throws:

[ErrorInfo](#)

agent_mget_statistics

```
public java.util.LinkedList agent_mget_statistics(java.util.LinkedList agents)
    throws ErrorInfo
```

Get statistics for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of SNMP stats numbers (total discarded error GET GETNEXT SET BULK TRAP GETVAR GETNEXTVAR SETVAR BULKVAR)

Throws:
`ErrorInfo`

agent_mget_drops

```
public java.util.LinkedList agent_mget_drops(java.util.LinkedList agents)
    throws ErrorInfo
```

Get drops for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of drop numbers

Throws:

`ErrorInfo`

agent_mget_changed

```
public java.util.LinkedList agent_mget_changed(java.util.LinkedList agents)
    throws ErrorInfo
```

Get changed flags for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of changed flag numbers

Throws:

`ErrorInfo`

agent_mget_config_changed

```
public java.util.LinkedList agent_mget_config_changed(java.util.LinkedList agents)
    throws ErrorInfo
```

Get config_changed flags for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of config changed flag numbers

Throws:

`ErrorInfo`

agent_mget_state_changed

```
public java.util.LinkedList agent_mget_state_changed(java.util.LinkedList agents)
    throws ErrorInfo
```

Get state_changed flags for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of state changed flag numbers

Throws:

[ErrorInfo](#)

agent_mget_trace

```
public java.util.LinkedList agent_mget_trace(java.util.LinkedList agents)
    throws ErrorInfo
```

Get trace flags for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of trace flag numbers

Throws:

[ErrorInfo](#)

agent_mget_validate

```
public java.util.LinkedList agent_mget_validate(java.util.LinkedList agents)
    throws ErrorInfo
```

Get validate flags for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of validate flag numbers

Throws:

[ErrorInfo](#)

agent_mget_owner

```
public java.util.LinkedList agent_mget_owner(java.util.LinkedList agents)
    throws ErrorInfo
```

Get owners for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of owner name strings

Throws:

[ErrorInfo](#)

agent_mget_privdir

```
public java.util.LinkedList agent_mget_privdir(java.util.LinkedList agents)
    throws ErrorInfo
```

Get privdirs for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of private directory strings

Throws:

[ErrorInfo](#)

agent_mget_oiddir

```
public java.util.LinkedList agent_mget_oiddir(java.util.LinkedList agents)
    throws ErrorInfo
```

Get OidDir filenames for specified agents (See [mimic agent mget](#))

Parameters:

agents - (IN) list of agents

Returns:

LinkedList list of OidDir filename strings

Throws:

[ErrorInfo](#)

agent_mget_inform_timeout

```
public java.util.LinkedList agent_mget_inform_timeout(java.util.LinkedList agents)
```

throws `ErrorInfo`

Get inform timeout for specified agents (See `mimic agent mget`)

Parameters:

agents - (IN) list of agents

Returns:

`LinkedList` list of inform timeout numbers

Throws:

`ErrorInfo`

agent_mget_inform_retries

```
public java.util.LinkedList agent_mget_inform_retries(java.util.LinkedList agents)
                                                    throws ErrorInfo
```

Get inform retries for specified agents (See `mimic agent mget`)

Parameters:

agents - (IN) list of agents

Returns:

`LinkedList` list of inform retry numbers

Throws:

`ErrorInfo`

agent_mset_interface

```
public void agent_mset_interface(java.util.LinkedList agents,
                                java.util.LinkedList interfaces)
                                throws ErrorInfo
```

Set interfaces for specified agents

Parameters:

agents - (IN) list of agents

interfaces - (IN) list of interface strings

Throws:

`ErrorInfo`

agent_mset_host

```
public void agent_mset_host(java.util.LinkedList agents,
                             java.util.LinkedList hosts)
                             throws ErrorInfo
```


Set ip addresses for specified agents

Parameters:

agents - (IN) list of agents

hosts - (IN) list of ip-address strings

Throws:

`ErrorInfo`

agent_mset_mask

```
public void agent_mset_mask(java.util.LinkedList agents,  
                             java.util.LinkedList masks)  
    throws ErrorInfo
```

Set subnet masks for specified agents

Parameters:

agents - (IN) list of agents

masks - (IN) list of subnet mask strings

Throws:

`ErrorInfo`

agent_mset_port

```
public void agent_mset_port(java.util.LinkedList agents,  
                             java.util.LinkedList ports)  
    throws ErrorInfo
```

Set port numbers for specified agents

Parameters:

agents - (IN) list of agents

ports - (IN) list of port numbers

Throws:

`ErrorInfo`

agent_mset_read

```
public void agent_mset_read(java.util.LinkedList agents,  
                             java.util.LinkedList readcomms)  
    throws ErrorInfo
```

Set read communities for specified agents

Parameters:

agents - (IN) list of agents

readcomms - (IN) list of community strings

Throws:

`ErrorInfo`

agent_mset_write

```
public void agent_mset_write(java.util.LinkedList agents,
                             java.util.LinkedList writecomms)
    throws ErrorInfo
```

Set write communities for specified agents

Parameters:

agents - (IN) list of agents

writecomms(IN) - list of community strings

Throws:

`ErrorInfo`

agent_mset_protocol

```
public void agent_mset_protocol(java.util.LinkedList agents,
                                java.util.LinkedList protocols)
    throws ErrorInfo
```

Set protocols for specified agents

Parameters:

agents - (IN) list of agents

protocols - (IN) list of list of protocol strings

Throws:

`ErrorInfo`

agent_mset_delay

```
public void agent_mset_delay(java.util.LinkedList agents,
                             java.util.LinkedList delays)
    throws ErrorInfo
```

Set delay for specified agents

Parameters:

agents - (IN) list of agents

delays - (IN) list of delays numbers

Throws:

`ErrorInfo`

agent_mset_starttime

```
public void agent_mset_starttime(java.util.LinkedList agents,  
                                java.util.LinkedList starttimes)  
    throws ErrorInfo
```

Set start time for specified agents

Parameters:

agents - (IN) list of agents

starttimes - (IN) list of start time numbers

Throws:

`ErrorInfo`

agent_mset_mibs

```
public void agent_mset_mibs(java.util.LinkedList agents,  
                            java.util.LinkedList mibs)  
    throws ErrorInfo
```

Set MIBs for specified agents

Parameters:

agents - (IN) list of agents

mibs - (IN) list of list of MIB triplet strings

Throws:

`ErrorInfo`

agent_mset_trace

```
public void agent_mset_trace(java.util.LinkedList agents,  
                             java.util.LinkedList flags)  
    throws ErrorInfo
```

Set trace flags for specified agents

Parameters:

agents - (IN) list of agents

flags - (IN) list of trace flag numbers

Throws:

`ErrorInfo`

agent_mset_pdusize

```
public void agent_mset_pdusize(java.util.LinkedList agents,  
                               java.util.LinkedList pduSizes)  
    throws ErrorInfo
```

Set PDU size for specified agents

Parameters:

agents - (IN) list of agents

pduSizes - (IN) list of PDU size numbers

Throws:

[ErrorInfo](#)

agent_mset_drops

```
public void agent_mset_drops(java.util.LinkedList agents,  
                             java.util.LinkedList drops)  
    throws ErrorInfo
```

Set drops for specified agents

Parameters:

agents - (IN) list of agents

drops - (IN) list of drop numbers

Throws:

[ErrorInfo](#)

agent_mset_validate

```
public void agent_mset_validate(java.util.LinkedList agents,  
                                java.util.LinkedList flags)  
    throws ErrorInfo
```

Set validate for specified agents

Parameters:

agents - (IN) list of agents

flags - (IN) list of flag numbers

Throws:

[ErrorInfo](#)

agent_mset_oiddir

```
public void agent_mset_oiddir(java.util.LinkedList agents,
                             java.util.LinkedList oiddirs)
    throws ErrorInfo
```

Set OidDir filename for specified agents

Parameters:

agents - (IN) list of agents

oiddirs - (IN) list of OidDir filename strings

Throws:

[ErrorInfo](#)

agent_mset_inform_timeout

```
public void agent_mset_inform_timeout(java.util.LinkedList agents,
                                       java.util.LinkedList timeout)
    throws ErrorInfo
```

Set inform timeout for specified agents

Parameters:

agents - (IN) list of agents

timeout - (IN) list of timeout numbers

Throws:

[ErrorInfo](#)

agent_mset_inform_retries

```
public void agent_mset_inform_retries(java.util.LinkedList agents,
                                       java.util.LinkedList retries)
    throws ErrorInfo
```

Set inform retries for specified agents

Parameters:

agents - (IN) list of agents

retries - (IN) list of retry numbers

Throws:

[ErrorInfo](#)

store_set

```
public void store_set(java.lang.String var,
                     java.lang.String val,
```

```
        int persist)
    throws ErrorInfo
```

Set (and start) storage with a given variable and value (See [mimic store set](#))

Parameters:

var - (IN) variable to set

val - (IN) value to set

persist - (IN) persistence

Throws:

[ErrorInfo](#)

store_append

```
public void store_append(java.lang.String var,
                        java.lang.String val,
                        int persist)
    throws ErrorInfo
```

Append (and start) storage with a given variable and value (See [mimic store append](#))

Parameters:

var - (IN) variable to set

val - (IN) value to set

persist - (IN) persistence

Throws:

[ErrorInfo](#)

store_lreplace

```
public void store_lreplace(java.lang.String var,
                          int index,
                          java.lang.String val)
    throws ErrorInfo
```

Treat variable as a list and modify entry for given index with given value (See [mimic store lreplace](#))

Parameters:

var - (IN) variable to set

index - (IN) variable to set

val - (IN) value to set

Throws:

[ErrorInfo](#)

store_mlreplace

```
public void store_mlreplace(java.util.LinkedList varlist,  
                           java.util.LinkedList indexlist,  
                           java.util.LinkedList vallist)  
    throws ErrorInfo
```

Treat each variable as a list and modify entry for given index with given value (See [mimic store mlreplace](#))

Parameters:

varlist - (IN) variable to set

indexlist - (IN) variable to set

vallist - (IN) value to set

Throws:

[ErrorInfo](#)

store_unset

```
public void store_unset(java.lang.String var)  
    throws ErrorInfo
```

Unset a variable-value pair (See [mimic store unset](#))

Parameters:

var - (IN) variable to unset

Throws:

[ErrorInfo](#)

store_list

```
public java.util.LinkedList store_list()  
    throws ErrorInfo
```

List variables (See [mimic store list](#))

Returns:

LinkedList list of variables

Throws:

[ErrorInfo](#)

store_exists

```
public int store_exists(java.lang.String var)  
    throws ErrorInfo
```

Examine existence (See [mimic store exists](#))

Returns:

1 if exists, 0 otherwise

Throws:

`ErrorInfo`

store_persists

```
public int store_persists(java.lang.String var)
                        throws ErrorInfo
```

Examine persistence (See [mimic store persists](#))

Returns:

1 if is persistent, 0 otherwise

Throws:

`ErrorInfo`

store_get

```
public java.lang.String store_get(java.lang.String var)
                        throws ErrorInfo
```

Get variable value (See [mimic store get](#))

Returns:

value

Throws:

`ErrorInfo`

store_mget

```
public java.util.LinkedList store_mget(java.util.LinkedList vars)
                        throws ErrorInfo
```

Get multiple variable values (See [mimic store mget](#))

Returns:

LinkedList list of variables

Throws:

`ErrorInfo`

trap_config_list


```
public java.util.LinkedList trap_config_list()
    throws ErrorInfo
```

List variables of trap destination (See [mimic trap](#))

Returns:

LinkedList list of configured traps

Throws:

`ErrorInfo`

trap_config_add

```
public void trap_config_add(java.lang.String destination,
    int port)
    throws ErrorInfo
```

add a trap destination (See [mimic trap](#))

Throws:

`ErrorInfo`

trap_config_delete

```
public void trap_config_delete(java.lang.String destination,
    int port)
    throws ErrorInfo
```

delete a trap destination (See [mimic trap](#))

Throws:

`ErrorInfo`

dump

```
public void dump()
```

Set start time for specified agents

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class StringParser

java.lang.Object
 Mimic.StringParser

```
public class StringParser
  extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

StringParser()
 Creates a StringParser instace

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method and Description
int	get_num_tokens () Get the number of tokens detected in the parsed string
java.lang.String	get_token (int index) Get the index'th string token
void	tokenize (java.lang.String str) Parse the supplied string and split it into tokens separated by spaces, surrounded in quotes and/or surrounded in braces "{...}"
void	tokenize (java.lang.String str, char sep) Parse the supplied string and split it into tokens separated by sep, surrounded in quotes and/or surrounded in braces "{...}"

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

StringParser

```
public StringParser()
```

Creates a StringParser instance

Method Detail

get_num_tokens

```
public int get_num_tokens()
```

Get the number of tokens detected in the parsed string

Returns:

int - number of tokens

get_token

```
public java.lang.String get_token(int index)
                               throws ErrorInfo
```

Get the index'th string token

Parameters:

index - (IN) index of token to retrieve

Returns:

String - string token

Throws:

[ErrorInfo](#) - on error

tokenize

```
public void tokenize(java.lang.String str,
                    char sep)
                   throws ErrorInfo
```

Parse the supplied string and split it into tokens separated by sep, surrounded in quotes and/or surrounded in braces "{...}"

Parameters:

`str` - (IN) string to be broken up

`sep` - (IN) separator

Throws:

`ErrorInfo` - on error

tokenize

```
public void tokenize(java.lang.String str)
    throws ErrorInfo
```

Parse the supplied string and split it into tokens separated by spaces, surrounded in quotes and/or surrounded in braces "{...}"

Parameters:

`str` - (IN) string to be broken up

Throws:

`ErrorInfo` - on error

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class Utils

java.lang.Object

Mimic.Utils

```
public class Utils
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

Utils()**Method Summary**

All Methods

Static Methods

Concrete Methods

Modifier and Type

Method and Description

static java.lang.String

LinkedListToString(java.util.LinkedList list, java.lang.String separator)

Creates a string from list, separating entries using the separator string provided.

static java.util.LinkedList

SplitToIntegerList(java.lang.String string)

Creates a linked list of Integers from passed String

static java.util.LinkedList

SplitToLongList(java.lang.String string)

Creates a linked list of Longs from passed String

static java.util.LinkedList

SplitToOidSubidList(java.lang.String string)Creates a linked list of positive Longs from passed String Throws an **ErrorInfo** exception if the numbers are negative

static java.util.LinkedList

SplitToStringList(java.lang.String string)

Creates a linked list of tokens from passed string

static java.util.LinkedList

SplitToStringList(java.lang.String string, char sep)

Creates a linked list of tokens from passed string

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

Utils

```
public Utils()
```

Method Detail

LinkedListToString

```
public static java.lang.String LinkedListToString(java.util.LinkedList list,
                                                  java.lang.String separator)
```

Creates a string from list, separating entries using the separator string provided.

Parameters:

`list` - (IN) list of string tokens

`separator` - (IN) separator string to use

Returns:

`String` - string of concatenated tokens

SplitToStringList

```
public static java.util.LinkedList SplitToStringList(java.lang.String string,
                                                     char sep)
    throws ErrorInfo
```

Creates a linked list of tokens from passed string

Parameters:

`string` - (IN) string to be broken up

`sep` - (IN) optional separator, by default space

Returns:

`LinkedList` - list of `String` tokens

Throws:

[ErrorInfo](#) - on error

SplitToStringList

```
public static java.util.LinkedList SplitToStringList(java.lang.String string)
                                     throws ErrorInfo
```

Creates a linked list of tokens from passed string

Parameters:

string - (IN) string to be broken up

Returns:

LinkedList - list of String tokens

Throws:

[ErrorInfo](#) - on error

SplitToIntegerList

```
public static java.util.LinkedList SplitToIntegerList(java.lang.String string)
                                     throws ErrorInfo
```

Creates a linked list of Integers from passed String

Parameters:

string - (IN) string to be broken up

Returns:

LinkedList - list of Integer tokens

Throws:

[ErrorInfo](#) - on error

SplitToLongList

```
public static java.util.LinkedList SplitToLongList(java.lang.String string)
                                     throws ErrorInfo
```

Creates a linked list of Longs from passed String

Parameters:

string - (IN) string to be broken up

Returns:

LinkedList - list of Integer tokens

Throws:

[ErrorInfo](#) - on error

SplitToOidSubidList

```
public static java.util.LinkedList SplitToOidSubidList(java.lang.String string)
    throws ErrorInfo
```

Creates a linked list of positive Longs from passed String Throws an [ErrorInfo](#) exception if the numbers are negative

Parameters:

string - (IN) string to be broken up

Returns:

LinkedList - list of Long tokens

Throws:

[ErrorInfo](#) - on error

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Mimic

Class Valuespace

java.lang.Object

Mimic.Valuespace

```
public class Valuespace
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description

Valuespace(Agent agent)

Creates a valuespace for the agent.

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type

Method and Description

void

add(java.lang.String object, **Oid** instance)Add a new instance (for tables only) (See **mimic value add**)

java.lang.String

eval_value(java.lang.String object, **Oid** instance)Evaluate the value for object.instance (See **mimic value eval**)

java.util.LinkedList

get_info(java.lang.String object)

Get MIB information for object.

java.util.LinkedList

get_instances(java.lang.String object)Get instances of an object (See **mimic value instances**)

java.lang.String

get_mib(java.lang.String object)

Get the MIB containing a given object

java.util.LinkedList

get_minfo(java.util.LinkedList objects)Get MIB information for multiple objects (See **mimic value minfo**)

java.lang.String

get_name(**Oid** oid)

Get a name for an OID (See **mimic value name**)

java.util.LinkedList **get_objects**()
Get the children objects at the current position (See **mimic value list**)

Oid **get_oid**(java.lang.String object)
Get an OID for a name (See **mimic value oid**)

java.lang.String **get_pos**()
Gets the current position in agent's OID tree.

java.lang.String **get_state**(java.lang.String object)
Get the state of a given object

java.lang.String **get_value**(java.lang.String object, **Oid** instance,
java.lang.String variable)
Get value for object.instance.variable (See **mimic value get**)

java.util.LinkedList **get_variables**(java.lang.String object, **Oid** instance)
Get variables for object.instance (See **mimic value variables**)

java.util.LinkedList **meval_value**(java.util.LinkedList objects,
java.util.LinkedList instances)
Evaluate the value for multiple object.instance

java.util.LinkedList **mget_value**(java.util.LinkedList objects,
java.util.LinkedList instances,
java.util.LinkedList variables)
Get values for multiple object.instance.variable (See **mimic value mget**)

void **mset_value**(java.util.LinkedList objects,
java.util.LinkedList instances,
java.util.LinkedList variables, java.util.LinkedList values)
set the value for multiple object.instance.variable (See **mimic value mset**)

void **munset_value**(java.util.LinkedList objects,
java.util.LinkedList instances,
java.util.LinkedList variables)
unset the value for multiple object.instance.variable (See **mimic value munset**)

void **remove**(java.lang.String object, **Oid** instance)
Remove an existing instance (table only) (See **mimic value remove**)

java.lang.String **set_pos**(java.lang.String pos)
Set the current position in agent's OID tree (See **mimic value pos**)

void **set_state**(java.lang.String object, int state)
Set the value for object.instance.variable (See **mimic value state**)

void **set_value**(java.lang.String object, **Oid** instance,
java.lang.String variable, java.lang.String value)
Set the value for object.instance.variable (See **mimic value set**)

java.util.LinkedList `split_oid(Oid oid)`

Split an OID into object and instance (See **mimic value split**)

void

`unset_value(java.lang.String object, Oid instance, java.lang.String variable)`

Unset the value for object.instance.variable (See **mimic value unset**)

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Valuespace

```
public Valuespace(Agent agent)
```

Creates a valuespace for the agent.

Parameters:

`agent` - (IN) agent number

Method Detail

get_pos

```
public java.lang.String get_pos()  
    throws ErrorInfo
```

Gets the current position in agent's OID tree.

Returns:

NONE

Throws:

[ErrorInfo](#) - on error

set_pos

```
public java.lang.String set_pos(java.lang.String pos)  
    throws ErrorInfo
```

Set the current position in agent's OID tree (See **mimic value pos**)

Parameters:

pos - (IN) position to be set

Returns:

String - position actually set

Throws:

`ErrorInfo` - on error

get_objects

```
public java.util.LinkedList get_objects()  
                               throws ErrorInfo
```

Get the children objects at the current position (See [mimic value list](#))

Returns:

LinkedList - list of children object name strings.

Throws:

`ErrorInfo` - on error

add

```
public void add(java.lang.String object,  
                Oid instance)  
               throws ErrorInfo
```

Add a new instance (for tables only) (See [mimic value add](#))

Parameters:

object - (IN) table entry object (e.g. `ifEntry`)

instance - (IN) instance oid to add

Throws:

`ErrorInfo` - on error

remove

```
public void remove(java.lang.String object,  
                   Oid instance)  
                  throws ErrorInfo
```

Remove an existing instance (table only) (See [mimic value remove](#))

Parameters:

object - (IN) table entry object (e.g. `ifEntry`)

instance - (IN) instance oid to add

Throws:

`ErrorInfo` - on error

get_instances

```
public java.util.LinkedList get_instances(java.lang.String object)
                                   throws ErrorInfo
```

Get instances of an object (See [mimic value instances](#))

Parameters:

object - (IN) leaf object (e.g. ifDescr)

Returns:

LinkedList - list of instance strings

Throws:

`ErrorInfo` - on error

get_variables

```
public java.util.LinkedList get_variables(java.lang.String object,
                                       Oid instance)
                                   throws ErrorInfo
```

Get variables for object.instance (See [mimic value variables](#))

Parameters:

object - (IN) leaf object (e.g. ifDescr)

instance - (IN) instance oid

Returns:

LinkedList - list of instance strings

Throws:

`ErrorInfo` - on error

get_value

```
public java.lang.String get_value(java.lang.String object,
                                  Oid instance,
                                  java.lang.String variable)
                                   throws ErrorInfo
```

Get value for object.instance.variable (See [mimic value get](#))

Parameters:

object - (IN) leaf object (e.g. ifDescr)

instance - (IN) instance oid

variable - (IN) variable name

Returns:

String - value string

Throws:

ErrorInfo - on error

mget_value

```
public java.util.LinkedList mget_value(java.util.LinkedList objects,
                                       java.util.LinkedList instances,
                                       java.util.LinkedList variables)
    throws ErrorInfo
```

Get values for multiple object.instance.variable (See [mimic value mget](#))

Parameters:

objects - (IN) list of leaf objects (e.g. ifDescr)

instances - (IN) list of instance oids

variables - (IN) list of variable names

Returns:

LinkedList - list of value strings

Throws:

ErrorInfo - on error

set_value

```
public void set_value(java.lang.String object,
                      Oid instance,
                      java.lang.String variable,
                      java.lang.String value)
    throws ErrorInfo
```

Set the value for object.instance.variable (See [mimic value set](#))

Parameters:

object - (IN) leaf object (e.g. ifDescr)

instance - (IN) instance oid

variable - (IN) variable name

value - (IN) value string

Throws:

ErrorInfo - on error

mset_value

```
public void mset_value(java.util.LinkedList objects,
                      java.util.LinkedList instances,
                      java.util.LinkedList variables,
                      java.util.LinkedList values)
    throws ErrorInfo
```

set the value for multiple object.instance.variable (See [mimic value mset](#))

Parameters:

objects - (IN) list of leaf objects (e.g. ifDescr)

instances - (IN) list of instance oids

variables - (IN) list of variable names

values - (IN) list of value strings

Throws:

[ErrorInfo](#) - on error

unset_value

```
public void unset_value(java.lang.String object,
                       Oid instance,
                       java.lang.String variable)
    throws ErrorInfo
```

Unset the value for object.instance.variable (See [mimic value unset](#))

Parameters:

object - (IN) leaf object (e.g. ifDescr)

instance - (IN) instance oid

variable - (IN) variable name

Throws:

[ErrorInfo](#) - on error

munset_value

```
public void munset_value(java.util.LinkedList objects,
                        java.util.LinkedList instances,
                        java.util.LinkedList variables)
    throws ErrorInfo
```

unset the value for multiple object.instance.variable (See [mimic value munset](#))

Parameters:

objects - (IN) list of leaf objects (e.g. ifDescr)

instances - (IN) list of instance oids
variables - (IN) list of variable names

Throws:

`ErrorInfo` - on error

eval_value

```
public java.lang.String eval_value(java.lang.String object,  
                                   Oid instance)  
    throws ErrorInfo
```

Evaluate the value for object.instance (See [mimic value eval](#))

Parameters:

object - (IN) leaf object (e.g. ifDescr)

instance - (IN) instance oid

Returns:

NONE

Throws:

`ErrorInfo` - on error

meval_value

```
public java.util.LinkedList meval_value(java.util.LinkedList objects,  
                                         java.util.LinkedList instances)  
    throws ErrorInfo
```

Evaluate the value for multiple object.instance

Parameters:

objects - (IN) list of leaf objects (e.g. ifDescr)

instances - (IN) list of instance oids

Returns:

LinkedList - list of value strings

Throws:

`ErrorInfo` - on error

get_oid

```
public Oid get_oid(java.lang.String object)  
    throws ErrorInfo
```

Get an OID for a name (See [mimic value oid](#))

Parameters:

object - (IN) object name (e.g. ifDescr)

Returns:

Oid - oid for given object

Throws:

`ErrorInfo` - on error

get_name

```
public java.lang.String get_name(Oid oid)
                           throws ErrorInfo
```

Get a name for an OID (See [mimic value name](#))

Parameters:

oid - (IN) object OID (e.g. 1.3.6.1.2.1.1)

Returns:

String - object name

Throws:

`ErrorInfo` - on error

get_mib

```
public java.lang.String get_mib(java.lang.String object)
                           throws ErrorInfo
```

Get the MIB containing a given object

Parameters:

object - (IN) object name (e.g. ifDescr)

Returns:

String - MIB name

Throws:

`ErrorInfo` - on error

get_info

```
public java.util.LinkedList get_info(java.lang.String object)
                           throws ErrorInfo
```

Get MIB information for object. MIB information is returned as a list of string tokens usually structured as :
token[0] = (noaccess | readonly | readwrite), token[1] = (branch | leaf), token[2] = (index | timeticks | counter | counter64 | octetstring ...), token[3...] = (index list for tables | size constraints for leaves) (See [mimic value info](#))

Parameters:

object - (IN) object name (e.g. ifDescr)

Returns:

LinkedList - list of strings (MIB details)

Throws:

`ErrorInfo` - on error

get_minfo

```
public java.util.LinkedList get_minfo(java.util.LinkedList objects)
                                throws ErrorInfo
```

Get MIB information for multiple objects (See [mimic value minfo](#))

Parameters:

objects - (IN) list of object names (e.g. ifDescr)

Returns:

LinkedList - list of list of strings (MIB details)

Throws:

`ErrorInfo` - on error

split_oid

```
public java.util.LinkedList split_oid(Oid oid)
                                throws ErrorInfo
```

Split an OID into object and instance (See [mimic value split](#))

Parameters:

oid - (IN) OID to split

Returns:

LinkedList - leaf OID and index OID

Throws:

`ErrorInfo` - on error

set_state

```
public void set_state(java.lang.String object,
                     int state)
                    throws ErrorInfo
```

Set the value for object.instance.variable (See [mimic value state](#))

Parameters:

object - (IN) leaf object (e.g. ifDescr)

state - (IN) state int

Throws:

[ErrorInfo](#) - on error

get_state

```
public java.lang.String get_state(java.lang.String object)
    throws ErrorInfo
```

Get the state of a given object

Parameters:

object - (IN) object name (e.g. ifDescr)

Returns:

String - enabled or disabled

Throws:

[ErrorInfo](#) - on error

[PACKAGE](#) **[CLASS](#)** [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

[SUMMARY: NESTED | FIELD | CONSTR | METHOD](#) [DETAIL: FIELD | CONSTR | METHOD](#)

Package Mimic

Class Summary

Class	Description
Agent	
Client	
Cloud	
Debug	
debugSession	
ErrorInfo	
Mgmt	
Oid	
Pod	
SearchPath	
Session	
StringParser	
Utils	
Valuespace	

Hierarchy For Package Mimic

Class Hierarchy

- java.lang.Object
 - Mimic.Agent
 - Mimic.Client
 - Mimic.Cloud
 - Mimic.Debug
 - Mimic.Mgmt
 - Mimic.Oid
 - Mimic.Pod
 - Mimic.SearchPath
 - Mimic.Session
 - Mimic.debugSession
 - Mimic.StringParser
 - java.lang.Throwable (implements java.io.Serializable)
 - Mimic.ErrorInfo
 - Mimic.Utils
 - Mimic.Valuespace

Deprecated API

Contents

A B C D E F G H I L M N O P R S T U V _

A

`add(String, Oid)` - Method in class `Mimic.Valuespace`

Add a new instance (for tables only) (See [mimic value add](#))

`add_access(String, String, String)` - Method in class `Mimic.Session`

Add user's access privilege for agents (See [mimic access add](#))

`add_ipalias(String, int, String, String)` - Method in class `Mimic.Agent`

Add a new ipalias (See [mimic agent ipalias add](#))

`add_timer_script(String, int, String)` - Method in class `Mimic.Agent`

Add (and start) timerscript with given interval and arguments (See [mimic timer script add](#))

`add_timer_script(String, int, String)` - Method in class `Mimic.Session`

Add (and start) timerscript with given interval and arguments (See [mimic timer script add](#))

`Agent` - Class in `Mimic`

`Agent(int, Session)` - Constructor for class `Mimic.Agent`

Creates an agent object for a given session

`AGENT_HALTED` - Static variable in class `Mimic.Agent`

agent's state : halted

`agent_mget_changed(LinkedList)` - Method in class `Mimic.Session`

Get changed flags for specified agents (See [mimic agent mget](#))

`agent_mget_config_changed(LinkedList)` - Method in class `Mimic.Session`

Get config_changed flags for specified agents (See [mimic agent mget](#))

`agent_mget_delay(LinkedList)` - Method in class `Mimic.Session`

Get delays for specified agents (See [mimic agent mget](#))

`agent_mget_drops(LinkedList)` - Method in class `Mimic.Session`

Get drops for specified agents (See [mimic agent mget](#))

`agent_mget_host(LinkedList)` - Method in class `Mimic.Session`

Get primary addresses for specified agents (See [mimic agent mget](#))

`agent_mget_inform_retries(LinkedList)` - Method in class `Mimic.Session`

Get inform retries for specified agents (See [mimic agent mget](#))

`agent_mget_inform_timeout(LinkedList)` - Method in class `Mimic.Session`

Get inform timeout for specified agents (See [mimic agent mget](#))

`agent_mget_interface(LinkedList)` - Method in class `Mimic.Session`

Get interfaces for specified agents (See [mimic agent mget](#))

`agent_mget_mask(LinkedList)` - Method in class `Mimic.Session`

Get masks for specified agents (See [mimic agent mget](#))

`agent_mget_mibs(LinkedList)` - Method in class `Mimic.Session`

Get MIB triplets for specified agents (See [mimic agent mget](#))

`agent_mget_oiddir(LinkedList)` - Method in class `Mimic.Session`

Get OidDir filenames for specified agents (See [mimic agent mget](#))

`agent_mget_owner(LinkedList)` - Method in class `Mimic.Session`

Get owners for specified agents (See [mimic agent mget](#))

`agent_mget_pdu_size(LinkedList)` - Method in class `Mimic.Session`

Get pdu_sizes for specified agents (See [mimic agent mget](#))

agent_mget_port(LinkedList) - Method in class Mimic.Session
 Get port numbers for specified agents (See [mimic agent mget](#))

agent_mget_privdir(LinkedList) - Method in class Mimic.Session
 Get privdirs for specified agents (See [mimic agent mget](#))

agent_mget_protocols(LinkedList) - Method in class Mimic.Session
 Get protocols for specified agents (See [mimic agent mget](#))

agent_mget_read_community(LinkedList) - Method in class Mimic.Session
 Get read communities for specified agents (See [mimic agent mget](#))

agent_mget_scen(LinkedList) - Method in class Mimic.Session
 Get scen for specified agents (See [mimic agent mget](#))

agent_mget_sim(LinkedList) - Method in class Mimic.Session
 Get sim name for specified agents (See [mimic agent mget](#))

agent_mget_starttime(LinkedList) - Method in class Mimic.Session
 Get starttimes for specified agents (See [mimic agent mget](#))

agent_mget_state(LinkedList) - Method in class Mimic.Session
 Get states for specified agents (See [mimic agent mget](#))

agent_mget_state_changed(LinkedList) - Method in class Mimic.Session
 Get state_changed flags for specified agents (See [mimic agent mget](#))

agent_mget_statistics(LinkedList) - Method in class Mimic.Session
 Get statistics for specified agents (See [mimic agent mget](#))

agent_mget_trace(LinkedList) - Method in class Mimic.Session
 Get trace flags for specified agents (See [mimic agent mget](#))

agent_mget_validate(LinkedList) - Method in class Mimic.Session
 Get validate flags for specified agents (See [mimic agent mget](#))

agent_mget_write_community(LinkedList) - Method in class Mimic.Session
 Get write communities for specified agents (See [mimic agent mget](#))

agent_mset_delay(LinkedList, LinkedList) - Method in class Mimic.Session
 Set delay for specified agents

agent_mset_drops(LinkedList, LinkedList) - Method in class Mimic.Session
 Set drops for specified agents

agent_mset_host(LinkedList, LinkedList) - Method in class Mimic.Session
 Set ip addresses for specified agents

agent_mset_inform_retries(LinkedList, LinkedList) - Method in class Mimic.Session
 Set inform retries for specified agents

agent_mset_inform_timeout(LinkedList, LinkedList) - Method in class Mimic.Session
 Set inform timeout for specified agents

agent_mset_interface(LinkedList, LinkedList) - Method in class Mimic.Session
 Set interfaces for specified agents

agent_mset_mask(LinkedList, LinkedList) - Method in class Mimic.Session
 Set subnet masks for specified agents

agent_mset_mibs(LinkedList, LinkedList) - Method in class Mimic.Session
 Set MIBs for specified agents

agent_mset_oiddir(LinkedList, LinkedList) - Method in class Mimic.Session
 Set OidDir filename for specified agents

agent_mset_pdu_size(LinkedList, LinkedList) - Method in class Mimic.Session
 Set PDU size for specified agents

agent_mset_port(LinkedList, LinkedList) - Method in class Mimic.Session
 Set port numbers for specified agents

agent_mset_protocol(LinkedList, LinkedList) - Method in class Mimic.Session
 Set protocols for specified agents

agent_mset_read(LinkedList, LinkedList) - Method in class Mimic.Session
Set read communities for specified agents

agent_mset_starttime(LinkedList, LinkedList) - Method in class Mimic.Session
Set start time for specified agents

agent_mset_trace(LinkedList, LinkedList) - Method in class Mimic.Session
Set trace flags for specified agents

agent_mset_validate(LinkedList, LinkedList) - Method in class Mimic.Session
Set validate for specified agents

agent_mset_write(LinkedList, LinkedList) - Method in class Mimic.Session
Set write communities for specified agents

AGENT_NOT_CONFIG - Static variable in class Mimic.Agent
agent's state : not configured

AGENT_PAUSED - Static variable in class Mimic.Agent
agent's state : paused

AGENT_RUNNING - Static variable in class Mimic.Agent
agent's state : running

AGENT_STOPPED - Static variable in class Mimic.Agent
agent's state : stopped

AGENT_STOPPING - Static variable in class Mimic.Agent
agent's state : stopping (or starting)

agent_store_append(String, String, int) - Method in class Mimic.Agent
Append (and start) storage with a given variable and value (See [mimic agent store append](#))

agent_store_copy(int) - Method in class Mimic.Agent
Copy variable store from other agent (See [mimic agent store copy](#))

agent_store_exists(String) - Method in class Mimic.Agent
Examine existence of a store variable (See [mimic agent store exists](#))

agent_store_get(String) - Method in class Mimic.Agent
Return variable value (See [mimic agent store get](#))

agent_store_list() - Method in class Mimic.Agent
List variables (See [mimic agent store list](#))

agent_store_lreplace(String, int, String) - Method in class Mimic.Agent
Treat variable as a list and modify entry for given index with given value (See [mimic agent store lreplace](#))

agent_store_mget(LinkedList) - Method in class Mimic.Agent
Get multiple variable values (See [mimic agent store mget](#))

agent_store_mlreplace(LinkedList, LinkedList, LinkedList) - Method in class Mimic.Agent
Treat each variable as a list and modify entry for given index with given value (See [mimic agent store mlreplace](#))

agent_store_persists(String) - Method in class Mimic.Agent
Examine persistence of a store variable (See [mimic agent store persists](#))

agent_store_set(String, String, int) - Method in class Mimic.Agent
Set (and start) storage with a given variable and value (See [mimic agent store set](#))

agent_store_unset(String) - Method in class Mimic.Agent
Unset a variable-value pair (See [mimic agent store unset](#))

B

BOTH - Static variable in class Mimic.SearchPath
Both shared and private areas

C

cfg_load(String) - Method in class Mimic.Session

Loads the specified configuration file

cfg_load(String, String) - Method in class Mimic.Session

Loads the specified configuration file

cfg_load(String, String, int) - Method in class Mimic.Session

Loads the specified configuration file

cfg_load(String, String, int, int) - Method in class Mimic.Session

/ Loads the specified configuration file**

cfg_new() - Method in class Mimic.Session

Load a new blank configuration

cfg_new(String) - Method in class Mimic.Session

clear configuration for specified agents

cfg_save() - Method in class Mimic.Session

Saves the current configuration

cfg_save(String) - Method in class Mimic.Session

Saves the current configuration

cfg_saveas(String) - Method in class Mimic.Session

Saves the current configuration as specified filename

cfg_saveas(String, String) - Method in class Mimic.Session

Saves the current configuration as specified filename

Client - Class in Mimic

Client() - Constructor for class Mimic.Client

Creates an default MIMIC client object

close_session(Session) - Method in class Mimic.Client

Closes an already open session (See [mimic session close](#))

Cloud - Class in Mimic

Cloud() - Constructor for class Mimic.Cloud

configure(String, LinkedList) - Method in class Mimic.Agent

Configure agent

connect() - Method in class Mimic.debugSession

connect() - Method in class Mimic.Session

Connects to the MIMIC daemon at the configured host and port

D

Debug - Class in Mimic

Debug() - Constructor for class Mimic.Debug

debugSession - Class in Mimic

debugSession(String, int) - Constructor for class Mimic.debugSession

del_access(String) - Method in class Mimic.Session

Delete user's access privilege (See [mimic access del](#))

del_ipalias(String, int) - Method in class Mimic.Agent

Delete a configured ipalias (See [mimic agent ipalias delete](#))

del_timer_script(String) - Method in class Mimic.Agent

Delete a running script with specified name (See [mimic timer script delete](#))
del_timer_script(String) - Method in class Mimic.Session

Delete a running script with specified name (See [mimic timer script delete](#))
diag(String) - Method in class Mimic.Session

Execute specified diagnostic code (used only for debugging)
diag(int) - Method in class Mimic.Session

Execute specified diagnostic code (used only for debugging)
disable() - Static method in class Mimic.Debug

Disables the display of debugging messages (using Debug.print) for the whole package.
disconnect() - Method in class Mimic.debugSession

disconnect() - Method in class Mimic.Session

Disconnects from the MIMIC daemon
dispatch_msg(int, String) - Method in class Mimic.debugSession

dispatch_msg(int, String) - Method in class Mimic.Session

Dispatches a request to the MIMIC daemon, retrieves and returns the reply from the same
dump() - Method in class Mimic.Client

diagnostic dump
dump() - Method in class Mimic.Session

Set start time for specified agents
dumpMarkers() - Method in class Mimic.debugSession

E

enable() - Static method in class Mimic.Debug

Enables the display of debugging messages (using Debug.print) for the whole package.
ErrorInfo - Class in Mimic

ErrorInfo(int, String) - Constructor for class Mimic.ErrorInfo

Creates the ErrorInfo exception object thrown by the Mimic package on error conditions.
eval_value(String, Oid) - Method in class Mimic.Valuespace

Evaluate the value for object.instance (See [mimic value eval](#))

F

FAILURE - Static variable in class Mimic.ErrorInfo

returned by get_error_code() indicating that no error occurred
find(String, int) - Static method in class Mimic.SearchPath

Finds a file relative to the directories in the searchpath
find_by_id(Session, String) - Static method in class Mimic.Pod

from_add(String, int) - Method in class Mimic.Agent

Add a new 'from' entry (See [mimic agent from add](#))
from_del(String, int) - Method in class Mimic.Agent

Delete a configured 'from' entry (See [mimic agent from delete](#))
from_list() - Method in class Mimic.Agent

List the 'from' entry list for the agent (See [mimic agent from list](#))

G

`get_access_aclddb()` - Method in class `Mimic.Session`
Get data base for access control (See `mimic access get aclddb`)

`get_access_admindir()` - Method in class `Mimic.Session`
Get access administrative directory (See `mimic access get adminidir`)

`get_access_adminuser()` - Method in class `Mimic.Session`
Get mimic access administrator (See `mimic access get adminuser`)

`get_access_enabled()` - Method in class `Mimic.Session`
Get access control enabled value (See `mimic access get enabled`)

`get_active_data_list()` - Method in class `Mimic.Session`
Get list of running agents with their statistics (See `mimic get`)

`get_active_data_list_Long()` - Method in class `Mimic.Session`
Get list of running agents with their statistics (See `mimic get`)

`get_active_list()` - Method in class `Mimic.Session`
Get the list of running agents (See `mimic get`)

`get_agent(int)` - Method in class `Mimic.Session`
Gets the agent object

`get_agent_no()` - Method in class `Mimic.Agent`
Get agent's number

`get_agent_position(int)` - Method in class `Mimic.Pod`

`get_cfg_file_changed()` - Method in class `Mimic.Session`
Get flag indicating if config file has changed (See `mimic get`)

`get_cfgfile()` - Method in class `Mimic.Session`
Get the currently loaded configuration file (See `mimic get`)

`get_changed()` - Method in class `Mimic.Agent`
Get the changed flag (See `mimic agent get`)

`get_changed_config_list()` - Method in class `Mimic.Session`
Get list of agents whose configuration has changed (See `mimic get`)

`get_changed_state_list()` - Method in class `Mimic.Session`
Get a list of agents whose state has changed (See `mimic get`)

`get_clients()` - Method in class `Mimic.Session`
Get the number of connected clients (See `mimic get`)

`get_config_changed()` - Method in class `Mimic.Agent`
Get the config changed flag (See `mimic agent get`)

`get_configured_list()` - Method in class `Mimic.Session`
Get the list of configured agents (See `mimic get`)

`get_delay()` - Method in class `Mimic.Agent`
Get the delay (See `mimic agent get`)

`get_drops()` - Method in class `Mimic.Agent`
Get the drop rate (See `mimic agent get`)

`get_error_code()` - Method in class `Mimic.ErrorInfo`
Get the error code for this exception

`get_error_msg()` - Method in class `Mimic.ErrorInfo`
Get the error message string for this exception

`get_expiration(int)` - Method in class `Mimic.Session`
Get expiration of license within specified days (See `mimic get`)

`get_host()` - Method in class `Mimic.Agent`
Get the primary ip address (See `mimic agent get`)

`get_host()` - Method in class `Mimic.Session`
Gets the host address of the MIMIC daemon

`get_info(String)` - Method in class `Mimic.Valuespace`
Get MIB information for object.

`get_inform_retries()` - Method in class `Mimic.Agent`
Get the inform retries (See `mimic agent get`)

`get_inform_timeout()` - Method in class `Mimic.Agent`
Get the inform timeout (See `mimic agent get`)

`get_instances(String)` - Method in class `Mimic.Valuespace`
Get instances of an object (See `mimic value instances`)

`get_interface()` - Method in class `Mimic.Agent`
Get the interface for the primary ipalias (See `mimic agent get`)

`get_interfaces()` - Method in class `Mimic.Session`
Get the interfaces (See `mimic get`)

`get_last()` - Method in class `Mimic.Session`
Get the last configured agent (See `mimic get`)

`get_licensing()` - Method in class `Mimic.Session`
Get licensing info where MIMIC is running.

`get_log()` - Method in class `Mimic.Session`
Get the current logfile (See `mimic get`)

`get_mask()` - Method in class `Mimic.Agent`
Get the primary subnet mask (See `mimic agent get`)

`get_max()` - Method in class `Mimic.Session`
Get maximum agents for session (See `mimic get`)

`get_mib(String)` - Method in class `Mimic.Valuespace`
Get the MIB containing a given object

`get_mibs()` - Method in class `Mimic.Agent`
Get the MIB triplets (See `mimic agent get`)

`get_minfo(LinkedList)` - Method in class `Mimic.Valuespace`
Get MIB information for multiple objects (See `mimic value minfo`)

`get_name(Oid)` - Method in class `Mimic.Valuespace`
Get a name for an OID (See `mimic value name`)

`get_netaddr()` - Method in class `Mimic.Session`
Get network address of host where MIMIC is running.

`get_netdev()` - Method in class `Mimic.Session`
Get default network device of host where MIMIC is running.

`get_nth_agent(Session, int, int)` - Static method in class `Mimic.Pod`

`get_nth_agent(int, int)` - Method in class `Mimic.Pod`

`get_nth_agent(int)` - Method in class `Mimic.Pod`

`get_num_tokens()` - Method in class `Mimic.StringParser`
Get the number of tokens detected in the parsed string

`get_objects()` - Method in class `Mimic.Valuespace`
Get the children objects at the current position (See `mimic value list`)

`get_oid(String)` - Method in class `Mimic.Valuespace`
Get an OID for a name (See `mimic value oid`)

`get_oiddir()` - Method in class `Mimic.Agent`
Get the `OidDir` filename (See `mimic agent get`)

`get_owner()` - Method in class `Mimic.Agent`
Get the owner (See `mimic agent get`)

get_pdu_size() - Method in class Mimic.Agent

Get the maximum PDU size (See [mimic agent get](#))

get_pod(Session, int) - Static method in class Mimic.Pod

get_pod_agents(Session, int) - Method in class Mimic.Pod

get_pod_assigned(Session, String) - Static method in class Mimic.Pod

get_pod_number(Session, int) - Static method in class Mimic.Pod

get_port() - Method in class Mimic.Agent

Get the port number (See [mimic agent get](#))

get_port() - Method in class Mimic.Session

Gets the management port number of the MIMIC daemon

get_pos() - Method in class Mimic.Valuespace

Gets the current position in agent's OID tree.

get_privdir() - Method in class Mimic.Agent

Get the private directory (See [mimic agent get](#))

get_privdir() - Method in class Mimic.Session

Gets the private directory for the session The Simulator private directory can be retrieved with [get_sim_privdir\(\)](#).

get_product() - Method in class Mimic.Session

Get licensed product number (See [mimic get](#))

get_protocols() - Method in class Mimic.Agent

Get the list of protocols (See [mimic agent get](#))

get_protocols() - Method in class Mimic.Session

Get all the protocols supported by the simulator (See [mimic get](#))

get_read_community() - Method in class Mimic.Agent

Get the read community (See [mimic agent get](#))

get_scen() - Method in class Mimic.Agent

Get the simulation scen (See [mimic agent get](#))

get_session() - Method in class Mimic.Agent

Get agent's session

get_sim() - Method in class Mimic.Agent

Get the sim name (See [mimic agent get](#))

get_sim_privdir() - Method in class Mimic.Session

Get the simulator's private directory (See [mimic get](#))

get_starttime() - Method in class Mimic.Agent

Get the start time (See [mimic agent get](#))

get_state() - Method in class Mimic.Agent

Get the agent state (See [mimic agent get](#))

get_state(String) - Method in class Mimic.Valuespace

Get the state of a given object

get_state_changed() - Method in class Mimic.Agent

Get the state changed flag (See [mimic agent get](#))

get_state_descr(int) - Static method in class Mimic.Agent

Get the description of the specified agent state (See [mimic agent get](#))

get_statistics() - Method in class Mimic.Agent

Get the list of statistics (See [mimic agent get](#))

get_statistics_Long() - Method in class Mimic.Agent

Get the list of statistics (Long) (See [mimic agent get](#))

get_statistics_String() - Method in class Mimic.Agent
 Get the list of statistics (String) (See [mimic agent get](#))

get_stop_time(Session, int) - Static method in class Mimic.Pod

get_token(int) - Method in class Mimic.StringParser
 Get the index'th string token

get_trace() - Method in class Mimic.Agent
 Get the trace flag (See [mimic agent get](#))

get_user() - Method in class Mimic.Session
 Gets the username for the session

get_validate() - Method in class Mimic.Agent
 Get the validate flag (See [mimic agent get](#))

get_value(String, Oid, String) - Method in class Mimic.Valuespace
 Get value for object.instance.variable (See [mimic value get](#))

get_valuespace() - Method in class Mimic.Agent
 Get the valuespace object for this agent.

get_variables(String, Oid) - Method in class Mimic.Valuespace
 Get variables for object.instance (See [mimic value variables](#))

get_version() - Method in class Mimic.Session
 Get the current MIMIC daemon version (See [mimic get](#))

get_write_community() - Method in class Mimic.Agent
 Get the write community (See [mimic agent get](#))

getAccessStatus(Session, int) - Static method in class Mimic.Pod

getDemoKeyPrefix(Session) - Static method in class Mimic.Pod

getDemoThrottle(Session) - Static method in class Mimic.Pod

getDemoThrottleException(Session) - Static method in class Mimic.Pod

getDemoThrottleFlushPeriod(Session) - Static method in class Mimic.Pod

getExceptionStackTraceAsString(Exception) - Static method in class Mimic.debugSession

getExpiredKeys(Session) - Static method in class Mimic.Pod

getExpireFlushFrequency(Session) - Static method in class Mimic.Pod

getInUseStatus(Session, int) - Static method in class Mimic.Pod

getLocalizedMessage() - Method in class Mimic.ErrorInfo

getMessage() - Method in class Mimic.ErrorInfo

getMessage(Session, int, String) - Static method in class Mimic.Pod

getPodMessage(Session, int) - Static method in class Mimic.Pod

getPoolName(Session, int) - Static method in class Mimic.Pod

getProxyServer(Session, int) - Static method in class Mimic.Pod

getTerminalServer(Session, int) - Static method in class Mimic.Pod

getVlabType(Session, int) - Static method in class Mimic.Pod

H

halt() - Method in class Mimic.Agent

Halt agent (from running) (See [mimic agent halt](#))

I

is_cloud_var_set(Session, String) - Static method in class Mimic.Cloud

isDebugEnabled() - Static method in class Mimic.Client

isDemoThrottleOn(Session) - Static method in class Mimic.Pod

isExpireTrackOn(Session) - Static method in class Mimic.Pod

L

LinkedListToString(LinkedList, String) - Static method in class Mimic.Utils

Creates a string from list, separating entries using the separator string provided.

list_access() - Method in class Mimic.Session

List accesses assigned to users (See [mimic access list](#))

list_ipaliases() - Method in class Mimic.Agent

List the ipaliases for the agent (See [mimic agent ipalias list](#))

list_sessions() - Method in class Mimic.Client

returns list of all the open sessions (See [mimic session list](#))

list_timer_scripts() - Method in class Mimic.Agent

List the currently running timer scripts (See [mimicsh_timer_script_list](#))

list_timer_scripts() - Method in class Mimic.Session

List the currently running timer scripts (See [mimic timer script list](#))

load_access(String) - Method in class Mimic.Session

Load access control file (See [mimic access load](#))

M

mark(String, Object) - Method in class Mimic.debugSession

meval_value(LinkedList, LinkedList) - Method in class Mimic.Valuespace

Evaluate the value for multiple object.instance

mget_value(LinkedList, LinkedList, LinkedList) - Method in class Mimic.Valuespace

Get values for multiple object.instance.variable (See [mimic value mget](#))

Mgmt - Class in Mimic

Mgmt() - Constructor for class Mimic.Mgmt

MGMT_DIAG_DISABLE_ERRORS - Static variable in class Mimic.Mgmt

MGMT_DIAG_ENABLE_ERRORS - Static variable in class Mimic.Mgmt

MGMT_PORT - Static variable in class Mimic.Mgmt

default management port

Mimic - package Mimic

mset_value(LinkedList, LinkedList, LinkedList, LinkedList) - Method in class Mimic.Valuespace

set the value for multiple object.instance.variable (See mimic value mset)

munset_value(LinkedList, LinkedList, LinkedList) - Method in class Mimic.Valuespace

unset the value for multiple object.instance.variable (See mimic value munset)

N

number_of_agents(Session, int) - Static method in class Mimic.Pod

number_of_agents(int) - Method in class Mimic.Pod

number_of_agents() - Method in class Mimic.Pod

number_of_pods(Session) - Static method in class Mimic.Pod

number_of_pods() - Method in class Mimic.Pod

O

Oid - Class in Mimic

Oid() - Constructor for class Mimic.Oid

Creates an empty Oid object

Oid(String) - Constructor for class Mimic.Oid

Creates an Oid object based on passed string

open_session(String, int) - Method in class Mimic.Client

Opens a new session to the host (See mimic session open)

P

pause_at(int) - Method in class Mimic.Agent

Pause agent at given time (from running) (See mimic agent pause)

pause_now() - Method in class Mimic.Agent

Pause agent now (from running) (See mimic agent pause)

Pod - Class in Mimic

Pod(int, Session) - Constructor for class Mimic.Pod

Creates an pod object for a given session

pod_init_done(Session) - Static method in class Mimic.Pod

pod_reset(Session) - Static method in class Mimic.Pod

print(String) - Static method in class Mimic.Debug

Prints the debug message if debugging is enabled.

PRIVATE - Static variable in class Mimic.SearchPath

Private area only

privpath(String) - Static method in class Mimic.SearchPath

Get the private path for a given path

protocol_msg(String, String) - Method in class Mimic.Agent

Pass agent's message string to the protocol

protocol_msg(String, String) - Method in class Mimic.Session

Pass session's message string to the protocol

R

remove() - Method in class Mimic.Agent

Delete a configured agent (See [mimic agent remove](#))

remove(String, Oid) - Method in class Mimic.Valuespace

Remove an existing instance (table only) (See [mimic value remove](#))

resume() - Method in class Mimic.Agent

Resume agent (from paused/halted) (See [mimic agent resume](#))

S

save() - Method in class Mimic.Agent

Save the agent's valuespace changes (See [mimic agent save](#))

save_access(String) - Method in class Mimic.Session

Save access control file (See [mimic access save](#))

SearchPath - Class in Mimic

SearchPath() - Constructor for class Mimic.SearchPath

Session - Class in Mimic

Session(String, int) - Constructor for class Mimic.Session

Creates a session for given host and port

sessionID - Variable in class Mimic.debugSession

set_access_acldb(String) - Method in class Mimic.Session

Set access control's data base (See [mimic access set acldb](#))

set_access_adminidir(String) - Method in class Mimic.Session

Set access administrative directory.

set_access_adminuser(String) - Method in class Mimic.Session

Set access administrator.

set_access_enabled(int) - Method in class Mimic.Session

Set access control's enabled value (See [mimic access set enabled](#))

set_attr(Session, String, int, String, int) - Static method in class Mimic.Pod

set_delay(int) - Method in class Mimic.Agent

Set the delay (See [mimic agent set](#))

set_drops(int) - Method in class Mimic.Agent

Set the drop rate (See [mimic agent set](#))

set_host(String) - Method in class Mimic.Agent

Set the primary ip address (See [mimic agent set](#))

set_inform_retries(int) - Method in class Mimic.Agent

Set the inform retries (See [mimic agent set](#))

set_inform_timeout(int) - Method in class Mimic.Agent

Set the inform timeout (See [mimic agent set](#))

set_interface(String) - Method in class Mimic.Agent

Set the interface for primary ipalias (See [mimic agent set](#))

set_log(String) - Method in class Mimic.Session

Set a new logfile (See [mimic set](#))

set_mask(String) - Method in class Mimic.Agent

Set the primary subnet mask (See [mimic agent set](#))
set_mibs(LinkedList) - Method in class Mimic.Agent
Set the MIB triplets (See [mimic agent set](#))
set_oiddir(String) - Method in class Mimic.Agent
Set the OidDir filename (See [mimic agent set](#))
set_pdu_size(int) - Method in class Mimic.Agent
Set the maximum PDU size (See [mimic agent set](#))
set_port(int) - Method in class Mimic.Agent
Set the port number (See [mimic agent set](#))
set_pos(String) - Method in class Mimic.Valuespace
Set the current position in agent's OID tree (See [mimic value pos](#))
set_protocols(LinkedList) - Method in class Mimic.Agent
Set the list of protocols supported (See [mimic agent set](#))
set_read_community(String) - Method in class Mimic.Agent
Set the read community (See [mimic agent set](#))
set_starttime(int) - Method in class Mimic.Agent
Set the starttime (See [mimic agent set](#))
set_state(String, int) - Method in class Mimic.Valuespace
Set the value for object.instance.variable (See [mimic value state](#))
set_trace(int) - Method in class Mimic.Agent
Set the trace flag (See [mimic agent set](#))
set_validate(int) - Method in class Mimic.Agent
Set the validate flag (See [mimic agent set](#))
set_value(String, Oid, String, String) - Method in class Mimic.Valuespace
Set the value for object.instance.variable (See [mimic value set](#))
set_write_community(String) - Method in class Mimic.Agent
Set the write community (See [mimic agent set](#))
setDebugEnable(boolean) - Static method in class Mimic.Client
setInUseStatus(Session, int, int) - Static method in class Mimic.Pod
setMessage(Session, int, String, String) - Static method in class Mimic.Pod
SHARED - Static variable in class Mimic.SearchPath
Shared area only
split_oid(Oid) - Method in class Mimic.Valuespace
Split an OID into object and instance (See [mimic value split](#))
SplitToIntegerList(String) - Static method in class Mimic.Utills
Creates a linked list of Integers from passed String
SplitToLongList(String) - Static method in class Mimic.Utills
Creates a linked list of Longs from passed String
SplitToOidSubidList(String) - Static method in class Mimic.Utills
Creates a linked list of positive Longs from passed String Throws an `ErrorInfo` exception if the numbers are negative
SplitToStringList(String, char) - Static method in class Mimic.Utills
Creates a linked list of tokens from passed string
SplitToStringList(String) - Static method in class Mimic.Utills
Creates a linked list of tokens from passed string
start() - Method in class Mimic.Agent
Start agent (from stopped) (See [mimic agent start](#))

`start_all_agents()` - Method in class `Mimic.Session`

Starts all configured agents

`start_ipalias(String, int)` - Method in class `Mimic.Agent`

Start the configured ipalias (See [mimic agent ipalias start](#))

`status_ipalias(String, int)` - Method in class `Mimic.Agent`

Status of the configured ipalias for the agent (See [mimic agent ipalias status](#))

`stop()` - Method in class `Mimic.Agent`

Stop agent (from running, paused, halted) (See [mimic agent stop](#))

`stop_all_agents()` - Method in class `Mimic.Session`

Stops all running agents

`stop_ipalias(String, int)` - Method in class `Mimic.Agent`

Stop the configured ipalias (See [mimic agent ipalias stop](#))

`store_append(String, String, int)` - Method in class `Mimic.Session`

Append (and start) storage with a given variable and value (See [mimic store append](#))

`store_exists(String)` - Method in class `Mimic.Session`

Examine existance (See [mimic store exists](#))

`store_get(String)` - Method in class `Mimic.Session`

Get variable value (See [mimic store get](#))

`store_list()` - Method in class `Mimic.Session`

List variables (See [mimic store list](#))

`store_lreplace(String, int, String)` - Method in class `Mimic.Session`

Treat variable as a list and modify entry for given index with given value (See [mimic store lreplace](#))

`store_mget(LinkedList)` - Method in class `Mimic.Session`

Get multiple variable values (See [mimic store mget](#))

`store_mlreplace(LinkedList, LinkedList, LinkedList)` - Method in class `Mimic.Session`

Treat each variable as a list and modify entry for given index with given value (See [mimic store mlreplace](#))

`store_persists(String)` - Method in class `Mimic.Session`

Examine persistence (See [mimic store persists](#))

`store_save()` - Method in class `Mimic.Session`

Set store variables persistent saving to disk (See [mimic set](#))

`store_set(String, String, int)` - Method in class `Mimic.Session`

Set (and start) storage with a given variable and value (See [mimic store set](#))

`store_unset(String)` - Method in class `Mimic.Session`

Unset a variable-value pair (See [mimic store unset](#))

`StringParser` - Class in `Mimic`

`StringParser()` - Constructor for class `Mimic.StringParser`

Creates a `StringParser` instance

`SUCCESS` - Static variable in class `Mimic.ErrorInfo`

returned by `get_error_code()` indicating that some error occurred

T

`terminate()` - Method in class `Mimic.Session`

Terminates the connected MIMIC daemon

`tokenize(String, char)` - Method in class `Mimic.StringParser`

Parse the supplied string and split it into tokens separated by `sep`, surrounded in quotes and/or surrounded in braces "{...}"

`tokenize(String)` - Method in class `Mimic.StringParser`

Parse the supplied string and split it into tokens separated by spaces, surrounded in quotes and/or surrounded in braces "{...}"

toString() - Method in class Mimic.Oid

returns a string representation of the object

trap_config_add(String, int) - Method in class Mimic.Agent

Set the trap destination (See [mimic agent trap config add](#))

trap_config_add(String, int) - Method in class Mimic.Session

add a trap destination (See [mimic trap](#))

trap_config_del(String, int) - Method in class Mimic.Agent

Remove the trap destination from list (See [mimic agent trap config delete](#))

trap_config_delete(String, int) - Method in class Mimic.Session

delete a trap destination (See [mimic trap](#))

trap_config_list() - Method in class Mimic.Agent

Get the trap destination list (See [mimic agent trap config list](#))

trap_config_list() - Method in class Mimic.Session

List variables of trap destination (See [mimic trap](#))

trap_list() - Method in class Mimic.Agent

Get the trap list

U

unset_value(String, Oid, String) - Method in class Mimic.Valuespace

Unset the value for object.instance.variable (See [mimic value unset](#))

Utils - Class in Mimic

Utils() - Constructor for class Mimic.Utils

V

Valuespace - Class in Mimic

Valuespace(Agent) - Constructor for class Mimic.Valuespace

Creates a valuespace for the agent.

–

_get_cloud_help_params(Session, String) - Static method in class Mimic.Cloud

_get_configured_pods(Session) - Static method in class Mimic.Pod

_get_console_params(Session, int, String) - Static method in class Mimic.Cloud

_get_external_console_params(Session, int, String) - Static method in class Mimic.Cloud

_get_external_telnet_params(Session, int, String) - Static method in class Mimic.Cloud

_get_nth_pod(Session, int) - Static method in class Mimic.Pod

_get_pod_id(Session, int) - Static method in class Mimic.Pod

_get_ssh_params(Session, int, String) - Static method in class Mimic.Cloud

_get_telnet_params(Session, int, String) - Static method in class Mimic.Cloud

A B C D E F G H I L M N O P R S T U V _

[PACKAGE](#) [CLASS](#) [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV](#) [NEXT](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

How This API Document Is Organized

This API (Application Programming Interface) document has pages corresponding to the items in the navigation bar, described as follows.

Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. This page can contain six categories:

- Interfaces (*italic*)
- Classes
- Enums
- Exceptions
- Errors
- Annotation Types

Class/Interface

Each class, interface, nested class and nested interface has its own separate page. Each of these pages has three sections consisting of a class/interface description, summary tables, and detailed member descriptions:

- Class inheritance diagram
- Direct Subclasses
- All Known Subinterfaces
- All Known Implementing Classes
- Class/interface declaration
- Class/interface description
- Nested Class Summary
- Field Summary
- Constructor Summary
- Method Summary
- Field Detail
- Constructor Detail
- Method Detail

Each summary entry contains the first sentence from the detailed description for that item. The summary entries are alphabetical, while the detailed descriptions are in the order they appear in the source code. This preserves the logical groupings established by the programmer.

Annotation Type

Each annotation type has its own separate page with the following sections:

- Annotation Type declaration
- Annotation Type description
- Required Element Summary
- Optional Element Summary

- [Element Detail](#)

Enum

Each enum has its own separate page with the following sections:

- Enum declaration
- Enum description
- Enum Constant Summary
- Enum Constant Detail

Tree (Class Hierarchy)

There is a [Class Hierarchy](#) page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. The classes are organized by inheritance structure starting with `java.lang.Object`. The interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on "Tree" displays the hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking "Tree" displays the hierarchy for only that package.

Deprecated API

The [Deprecated API](#) page lists all of the API that have been deprecated. A deprecated API is not recommended for use, generally due to improvements, and a replacement API is usually given. Deprecated APIs may be removed in future implementations.

Index

The [Index](#) contains an alphabetic list of all classes, interfaces, constructors, methods, and fields.

Prev/Next

These links take you to the next or previous class, interface, package, or related page.

Frames/No Frames

These links show and hide the HTML frames. All pages are available with or without frames.

All Classes

The [All Classes](#) link shows all classes and interfaces except non-static nested types.

Serialized Form

Each serializable or externalizable class has a description of its serialization fields and methods. This information is of interest to re-implementors, not to developers using the API. While there is no link in the navigation bar, you can get to this information by going to any serialized class and clicking "Serialized Form" in the "See also" section of the class description.

Constant Field Values

The [Constant Field Values](#) page lists the static final fields and their values.

This help file applies to API documentation generated using the standard doclet.

[PACKAGE](#) [CLASS](#) [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)

[PREV](#) [NEXT](#) [FRAMES](#) [NO FRAMES](#) [ALL CLASSES](#)

□ Table of contents

□

Visual Studio 2013 Compatibility

15 minutes to read

In this article

When you are considering whether you should move to Visual Studio 2013, you can use this document to find out which solutions, projects, files, and other assets that you created in Visual Studio 2012 or Visual Studio 2010 SP1 will run without modification in Visual Studio 2013. You might also reach this page if you encounter a project that won't open in your Visual Studio installation; sometimes this is because the project is not supported in your version of Visual Studio, and sometimes you just need to install the an SDK (for example, you need to install the Azure SDK for .NET to open Cloud Service projects and Azure Resource Manager projects). Many widely used assets behave the same in Visual Studio 2013 and the two earlier versions. For example, in Visual Studio 2013, you can open a project that was created in Visual Studio 2012, change it, and then reopen it in Visual Studio 2013; your changes persist and the project behaves the same as it does in Visual Studio 2012. The same is true for many assets that were created in Visual Studio 2010 SP1.

If you use Visual Studio 2013 together with Visual Studio 2012 or Visual Studio 2010 SP1, you can create and modify projects and files in any of the three versions. You can transfer projects and files among the versions as long as you don't add features that are not supported by one of the versions.

Projects

The following list describes support in Visual Studio 2013 for projects that were created in Visual Studio 2012 or Visual Studio 2010 SP1. You can use this list to help determine whether you can open a project as-is in Visual Studio 2013, Visual Studio 2012, or Visual Studio 2010 SP1, or whether you have to modify it to ensure compatibility.

Type of Project Compatibility

Windows Store app development is supported only on Windows 8.1. Existing Windows 8 projects can continue to be serviced, but new Windows 8 projects cannot be created.

Windows 8.1 projects can depend only on certain types of references. For more information, see [Managing Project References](#).

Windows Store apps

Note

Windows 8.1 projects that you create by using Visual Studio 2013 cannot be opened in Visual Studio 2012. That is because Windows 8.1 projects created by using Visual Studio 2013 target Visual Studio 2013, and Visual Studio 2012 supports only Windows 8 projects that target Windows 8.

.NET Framework 4.5.1	You can create and use these projects in Visual Studio 2013 after you install the appropriate multi-targeting pack. These projects are not supported in Visual Studio 2010 SP1.
.NET Framework 4.5	You can create and open these projects in Visual Studio 2013 and Visual Studio 2012, but not in Visual Studio 2010 SP1. For more information, see Migration Guide to the .NET Framework 4.5
BizTalk	BizTalk server projects are not compatible with Visual Studio 2013.
C#/Visual Basic Silverlight 4 Application or Class Library	If you allow Visual Studio to update the project automatically, you can open it in either Visual Studio 2013 or Visual Studio 2012.
C#/Visual Basic Webform or Windows Form	You can open the project in Visual Studio 2013 and Visual Studio 2012.
Visual Basic 6 and Visual C++ 6	Visual Studio 2012 and Visual Studio 2013 do not support debugging applications built with Visual Basic 6 or Visual C++ 6; to debug these applications, use earlier versions of Visual Studio.
Coded UI test	If you allow Visual Studio to update the project automatically, you can open it in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1.
F#	If you allow Visual Studio to upgrade a project that was created in Visual Studio 2010 SP1, you can open it in Visual Studio 2013 and Visual Studio 2012. However, you can't upgrade a Silverlight project that was created in an older version of Visual Studio to Visual Studio 2013. Instead, you must create a Silverlight project in Visual Studio 2013 and then copy your code into it. Silverlight projects that you create in Visual Studio 2013 target Silverlight 5.
LightSwitch	If you allow Visual Studio to upgrade the project automatically, you can open it in Visual Studio 2013 only.
Local Database Cache	The Local Database Cache template and the Configure Data Synchronization dialog box are not included in Visual Studio 2013. You can use Visual Studio 2013 to open and run projects that were created in Visual Studio 2010 if Microsoft Synchronization Services v1.0 is installed, but if you want to update them in Visual Studio 2013, you must make all changes manually in code. As an alternative, you can continue to use Visual Studio 2010 to maintain and update these projects. For new development, target the new synchronization model that's provided by the Microsoft Sync Framework. For information, see Microsoft Sync Framework Developer Center
Model-View-Controller framework	Visual Studio 2010 SP1 supports only MVC 2 and MVC 3, Visual Studio 2012 supports only MVC 3 and MVC 4, and Visual Studio 2013 supports only MVC 4. For information about how to automatically upgrade from MVC 2 to MVC 3, see ASP.NET MVC 3 Application Upgrader . For information about how to manually upgrade from MVC 2 to MVC 3, see Upgrading an ASP.NET MVC 2 Project to ASP.NET MVC 3 Tools Update . For information about how to manually upgrade from MVC3 to MVC 4, see Upgrading an ASP.NET MVC 3 Project to

[ASP.NET MVC 4](#). If your project targets the .NET Framework 3.5 SP1, you must retarget it to use the .NET Framework 4.

If you allow Visual Studio to update the project automatically, you can open it in Visual Studio 2013, Visual Studio 2012, or Visual Studio 2010 SP1.

When Team Foundation builds a modeling project, it tries to validate the layers in the project. In Visual Studio 2013, Team Foundation Build can't validate the layers for a modeling project that was created in Visual Studio 2010 SP1. However, in Visual Studio 2010 SP1, Team Foundation Build can validate the layers in a modeling project that was created in Visual Studio 2013.

If the same version of the runtime or tools is installed on the computers that are running Visual Studio 2013, Visual Studio 2012, or Visual Studio 2010 SP1, you can open this project in all three versions.

This project can't be opened in Visual Studio 2013 because it doesn't support that project type. We recommend that you use InstallShield Limited Edition for Visual Studio (ISLE), a free deployment solution that directly supports most Windows platforms and application runtimes. You can also use ISLE to import data and settings from Visual Studio Installer projects. .

If you upgrade the project to target Office 2013 and the .NET Framework 4, you can open this project in Visual Studio 2013, Visual Studio 2012, or Visual Studio 2010 SP1.

If the project targets the .NET Framework 4, you can open it in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1. All other projects require a one-way upgrade.

If you upgrade the project, you can open it in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1.

This project can't be opened in Visual Studio 2013. However, if you manually upgrade the project to SharePoint 2010, you can open it in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1. For more information about how to upgrade SharePoint 2007, see [Migrating from SharePoint 2007 to SharePoint 2010 for the IT Pro](#), [Migrating a 2007 Workflow to Visual Studio & SharePoint 2010](#), and [SharePoint Enterprise Search Migration Tool for SharePoint Server 2010](#).

You can open the project in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1.

If you allow Visual Studio to upgrade the project to WPF 4.5/Silverlight 5, you can open it in Visual Studio 2012 and Visual Studio 2013.

You can open the project in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1. If you have a database file (.mdf) that was created in an earlier version of SQL Server, you must upgrade it to SQL Server 2012 before you can use it with SQL Server Express LocalDB, but the database is no longer compatible with earlier versions of SQL Server. If you don't upgrade, you can continue to work with the database in Visual Studio 2013 by installing and using SQL Server 2008 Express on the same computer. For more information,

Modeling

MPI/Cluster
Debugging

MSI setup
(.vdproj)

Office 2007 VSTO

Office 2010 VSTO

Rich Internet
Applications

SharePoint 2007

SharePoint 2010

SketchFlow

SQL Server 2008
Express database

see [How to: Upgrade to LocalDB or Continue with SQL Server Express](#).

SQL Server 2008 R2 Express	<p>If SQL Server 2008 R2 Express is installed on the computers that are running Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1, you can open the project in all three versions.</p> <p>You can open the project in Visual Studio 2013 and Visual Studio 2012. For local mode only (that is, when not connected to SQL Server), you won't get the design-time experience for controls that are associated with the viewer in Visual Studio 2010, but the project will function correctly at runtime.</p>
SQL Server Report Project	<p>Warning</p> <p>If you add a feature that's specific to Visual Studio 2013, the report schema is upgraded automatically and you can no longer open the project in Visual Studio 2012.</p>
Unit tests	<p>You can use Microsoft Test Manager in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1 to open tests that were created in any of these versions.</p>
Visual C++	<p>You can use Visual Studio 2013 to open a C++ project that was created in Visual Studio 2012 or Visual Studio 2010 SP1. If you want to use the Visual Studio 2013 build environment to build a project that was created in Visual Studio 2012, you must have both versions of Visual Studio installed on the same computer. For more information, see How to: Upgrade Visual C++ Projects to Visual Studio 2013.</p>
Visual Studio 2010 web	<p>If you allow Visual Studio to upgrade the project automatically, you can open it in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1.</p> <p>If you convert the project to a SQL Server Data Tools Database project, you can open it in Visual Studio 2013. However, Visual Studio 2013 doesn't support these artifacts:</p> <ul style="list-style-type: none">• unit tests• data-generation plans• data-comparison files
Visual Studio 2010 Database (.dbproj)	<ul style="list-style-type: none">• custom rule extensions for static code analysis• server.sqlsettings• .sqlcmd files• custom deployment extensions• partial projects (.files)
Visual Studio 2010 Visual Database Tools	<p>If you install SQL Server Data Tools, you can open the project in Visual Studio 2010 SP1 after the conversion. For more information, see Microsoft SQL Server Data Tools.</p>
Visual Studio Lab	<p>You can open this project in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1.</p> <p>You can use Microsoft Test Manager, Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1 to open environments that were created in any of these versions. However,</p>

Management	the version of Microsoft Test Manager must match the version of Team Foundation Server before you can create environments.
Visual Studio Macro	This project can't be opened in Visual Studio 2013 because it doesn't support the project type.
Visual Studio SDK/VSIX	After you upgrade a Visual Studio SDK project to Visual Studio 2013, it can't be opened in Visual Studio 2012. For more information, see How to: Migrate VSPackages to Visual Studio 2013 .
Microsoft Azure Tools for Visual Studio, including Cloud Service projects (.ccproj) and Azure Resource Manager projects (.deployproj).	If you're using Microsoft Azure Tools for Visual Studio version 2.1, you can open the project in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1. For projects that target earlier versions, if you allow Visual Studio to upgrade the project to version 2.1, you can open it in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1. For Azure Resource Manager projects (.deployproj), you need Azure SDK for .NET 2.6 .
Windows Communication Foundation, Windows Presentation Foundation	You can open this project in Visual Studio 2013, Visual Studio 2012, and Visual Studio 2010 SP1.
Windows Mobile	This project can't be opened in Visual Studio 2013 because it doesn't support the project type.
Windows Phone 7.1	If you allow Visual Studio to upgrade the project to Windows Phone 8.0, you can open it in Visual Studio 2012 and Visual Studio 2013.
Other	You can open most other types of projects in Visual Studio 2012, Visual Studio 2013, and Visual Studio 2010 SP1.
FrontPage Web Sites	This project can't be opened in Visual Studio 2013 because it doesn't support the project type. If you allow Visual Studio to update the project automatically, you can open it in Visual Studio 2013, Visual Studio 2012, or Visual Studio 2010 SP1.
Portable Class Library	<ul style="list-style-type: none"> • Projects that targeted Silverlight 4 will target Silverlight 5. • Projects that targeted Windows Phone 7.0 or Windows Phone 7.5 will target Windows Phone 8. • Projects that targeted Xbox 360 will no longer target Xbox 360.

Troubleshooting project compatibility issues

Here are some things you can do when a project won't open in Visual Studio 2013:

- If you try to open a project that isn't supported in Visual Studio 2013 and for which the associated version of Visual Studio isn't installed, a message that the project type isn't supported might appear and the project type might be listed in the **Review Project and Solution Changes** dialog box under **Unsupported projects**. To resolve this issue, open the programs and features page in the **Windows Control Panel**, select **Visual Studio**, and then choose **Change, Repair**. Then you can install the missing version.
- If you try to open a project for a desktop app in Visual Studio Express 2013 for Windows, an error occurs and one of these messages is displayed: "This edition of Visual Studio only supports Windows 8.1 apps" or "This project is incompatible with the current edition of Visual Studio." Visual Studio Express 2013 for Windows is restricted to the development, testing, and deployment of Windows Store apps designed for Windows 8.1. To open a desktop app project, you must use an edition of Visual Studio that supports that project type.
For more information about the Visual Studio editions, see [Microsoft Visual Studio Products](#)
- If you try to open a Windows Store App project in Visual Studio Express 2013 for Windows Desktop, an error occurs. Visual Studio Express 2013 for Windows Desktop cannot be used to build Windows Store apps. If you want to build Windows Store apps, you can also install Visual Studio Express 2013 for Windows. Or, to develop apps for all Microsoft platforms and the web, try Visual Studio Professional 2013.
- If a project requires features that are specific to Visual Studio 2013, it can't be opened in an earlier version.
- If you're using Visual Studio 2012 and you want to open a project that was created in Visual Studio 2013, you might be able to customize the project system to incorporate features of Visual Studio 2013. For information about how to do this, see [How to: Modify a Project System So That Projects Load in Multiple Versions of Visual Studio](#).

For additional troubleshooting information, see the [Visual Studio 2013 Compatibility](#) KB article.

Files

The following list identifies whether Visual Studio 2013 supports each type of file, whether you can open the file in Visual Studio 2012 and Visual Studio 2010 SP1, and whether you have to modify it to ensure compatibility.

Type of File	Compatibility
AppManifest, Inbrowsersettings, OutOfBrowserSettingsSP1. (.xml files)	You can open these files in Visual Studio 2012, Visual Studio 2013, and Visual Studio 2010
BizTalk flat file	You can add these schemas to a BizTalk 2013 project in Visual Studio 2013. To use Visual Studio 2013 with BizTalk 2010 projects that have flat file schemas, install BizTalk 2013 on

schemas	the computer that has Visual Studio 2013. The first time you open the BizTalk 2010 project, it is automatically upgraded to the BizTalk 2013 or Visual Studio 2013 project system.
Client Report Definition (.rdlc) files	You can open these files in Visual Studio 2013, and the schema is automatically upgraded if you add Visual Studio 2013 features and controls.
Code analysis rule sets	You can open these files in Visual Studio 2012, Visual Studio 2013, and Visual Studio 2010 SP1.
Data-tier application package files	You can open these files in Visual Studio 2013 if they're version 2.0 or version 2.5.
Debugger dump files	You can open these files in Visual Studio 2012, Visual Studio 2013, and Visual Studio 2010 SP1.
Directed Graph Markup Language (DGML) diagram files	You can open these files in Visual Studio 2012, Visual Studio 2013, and Visual Studio 2010 SP1 without changing the file.
Entity Data Model (EDMX) files	In Visual Studio 2013, you can open an EDMX file that targets the .NET Framework 4.5 or the .NET Framework 4 without changing the file.
Profiler report files	You can open Profiler report files (.vsp, .vsps, .psess, and .vspf) in Visual Studio 2012 and Visual Studio 2013. A .vspx file can't be opened in Visual Studio 2010 SP1.
Solution (.suo) file	You can use Visual Studio 2013 to open a solution file that was created in Visual Studio 2012 or Visual Studio 2010 SP1
SQL Server Compact Edition	Visual Studio 2013 doesn't support SQL Server Compact Edition.
SQLX files	To open these files in Visual Studio 2013, you must perform a one-way upgrade, deploy the .sqlx file on the target version of Visual Studio, and then rebuild the file in the .dacpac format.
IntelliTrace log files from Visual Studio 2010	You can open these files in Visual Studio 2012, Visual Studio 2013, and Visual Studio 2010 SP1.
JavaScript Memory Analyzer (.diagsession) files	Files created by older versions of Visual Studio can be viewed in Visual Studio 2013. However, depending on the information gathered, files created in Visual Studio 2013 may not open in Visual Studio 2012 or Visual Studio 2010 SP1.

Integration assets

You might encounter compatibility issues if you use clients and servers from different versions of Visual Studio Team Foundation Server.

Kind of Integration Compatibility

Code Review and My Work The Code Review and My Work features won't work if you connect a client of Team Foundation to Visual Studio Team Foundation Server 2010.

Visual Studio Express A 64-bit environment such as MSBuild or Team Foundation Build can't be used to build 2012 for Windows 8 Windows Store apps that are created in Visual Studio Express 2013 for Windows.

See Also

Tasks

[How to: Modify a Project System So That Projects Load in Multiple Versions of Visual Studio](#)

□

□ Theme

[Previous Version Docs](#) [Blog](#) [Contribute](#) [Privacy & Cookies](#) [Terms of Use](#) [Site Feedback](#) [Trademarks](#)

© Microsoft 2020

```
#ifndef _MIMIC_ACTION_HH_
#define _MIMIC_ACTION_HH_
```

```
/*
Copyright (c) 2002 by Gambit Communications, Inc.
All Rights Reserved
*/
```

```
/*
 * This file publishes the MIMIC action API for creating C/C++
 * action modules that can loaded by the simulator.
 */
```

```
#include "mimic_exports.hh"
#include "mimic_api.hh"
```

```
/*
 * The following define is used to verify the build version of a DLL
 * with respect to the simulator. This ensures that the two sides are
 * using the same API definition. This ID will be changed when any of
 * the API message identifiers, data structures, etc. change.
 */
```

```
#define ACTION_BUILD_ID          "1.00"
```

```
/*-----*/
```

```
/* buffer sizes */
```

```
/*-----*/
```

```
#define SML_BUF_SIZE          (1024/4)
#define MED_BUF_SIZE         (1*1024)
#define LRG_BUF_SIZE         (8*1024)
#define HUG_BUF_SIZE         (31*1024)
```

```
/*-----*/
```

```
/* msgid passed to 'process_message()' */
```

```
/*-----*/
```

```
#define ACTION_MODULE_REGISTER      0x1001
#define ACTION_MODULE_INIT          0x1002
#define ACTION_MODULE_UNINIT        0x1003
#define ACTION_MODULE_RUN            0x1004
```

```
/*-----*/
```

```
/* return values from 'process_message()' */
```

```
/*-----*/
```

```
#define ACTION_FAILURE            -1
#define ACTION_SUCCESS            0
```

```
/*-----*/
```

```
/* structures used for module messaging */
```

```
/*-----*/
```

```
typedef struct _action_generic_t  
{  
    Mimic::DynStore    ret_msg;    // OUT - dynamic return message  
    Mimic::Session    *session;    // IN/OUT - C++ API  
} action_generic_t;
```

```
typedef struct _action_module_register_t  
{  
    Mimic::DynStore    ret_msg;    // OUT - dynamic return message  
    Mimic::Session    *session;    // IN/OUT - C++ API  
    Mimic::DynStore    version;    // OUT - build ID verification  
} action_module_register_t;
```

```
typedef struct _action_module_init_t  
{  
    Mimic::DynStore    ret_msg;    // OUT - dynamic return message  
    Mimic::Session    *session;    // IN/OUT - C++ API  
} action_module_init_t;
```

```
typedef struct _action_module_uninit_t  
{  
    Mimic::DynStore    ret_msg;    // OUT - dynamic return message  
    Mimic::Session    *session;    // IN/OUT - C++ API  
} action_module_uninit_t;
```

```
typedef struct _action_module_run_t  
{  
    Mimic::DynStore    ret_msg;    // OUT - dynamic return message  
    Mimic::Session    *session;    // IN/OUT - C++ API  
    Mimic::ActionGlobals *globals;    // IN/OUT - global variables  
} action_module_run_t;
```

```
#endif /* _MIMIC_ACTION_HH_ */
```

```
#ifndef _MIMIC_AGENT_HH
#define _MIMIC_AGENT_HH

////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////////////////////////////////
```

```
#include <algorithm>
```

```
#include "mimic_errorinfo.hh"
#include "mimic_valuespace.hh"
```

```
/** agent's state : running */
#define AGENT_RUNNING 1
/** agent's state : stopped */
#define AGENT_STOPPED 2
/** agent's state : halted */
#define AGENT_HALTED 3
/** agent's state : paused */
#define AGENT_PAUSED 4
/** agent's state : not configured */
#define AGENT_NOT_CONFIG 5
/** agent's state : stopping (or starting) */
#define AGENT_STOPPING 6
```

```
namespace Mimic {
class Agent
{
private:
    // POLICY: disallow copy and assignment
    Agent (const Agent& other);
    Agent& operator = (const Agent& other);
public :
    //member functions
    //Creates an agent object for a given session.
    Agent (int agent_no,Session* session);

    // state change functions
    //Start agent (from stopped)
    void start () throw (ErrorInfo);

    //Stop agent (from running, paused, halted)
    void stop () throw (ErrorInfo);

    //Pause agent at given time (from running)
    void pause_at ( int time ) throw (ErrorInfo);

    //Pause agent now (from running)
    void pause_now() throw (ErrorInfo);

    //Halt agent (from running)
    void halt () throw (ErrorInfo);
```

```
//Resume agent (from paused/halted)
void resume () throw (ErrorInfo);

// config functions
// Configure agent
void configure ( std::string address, std::list<std::string>& mibs ) throw (ErrorInfo);

// Delete a configured agent
void remove () throw (ErrorInfo);

// get functions
//Get agent's number
int get_agent_no ();

// Get agent's session
Session* get_session () throw (ErrorInfo);

// Get the interface for the primary ipalias
std::string get_interface () throw (ErrorInfo);

//Get the primary ip address
std::string get_host () throw (ErrorInfo);

// Get the primary subnet mask
std::string get_mask () throw (ErrorInfo);

// Get the port number
int get_port () throw (ErrorInfo);

// Get the read community
std::string get_read_community () throw (ErrorInfo);

//Get the write community
std::string get_write_community () throw (ErrorInfo);

//Get the MIB triplets
void get_mibs (std::list<std::string>& mibs) throw (ErrorInfo);

//Get the simulation name
std::string get_sim () throw (ErrorInfo);

//Get the simulation scenario
std::string get_scen () throw (ErrorInfo);

//Get the maximum PDU size
int get_pdu_size () throw (ErrorInfo);

//Get the list of protocols
void get_protocols (std::list<std::string>& protocols) throw (ErrorInfo);

//Get the delay
int get_delay () throw (ErrorInfo);
```

```
//Get the start time
int get_starttime () throw (ErrorInfo);

//Get the agent state
int get_state ()throw (ErrorInfo);

//Get the list of statistics
void get_statistics (std::list<std::string>& stats) throw (ErrorInfo);

// this is OBSOLETE
void get_statistics (std::list<int>& stats) throw (ErrorInfo);

//Get the drop rate
int get_drops () throw (ErrorInfo);

//Get the changed flag
int get_changed () throw (ErrorInfo);

//Get the config changed flag
int get_config_changed () throw (ErrorInfo);

//Get the state changed flag
int get_state_changed () throw (ErrorInfo);

//Get the trace flag
int get_trace () throw (ErrorInfo);

//Get the validate flag
int get_validate () throw (ErrorInfo);

//Get the owner
std::string get_owner () throw (ErrorInfo);

//Get the private directory
std::string get_privdir () throw (ErrorInfo);

//Get the OidDir filename
std::string get_oiddir () throw (ErrorInfo);

//Get the inform timeout
int get_inform_timeout () throw (ErrorInfo);

//Get the inform retries
int get_inform_retries () throw (ErrorInfo);

//Get the trap list
void trap_list (std::list<std::string>& trap) throw (ErrorInfo);

//Get the trap destination list
void trap_config_list (std::list<std::string>& trap) throw (ErrorInfo);

//Set the trap destination
void trap_config_add (std::string address,int port) throw (ErrorInfo);
```

```
//Remove the trap destination from list
void trap_config_del (std::string address,int port) throw (ErrorInfo);

//set functions
//Set the interface for primary ipalias
void set_interface ( std::string iface ) throw (ErrorInfo);

//Set the primary ip address
void set_host ( std::string host ) throw (ErrorInfo);

//Set the primary subnet mask
void set_mask (std::string mask ) throw (ErrorInfo);

//Set the port number
void set_port ( int port ) throw (ErrorInfo);

//Set the read community
void set_read_community ( std::string readcomm ) throw (ErrorInfo);

//Set the write community
void set_write_community ( std::string writecomm ) throw (ErrorInfo);

//Set the list of protocols supported
void set_protocols ( std::list<std::string>& protocols ) throw (ErrorInfo);

//Set the delay
void set_delay ( int delay ) throw (ErrorInfo);

//Set the starttime
void set_starttime ( int starttime ) throw (ErrorInfo);

//Set the MIB triplets
void set_mibs ( std::list<std::string>& mibs ) throw (ErrorInfo);

//Set the trace flag
void set_trace ( int flag ) throw (ErrorInfo);

//Set the maximum PDU size
void set_pdu_size ( int pdu_size ) throw (ErrorInfo);

//Set the drop rate
void set_drops ( int drops ) throw (ErrorInfo);

//Set the validate flag
void set_validate ( int flag ) throw (ErrorInfo);

//Set the OidDir filename
void set_oiddir ( std::string oiddir ) throw (ErrorInfo);

//Set the inform timeout
void set_inform_timeout (int timeout) throw (ErrorInfo);

//Set the inform retries
```

```
void set_inform_retries (int retries) throw (ErrorInfo);

//persistence
void save () throw (ErrorInfo);

//IPAlias functions
//Add a new ipalias
void add_ipalias ( std::string address, int port, std::string mask, std::string iface ) throw (ErrorInfo);

//Delete a configured ipalias
void del_ipalias ( std::string address, int port ) throw (ErrorInfo);

//Start the configured ipalias
void start_ipalias ( std::string address, int port ) throw (ErrorInfo);

//Stop the configured ipalias
void stop_ipalias ( std::string address, int port ) throw (ErrorInfo);

//Get status of ipalias
int status_ipalias ( std::string address, int port ) throw (ErrorInfo);

//List the ipaliases for the agent
void list_ipaliases (std::list<std::string>& ipaliases) throw (ErrorInfo);

//Add (and start) timerscript with given interval and arguments
void add_timer_script (std::string script,int interval,std::string args = "") throw (ErrorInfo);

//Delete a running script with specified name
void del_timer_script (std::string script) throw (ErrorInfo);

//List the currently running timer scripts
void list_timer_scripts (std::list<std::string>& scripts) throw (ErrorInfo);

//protocol messages
std::string protocol_msg ( std::string protocol, std::string msg ) throw (ErrorInfo);

//valuespace access
Valuespace& get_valuespace () ;

//system get
std::string get_state_descr (int state);

//Set (and start) storage with a given variable and value
void agent_store_set (std::string var,std::string val,int persist = 0) throw (ErrorInfo);

//Append (and start) storage with a given variable and value
void agent_store_append (std::string var,std::string val,int persist = 0) throw (ErrorInfo);

//Unset the variable
void agent_store_unset(std::string var) throw (ErrorInfo);

//List variables
void agent_store_list(std::list<std::string>& store_list) throw (ErrorInfo);
```



```
//Examine existance
int agent_store_exists(std::string var) throw (ErrorInfo);

//Examine persistence
int agent_store_persists(std::string var) throw (ErrorInfo);

//get value
std::string agent_store_get(std::string var) throw (ErrorInfo);

//Get from entry list
void from_list(std::list<std::string>& from) throw (ErrorInfo);

//Add from entry
void from_add(std::string host, int port) throw (ErrorInfo);

//Delete from entry
void from_del(std::string host, int port) throw (ErrorInfo);

//Destructor
~Agent();
private :

//data members
int _agent_no;
Session *_session;
Valuespace *_valuespace;

//member functions
std::string agent_dispatch_msg(int req_id,std::string req_msg) throw (ErrorInfo);
};
}
#endif
```

```
#ifndef _MIMIC_API_HH
#define _MIMIC_API_HH
```

```
////////////////////////////////////
// Copyright (c) 1995-2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#include "mimic_client.hh"
#include "mimic_session.hh"
#include "mimic_apiutils.hh"
```

```
#endif
```

```
#ifndef _MIMIC_APIUTILS_HH
#define _MIMIC_APIUTILS_HH
/*****
Copyright (c) 1995-2002 by Gambit Communications, Inc.
All Rights Reserved
*****/
```

```
/*
 * mimic_apiutils.hh
 *
 */
//-----
```

```
#include <list>
#include <exception>
#include <iterator>
#include <string>
#include <iostream>
#include <iterator>
```

```
#include "mimic_errorinfo.hh"
```

```
////////////////////////////////////n
// Description:
//   Converts string to integer .
// Return values:
//   Converted number on suces,-1 on error
// Side effects:
//   NONE
////////////////////////////////////
```

```
int parse_int ( std::string str);
```

```
////////////////////////////////////
// Description:
//   Converts integer to string.
// Return values:
//   Converted string
// Side effects:
//   NONE
////////////////////////////////////
```

```
std::string valueof (int number);
```

```
////////////////////////////////////
// Description:
//   Creates a string from list, separating entries using
//   the separator string provided.
// Return values:
//   string of concatenated tokens.
// Side effects:
//   NONE
////////////////////////////////////
```

```
std::string linkedlist_to_string ( std::list<int>& intlist, std::string separator);
```

```
////////////////////////////////////
```

```
// Description:
```

```
//   Creates a linked list of tokens from passed string.
```

```
// Return values:
```

```
//   list of String tokens.
```

```
// Side effects:
```

```
//   NONE
```

```
////////////////////////////////////
```

```
std::list<std::string> split_to_stringlist ( std::string str, char separator = ' ' )  
    throw (Mimic::ErrorInfo);
```

```
////////////////////////////////////
```

```
// Description:
```

```
//   Creates a linked list of Integers from passed String.
```

```
// Return values:
```

```
//   list of Integer tokens.
```

```
// Side effects:
```

```
//   NONE
```

```
////////////////////////////////////
```

```
std::list<int> split_to_integerlist ( std::string str ) throw(Mimic::ErrorInfo);
```

```
std::list<unsigned long long> split_to_unsigned_long_long_list ( std::string str ) throw(Mimic::ErrorInfo);
```

```
////////////////////////////////////
```

```
// Description:
```

```
//   Creates a linked list of positive Longs from passed String.
```

```
// Return values:
```

```
//   list of Integer tokens.
```

```
// Side effects:
```

```
//   NONE
```

```
////////////////////////////////////
```

```
std::list<unsigned long> split_to_oidsubidlist ( std::string str ) throw (Mimic::ErrorInfo);
```

```
template <class InputIterator>
```

```
InputIterator
```

```
get(
```

```
    InputIterator first,
```

```
    int index
```

```
)
```

```
{
```

```
    if (index == 0)
```

```
    ;
```

```
    else
```

```
    {
```

```
        for (int i = 0; i < index; ++i)
```

```
        {
```

```
            ++first;
```

```
        }
```

```
    }
```

```
    return first;
```

```
}

/////////////////////////////////////////////////////////////////
// Description:
//   Sets a sublist within a list of lists at perticular index.
// Return values:
//   NONE
// Side effects:
//   NONE
/////////////////////////////////////////////////////////////////
template <class T>
void set_list(
    std::list< std::list<T> >& main_list,
    std::list<T>& sub_list,
    int index
)
{
    main_list.insert(get(main_list.begin(),index),sub_list);
}

#endif
```

```
#ifndef _MIMIC_CLIENT_HH
#define _MIMIC_CLIENT_HH
```

```
////////////////////////////////////
// Copyright (c) 1995-2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#include <string>
#include <iostream>
#include <exception>
#include <list>
```

```
#include "mimic_errorinfo.hh"
```

```
namespace Mimic
```

```
{
class Session;
class Client
```

```
{
public:
//Creates an default MIMIC client object
Client();
~Client();
```

```
//Opens a new session to the host
Session& open_session(std::string host = "localhost", long port = 9797) throw (ErrorInfo);
Session& open_session(std::string host = "localhost", void* port = NULL) throw (ErrorInfo);
```

```
//Closes an already open session
void close_session(Session& session) throw (ErrorInfo);
```

```
//returns list of all the open sessions
std::list<Session*>& list_sessions ();
```

```
//diagnostic dump
void dump ();
```

```
private:
// data members
std::list<Session*> _sessions;
```

```
//disallow copy & assignment
Client& operator=(const Client& other);
Client(const Client& other);
```

```
};
}
#endif
```

```

#ifndef _ERRORINFO_HH
#define _ERRORINFO_HH

////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2001 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////////////////////////////////
#include <list>
#include <string>
#include <string.h>

#define MAX_BUFF_SIZE 4096
/** returned by get_error_code() indicating that some error occurred */
#define MIMIC_SUCCESS 0
/** returned by get_error_code() indicating that no error occurred */
#define MIMIC_FAILURE -1

/** returned by get_error_code() indicating that no error occurred */
#define MIMIC_WARN 1

#define API_APPEND_DATE_FILENAME() \
    date_time_append (tmp,msg.c_str(),filename.c_str())

namespace Mimic {
class ErrorInfo
{
public :

    //Creates the ErrorInfo exception object thrown by the
    //Mimic package on error conditions.
    ErrorInfo (int err_code,std::string err_msg);

    //copy constructor
    ErrorInfo (const Mimic::ErrorInfo& other);

    //Assingment operator
    ErrorInfo& operator= (const Mimic::ErrorInfo& other);

    //Get the error code for this exception
    int get_error_code ();

    // Gets the error message string for this exception.
    std::string get_error_msg ();

    //Prepend error strings at each level as the stack unwinds.
    void add (std::string msg,std::string filename);

    //Destructor
    ~ErrorInfo ();
private:
    //data members
    int _code;

```

```
std::list<std::string> _msg;
```

```
//to add datetime
```

```
void date_time_append (char tmp[],const char* msg,const char* filename);
```

```
};  
}
```

```
#endif
```



```
#ifndef _MIMIC_EXPORTS_H_
#define _MIMIC_EXPORTS_H_
```

```
/*
Copyright (c) 2002 by Gambit Communications, Inc.
All Rights Reserved
*/
```

```
/*
 * This header exports wrappers to various libgci utility classes
 * which are useful while writing C/C++ action modules. This would
 * not pull in any internal headers i.e. use forward declares.
 */
```

```
//
// forward declarations of internal classes.
// NOTE: avoids pulling in their declarative headers
//
class DynStore;
class ActionGlobals;
```

```
//
// namespace "Mimic" that is used to expose internal libgci
// classes using thin class wrappers.
//
#ifdef __cplusplus
namespace Mimic {
#endif
```

```
//
// exposes parts of libgci "DynStore"
//
class DynStore
{
public:
    DynStore ();
    ~DynStore ();

    int clear ();

    int sprintf (          // NOTE: 16K at a time
        const char* fmt,
        ...
    );
    int append (
        const char* buf
    );
    int append (
        const char* buf,
        int n
    );
};
```

```

int prepend (
    const char* buf
);

char* get_string () const;
int get_length () const;

private:
    // POLICY disallow copy and assignment by default.
    DynStore ( const DynStore& d );
    DynStore& operator= ( const DynStore& d );

private:
    ::DynStore *_dynstore;
};

//
// exposes parts of libgci "ActionGlobals"
//
class ActionGlobals
{
public:
    ActionGlobals (::ActionGlobals *g);
    ~ActionGlobals ();

    char* get (
        const char *name
    ) const;

    int set (
        const char *name,
        const char *value
    );

    int get_vars(
        DynStore& vars
    );

private:
    // POLICY disallow copy and assignment by default.
    ActionGlobals ( const ActionGlobals& g );
    ActionGlobals& operator= ( const ActionGlobals& g );

private:
    ::ActionGlobals *_globals;
};

#ifdef __cplusplus
} // namespace
#endif

#endif // _MIMIC_EXPORTS_H_

```



```
#ifndef _MIMIC_HELPER_HH
#define _MIMIC_HELPER_HH
```

```
////////////////////////////////////
// Copyright (c) 1995-2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#include "mimic_api.hh"
```

```
void wait_for_agent_state (Mimic::Session* session, int agent, int state) throw (Mimic::ErrorInfo);
```

```
#endif
```

```

#ifndef _MIMIC_SESSION_HH
#define _MIMIC_SESSION_HH

////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////////////////////////////////

#include <string>
#include <iostream>
#include <exception>
#include <list>
#include <algorithm>

#include "mimic_errorinfo.hh"
#include "mimic_agent.hh"
#include "mimic_client.hh"

extern void mimic_debug_runtime_load ();

class CriticalRegion;
class MgmtTransport;
namespace Mimic {
class Session
{
private:
    // POLICY: disallow copy and assignment
    Session (const Session& other);
    Session& operator = (const Session& other);
public:
    //Gets the host address of the MIMIC daemon
    std::string get_host ();

    //Gets the management port number of the MIMIC daemon
    long get_port ();

    //Gets the username for the session
    std::string get_user ();

    //Gets the private directory for the session
    std::string get_privdir ();

    //Connects to the MIMIC daemon at the configured host and port
    void connect () throw (ErrorInfo);

    //Disconnects from the MIMIC daemon
    void disconnect () throw (ErrorInfo);

    //Load a new blank configuration
    void cfg_new (std::list<int>* agents = NULL) throw (ErrorInfo);

    //Loads the specified configuration file

```

```

void cfg_load (std::string file,std::list<int>* agents = NULL, int start=0, int merge=0) throw (ErrorInfo);

//Saves the current configuration
void cfg_save (std::list<int>* agents = NULL) throw (ErrorInfo);

//Saves the current configuration as specified filename
void cfg_saveas (std::string file,std::list<int>* agents = NULL) throw (ErrorInfo);

//Terminates the connected MIMIC daemon
void terminate () throw (ErrorInfo);

//Starts all configured agents
void start_all_agents () throw (ErrorInfo);

//Stops all running agents
void stop_all_agents () throw (ErrorInfo);

//Gets the agent object
Agent& get_agent (int agentno) throw (ErrorInfo);

//Get maximum agents for session
int get_max () throw (ErrorInfo);

//Get the last configured agent
int get_last () throw (ErrorInfo);

//Get the current MIMIC daemon version
std::string get_version () throw (ErrorInfo);

//Get the number of connected clients
int get_clients () throw (ErrorInfo);

//Get the currently loaded configuration file
std::string get_cfgfile () throw (ErrorInfo);

//Get the list of configured agents
void get_configured_list (std::list<int>& agentid) throw (ErrorInfo);

//Get the list of running agents
void get_active_list (std::list<int>& agentid) throw (ErrorInfo);

//Get list of running agents with their statistics
void get_active_data_list (std::list< std::list<unsigned long long> >& agentstats) throw (ErrorInfo);
// OBSOLETE
void get_active_data_list (std::list< std::list<int> >& agentstats) throw (ErrorInfo);

//Get list of agents whose configuration has changed
void get_changed_config_list (std::list<int>& agentid) throw (ErrorInfo);

//Get a list of agents whose state has changed
void get_changed_state_list (std::list< std::list<int> >& agentstate) throw (ErrorInfo);

//Get the current logfile
std::string get_log () throw (ErrorInfo);

```

```
//Get the interfaces
void get_interfaces (std::list<std::string>& iface) throw (ErrorInfo);

//Get the simulator's private directory
std::string get_sim_privdir () throw (ErrorInfo);

//Get the flag indicating if config file changed
int get_cfg_file_changed () throw (ErrorInfo);

//Get all the protocols supported by the simulator
void get_protocols (std::list<std::string>& protocols) throw (ErrorInfo);

//Get the licensed product
int get_product () throw (ErrorInfo);

//Get the Mimic server host network address
std::string get_netaddr () throw (ErrorInfo);

//Get the Mimic server host network device
std::string get_netdev () throw (ErrorInfo);

//Get the Mimic server license expiration
std::string get_expiration (int days) throw (ErrorInfo);

//Get the Mimic licensing
std::string get_licensing () throw (ErrorInfo);

//Get Mimic thread info
void get_threads (std::list <std::list<std::string> >& threads) throw (ErrorInfo);

//Set a new logfile
void set_log (std::string logfile) throw (ErrorInfo);

//Add (and start) timerscript with given interval and arguments
void add_timer_script (std::string script,int interval,std::string args = "") throw (ErrorInfo);

//Delete a running script with specified name
void del_timer_script (std::string script) throw (ErrorInfo);

//List the currently running timer scripts
void list_timer_scripts (std::list<std::string>& scripts) throw (ErrorInfo);

//Pass session's message string to the protocol
std::string protocol_msg (std::string protocol,std::string msg) throw (ErrorInfo);

//Execute specified diagnostic code (used only for debugging)
void diag (int diagid) throw (ErrorInfo);

//Mget
std::list<std::string> mget(std::list<std::string>& cmds)throw(ErrorInfo);

//Get interfaces for specified agents
void agent_mget_interface (std::list<int>& agents,std::list<std::string>& iface) throw (ErrorInfo);
```

```
//Get primary addresses for specified agents
void agent_mget_host (std::list<int>& agents,std::list<std::string>& host) throw (ErrorInfo);

//Get masks for specified agents
void agent_mget_mask (std::list<int>& agents,std::list<std::string>& mask) throw (ErrorInfo);

//Get port numbers for specified agents
void agent_mget_port (std::list<int>& agents,std::list<int>& port) throw (ErrorInfo);

//Get read communities for specified agents
void agent_mget_read_community (std::list<int>& agents,std::list<std::string>& readcomm) throw (ErrorInfo);

//Get write communities for specified agents
void agent_mget_write_community (std::list<int>& agents,std::list<std::string>& writecomm) throw (ErrorInfo);

//Get MIB triplets for specified agents
void agent_mget_mibs (std::list<int>& agents,std::list< std::list<std::string> >& mibs) throw (ErrorInfo);

//Get simulation name for specified agents
void agent_mget_sim (std::list<int>& agents,std::list<std::string>& sim) throw (ErrorInfo);

//Get simulation scenario for specified agents
void agent_mget_scen (std::list<int>& agents,std::list<std::string>& scen) throw (ErrorInfo);

//Get pdu sizes for specified agents
void agent_mget_pdu_size (std::list<int>& agents,std::list<int>& pdu_size) throw (ErrorInfo);

//Get protocols for specified agents
void agent_mget_protocols (std::list<int>& agents,std::list< std::list<std::string> >& protocols) throw (ErrorInfo);

//Get delays for specified agents
void agent_mget_delay (std::list<int>& agents,std::list<int>& delays) throw (ErrorInfo);

//Get starttimes for specified agents
void agent_mget_starttime (std::list<int>& agents,std::list<int>& starttimes) throw (ErrorInfo);

//Get states for specified agents
void agent_mget_state (std::list<int>& agents,std::list<int>& state) throw (ErrorInfo);

//Get statistics for specified agents
void agent_mget_statistics (std::list<int>& agents,std::list<std::string>& stats) throw (ErrorInfo);

//Get drops for specified agents
void agent_mget_drops (std::list<int>& agents,std::list<int>& drops) throw (ErrorInfo);

//Get changed flags for specified agents
void agent_mget_changed (std::list<int>& agents,std::list<int>& flags) throw (ErrorInfo);

//Get config_changed flags for specified agents
void agent_mget_config_changed (std::list<int>& agents,std::list<int>& flags) throw (ErrorInfo);

//Get state_changed flags for specified agents
void agent_mget_state_changed (std::list<int>& agents,std::list<int>& flags) throw (ErrorInfo);
```



```
//Get trace flags for specified agents
void agent_mget_trace (std::list<int>& agents,std::list<int>& flags) throw (ErrorInfo);

//Get validate flags for specified agents
void agent_mget_validate (std::list<int>& agents,std::list<int>& flags) throw (ErrorInfo);

//Get owners for specified agents
void agent_mget_owner (std::list<int>& agents,std::list<std::string>& owner) throw (ErrorInfo);

//Get privdirs for specified agents
void agent_mget_privdir (std::list<int>& agents,std::list<std::string>& privdir) throw (ErrorInfo);

//Get OidDir filenames for specified agents
void agent_mget_oiddir (std::list<int>& agents,std::list<std::string>& oiddir) throw (ErrorInfo);

//Get inform timeout for specified agents
void agent_mget_inform_timeout (std::list<int>& agents,std::list<int>& timeout) throw (ErrorInfo);

//Get inform retries for specified agents
void agent_mget_inform_retries (std::list<int>& agents,std::list<int>& retries) throw (ErrorInfo);

//Set interfaces for specified agents
void agent_mset_interface (std::list<int>& agents,std::list<std::string>& interfaces) throw (ErrorInfo);

//Set ip addresses for specified agents
void agent_mset_host (std::list<int>& agents,std::list<std::string>& hosts) throw (ErrorInfo);

//Set subnet masks for specified agents
void agent_mset_mask (std::list<int>& agents,std::list<std::string>& masks) throw (ErrorInfo);

//Set port numbers for specified agents
void agent_mset_port (std::list<int>& agents,std::list<int>& ports) throw (ErrorInfo);

//Set read communities for specified agents
void agent_mset_read (std::list<int>& agents,std::list<std::string>& readcomms) throw (ErrorInfo);

//Set write communities for specified agents
void agent_mset_write (std::list<int>& agents,std::list<std::string>& writecomms) throw (ErrorInfo);

//Set protocols for specified agents
void agent_mset_protocol (std::list<int>& agents,std::list<std::string>& protocols) throw (ErrorInfo);

//Set delay for specified agents
void agent_mset_delay (std::list<int>& agents,std::list<int>& delays) throw (ErrorInfo);

//Set start time for specified agents
void agent_mset_starttime (std::list<int>& agents,std::list<int>& starttimes) throw (ErrorInfo);

//Set MIBs for specified agents
void agent_mset_mibs (std::list<int>& agents,std::list<std::string>& mibs) throw (ErrorInfo);

//Set trace flags for specified agents
void agent_mset_trace (std::list<int>& agents,std::list<int>& flags) throw (ErrorInfo);
```

```
//Set PDU size for specified agents
void agent_mset_pdu_size (std::list<int>& agents, std::list<int>& pdu_sizes) throw (ErrorInfo);

//Set drops for specified agents
void agent_mset_drops (std::list<int>& agents, std::list<int>& drops) throw (ErrorInfo);

//Set validate for specified agents
void agent_mset_validate (std::list<int>& agents, std::list<int>& flags) throw (ErrorInfo);

//Set OidDir filename for specified agents
void agent_mset_oiddir (std::list<int>& agents, std::list<std::string>& oiddirs) throw (ErrorInfo);

//Set inform timeout for specified agents
void agent_mset_inform_timeout (std::list<int>& agents, std::list<int>& timeout) throw (ErrorInfo);

//Set inform retries for specified agents
void agent_mset_inform_retries (std::list<int>& agents, std::list<int>& retries) throw (ErrorInfo);

//Debug dump
void dump ();

//Add user's access privilege for agents
void add_access (std::string user, std::string agents, std::string mask) throw (ErrorInfo);

//Delete user's access privilege
void del_access (std::string user) throw (ErrorInfo);

//List accesses assigned to users
void list_access (std::list<std::string>& access_list) throw (ErrorInfo);

//Get mimic access administrator
std::string get_access_adminuser() throw (ErrorInfo);

//Get access administrative directory
std::string get_access_admindir() throw (ErrorInfo);

//Get data base for access control
std::string get_access_acldb() throw (ErrorInfo);

//Get access control enabled value
int get_access_enabled() throw (ErrorInfo);

//Set access administrator.
void set_access_adminuser (std::string user) throw (ErrorInfo);

//Set access administrative directory.
void set_access_admindir (std::string dir) throw (ErrorInfo);

//Set access administrative data base
void set_access_acldb (std::string acldb) throw (ErrorInfo);

//Set access control's enabled value
void set_access_enabled (int arg) throw (ErrorInfo);
```

```
//Save access control file
void save_access (std::string file = "") throw (ErrorInfo);

//Load access control file
void load_access (std::string file = "") throw (ErrorInfo);

//Set (and start) storage with a given variable and value
void store_set (std::string var,std::string val,int persist) throw (ErrorInfo);

//Append (and start) storage with a given variable and value
void store_append (std::string var,std::string val,int persist) throw (ErrorInfo);

//Unset a variable-value pair
void store_unset (std::string var) throw (ErrorInfo);

//List variables
void store_list (std::list<std::string>& store_list) throw (ErrorInfo);

//Examine existance
int store_exists (std::string var) throw (ErrorInfo);

//Examine persistence
int store_persists (std::string var) throw (ErrorInfo);

//get value
std::string store_get (std::string var) throw (ErrorInfo);

//Trap Configure
void trap_config_list (std::list<std::string>& trap_list) throw (ErrorInfo);

//Trap Add
void trap_config_add (std::string destination,int port) throw (ErrorInfo);

//Trap Delete
void trap_config_delete (std::string destination,int port) throw (ErrorInfo);

// Action script support
int action_version () throw (ErrorInfo);

std::string action_auth () throw (ErrorInfo);

std::string action_varbinds () throw (ErrorInfo);

int action_reqid () throw (ErrorInfo);

std::string action_from () throw (ErrorInfo);

std::string action_to () throw (ErrorInfo);

void action_jump (std::string oid) throw (ErrorInfo);

// Destructor
~Session();
```

```

private:
    // data members
    std::string _host;
#ifdef MIMIC_INTERNAL
    void* _internal;
#else
    long _port;
#endif
    MgmtTransport* _socket;
    std::string _user;
    std::string _privdir;
    int _nagents;
    Agent** _agents;

    CriticalRegion* _dispatch_crit;

    Session ();

    //Creates a session for given host and port
    Session (std::string host, long port) throw (ErrorInfo);
    Session (std::string host, void* internal) throw (ErrorInfo);

    //Only Client can instantiate session
    friend Session& Client::open_session(std::string host, long port) throw (ErrorInfo);
    friend Session& Client::open_session(std::string host, void* internal) throw (ErrorInfo);

    //Dispatches a request to the MIMIC daemon, retrieves and
    // returns the reply from the same
public:
    std::string dispatch_msg (int req_id,std::string req_msg) throw (ErrorInfo);

};
}
#endif

```

```
#ifndef _MIMIC_VALUESPACE_HH
#define _MIMIC_VALUESPACE_HH
```

```
////////////////////////////////////
// Copyright (c) 1995-2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#include <string>
#include <iostream>
#include <exception>
#include <list>
```

```
#include "mimic_errorinfo.hh"
```

```
namespace Mimic {
class Agent;
class Session;
class Valuespace
{
private:
    // POLICY: disallow copy and assignment
    Valuespace (const Valuespace& other);
    Valuespace& operator = (const Valuespace& other);
public :
    //data member methods
    //Creates a valuespace for the agent.
    Valuespace (Agent* agent);

    //Gets the current position in agent's OID tree.
    std::string get_pos () throw(ErrorInfo);

    //Set the current position in agent's OID tree.
    std::string set_pos(std::string pos) throw (ErrorInfo);

    //Get the children objects at the current position.-ver list name objects
    void get_objects (std::list<std::string>& objects) throw (ErrorInfo);

    //Add a new instance (for tables only)
    void add(std::string object, std::string instance) throw(ErrorInfo);

    //Remove an existing instance (table only)
    void remove(std::string object, std::string instance) throw(ErrorInfo);

    //Get instances of an object -ver list name instances
    void get_instances(std::string object,std::list<std::string>& instances) throw(ErrorInfo);

    //Get variables for object.instance -ver list name variables
    void get_variables (std::string object, std::string instance,std::list<std::string>& variables) throw (ErrorInfo);

    //Get value for object.instance.variable
    std::string get_value (std::string object, std::string instance,std::string variable) throw(ErrorInfo);
```

```

//Get values for multiple object.instance.variable -ver list name values
void mget_value (std::list<std::string>& objects,std::list<std::string>& instances,std::list<std::string>&
variables,std::list<std::string>& values) throw (ErrorInfo);

//Set the value for object.instance.variable
void set_value (std::string object,std::string instance,std::string variable,std::string value ) throw (ErrorInfo);

//set the value for mutliple object.instance.variable
void mset_value (std::list<std::string>& objects,std::list<std::string>& instances,std::list<std::string>&
variables,std::list<std::string>& values) throw (ErrorInfo);

//Unset the value for object.instance.variable
void unset_value (std::string object, std::string instance,std::string variable) throw (ErrorInfo);

//unset the value for mutliple object.instance.variable
void munset_value (std::list<std::string>& objects,std::list<std::string>& instances,std::list<std::string>&
variables) throw (ErrorInfo);

//Evaluate the value for object.instance
std::string eval_value (std::string object,std::string instance) throw (ErrorInfo);

//Evaluate the value for multiple object.instance -ver list name values
void meval_value (std::list<std::string>& objects,std::list<std::string>& instances,std::list<std::string>& values)
throw (ErrorInfo);

//Get an OID for a name
std::string get_oid(std::string object) throw(ErrorInfo);

//Get a name for an OID
std::string get_name (std::string oid) throw(ErrorInfo);

//Get the MIB conatining a given object
std::string get_mib (std::string object) throw(ErrorInfo);

//Get MIB information for object. MIB information is returned -ver list name mibs
void get_info (std::string object,std::list<std::string>& mibs) throw(ErrorInfo);

//Get MIB information for multiple objects -ver list name mibs
void get_minfo (std::list<std::string>& objects,std::list< std::list<std::string > >& mibs) throw(ErrorInfo);

//Split an OID into object and instance -ver list name oids
void split_oid (std::string oid,std::list<std::string>& oids) throw(ErrorInfo);

//Get/Set state of the object
std::string get_state (std::string object) throw(ErrorInfo);
void set_state (std::string object, int state) throw(ErrorInfo);

//Destructor
~Valuespace();
private :
int      _agent_no;
Session* _session;

std::string valuespace_dispatch_msg(int req_id,std::string req_msg) throw (ErrorInfo);

```

```
};  
}  
#endif
```

```
////////////////////////////////////
// Copyright (c) 2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#ifdef WIN32
#pragma warning(disable:4786)
#pragma warning(disable:4290)
#endif
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>          // for va_*
```

```
#include "mimic_action.hh"
```

```
// defines
#ifdef WIN32
#define vsnprintf _vsnprintf
#endif
```

```
// toggle following flag to enable/disable debugging
static int _debug_on = 0;
```

```
static void _debug ( const char* fmt, ... )
{
    if ( !_debug_on )
        return;

    char tmp[16*1024];    // NOTE: fixed size, 16K at a time

    // create string in tmp buffer
    va_list ap;
    va_start (ap, fmt);
    vsnprintf (tmp, sizeof (tmp), fmt, ap);
    va_end (ap);

    fprintf ( stderr, tmp );
}
```

```
static int
_on_action_module_register (
    action_module_register_t *data
)
{
    // verify version for the simulator
    data->version.sprintf ( ACTION_BUILD_ID );

    return ACTION_SUCCESS;
}
```



```

static int
_on_action_module_init (
    action_module_init_t *data
)
{
    // nothing TODO

    return ACTION_SUCCESS;
}

static int
_on_action_module_uninit (
    action_module_uninit_t *data
)
{
    // nothing TODO

    return ACTION_SUCCESS;
}

static int
_on_action_module_run (
    action_module_run_t *data
)
{
    _debug ( "entryStatus.dll: run invoked.\n");

    int value = atoi(data->globals->get("gCurrentValue"));
    _debug ( "entryStatus.dll: value = %d\n", value );

    // check valid value range
    if ( value < 1 || value > 4 )
    {
        _debug ( "entryStatus.dll: invalid value being set. ignoring...\n" );
        data->globals->set ( "gError", "3" );
        return ACTION_SUCCESS;
    }

    // create
    if ( value == 2 )
    {
        _debug ( "entryStatus.dll: create and go...\n" );
        data->globals->set ( "gCurrentValue", "1" );
        return ACTION_SUCCESS;
    }

    // delete
    if ( value == 4 )
    {
        _debug ( "entryStatus.dll: delete...\n" );

        // get agent and its valuespace
        int nagent = atoi(data->globals->get("gCurrentAgent"));
        _debug ( "entryStatus.dll: agent = %d\n", nagent );
    }
}

```

```

Mimic::Agent& agent = data->session->get_agent(nagent);
_debug ( "entryStatus.dll: step1\n");
Mimic::Valuespace& valuespace = agent.get_valuespace();
_debug ( "entryStatus.dll: step2\n");

// position at the xxxTable point in the table
std::string object = data->globals->get("gCurrentObject");
_debug ( "entryStatus.dll: object = %s\n", object.c_str() );
valuespace.set_pos ( object );
valuespace.set_pos ( ".." );
valuespace.set_pos ( ".." );
std::list<std::string> children;
valuespace.get_objects ( children );
std::string entry_name = *(get(children.begin(),0));
_debug ( "entryStatus.dll: entry = %s\n", entry_name.c_str() );

// delete the row
std::string instance = data->globals->get("gCurrentInstance");
valuespace.remove ( entry_name, instance );
}

return ACTION_SUCCESS;
}

```

```

#ifdef __cplusplus
extern "C" {
#endif

```

```

//////////
// DLL entry point
//////////

```

```

#ifdef WIN32
__declspec( dllexport )
#endif
int
process_action_message (
    int msg_id,
    void* msg_data
)
{
    switch( msg_id )
    {
        case ACTION_MODULE_REGISTER:
        {
            return _on_action_module_register (
                (action_module_register_t*)msg_data
            );
        }

        case ACTION_MODULE_INIT:
        {
            return _on_action_module_init (

```

```
        (action_module_init_t*)msg_data
    );
}

case ACTION_MODULE_UNINIT:
{
    return _on_action_module_uninit (
        (action_module_uninit_t*)msg_data
    );
}

case ACTION_MODULE_RUN:
{
    return _on_action_module_run (
        (action_module_run_t*)msg_data
    );
}
}

return ACTION_SUCCESS; // default success
}

#ifdef __cplusplus
}
#endif
```

```
////////////////////////////////////  
// Copyright (c) 2004 by Gambit Communications, Inc.  
// All Rights Reserved  
////////////////////////////////////
```

```
// MIMIC Sample Action DLL.  
// The dll needs to be copied to a particular simulation directory under  
// data/sim and then associate it with ifAdminStatus variable from the  
// ifTable for the SNMP-SET request. When the ifAdminStatus is changed from  
// '2' to '1' then the ifOperStatus is also changed to '1' and a linkUp  
// trap is sent out. Similarly when the change is from '1' to '2' a linkDown  
// trap is sent and ifOperStatus is set to '2'.
```

```
#ifdef WIN32  
#pragma warning(disable:4786)  
#pragma warning(disable:4290)  
#endif
```

```
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>
```

```
#include "mimic_action.hh"
```

```
static int  
_on_action_module_register (  
    action_module_register_t *data  
)  
{  
    data->version.sprintf ( ACTION_BUILD_ID );  
    return ACTION_SUCCESS;  
}
```

```
static int  
_on_action_module_init (  
    action_module_init_t *data  
)  
{  
    return ACTION_SUCCESS;  
}
```

```
static int  
_on_action_module_uninit (  
    action_module_uninit_t *data  
)  
{  
    return ACTION_SUCCESS;  
}
```

```
static int  
_on_action_module_run (  
    action_module_run_t *data  
)
```

```

{

const int agtid = atoi( data->globals->get("gCurrentAgent" ) );
Mimic::Agent& agent = data->session->get_agent(agtid);

const char* cur_ins = data->globals->get("gCurrentInstance");
Mimic::Valuespace& valuespace = agent.get_valuespace();

const char* cur_value = valuespace.eval_value( "ifAdminStatus", cur_ins ).c_str();
const char* set_value = data->globals->get("gCurrentValue");

if ( strcmp( cur_value, set_value) == 0 )
{
    // Nothing has changed, so no need to generate trap
    return ACTION_SUCCESS;
}

if ( strcmp ( set_value, "1" ) == 0 )
{
    // set ifOperStatus=1 and send a linkUp trap
    valuespace.set_value( "ifOperStatus", cur_ins, "v", "1" );
    valuespace.set_pos( "traps" );
    valuespace.set_value( "linkUp", "0", "r", "1" );
    valuespace.set_value( "linkUp", "0", "tu", "1" );
    valuespace.set_value( "linkUp", "0", "c", "1" );
    return ACTION_SUCCESS;
}

if ( strcmp ( set_value, "2" ) == 0 )
{
    // set ifOperStatus=2 and send a linkDown trap
    valuespace.set_value( "ifOperStatus", cur_ins, "v", "2" );
    valuespace.set_pos( "traps" );
    valuespace.set_value( "linkDown", "0", "r", "1" );
    valuespace.set_value( "linkDown", "0", "tu", "1" );
    valuespace.set_value( "linkDown", "0", "c", "1" );
}

// ignore set_value 3

return ACTION_SUCCESS;
}

#ifdef __cplusplus
extern "C" {
#endif

//////////
// DLL entry point
//////////

#ifdef WIN32
__declspec( dllexport )

```

```

#endif
int
process_action_message (
    int msg_id,
    void* msg_data
)
{
    switch( msg_id )
    {
        case ACTION_MODULE_REGISTER:
        {
            return _on_action_module_register (
                (action_module_register_t*)msg_data
            );
        }

        case ACTION_MODULE_INIT:
        {
            return _on_action_module_init (
                (action_module_init_t*)msg_data
            );
        }

        case ACTION_MODULE_UNINIT:
        {
            return _on_action_module_uninit (
                (action_module_uninit_t*)msg_data
            );
        }

        case ACTION_MODULE_RUN:
        {
            return _on_action_module_run (
                (action_module_run_t*)msg_data
            );
        }
    }

    return ACTION_SUCCESS; // default success
}

#ifdef __cplusplus
}
#endif

```

```
/*
*****
Copyright (c) 1995-2002 by Gambit Communications, Inc.
All Rights Reserved
*****
*/
```

```
/*
 * Helper functions
 */
//-----
```

```
#ifdef WIN32
 #pragma warning(disable:4786)
 #pragma warning(disable:4290)
#endif
```

```
#include <time.h>
#include <iostream>
using namespace std;
#include <stdio.h>
```

```
#include "mimic_helper.hh"
#include "mimic_api.hh"
```

```
#ifdef WIN32
#define sleep(s) _sleep(s*1000)
#else
#include <unistd.h>
#endif
```

```
////////////////////////////////////n
// Description:
//   Wait for the specified agent state
// Return values:
//   NONE
// Side effects:
//   NONE
////////////////////////////////////
```

```
void
wait_for_agent_state (
    Mimic::Session* session,
    int agent_no,
    int status
) throw (Mimic::ErrorInfo)
{
    Mimic::Agent& agent = session->get_agent(agent_no);
    while (agent.get_state() != status)
    {
        sleep(1);
    }
}
```

```
////////////////////////////////////
// Copyright (c) 2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#ifdef WIN32
#pragma warning(disable:4786)
#pragma warning(disable:4290)
#endif
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>          // for va_*
```

```
#include "mimic_action.hh"
```

```
// defines
#ifdef WIN32
#define vsnprintf _vsnprintf
#endif
```

```
// toggle following flag to enable/disable debugging
static int _debug_on;
```

```
static void _debug ( const char* fmt, ... )
{
    if ( !_debug_on )
        return;

    static char tmp[16*1024];    // NOTE: fixed size, 16K at a time
    static short prefix_len = 0;

    if(prefix_len == 0)
    {
```

```
#ifdef WIN32
        strcpy(tmp, "rowStatus.dll: " );
#else
        strcpy(tmp, "rowStatus.so: " );
#endif
        prefix_len = strlen(tmp);
    }
```

```
// create string in tmp buffer
va_list ap;
va_start (ap, fmt);
vsnprintf (&tmp[prefix_len], (sizeof(tmp)-prefix_len), fmt, ap);
va_end (ap);
```



```

    fputs ( tmp, stderr );
    fputs ( "\n", stderr);
    fflush(stderr);
}

static int
_on_action_module_register (
    action_module_register_t *data
)
{
    // verify version for the simulator
    data->version.sprintf ( ACTION_BUILD_ID );

    return ACTION_SUCCESS;
}

static int
_on_action_module_init (
    action_module_init_t *data
)
{
    int exists;
    exists = data->session->store_exists ("rowStatus::debug");
    if(!exists)
        return ACTION_SUCCESS;

    std::string ret_val = data->session->store_get("rowStatus::debug");

    if (ret_val.compare("1") == 0)
    {
        _debug_on = 1;
    }
    else
    {
        _debug_on = 0;
    }

    return ACTION_SUCCESS;
}

static int
_on_action_module_uninit (
    action_module_uninit_t *data
)
{
    // nothing TODO

    return ACTION_SUCCESS;
}

static int
_on_action_module_run (

```

```

action_module_run_t *data
)
{

// Print the value of the Current Agent
int agent_val = atoi(data->globals->get("gCurrentAgent"));
_debug("Agent number = %d", agent_val);

// Initializing vspace
Mimic::Agent& agent = data->session->get_agent(agent_val);
Mimic::Valuespace& vspace = agent.get_valuespace();
_debug ( "run invoked.");

// Print the value of the gCurrentValue
int value = atoi(data->globals->get("gCurrentValue"));
_debug ( "value = %d", value );

// check valid value range
if ( value < 1 || value > 6 || value == 3 )
{
    _debug ( "Invalid value being set. ignoring..." );
    data->globals->set ( "gError", "3" );
    return ACTION_SUCCESS;
}

//Get the Current Object
std::string cur_object = data->globals->get("gCurrentObject");
_debug ( "RowStatus object = %s", cur_object.c_str() );

std::string eval_val;
// Get the Current Instance
std::string cur_instance = data->globals->get("gCurrentInstance");
_debug ( "Current Instance = %s", cur_instance.c_str());

// active
if ( value == 1 )
{
    eval_val = vspace.eval_value(cur_object, cur_instance);
    _debug("Evaluated Value = %s", eval_val.c_str());

    if ((eval_val.compare("3") == 0))
    {
        _debug ( "Inconsistent Value..." );
        data->globals->set ( "gError", "12" );
        return ACTION_SUCCESS;
    }

    _debug ( "active..." );
    return ACTION_SUCCESS;
}
}

```

```

// notInService
if ( value == 2 )
{
    eval_val = vspace.eval_value(cur_object, cur_instance);
    _debug("Evaluated Value = %s", eval_val.c_str());

    if ((eval_val.compare("3") == 0))
    {
        _debug ( "Inconsistent Value..." );
        data->globals->set ( "gError", "12" );
        return ACTION_SUCCESS;
    }

    _debug ( "notInService..." );
    return ACTION_SUCCESS;
}

// createAndGo
if ( value == 4 )
{
    eval_val = vspace.eval_value(cur_object, cur_instance);
    _debug("Evaluated Value = %s", eval_val.c_str());

    if ((eval_val.compare("1") == 0) || (eval_val.compare("3") == 0))
    {
        _debug ( "Inconsistent Value..." );
        data->globals->set ( "gError", "12" );
        return ACTION_SUCCESS;
    }

    _debug ( "create and go..." );
    data->globals->set ( "gCurrentValue", "1" );
    return ACTION_SUCCESS;
}

// createAndWait
if ( value == 5 )
{
    eval_val = vspace.eval_value(cur_object, cur_instance);
    _debug("Evaluated Value = %s", eval_val.c_str());

    if ((eval_val.compare("1") == 0) || (eval_val.compare("3") == 0))
    {
        _debug ( "Inconsistent Value..." );
        data->globals->set ( "gError", "12" );
        return ACTION_SUCCESS;
    }

    _debug ( "create and wait..." );
    data->globals->set ( "gCurrentValue", "3" );
}

```

```

    return ACTION_SUCCESS;
}

// delete
if ( value == 6 )
{
    _debug ( "delete..." );

    // get agent and its valuespace
    int nagent = atoi(data->globals->get("gCurrentAgent"));
    _debug ( "agent = %d", nagent );
    Mimic::Agent& agent = data->session->get_agent(nagent);
    _debug ( "step1");
    Mimic::Valuespace& valuespace = agent.get_valuespace();
    _debug ( "step2");

    // position at the xxxTable point in the table
    std::string object = data->globals->get("gCurrentObject");
    _debug ( "object = %s", object.c_str() );
    valuespace.set_pos ( object );
    valuespace.set_pos ( ".." );
    valuespace.set_pos ( ".." );
    std::list<std::string> children;
    valuespace.get_objects ( children );
    std::string entry_name = *(get(children.begin(),0));
    _debug ( "entry = %s", entry_name.c_str() );

    // delete the row
    std::string instance = data->globals->get("gCurrentInstance");
    valuespace.remove ( entry_name, instance );
}

return ACTION_SUCCESS;
}

```

```

#ifdef __cplusplus
extern "C" {
#endif

```

```

//////////
// DLL entry point
//////////

```

```

#ifdef WIN32
__declspec( dllexport )
#endif
int
process_action_message (
    int msg_id,
    void* msg_data
)
{
    switch( msg_id )

```

```
{
  case ACTION_MODULE_REGISTER:
  {
    return _on_action_module_register (
      (action_module_register_t*)msg_data
    );
  }

  case ACTION_MODULE_INIT:
  {
    return _on_action_module_init (
      (action_module_init_t*)msg_data
    );
  }

  case ACTION_MODULE_UNINIT:
  {
    return _on_action_module_uninit (
      (action_module_uninit_t*)msg_data
    );
  }

  case ACTION_MODULE_RUN:
  {
    return _on_action_module_run (
      (action_module_run_t*)msg_data
    );
  }
}

return ACTION_SUCCESS; // default success
}

#ifdef __cplusplus
}
#endif
```

```
////////////////////////////////////
// Copyright (c) 2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#ifdef WIN32
#pragma warning(disable:4786)
#pragma warning(disable:4290)
#endif
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>          // for va_*
```

```
#include "mimic_action.hh"
```

```
// defines
#ifdef WIN32
#define vsnprintf _vsnprintf
#endif
```

```
// toggle following flag to enable/disable debugging
static int _debug_on = 0;
```

```
static void _debug ( const char* fmt, ... )
{
    if ( !_debug_on )
        return;

    char tmp[16*1024];    // NOTE: fixed size, 16K at a time

    // create string in tmp buffer
    va_list ap;
    va_start (ap, fmt);
    vsnprintf (tmp, sizeof (tmp), fmt, ap);
    va_end (ap);

    fprintf ( stderr, tmp );
}
```

```
static int
_on_action_module_register (
    action_module_register_t *data
)
{
    // verify version for the simulator
    data->version.sprintf ( ACTION_BUILD_ID );

    return ACTION_SUCCESS;
}
```

```

static int
_on_action_module_init (
    action_module_init_t *data
)
{
    // nothing TODO

    return ACTION_SUCCESS;
}

```

```

static int
_on_action_module_uninit (
    action_module_uninit_t *data
)
{
    // nothing TODO

    return ACTION_SUCCESS;
}

```

```

static int
_on_action_module_run (
    action_module_run_t *data
)
{
    _debug ( "start.dll: run invoked.\n");

    //
    // get agent and its valuespace
    //
    int nagent = atoi(data->globals->get("gCurrentAgent"));
    _debug ( "start.dll: agent = %d\n", nagent );
    Mimic::Agent& agent = data->session->get_agent(nagent);
    Mimic::Valuespace& valuespace = agent.get_valuespace();

    //
    // position at "traps" and set trap parameters
    //
    _debug ( "start.dll: sending a coldStart trap...\n" );
    valuespace.set_pos ( "traps" );
    valuespace.set_value ( "coldStart", "0", "r", "1" );
    valuespace.set_value ( "coldStart", "0", "tu", "1" );
    valuespace.set_value ( "coldStart", "0", "c", "1" );

    //
    // manipulate ifPhysAddress to distinct values based on
    // the agent number and ifIndex
    //
    _debug ( "start.dll: manipulating ifPhysAddress...\n" );
    unsigned byte1 = 0xfe; // common prefix
    unsigned byte2 = 0xdc; // ...
    unsigned byte3 = nagent/255; // based on agent
    unsigned byte4 = nagent%255; // ...
    valuespace.set_pos ( "ifEntry" );
}

```

```

std::list<std::string> inst_list;
valuespace.get_instances( "ifIndex", inst_list );
for ( unsigned int i=0; i<inst_list.size(); i++ )
{
    std::string& inst = *( get(inst_list.begin(),i) );

    int nifIndex = atoi(inst.c_str());

    unsigned byte5 = nifIndex/255; // based on ifIndex
    unsigned byte6 = nifIndex%255; // ...

    char phys_addr[256];
    sprintf ( phys_addr, "\\x%02x %02x %02x %02x %02x %02x",
        byte1, byte2, byte3, byte4, byte5, byte6 );

    valuespace.set_value ( "ifPhysAddress", inst, "v", phys_addr );
}

_debug ( "start.dll: success.\n");
return ACTION_SUCCESS;
}

```

```

#ifdef __cplusplus
extern "C" {
#endif

```

```

//////////
// DLL entry point
//////////

```

```

#ifdef WIN32
__declspec( dllexport )
#endif
int
process_action_message (
    int msg_id,
    void* msg_data
)
{
    switch( msg_id )
    {
        case ACTION_MODULE_REGISTER:
        {
            return _on_action_module_register (
                (action_module_register_t*)msg_data
            );
        }

        case ACTION_MODULE_INIT:
        {
            return _on_action_module_init (
                (action_module_init_t*)msg_data
            );
        }
    }
}

```



```
    }

    case ACTION_MODULE_UNINIT:
    {
        return _on_action_module_uninit (
            (action_module_uninit_t*)msg_data
        );
    }

    case ACTION_MODULE_RUN:
    {
        return _on_action_module_run (
            (action_module_run_t*)msg_data
        );
    }
}

return ACTION_SUCCESS; // default success
}

#ifdef __cplusplus
}
#endif
```

```
////////////////////////////////////
// Copyright (c) 2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#ifdef WIN32
#pragma warning(disable:4786)
#pragma warning(disable:4290)
#endif
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
#include "mimic_action.hh"
```

```
class Dummy {};
```

```
static void _generate_exception ()
```

```
{
    static int ncall = 0;
    ncall++;

    if ( ncall == 1 )
    {
        fprintf ( stderr, "    - exercising exception handling (ErrorInfo exception) :\n" );
        fflush ( stderr );

        // generate a dummy exception
        // i.e. unknown to simulator
        throw Mimic::ErrorInfo (
            MIMIC_FAILURE,
            "a test exception"
        );
    }
    if ( ncall == 2 )
    {
        fprintf ( stderr, "    - exercising exception handling (unknown C++ exception) :\n" );
        fflush ( stderr );

        // generate a dummy exception
        // i.e. unknown to simulator
        throw Dummy();
    }
}
```

```
// on Unices following exception causes appropriate
// signals to be generated and not result in any
// catchable exceptions.
```

```
#ifdef WIN32
    if ( ncall == 3 )
    {
        fprintf ( stderr, "    - exercising exception handling (NULL pointer access) :\n" );
    }
}
```

```

    fflush ( stderr );

    // write to NULL pointer
    char *cp = NULL;
    cp[0] = 'a';
}
if ( ncall == 4 )
{
    fprintf ( stderr, "    - exercising exception handling (divide by zero) :\n" );

    // integer divide by zero
    int a = 0;
    a = 1 / a;
}
#endif
}

static int
_on_action_module_register (
    action_module_register_t *data
)
{
    fprintf ( stderr, "INFO - received message : ACTION_MODULE_REGISTER.\n" );

    // verify version for the simulator
    data->version.sprintf ( ACTION_BUILD_ID );

    fprintf ( stderr, "    - done.\n" );
    fflush ( stderr );

    return ACTION_SUCCESS;
}

static int
_on_action_module_init (
    action_module_init_t *data
)
{
    fprintf ( stderr, "INFO - received message - ACTION_MODULE_INIT. ***\n" );

    fprintf ( stderr, "    - exercising application global commands :\n" );
    fprintf ( stderr, "    - mimic get version = %s\n", data->session->get_version().c_str() );
    fprintf ( stderr, "    - mimic get cfgfile = %s\n", data->session->get_cfgfile().c_str() );
    fprintf ( stderr, "    - mimic get clients = %d\n", data->session->get_clients() );
    fprintf ( stderr, "    - mimic get last   = %d\n", data->session->get_last() );
    fprintf ( stderr, "    - mimic get max    = %d\n", data->session->get_max() );

    fprintf ( stderr, "    - done.\n" );
    fflush ( stderr );

    return ACTION_SUCCESS;
}

static int

```

```

_on_action_module_uninit (
    action_module_uninit_t *data
)
{
    fprintf ( stderr, "INFO - received message - ACTION_MODULE_UNINIT. ***\n" );

    // TODO cleanup

    fprintf ( stderr, "    - done.\n" );
    fflush ( stderr );

    return ACTION_SUCCESS;
}

static int
_on_action_module_run (
    action_module_run_t *data
)
{
    fprintf ( stderr, "INFO - received message - ACTION_MODULE_RUN. ***\n" );

    // print the globals here
    fprintf ( stderr, "    - passed in globals :\n" );
    fprintf ( stderr, "    - Agent    = %s\n", data->globals->get("gCurrentAgent") );
    fprintf ( stderr, "    - Request  = %s\n", data->globals->get("gCurrentRequest") );
    fprintf ( stderr, "    - PduInfo  = %s\n", data->globals->get("gPduInfo") );
    fprintf ( stderr, "    - Object   = %s\n", data->globals->get("gCurrentObject") );
    fprintf ( stderr, "    - Instance = %s\n", data->globals->get("gCurrentInstance") );
    fprintf ( stderr, "    - Value    = %s\n", data->globals->get("gCurrentValue") );
    fprintf ( stderr, "    - SharedDir = %s\n", data->globals->get("gSharedDir") );
    fprintf ( stderr, "    - Owner    = %s\n", data->globals->get("gOwner") );
    fprintf ( stderr, "    - PrivateDir = %s\n", data->globals->get("gPrivateDir") );
    fprintf ( stderr, "    - Error    = %s\n", data->globals->get("gError") );

    data->globals->set ( "gCurrentValue", "My Test Device" );

    Mimic::Agent& agent = data->session->get_agent(1);
    fprintf ( stderr, "    - exercising agent commands :\n" );
    fprintf ( stderr, "    - mimic agent get host = %s\n", agent.get_host().c_str() );
    fprintf ( stderr, "    - mimic agent get port = %d\n", agent.get_port() );

    Mimic::Valuespace& valuespace = agent.get_valuespace();
    fprintf ( stderr, "    - exercising agent valuespace commands :\n" );
    fprintf ( stderr, "    - mimic value get sysDescr 0 v = %s\n", valuespace.get_value( "sysDescr", "0", "v").c_str()
);
    fprintf ( stderr, "    - mimic value eval sysUpTime 0 = %s\n", valuespace.eval_value( "sysUpTime", "0").c_str()
);
    fflush ( stderr );

    _generate_exception ();

    fprintf ( stderr, "    - done.\n" );
    fflush ( stderr );
}

```

```

    return ACTION_SUCCESS;
}

#ifdef __cplusplus
extern "C" {
#endif

//////////
// DLL entry point
//////////

#ifdef WIN32
__declspec( dllexport )
#endif
int
process_action_message (
    int msg_id,
    void* msg_data
)
{
    switch( msg_id )
    {
        case ACTION_MODULE_REGISTER:
        {
            return _on_action_module_register (
                (action_module_register_t*)msg_data
            );
        }

        case ACTION_MODULE_INIT:
        {
            return _on_action_module_init (
                (action_module_init_t*)msg_data
            );
        }

        case ACTION_MODULE_UNINIT:
        {
            return _on_action_module_uninit (
                (action_module_uninit_t*)msg_data
            );
        }

        case ACTION_MODULE_RUN:
        {
            return _on_action_module_run (
                (action_module_run_t*)msg_data
            );
        }
    }

    return ACTION_SUCCESS; // default success
}

```

```
#ifdef __cplusplus
}
#endif
```

```

#ifdef WIN32
    #pragma warning(disable:4786)
    #pragma warning(disable:4290)
#endif

#include <time.h>
#include <iostream>
#include <stdio.h>

#include "mimic_api.hh"

#ifdef _WIN64
#include <windows.h>
#define sleep(s) Sleep(s*1000)
#else
#ifdef WIN32
#define sleep(s) _sleep(s*1000)
#else
#include <unistd.h>
#endif
#endif

std::string host = "localhost";
long port = 9797;
Mimic::Client *clientptr;
Mimic::Session* session;
int threadno = 1;

using namespace std;

/*int _thread_func (GciThread *handle)
{

    Mimic::Session *session = (Mimic::Session*)handle->get_data();
    try
    {
        cout << "MyThread[" << threadno << "] - starting...\n" ;
        for (int i=0; i<1000; i++ )
        {
            //
            // do some interaction
            //

            std::string cfgfile = session->get_cfgfile();

            std::list<int> cfgd_agents;
            session->get_configured_list(cfgd_agents);
            std::list<int> chngd_cfgd_agents;
            session->get_changed_config_list (chngd_cfgd_agents);

            std::list<std::string> interfaces;

```

```

    session->get_interfaces(interfaces );
    std::list<std::string> protocols;
    session->get_protocols (protocols);
    Mimic::Agent& agent = session->get_agent(threadno);
    std::string host = agent.get_host();

    if ( (i % 100) == 0 )
    {
        cout << "Agent [" << threadno << "].host = " << host << "\n";
    }

    // sleep a while
    try
    {

        GciThreadMgr::sleep (1,0);
    }
    catch (...)
    {
        break;
    }
}
    cout << "MyThread[" << threadno << "] - done.\n" ;
}
catch ( Mimic::ErrorInfo errorinfo )
{
    cout << "ERROR: " + errorinfo.get_error_msg() << "\n";
    return -1;
}
return 0;
}*/

static
void _init (
)

{
    mimic_debug_runtime_load ();
}

void
parse_args(
    int argc,
    char *args[]
) throw(Mimic::ErrorInfo)

{
    if ( argc <= 1 ) return;

    int idx = 1; //start from first argument

    while ( true )
    {
        if ( idx >= argc ) break;

```



```

std::string argvstr(args[idx]);
if(argvstr.compare("--host") == 0)
{
    idx++; if ( idx >= argc ) break;
    host = args[idx];
    idx++; if ( idx >= argc ) break;
    cout << "TestApp: host = " << host.c_str() << "\n";
}
else if( argvstr.compare("--port") == 0 )
{
    idx++; if ( idx >= argc ) break;
    port = parse_int( args[idx] );
    idx++; if ( idx >= argc ) break;
    cout << "TestApp: port = " << port << "\n";
}
else
{
    throw (
        Mimic::ErrorInfo(
            MIMIC_FAILURE,
            std::string("unknown option ") + argvstr
        )
    );
}
}
}

```

```

void
test_session_init (
    ) throw(Mimic::ErrorInfo)
{
    clientptr = new Mimic::Client();
    //session_list.push_back(&clientptr->open_session(host));
    session = &clientptr->open_session(host, port);
    cout << "test_session_init : Session : " << session << "\n";
    clientptr->dump();

    //session->cfg_load("agents.cfg");
    //session->cfg_saveas("foo.cfg");
    cout << "test_session_init over\n";
}

```

```

void
test_3510 (
    ) throw(Mimic::ErrorInfo)
{
    // Open a session
    Mimic::Session& session2 = clientptr->open_session( host, port );
    std::list<Mimic::Session*> &sessions = clientptr->list_sessions();
    if (sessions.size() != 2)
    {
        throw (

```

```
        Mimic::ErrorInfo(
            MIMIC_FAILURE,
            "client list should have 2 entries"
        )
    );
}
```

```
// Disconnect the session
clientptr->close_session( session2 );
sessions = clientptr->list_sessions();
if (sessions.size() != 1)
{
    throw (
        Mimic::ErrorInfo(
            MIMIC_FAILURE,
            "client list should have 1 entry"
        )
    );
}
```

```
// Start three new sessions on a new client
Mimic::Client *client2;
client2 = new Mimic::Client();
Mimic::Session& session3 = client2->open_session( host, port );
Mimic::Session& session4 = client2->open_session( host, port );
Mimic::Session& session5 = client2->open_session( host, port );
```

```
// Check that old client has 1 session
sessions = clientptr->list_sessions();
if (sessions.size() != 1)
{
    throw (
        Mimic::ErrorInfo(
            MIMIC_FAILURE,
            "client list should have 1 entry"
        )
    );
}
```

```
// Check that new client has 3 sessions
sessions = client2->list_sessions();
if (sessions.size() != 3)
{
    throw (
        Mimic::ErrorInfo(
            MIMIC_FAILURE,
            "client2 list should have 3 entries"
        )
    );
}
```

```
// Disconnect sessions
client2->close_session( session3 );
sessions = client2->list_sessions();
```

```

if (sessions.size() != 2)
{
    throw (
        Mimic::ErrorInfo(
            MIMIC_FAILURE,
            "client2 list should have 2 entries"
        )
    );
}
client2->close_session( session4 );
sessions = client2->list_sessions();
if (sessions.size() != 1)
{
    throw (
        Mimic::ErrorInfo(
            MIMIC_FAILURE,
            "client2 list should have 1 entry"
        )
    );
}
client2->close_session( session5 );
sessions = client2->list_sessions();
if (sessions.size() != 0)
{
    throw (
        Mimic::ErrorInfo(
            MIMIC_FAILURE,
            "client2 list should have 0 entries"
        )
    );
}
}

void
test_session_get(
    ) throw(Mimic::ErrorInfo)
{

//Mimic::Session *session = *(session_list.end());
std::string cfgfile = session->get_cfgfile();
cout<< "TestApp: config file = " << cfgfile.c_str() << "\n";

int max_agents = session->get_max();
cout << "TestApp: max agents = " << max_agents << "\n";

int last_agent = session->get_last();
cout << "TestApp: last agent = " << last_agent << "\n";

std::string version = session->get_version();
cout << "TestApp: version = " << version.c_str() << "\n";

int num_clients = session->get_clients();
cout << "TestApp: num clients = " << num_clients << "\n";

```

```

std::list<int> cfg_agents ;
session->get_configured_list(cfg_agents);
cout << "TestApp: configured agents = [";
for (size_t i = 0; i < cfg_agents.size(); i++)
{
    if (i > 0)
        cout << ", ";
    cout << *(get(cfg_agents.begin(),(int)i));
}
cout << "]\n";

{
std::list< std::list<unsigned long long> > activedata ;
session->get_active_data_list(activedata);
cout << "TestApp: active data :\n" ;
for (size_t i=0; i<activedata.size(); i++ )
{
    std::list<unsigned long long>& el = *(get(activedata.begin(),(int)i));

    cout << "  agent = " << *(get(el.begin(),0)) ;
    cout << ", stats = ";
    for ( size_t j=1; j<el.size(); j++ )
    {
        cout << " " << *(get(el.begin(),(int)j));
    }
    cout << "\n";
}
}

{
cout << "Testing obsolete session.get_active_data_list, returns incorrect values on large statistics\n";
std::list< std::list<int> > activedata ;
session->get_active_data_list(activedata);
cout << "TestApp: active data :\n" ;
for (size_t i=0; i<activedata.size(); i++ )
{
    std::list<int>& el = *(get(activedata.begin(),(int)i));

    cout << "  agent = " << *(get(el.begin(),0)) ;
    cout << ", stats = ";
    for ( size_t j=1; j<el.size(); j++ )
    {
        cout << " " << *(get(el.begin(),(int)j));
    }
    cout << "\n";
}
}

std::list<int> chcfg_agents;
session->get_changed_config_list(chcfg_agents);
cout << "TestApp: configured changed agents = [";
for (size_t i = 0 ; i < chcfg_agents.size() ; i++)
{
    if (i > 0)

```

```

        cout << ", ";
        cout << *(get(chcfg_agents.begin(),(int)i));
    }
    cout << "]\n";
    std::list< std::list<int> > chstate_agents;
    session->get_changed_state_list(chstate_agents);
    cout << "TestApp: state changes :\n";

    for (size_t i=0; i<chstate_agents.size(); i++ )
    {
        std::list<int>& el = *(get(chstate_agents.begin(),(int)i));

        cout << "  agent = " << *(get(el.begin(),0));
        cout << "  state = " << *(get(el.begin(),1)) << "\n";
    }

    std::string logfile = session->get_log();
    cout<< "TestApp: log file = " << logfile.c_str() << "\n";

    std::list<std::string> interfaces;
    session->get_interfaces(interfaces);
    cout<< "TestApp: interfaces = [";
    for (size_t i = 0; i < interfaces.size();i++)
    {
        if (i > 0)
            cout << ", ";
        cout << (*(get(interfaces.begin(),(int)i))).c_str();
    }
    cout << "]\n";

    std::list<std::string> protocols ;
    session->get_protocols(protocols);
    cout << "TestApp: protocols = [";
    for (size_t i = 0; i < protocols.size();i++)
    {
        if (i > 0)
            cout << ", ";
        cout << (*(get(protocols.begin(),(int)i))).c_str();
    }
    cout << "]\n";

    int product = session->get_product();
    cout << "TestApp: product = " << product << "\n";

    std::string netaddr = session->get_netaddr();
    cout << "TestApp: netaddr = " << netaddr << "\n";

    std::string netdev = session->get_netdev();
    cout << "TestApp: netdev = " << netdev << "\n";

    std::string expiration = session->get_expiration(5);
    cout << "TestApp: expiration(5) = " << expiration << "\n";

    std::string licensing = session->get_licensing();

```

```

cout << "TestApp: licensing = " << licensing << "\n";

std::list<std::list<std::string>> threads ;
session->get_threads(threads);
cout << "TestApp: threads = \n";
for (size_t i = 0;i < threads.size();i++)
{
    std::list<std::string>& el = *(get(threads.begin(),(int)i));

    for (size_t j = 0;j < el.size();j++)
    {
        const char* buff = (*(get(el.begin(),(int)j))).c_str();
        std::list<std::string> attr_list= split_to_stringlist (buff);

        cout << (*(get(attr_list.begin(),0))).c_str();
        cout << "=\n";
        cout << (*(get(attr_list.begin(),1))).c_str();
        cout << "\n";
    }
    cout << "\n";
}
cout << "]\n";

std::list<std::string> cmd_list,resp_list;
cmd_list.push_back("version");
cmd_list.push_back("max");
cmd_list.push_back("cfgfile");
cmd_list.push_back("active_list");
cmd_list.push_back("active_data_list");
cmd_list.push_back("protocols");
cmd_list.push_back("interfaces");
cmd_list.push_back("product");

resp_list = session->mget(cmd_list);
cout<<"TestApp: mget :\n" ;
for (size_t i=0; i<resp_list.size(); i++ )
{
    cout<< " " << (*(get(cmd_list.begin(),(int)i))).c_str() << " = ";
    cout<< (*(get(resp_list.begin(),(int)i))).c_str() << "\n";
}

// trap config add will cause error if entry already exists
// So, delete any existing ones..
std::list<std::string> trap_config_list;
session->trap_config_list (trap_config_list);
if (trap_config_list.size() > 0)
    cout<< "TestApp: Deleting existing trap config entries\n";
for(size_t i = 0;i < trap_config_list.size();i++)
{
    std::string& entry = (*(get(trap_config_list.begin(),(int)i)));
    int pos = (int)entry.find(',');
    std::string dest = "";
    std::string port = "";
    if (pos >= 0)

```

```

    {
        dest = entry.substr(0, pos);
        port = entry.substr(pos+1);
        session->trap_config_delete(dest,atoi(port.c_str()));
    }
}

cout<< "TestApp: trap config add = [192.9.240.24,162 192.9.200.16,162]\n";
session->trap_config_add ( "192.9.240.24", 162);
session->trap_config_add ( "192.9.200.16", 162);

```

```

trap_config_list.clear();
session->trap_config_list (trap_config_list);
cout<< "TestApp: trap_config_list = [";
for(size_t i = 0;i < trap_config_list.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(trap_config_list.begin(),(int)i))).c_str();
}
cout << "]\n";

```

```

session->trap_config_delete ( "192.9.200.16", 162);
cout<< "TestApp: removed trap destination 192.9.200.16,162\n";

```

```

std::list<std::string> trap_config_list1 ;
session->trap_config_list (trap_config_list1);
cout << "TestApp: trap_config_list = [";
for(size_t i = 0;i < trap_config_list1.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(trap_config_list1.begin(),(int)i))).c_str();
}
cout << "]\n";

```

```

void
test_session_timerscripts (
    ) throw (Mimic::ErrorInfo)
{
    try
    {
        //Mimic::Session *session = *(session_list.end());
        std::list<std::string> timerscripts;

        session->add_timer_script ( "timertest1.mtcl", 1000, "1 3 5" );

        session->list_timer_scripts (timerscripts);
        cout<<"TestApp: timer scripts :";
        for (size_t i=0; i<timerscripts.size(); i++ )
        {
            cout<<"    " << (*(get(timerscripts.begin(),(int)i ))).c_str();

```

```

}
//clock_t endwait;
//endwait = clock() + 5000 * CLK_TCK;
//while (clock() < endwait) { } ;
sleep (5);

session->add_timer_script ( "timertest1.mtcl", 2000, "7 9" );

session->list_timer_scripts (timerscripts);
cout<<"TestApp: timer scripts :";
for (size_t i=0; i<timerscripts.size(); i++ )
{
    cout<<"    " << (*(get(timerscripts.begin(),(int)i) )).c_str();
}
//clock_t endwait;
//endwait = clock() + 5000 * CLK_TCK;
//while (clock() < endwait) { } ;
sleep (5);
session->del_timer_script ( "timertest1.mtcl" );

session->list_timer_scripts (timerscripts);
cout<<"TestApp: timer scripts :";
for (size_t i=0; i<timerscripts.size(); i++ )
{
    cout<<"    " << (*(get(timerscripts.begin(),(int)i) )).c_str();
}
//clock_t endwait;
//endwait = clock() + 5000 * CLK_TCK;
//while (clock() < endwait) { } ;

sleep (5);
session->del_timer_script ( "timertest2.mtcl" );

session->list_timer_scripts (timerscripts);
cout<<"TestApp: timer scripts :";
for (size_t i=0; i<timerscripts.size(); i++ )
{
    cout<<"    " << (*(get(timerscripts.begin(),(int)i) )).c_str();
}
}
catch(Mimic::ErrorInfo e)
{
    cout << "ERROR: " << e.get_error_msg().c_str() << "\n";
}
//catch(...)
//{
    //cout << "ERROR: thread sleep interrupted" ;
    //return;
//}

}

void
test_session_protmsg (

```



```

) throw (Mimic::ErrorInfo)

{
//Mimic::Session *session = *(session_list.end());
std::string v3args = session->protocol_msg ( "snmpv3", "get args" );
cout<< "TestApp: snmpv3 arguments = " << v3args.c_str() << "\n";
}

void
test_session_diag (
) throw (Mimic::ErrorInfo)

{
//Mimic::Session *session = *(session_list.end());
session->diag( 2 );
cout<< "TestApp: check MIMICLog for dump agents\n" ;
}

void
test_session_agent_mget(
) throw (Mimic::ErrorInfo)

{
//Mimic::Session *session = *(session_list.end());
std::list<int> agents;
agents.push_back(1);
agents.push_back(3);
agents.push_back(5);

std::list<std::string> iface_list;
session->agent_mget_interface (agents,iface_list);
cout<<"TestApp: mget interfaces : " ;
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].iface = "
        << (*(get(iface_list.begin(),(int)i))).c_str() << "\n";
}

std::list<std::string> host_list;
session->agent_mget_host (agents,host_list);
cout<<"TestApp: mget host : " ;
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].host = "
        << (*(get(host_list.begin(),(int)i))).c_str() << "\n";
}

std::list<std::string> mask_list;
session->agent_mget_mask (agents,mask_list);
cout<< "TestApp: mget mask : " ;
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].mask = "
        << (*(get(mask_list.begin(),(int)i))).c_str() << "\n";
}

```

```

}

std::list<std::string> read_list;
session->agent_mget_read_community (agents,read_list);
cout<<"TestApp: mget read :";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<<" agent[" << *(get(agents.begin(),(int)i)) << "].read = "
        << *(get(read_list.begin(),(int)i)).c_str() << "\n";
}

std::list<std::string> write_list;
session->agent_mget_write_community (agents,write_list);
cout<<"TestApp: mget write :";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<<" agent[" << *(get(agents.begin(),(int)i)) << "].write = "
        << *(get(write_list.begin(),(int)i)).c_str() << "\n";
}

std::list< std::list<std::string> > mibs_list;
session->agent_mget_mibs ( agents,mibs_list );
cout<<"TestApp: mget MIBs :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<<" agent[" << *(get(agents.begin(),(int)i)) << "].mibs = ";
    std::list<std::string> mibs = *(get(mibs_list.begin(),(int)i));
    for (size_t j=0;j<mibs.size();j++)
    {
        cout<<" " << *(get(mibs.begin(),(int)j)).c_str();
    }
    cout << "\n";
}

std::list<std::string> sim_list;
session->agent_mget_sim (agents,sim_list);
cout<<"TestApp: mget sim :";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<<" agent[" << *(get(agents.begin(),(int)i)) << "].sim = "
        << *(get(sim_list.begin(),(int)i)).c_str() << endl;
}

std::list<std::string> scen_list;
session->agent_mget_scen (agents,scen_list);
cout<<"TestApp: mget scen :";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<<" agent[" << *(get(agents.begin(),(int)i)) << "].scen = "
        << *(get(scen_list.begin(),(int)i)).c_str() << endl;
}

std::list<int> pdu_size_list;
session->agent_mget_pdu_size(agents,pdu_size_list);

```

```

cout<<"TestApp: mget pdu size :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].pdu size = "
        << *(get(pdu_size_list.begin(),(int)i)) << "\n";
}

std::list< std::list<std::string> > protocols_list;
session->agent_mget_protocols( agents,protocols_list );
cout<< "TestApp: mget Protocols :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].protocols = ";
    std::list<std::string> protocols= *(get(protocols_list.begin(),(int)i));
    for (size_t j=0;j<protocols.size();j++)
    {
        cout<< " " << (*(get(protocols.begin(),(int)j))).c_str();
    }
    cout << "\n";
}

std::list<int> delay_list;
session->agent_mget_delay(agents,delay_list);
cout<<"TestApp: mget delay :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].delay = "
        << *(get(delay_list.begin(),(int)i)) << "\n";
}

std::list<int> starttime_list;
session->agent_mget_starttime(agents,starttime_list);
cout<<"TestApp: mget starttime :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].starttime = "
        << *(get(starttime_list.begin(),(int)i)) << "\n";
}

std::list<int> state_list;
session->agent_mget_state(agents,state_list);
cout<<"TestApp: mget state :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].state = "
        << *(get(state_list.begin(),(int)i)) << "\n";
}

std::list<std::string> statistics_list;
session->agent_mget_statistics (agents,statistics_list);
cout<<"TestApp: mget statistics :\n" ;
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].statistics = "

```

```

        << (*(get(statistics_list.begin(),(int)i)).c_str() << "\n";
    }

std::list<int> drops_list;
session->agent_mget_drops(agents,drops_list);
cout<<"TestApp: mget drops :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].drops = "
        << *(get(drops_list.begin(),(int)i)) << "\n";
}

std::list<int> informtimeout_list;
session->agent_mget_inform_timeout(agents,informtimeout_list);
cout<<"TestApp: mget informtimeout :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].inform_timeout = "
        << *(get(informtimeout_list.begin(),(int)i)) << "\n";
}

std::list<int> informretries_list;
session->agent_mget_inform_retries(agents,informretries_list);
cout<<"TestApp: mget informretries :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].inform_retries = "
        << *(get(informretries_list.begin(),(int)i)) << "\n";
}

std::list<int> chflag_list;
session->agent_mget_changed(agents,chflag_list);
cout<<"TestApp: mget changed :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].changed = "
        << *(get(chflag_list.begin(),(int)i)) << "\n";
}

std::list<int> cfgchflag_list;
session->agent_mget_config_changed(agents,cfgchflag_list);
cout<<"TestApp: mget config changed :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].config_changed = "
        << *(get(cfgchflag_list.begin(),(int)i)) << "\n";
}

std::list<int> stchflag_list;
session->agent_mget_state_changed(agents,stchflag_list);
cout<<"TestApp: mget state changed :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].statechanged = "

```

```

        << *(get(stchflag_list.begin(),(int)i)) << "\n";
    }

std::list<int> trace_list;
session->agent_mget_trace(agents,trace_list);
cout<<"TestApp: mget trace :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].trace = "
        << *(get(trace_list.begin(),(int)i)) << "\n";
}

std::list<int> validate_list;
session->agent_mget_validate(agents,validate_list);
cout<<"TestApp: mget validate :\n";
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].validate = "
        << *(get(validate_list.begin(),(int)i)) << "\n";
}

std::list<std::string> owner_list;
session->agent_mget_owner (agents,owner_list);
cout<<"TestApp: mget owner :\n" ;
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].owner = "
        << (*(get(owner_list.begin(),(int)i))).c_str() << "\n";
}

std::list<std::string> privdir_list;
session->agent_mget_privdir (agents,privdir_list);
cout<< "TestApp: mget privdir :\n" ;
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].privdir"
        << (*(get(privdir_list.begin(),(int)i))).c_str() << "\n";
}

std::list<std::string> oiddir_list;
session->agent_mget_oiddir(agents,oiddir_list);
cout << "TestApp mget: oiddir " ;
for (size_t i=0; i<agents.size(); i++ )
{
    cout<< " agent[" << *(get(agents.begin(),(int)i)) << "].oiddir"
        << (*(get(oiddir_list.begin(),(int)i))).c_str() << "\n";
}
}

void
test_session_agent_mset () throw (Mimic::ErrorInfo)
{

```

```
//Mimic::Session *session = *(session_list.end());
std::list<int> agents;
agents.push_back(1);
agents.push_back(3);
agents.push_back(5);
```

```
std::list<std::string> iface_list ;
iface_list.push_back( "if1" );
iface_list.push_back( "if2" );
iface_list.push_back( "if 3" );
session->agent_mset_interface ( agents, iface_list );
cout<< "TestApp: mset interfaces.\n";
```

```
std::list<std::string> host_list ;
host_list.push_back("10.11.12.1" );
host_list.push_back("10.11.12.2" );
host_list.push_back("10.11.12.2" );
session->agent_mset_host ( agents, host_list );
cout<< "TestApp: mset host.\n" ;
```

```
std::list<std::string> mask_list;
mask_list.push_back( "255.255.0.0" );
mask_list.push_back( "255.255.0.0" );
mask_list.push_back( "255.255.0.0" );
session->agent_mset_mask ( agents, mask_list );
cout<< "TestApp: mset mask.\n" ;
```

```
// TODO rest
```

```
std::list<std::string> prots_list ;
prots_list.push_back( "snmpv1,snmpv2c" );
prots_list.push_back( "snmpv2c,snmpv2" );
prots_list.push_back( "snmpv2,snmpv3" );
session->agent_mset_protocol ( agents, prots_list );
cout<< "TestApp: mset protocols.\n" ;
```

```
// TODO rest
```

```
}
```

```
void
test_session_varstore () throw (Mimic::ErrorInfo)
{
    try
    {
        std::list<std::string> list;
        session->store_list(list);
        cout<< "TestApp: session->store_list = [";
        for (size_t i =0;i<list.size();i++)
        {
            if (i > 0)
                cout << ", ";
```

```

    cout << (*(get(list.begin(),(int)i))).c_str();
}
cout << "]\n";
int exists;
exists = session->store_exists ("uwe");
cout<< "TestApp: session->store_exists (uwe) = " << exists<< "\n";

// test different combinations of strings to set and append
struct {
    char* val;
    char* appval;
} pairvals[] = { {"simple", "st"},
    {"ab c", "de f"},
    {"{", "}"},
    {"[", "}"},
    {"a{", "}"},
    {"[a", "b}"},
    {"{ a", "b }"},
    {"a {", "b }"},
    {"\"", "a"},
    {"a \\"", "b"},
    {NULL, NULL} };
for (int pairi = 0; true; pairi++)
{
    char* setval = pairvals[pairi].val;
    char* appval = pairvals[pairi].appval;

    if (setval == NULL)
        break;

    cout << "*** val " << setval << " appval " << appval << "\n";
    session->store_set("uwe", setval, 0);
    session->store_list(list);
    cout<< "TestApp: session->store_list = [";
    for (size_t i =0;i<list.size();i++)
    {
        if (i > 0)
            cout << ", ";
        cout << (*(get(list.begin(),(int)i))).c_str();
    }
    cout << "]\n";
    exists = session->store_exists ("uwe");
    cout<< "TestApp: session->store_exists (uwe) = " << exists<< "\n";
    int persists;
    persists = session->store_persist ("uwe");
    cout<< "TestApp: session->store_persist (uwe) = " << persists<< "\n";
    std::string val = session->store_get("uwe");
    cout<< "TestApp: session->store_get(uwe) = " << val.c_str() << "\n";
    session->store_append("uwe", appval, 0);
    val = session->store_get("uwe");
    cout<< "TestApp: session->store_get(uwe) = " << val.c_str() << "\n";
    session->store_unset("uwe");
    session->store_list(list);
    cout<< "TestApp: session->store_list = [";

```

```

        for (size_t i =0;i<list.size();i++)
        {
            if (i > 0)
                cout << ", ";
            cout << (*(get(list.begin(),(int)i))).c_str();
        }
        cout << "]\n";
    }
}
catch(Mimic::ErrorInfo e)
{
    cout << "ERROR: " << e.get_error_msg().c_str() << "\n";
}
}

void
test_agent_state_changes () throw (Mimic::ErrorInfo)
{
    try
    {
        //Mimic::Session *session = *(session_list.end());
        Mimic::Agent& agent = session->get_agent(1);

        session->cfg_load("agent.cfg"); // reload

        agent.start();
        cout << "TestApp: agent 1, started.\n" ;

        while (agent.get_state() != 1)
        {
            //clock_t endwait;
            //endwait = clock() + 1000 * CLK_TCK;
            //while (clock() < endwait) {} ;

            sleep(1);
        }
        cout << "TestApp: agent 1, state " + agent.get_state() << "\n";
        agent.pause_now();
        cout << "TestApp: agent 1, paused at NOW.\n" ;
        while (agent.get_state() != 4)
        {
            //clock_t endwait;
            //endwait = clock() + 1000 * CLK_TCK;
            //while (clock() < endwait) {} ;
            sleep(1);
        }

        agent.resume();
        cout << "TestApp: agent 1, resumed.\n" ;
        while (agent.get_state() != 1)
        {
            //clock_t endwait;
            //endwait = clock() + 1000 * CLK_TCK;
            //while (clock() < endwait) {} ;

```



```

        sleep(1);
    }

    agent.pause_at(1000000);
    cout << "TestApp: agent 1, paused at 1000000 secs.\n" ;
    while (agent.get_state() != 4)
    {
        //clock_t endwait;
        //endwait = clock() + 1000 * CLK_TCK;
        //while (clock() < endwait) {} ;
        sleep(1);
    }

    agent.resume();
    cout << "TestApp: agent 1, resumed.\n" ;
    while (agent.get_state() != 1)
    {
        //clock_t endwait;
        //endwait = clock() + 1000 * CLK_TCK;
        //while (clock() < endwait) {} ;
        sleep(1);
    }

    agent.halt();
    cout << "TestApp: agent 1, halted.\n" ;
    while (agent.get_state() != 3)
    {
        //clock_t endwait;
        //endwait = clock() + 1000 * CLK_TCK;
        //while (clock() < endwait) {} ;
        sleep(1);
    }

    agent.resume();
    cout << "TestApp: agent 1, resumed.\n" ;
    while (agent.get_state() != 1)
    {
        //clock_t endwait;
        //endwait = clock() + 1000 * CLK_TCK;
        //while (clock() < endwait) {} ;
        sleep(1);
    }

    agent.stop();
    cout << "TestApp: agent 1, stopped.\n" ;
    while (agent.get_state() != 2)
    {
        //clock_t endwait;
        //endwait = clock() + 1000 * CLK_TCK;
        //while (clock() < endwait) {} ;
        sleep(1);
    }
}
catch(Mimic::ErrorInfo e)

```

```

{
    cout << "ERROR: " << e.get_error_msg().c_str() << "\n";
}
//catch (...)
//{{
    //cout << "ERROR: thread sleep interrupted" ;
    //return;
//}}
}

```

```

void
test_agent_cfg () throw (Mimic::ErrorInfo)
{
    try
    {
        //Mimic::Session *session = *(session_list.end());
        Mimic::Agent& agent = session->get_agent(11);
        std::list<std::string> mibs ;
        mibs.push_back( "windows-nt-4.0.random,RFC1213-MIB,1" );
        mibs.push_back( "windows-nt-4.0.random,IF-MIB,1" );
        mibs.push_back( "windows-nt-4.0.random,microsoft/LanMgr-Mib-II-MIB,1" );

        agent.configure ( "192.9.215.11", mibs );
        cout<< "TestApp: agent 11, configured as a windows-nt 4.0 simulation\n" ;

        //clock_t endwait;
        //endwait = clock() + 15000 * CLK_TCK;
        //while (clock() < endwait) { } ;
        sleep(15);
        agent.remove();
        cout<< "TestApp: agent 11, deleted.\n" ;
    }
    catch(Mimic::ErrorInfo e)
    {
        cout << "ERROR: " << e.get_error_msg().c_str() << "\n";
    }
    //catch (...)
    //{
        //cout<< "ERROR: thread sleep interrupted" ;
        //return;
    //}
}

```

```

void
test_agent_set () throw (Mimic::ErrorInfo)
{

    //Mimic::Session *session = *(session_list.end());
    Mimic::Agent& agent = session->get_agent(1);
}

```

```

cout << "TestApp: agent 1, setting config values...\n" ;

agent.set_interface ( "testif" );

agent.set_host ( "1.2.3.4" );

agent.set_mask ( "255.255.0.0" );

agent.set_read_community ( "testread" );

agent.set_write_community ( "testwrite" );

std::list<std::string> mibs ;
mibs.push_back( "windows-nt-4.0.random,RFC1213-MIB,1" );
mibs.push_back( "windows-nt-4.0.random,IF-MIB,1" );
mibs.push_back( "windows-nt-4.0.random,microsoft/LanMgr-Mib-II-MIB,1" );
agent.set_mibs( mibs );

agent.set_pdu_size ( 2222 );

std::list<std::string> protocols ;
protocols.push_back( "snmpv1" );
protocols.push_back( "snmpv3" );
// protocols.push_back( "DHCP" );
agent.set_protocols( protocols );
agent.set_delay ( 1234 );
agent.set_starttime ( 56789 );
agent.set_drops ( 135 );
agent.set_trace ( 1 );
agent.set_validate ( 0xF0F0 );
agent.set_oiddir ( "test.dir" );
agent.set_inform_timeout ( 15 );
agent.set_inform_retries ( 20 );
agent.trap_config_add ( "192.9.240.4", 162);
agent.trap_config_add ( "192.9.200.5", 162);
}

void
test_agent_get() throw(Mimic::ErrorInfo)
{
    //Mimic::Session *session = *(session_list.end());
    Mimic::Agent& agent = session->get_agent(1);
    std::string iface = agent.get_interface ();
    cout<< "TestApp: agent[1].interface = " << iface.c_str() << "\n";
    std::string host = agent.get_host ();
    cout<< "TestApp: agent[1].host = " << host.c_str() << "\n";
    std::string mask = agent.get_mask ();
    cout<< "TestApp: agent[1].mask = " << mask.c_str() << "\n";
    std::string read = agent.get_read_community ();
    cout<< "TestApp: agent[1].read = " << read.c_str() << "\n";
    std::string write = agent.get_write_community ();
    cout<< "TestApp: agent[1].write = " << write.c_str() << "\n";
}

```

```

std::list<std::string> mibs ;
agent.get_mibs(mibs);
cout<< "TestApp: agent[1].MIBs = [";
for(size_t i = 0;i < mibs.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout<< (*(get(mibs.begin(),(int)i))).c_str() ;
}
cout << "\n";

std::string sim = agent.get_sim ();
cout<< "TestApp: agent[1].sim = " << sim.c_str() << "\n";

std::string scen = agent.get_scen ();
cout<< "TestApp: agent[1].scen = " << scen.c_str() << "\n";

int pdu_size = agent.get_pdu_size ();
cout<< "TestApp: agent[1].PDUsize = " << pdu_size << "\n";

std::list<std::string> protocols ;
agent.get_protocols(protocols);
cout<< "TestApp: agent[1].protocols = [";
for(size_t i = 0;i < protocols.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(protocols.begin(),(int)i))).c_str();
}
cout << "]\n";

int delay = agent.get_delay ();
cout<< "TestApp: agent[1].delay = " << delay << "\n";
int starttime = agent.get_starttime ();
cout<< "TestApp: agent[1].starttime = " << starttime << "\n";

int state = agent.get_state ();
cout<< "TestApp: agent[1].state = " << state << "\n";
std::list<std::string> statistics;
agent.get_statistics(statistics);

cout<< "TestApp: agent[1].statistics = [";
for(size_t i = 0;i < statistics.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << *(get(statistics.begin(),(int)i));
}
cout << "]\n";

int drops = agent.get_drops ();
cout<< "TestApp: agent[1].drops = " << drops << "\n";

int chflag = agent.get_changed ();

```

```

cout<< "TestApp: agent[1].changed = " << chflag << "\n";

int cfgchflag = agent.get_config_changed ();
cout<< "TestApp: agent[1].config_changed = " << cfgchflag << "\n";
int stchflag = agent.get_state_changed ();
cout<< "TestApp: agent[1].state_changed = " << stchflag << "\n";
int trace = agent.get_trace ();
cout<< "TestApp: agent[1].trace = " << trace << "\n";
int validate = agent.get_validate ();
cout<< "TestApp: agent[1].validate = " << validate << "\n";
std::string owner = agent.get_owner ();
cout<< "TestApp: agent[1].owner = " << owner.c_str() << "\n";

std::string privdir = agent.get_privdir ();
cout<< "TestApp: agent[1].privdir = " << privdir.c_str() << "\n";

std::string oiddir = agent.get_oiddir ();
cout<< "TestApp: agent[1].oiddir = " << oiddir.c_str() << "\n";

int inform_timeout = agent.get_inform_timeout ();
cout<< "TestApp: agent[1].inform_timeout = " << inform_timeout << "\n";

int inform_retries = agent.get_inform_retries ();
cout<< "TestApp: agent[1].inform_retries = " << inform_retries << "\n";

std::list<std::string> trap_config_list;
agent.trap_config_list (trap_config_list);
cout<< "TestApp: agent[1].trap_config_list = [";
for(size_t i = 0;i < trap_config_list.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(trap_config_list.begin(),(int)i))).c_str();
}
cout << "]\n";

agent.trap_config_del ( "192.9.200.5", 162);
cout<< "TestApp: agent 1, removed trap destination 192.9.200.5,162\n";

std::list<std::string> trap_config_list1 ;
agent.trap_config_list (trap_config_list1);
cout << "TestApp: agent[1].trap_config_list = [";
for(size_t i = 0;i < trap_config_list1.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(trap_config_list1.begin(),(int)i))).c_str();
}
cout << "]\n";
}

```

```

void
test_agent_ipalias () throw (Mimic::ErrorInfo)

```

```

{
try
{
//Mimic::Session *session = *(session_list.end());
Mimic::Agent& agent = session->get_agent(1);

// ipalias list should be empty
std::list<std::string> list0 ;
agent.list_ipaliases (list0);
cout << "TestApp: agent[1].ipaliases = [";
for (size_t i =0;i<list0.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(list0.begin(),(int)i))).c_str();
}
cout << "]\n";

agent.add_ipalias ( "192.9.216.1", 161, "", "" );
cout<< "TestApp: agent 1, added ipalias 192.9.216.1,161\n" ;

//clock_t endwait;
//endwait = clock() + 1000 * CLK_TCK;
//while (clock() < endwait) { } ;
sleep(1);
std::list<std::string> list1 ;
agent.list_ipaliases (list1);
cout << "TestApp: agent[1].ipaliases = [";
for (size_t i =0;i<list1.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(list1.begin(),(int)i))).c_str();
}
cout << "]\n";

agent.add_ipalias ( "192.9.216.2", 161, "", "" );
cout<< "TestApp: agent 1, added ipalias 192.9.216.2,161\n" ;
//endwait = clock() + 1000 * CLK_TCK;
//while (clock() < endwait) { } ;
sleep(1);

std::list<std::string> list2 ;
agent.list_ipaliases (list2);
cout << "TestApp: agent[1].ipaliases = [";
for (size_t i =0;i<list2.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(list2.begin(),(int)i))).c_str();
}
cout << "]\n";

agent.start();

```

```

cout<< "TestApp: agent 1, started.\n" ;
while (agent.get_state() != 1)
{
    //clock_t endwait;
    //endwait = clock() + 1000 * CLK_TCK;
    //while (clock() < endwait) { } ;
    sleep(1);
}
cout<< "TestApp: agent 1, state " << agent.get_state() << "\n";
sleep(10);
agent.stop_ipalias ( "192.9.216.1", 161 );
cout<< "TestApp: agent 1, stopped ipalias 192.9.216.1,161\n" ;
//clock_t endwait;
//endwait = clock() + 1000 * CLK_TCK;
//while (clock() < endwait) { } ;
sleep(5);
agent.start_ipalias ( "192.9.216.1", 161 );
cout<< "TestApp: agent 1, restarted ipalias 192.9.216.1,161\n" ;
//endwait = clock() + 1000 * CLK_TCK;
//while (clock() < endwait) { } ;
sleep(5);
agent.stop();

cout<< "TestApp: agent 1, stopped.\n" ;
//endwait = clock() + 1000 * CLK_TCK;
//while (clock() < endwait) { } ;
sleep(1);
agent.del_ipalias ( "192.9.216.1", 161 );
cout<< "TestApp: agent 1, deleted ipalias 192.9.216.1,161\n" ;
//endwait = clock() + 1000 * CLK_TCK;
//while (clock() < endwait) { } ;
sleep(1);
std::list<std::string> list3 ;
agent.list_ipaliases (list3);
cout << "TestApp: agent[1].ipaliases = [";
for (size_t i =0;i<list3.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(list3.begin(),(int)i))).c_str();
}
cout << "]\n";

agent.del_ipalias ( "192.9.216.2", 161 );
cout<< "TestApp: agent 1, deleted ipalias 192.9.216.2,161\n" ;

// ipalias list should be empty
std::list<std::string> list4 ;
agent.list_ipaliases (list4);
cout<< "TestApp: agent[1].ipaliases = [";
for (size_t i =0;i<list4.size();i++)
{
    if (i > 0)

```

```

        cout << ", ";
        cout << (*(get(list4.begin(),(int)i))).c_str();
    }
    cout << "]\n";
}
catch(Mimic::ErrorInfo e)
{
    cout << "ERROR: " << e.get_error_msg().c_str() << "\n";
}
//catch ( ... )
//{
    //cout<< "ERROR: thread sleep interrupted" ;
    //return;
//}
}

void
test_agent_varstore () throw (Mimic::ErrorInfo)
{
    try
    {
        //Mimic::Session *session = *(session_list.end());
        Mimic::Agent& agent = session->get_agent(1);
        std::list<std::string> list;
        agent.agent_store_list(list);
        cout<< "TestApp: agent[1].agent_store_list = [";
        for (size_t i =0;i<list.size();i++)
        {
            if (i > 0)
                cout << ", ";
            cout << (*(get(list.begin(),(int)i))).c_str();
        }
        cout << "]\n";
        int exists;
        exists = agent.agent_store_exists ("uwe");
        cout<< "TestApp: agent[1].agent_store_exists (uwe) = " << exists<< "\n";
        agent.agent_store_set("uwe", "ab c", 0);
        agent.agent_store_list(list);
        cout<< "TestApp: agent[1].agent_store_list = [";
        for (size_t i =0;i<list.size();i++)
        {
            if (i > 0)
                cout << ", ";
            cout << (*(get(list.begin(),(int)i))).c_str();
        }
        cout << "]\n";
        exists = agent.agent_store_exists ("uwe");
        cout<< "TestApp: agent[1].agent_store_exists (uwe) = " << exists<< "\n";
        int persists;
        persists = agent.agent_store_persists ("uwe");
        cout<< "TestApp: agent[1].agent_store_persists (uwe) = " << persists<< "\n";
        std::string val = agent.agent_store_get("uwe");
        cout<< "TestApp: agent[1].agent_store_get(uwe) = " << val.c_str() << "\n";
        agent.agent_store_append("uwe", "de f", 0);
    }
}

```



```

    val = agent.agent_store_get("uwe");
    cout<< "TestApp: agent[1].agent_store_get(uwe) = " << val.c_str() << "\n";
    agent.agent_store_unset("uwe");
    agent.agent_store_list(list);
    cout<< "TestApp: agent[1].agent_store_list = [";
    for (size_t i =0;i<list.size();i++)
    {
        if (i > 0)
            cout << ", ";
        cout << (*(get(list.begin(),(int)i))).c_str();
    }
    cout << "]\n";
}
catch(Mimic::ErrorInfo e)
{
    cout << "ERROR: " << e.get_error_msg().c_str() << "\n";
}
}

void
test_agent_valuespace () throw (Mimic::ErrorInfo)
{
    try
    {
        //Mimic::Session *session = *(session_list.end());
        Mimic::Agent& agent = session->get_agent(1);
        Mimic::Valuespace& vspace = agent.get_valuespace();

        session->cfg_load("agent.cfg"); // reload

        agent.start();
        cout<< "TestApp: agent 1, started." << "\n";
        while (agent.get_state() != 1)
        {
            //clock_t endwait;
            //endwait = clock() + 1000 * CLK_TCK;
            //while (clock() < endwait) { } ;
            sleep(1);
        }
        cout<< "TestApp: agent 1, state " << agent.get_state() << "\n";
        std::string pos1 = vspace.get_pos();
        cout<< "TestApp: agent[1].vspace.pos = " << pos1.c_str() << "\n";

        std::string pos2 = vspace.set_pos( "ifEntry" );
        cout<< "TestApp: agent[1].vspace.pos = " << pos2.c_str() << "\n";

        std::string state1 = vspace.get_state( "ifTable" );
        cout<< "TestApp: agent[1].vspace.state of ifTable = " << state1.c_str() << "\n";

        cout<< "TestApp: agent[1] disabling ifTable subtree" << "\n";
        vspace.set_state( "ifTable" , 0);

        std::string state2 = vspace.get_state( "ifDescr" );
        cout<< "TestApp: agent[1].vspace.state of ifDescr = " << state2.c_str() << "\n";
    }
}

```

```

std::list<std::string> ch_list;
vspace.get_objects(ch_list);
cout<< "TestApp: agent[1].vspace.objects = [";
for (size_t i=0;i<ch_list.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout<< (*(get(ch_list.begin(),(int)i))).c_str();
}
cout << "]\n";

//Oid oid;
//oid.sscanf("51");
std::string oid("51");
vspace.add( "ifEntry", oid );
cout<< "TestApp: agent[1].ifTable, added instance 51" << "\n";

std::list<std::string> inst_list;
vspace.get_instances("ifDescr",inst_list);
cout<< "TestApp: agent[1].ifDescr instances = [";
for (size_t i=0;i<inst_list.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout<< (*(get(inst_list.begin(),(int)i))).c_str();
}
cout << "]\n";

vspace.remove( "ifEntry", oid );
cout<< "TestApp: agent[1].ifTable, removed instance 51" << "\n";

std::list<std::string> var_list;
oid="1";
vspace.get_variables( "ifInOctets", oid,var_list);
cout<< "TestApp: agent[1].ifInOctets.1 variables = [";
for (size_t i=0;i<var_list.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout<< (*(get(var_list.begin(),(int)i))).c_str();
}
cout << "]\n";

oid="0";
std::string get_val = vspace.get_value( "sysDescr",oid, "v" );
cout<< "TestApp: agent[1].sysDescr.0 get value = " << get_val.c_str() << "\n";

std::list<std::string> objs1;
objs1.push_back( "sysDescr" );
objs1.push_back( "sysContact" );
std::list<std::string> insts1;
std::string oid1("0"),oid2("0");
//oid1.sscanf("0");oid2.sscanf("0");
insts1.push_back( oid1 );

```

```

insts1.push_back( oid2 );
std::list<std::string> vars1;
vars1.push_back( "v" );
vars1.push_back( "v" );
std::list<std::string> mget_val;
vspace.mget_value( objs1, insts1, vars1,mget_val);
cout << "TestApp: agent[1].sysDescr.0,sysContact.0 mget value = [";
for (size_t i=0;i<mget_val.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(mget_val.begin(),(int)i))).c_str();
}
cout << "]\n";

vspace.set_value( "sysContact", oid1, "v", "Prashant" );
cout << "TestApp: agent[1].sysContact.0 set value = " << "Prashant"<< "\n";

std::string eval_val = vspace.eval_value( "sysContact", oid1 );
cout<< "TestApp: agent[1].sysContact.0 eval value = " << eval_val.c_str() << "\n";

std::list<std::string> objs2;
objs2.push_back( "sysDescr" );
objs2.push_back( "sysContact" );
std::list<std::string> insts2;
insts2.push_back( oid1 );
insts2.push_back( oid2 );
std::list<std::string> meval_val;
vspace.meval_value(objs2, insts2,meval_val);
cout<< "TestApp: agent[1].sysDescr.0,sysContact.0 meval value = [";
for (size_t i=0;i<meval_val.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(meval_val.begin(),(int)i))).c_str();
}
cout << "]\n";

vspace.unset_value( "sysContact", oid1, "v" );
cout<< "TestApp: agent[1].sysContact.0 unset value " << "\n";
std::list<std::string> objs3;
objs3.push_back( "sysDescr" );
objs3.push_back( "sysContact" );
objs3.push_back( "linkUp" );
objs3.push_back( "linkUp" );
objs3.push_back( "linkUp" );
objs3.push_back( "linkUp" );
objs3.push_back( "linkDown" );
objs3.push_back( "linkDown" );
objs3.push_back( "linkDown" );
objs3.push_back( "linkDown" );
std::list<std::string> insts3;
insts3.push_back( oid1 );
insts3.push_back( oid1 );

```

```

insts3.push_back( oid1);
insts3.push_back( oid1);
insts3.push_back( oid1);
insts3.push_back( oid1);
insts3.push_back( oid1);
insts3.push_back( oid1);
insts3.push_back( oid1);
insts3.push_back( oid1);
std::list<std::string> vars3 ;
vars3.push_back( "v" );
vars3.push_back( "v" );
vars3.push_back( "v" );
vars3.push_back( "r" );
vars3.push_back( "tu" );
vars3.push_back( "c" );
vars3.push_back( "v" );
vars3.push_back( "r" );
vars3.push_back( "tu" );
vars3.push_back( "c" );
std::list<std::string> vals3;
vals3.push_back( "test system" );
vals3.push_back( "prashant juvekar" );
vals3.push_back( "1" );
vals3.push_back( "1" );
vals3.push_back( "1" );
vals3.push_back( "120" );
vals3.push_back( "1");
vals3.push_back( "1");
vals3.push_back( "1" );
vals3.push_back( "120" );
vspace.mset_value(objs3, insts3, vars3, vals3 );
cout<< "TestApp: mset value" << "\n";
vspace.mget_value( objs3, insts3, vars3,mget_val);
cout << "TestApp: mget value = [";
for (size_t i=0;i<mget_val.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(mget_val.begin(),(int)i))).c_str();
}
cout << "]\n";
vspace.munset_value(objs3, insts3, vars3 );
cout<< "TestApp: unset value" << "\n";
bool failed = false;
try
{
    vspace.mget_value( objs3, insts3, vars3,mget_val);
}
catch(Mimic::ErrorInfo e)
{
    failed = true;
}
if ( !failed )
{

```

```

cout << "TestApp: mget should have failed, instead it returned [";
for (size_t i=0;i<mget_val.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(mget_val.begin(),(int)i))).c_str();
}
cout << "]\n";
}

std::list<std::string> trap_list;
agent.trap_list(trap_list);
cout<< "TestApp: agent[1].traps = [";
for (size_t i=0;i<trap_list.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(trap_list.begin(),(int)i))).c_str();
}
cout << "]\n";

std::string oid3 = vspace.get_oid( "ifNumber" );
cout<< "TestApp: agent[1].ifNumber OID = " << oid3.c_str() << "\n";
std::string oid4("1.3.6.1.2.1");
//oid4.sscanf("1.3.6.1.2.1");
std::string name1 = vspace.get_name( oid4 );
cout<< "TestApp: agent[1].1.3.6.1.2.1 NAME = " << name1.c_str() << "\n";
std::string mib1 = vspace.get_mib( "ifOutOctets" );
cout<< "TestApp: agent[1].ifOutOctets MIB = " << mib1.c_str() << "\n";
std::list<std::string> info1;
vspace.get_info( "sysUpTime",info1 );
cout<< "TestApp: agent[1].sysUpTime info = [";
for (size_t i=0;i<info1.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout<< (*(get(info1.begin(),(int)i))).c_str();
}
cout << "]\n";

cout<< "TestApp: Testing with instances having subids larger than 2147483647.\n";
oid4="2147483650";
vspace.add( "ifEntry", oid4 );
cout<<"TestApp: agent[1].ifTable, added instance 2147483650" << "\n";
//verify inst_list
vspace.get_instances("ifDescr",inst_list );
cout<< "TestApp: agent[1].ifDescr instances = [";
for(size_t i=0;i<inst_list.size();i++)
{
    if (i > 0)
        cout << ", ";
    cout << (*(get(inst_list.begin(),(int)i))).c_str();
}
cout << "]\n";

```

```

vspace.remove( "ifEntry", oid4 );
cout<< "TestApp: agent[1].ifTable, removed instance 2147483650" << "\n";
agent.stop();
cout<< "TestApp: agent 1, stopped." << "\n";
while (agent.get_state() != 2)
{
    //clock_t endwait;
    //endwait = clock() + 1000 * CLK_TCK;
    //while (clock() < endwait) { } ;
    sleep(1);
}
cout<< "TestApp: agent 1, state " << agent.get_state() << "\n";
}
catch(Mimic::ErrorInfo e)
{
    cout << "ERROR: " << e.get_error_msg().c_str() << "\n";
}
//catch ( ... )
//{
    //cout<<"ERROR: thread sleep interrupted" ;
    //return;
//}
}

/*void
test_thread_safety () throw (Mimic::ErrorInfo)
{

    GciThreadMgr thread_mgr;
    GciThread *thread1,*thread2;
    session->cfg_load("agent.cfg"); // reload

    thread1 = thread_mgr.start_thread (
        1,
        _thread_func,
        (void*)&session,
        NULL,
        NULL
    );

    threadno++;
    thread2 = thread_mgr.start_thread (
        1,
        _thread_func,
        (void*)&session,
        NULL,
        NULL
    );

    while (true)
    {
        // sleep on...
        try

```

```

    {
        if ( thread1->done() && thread2->done() )
        {
            break;
        }
        //GciThreadMgr::sleep (10, 0);

    }
    catch (...)
    {
        break;
    }
}
*/
int main(int argc,char *argv[])
{
    _init();
    cout << "TestApp: starting...\n";
    try
    {
        parse_args (argc, argv);
        test_session_init ();

        test_3510 ();

        test_session_get();
        // test_session_timerscripts ();
        // test_session_access (): // Not implemented yet
        test_session_protmsg ();
        test_session_diag ();
        test_session_agent_mget();
        test_session_agent_mset ();
        test_session_agent_mget();
        test_session_varstore ();

        test_agent_cfg ();
        test_agent_set ();
        test_agent_get();
        test_agent_varstore ();
        test_agent_valuespace ();
        test_agent_ipalias ();
        test_agent_state_changes ();

        // test_thread_safety ();

        // test_session_uninit (); // Not implemented yet
    }
    catch ( Mimic::ErrorInfo errorinfo )
    {
        cout<< errorinfo.get_error_msg().c_str() << "\n";
        return -1;
    }
}

```

```
cout<< "TestApp: done.\n" ;
```

```
return 0;
```

```
}
```



```

////////////////////////////////////
// Copyright (c) 2006 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
//
// This is a sample code for timer scripts in C++
//
////////////////////////////////////
#ifdef WIN32
    #pragma warning(disable:4786)
    #pragma warning(disable:4290)
#endif

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "mimic_action.hh"

static int
_on_action_module_register (
    action_module_register_t *data
)
{
    fprintf ( stderr, "INFO - received message : ACTION_MODULE_REGISTER.\n" );

    // verify version for the simulator
    data->version.sprintf ( ACTION_BUILD_ID );

    return ACTION_SUCCESS;
}

static int
_on_action_module_init (
    action_module_init_t *data
)
{
    fprintf ( stderr, "INFO - received message - ACTION_MODULE_INIT. ***\n" );

    return ACTION_SUCCESS;
}

static int
_on_action_module_uninit (
    action_module_uninit_t *data
)
{
    // UWE: cannot do fprintf stderr in UNINIT for some reason
    // valgrind complains:
    //
    // ==14005== Invalid read of size 4
    // ==14005==   at 0x37C6F8EB: _on_action_module_uninit(_action_module_uninit_t*) (testtimer.cc:48)
    // ==14005==   by 0x37C6FEE6: process_action_message (testtimer.cc:132)

```

```

//
fprintf ( stderr, "INFO - received message - ACTION_MODULE_UNINIT. ***\n" );
// DO ANY cleanups here

return ACTION_SUCCESS;
}

static int
_on_action_module_run (
    action_module_run_t *data
)
{
    fprintf ( stderr, "INFO - received message - ACTION_MODULE_RUN. ***\n" );
    // visible variables
    fprintf ( stderr, "    - accessing variables :\n" );
    fprintf ( stderr, "    - gCurrentAgent = %s\n", data->globals->get("gCurrentAgent"));
    fprintf ( stderr, "    - gArguments = %s\n", data->globals->get("gArguments"));
    fprintf ( stderr, "    - argv = %s\n", data->globals->get("argv"));
    fprintf ( stderr, "    - argc = %s\n", data->globals->get("argc"));
    fprintf ( stderr, "    - gNextInterval = %s\n", data->globals->get("gNextInterval"));
    fprintf ( stderr, "    - gSharedDir = %s\n", data->globals->get("gSharedDir"));
    fprintf ( stderr, "    - gPrivateDir = %s\n", data->globals->get("gPrivateDir"));
    fprintf ( stderr, "    - gOwner = %s\n", data->globals->get("gOwner"));

    // some mimic global commands
    fprintf ( stderr, "    - exercising application global commands :\n" );
    fprintf ( stderr, "    - mimic get version = %s\n", data->session->get_version().c_str() );
    fprintf ( stderr, "    - mimic get cfgfile = %s\n", data->session->get_cfgfile().c_str() );
    fprintf ( stderr, "    - mimic get clients = %d\n", data->session->get_clients() );
    fprintf ( stderr, "    - mimic get last    = %d\n", data->session->get_last() );
    fprintf ( stderr, "    - mimic get max     = %d\n", data->session->get_max() );

    // some agent info
    Mimic::Agent& agent = data->session->get_agent(1);
    fprintf ( stderr, "    - exercising agent commands :\n" );
    fprintf ( stderr, "    - mimic agent get host = %s\n", agent.get_host().c_str() );
    fprintf ( stderr, "    - mimic agent get port = %d\n", agent.get_port() );

    fflush ( stderr );

    // cancel the timer
    fprintf ( stderr, "setting gNextInterval to -1\n" );
    data->globals->set("gNextInterval", "-1");

    return ACTION_SUCCESS;
}

#ifdef __cplusplus
extern "C" {
#endif

//////////
// DLL entry point

```

```
//////////
```

```
#ifdef WIN32
__declspec( dllexport )
#endif
int
process_action_message (
    int msg_id,
    void* msg_data
)
{
    switch( msg_id )
    {
        case ACTION_MODULE_REGISTER:
        {
            return _on_action_module_register (
                (action_module_register_t*)msg_data
            );
        }

        case ACTION_MODULE_INIT:
        {
            return _on_action_module_init (
                (action_module_init_t*)msg_data
            );
        }

        case ACTION_MODULE_UNINIT:
        {
            return _on_action_module_uninit (
                (action_module_uninit_t*)msg_data
            );
        }

        case ACTION_MODULE_RUN:
        {
            return _on_action_module_run (
                (action_module_run_t*)msg_data
            );
        }
    }

    return ACTION_SUCCESS; // default success
}

#ifdef __cplusplus
}
#endif
```

```
////////////////////////////////////
// Copyright (c) 2002 by Gambit Communications, Inc.
// All Rights Reserved
////////////////////////////////////
```

```
#ifdef WIN32
#pragma warning(disable:4786)
#pragma warning(disable:4290)
#endif
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>          // for va_*
```

```
#include "mimic_action.hh"
```

```
// defines
#ifdef WIN32
#define vsnprintf _vsnprintf
#endif
```

```
// toggle following flag to enable/disable debugging
static int _debug_on = 0;
```

```
// other globals
static int _time_value;
static int _nlMatrixSDTable_seeded;
static int _nlMatrixDSTable_seeded;
```

```
static void _debug ( const char* fmt, ... )
{
    if ( !_debug_on )
        return;

    char tmp[16*1024];    // NOTE: fixed size, 16K at a time

    // create string in tmp buffer
    va_list ap;
    va_start (ap, fmt);
    vsnprintf (tmp, sizeof (tmp), fmt, ap);
    va_end (ap);

    fprintf ( stderr, tmp );
}
```

```
static void _msg ( const char *prefix, const char* fmt, ... )
{
    char tmp[16*1024];    // NOTE: fixed size, 16K at a time

    // create string in tmp buffer
    va_list ap;
```

```

va_start (ap, fmt);
vsprintf (tmp, sizeof (tmp), fmt, ap);
va_end (ap);

fprintf ( stderr, "%-5s - ", prefix );
fprintf ( stderr, tmp );
}

static int
_on_action_module_register (
    action_module_register_t *data
)
{
    // verify version for the simulator
    data->version.sprintf ( ACTION_BUILD_ID );

    return ACTION_SUCCESS;
}

static int
_on_action_module_init (
    action_module_init_t *data
)
{
    _time_value = 0;
    _nlMatrixSDTable_seeded = 0;
    _nlMatrixDSTable_seeded = 0;

    return ACTION_SUCCESS;
}

static int
_on_action_module_uninit (
    action_module_uninit_t *data
)
{
    // nothing TODO

    return ACTION_SUCCESS;
}

static int
_update_nlMatrixSDTable (
    Mimic::Agent& agent,
    int timevalue
)
{
    Mimic::Valuespace& valuespace = agent.get_valuespace();

    // check if nlMatrixSDTable is supported
    std::string cur_pos = valuespace.set_pos ( "nlMatrixSDTable" );
    char *obj = (char*)cur_pos.c_str();
    char *tmp = strchr ( obj, '.' );
    if ( tmp != NULL ) obj = tmp+1;
}

```

```

if ( strcmp ( obj, "nlMatrixSDTable" ) != 0 )
{
    _msg ( "ERROR", "agent %d, nlMatrixSDTable not supported.\n",
        agent.get_agent_no ( ) );
    return -1;
}

_debug ( "timefilter.dll: nlMatrixSDTable.\n" );

// if first time, then seed the table
if ( !_nlMatrixSDTable_seeded )
{
    // TODO get following working
    // valuespace.set_value ( "nlMatrixSDPkts", "*", "r", "10" );
    // valuespace.set_value ( "nlMatrixSDPkts", "*", "tu", "1" );
    // valuespace.set_value ( "nlMatrixSDOctets", "*", "r", "1000" );
    // valuespace.set_value ( "nlMatrixSDOctets", "*", "tu", "1" );
    // valuespace.set_value ( "nlMatrixSDCreateTime", "*", "v", "100" );

    for ( int i=1; i<=2; i++ ) // ctl_idx
    {
        for ( int j=1; j<=3; j++ ) // prot_idx
        {
            for ( int k=1; k<=2; k++ ) // src_addr
            {
                for ( int l=1; l<=1; l++ ) // dest_addr
                {
                    char instance[256];
                    sprintf ( instance, "%d.0.%d.4.192.9.215.%d.4.10.1.101.%d", i, j, k, l );
                    valuespace.add ( "nlMatrixSDEntry", instance );

                    _debug ( "timefilter.dll:   added %s.\n", instance );
                }
            }
        }
    }

    _nlMatrixSDTable_seeded = 1;
    return 0;
}

std::list<std::string> inst_list;
valuespace.get_instances( "nlMatrixSDPkts", inst_list );

valuespace.remove ( "nlMatrixSDEntry", "*" );

for ( unsigned int i = 0; i < inst_list.size(); i++ )
{
    std::string inst = *(get(inst_list.begin(),i));

    int ctl_idx, dummy;
    char rest[256];

    sscanf ( inst.c_str(), "%d.%d.%s", &ctl_idx, &dummy, rest );
}

```

```

char instance[256];
sprintf ( instance, "%d.%d.%s", ctl_idx, timevalue, rest );
valuespace.add ( "nlMatrixSDEntry", instance );

_debug ( "timefilter.dll:  added %s.\n", instance );
}

return 0;
}

static int
_update_nlMatrixDSTable (
    Mimic::Agent& agent,
    int timevalue
)
{
    Mimic::Valuespace& valuespace = agent.get_valuespace();

    // check if nlMatrixDSTable is supported
    std::string cur_pos = valuespace.set_pos ( "nlMatrixDSTable" );
    char *obj = (char*)cur_pos.c_str();
    char *tmp = strrchr ( obj, '.' );
    if ( tmp != NULL ) obj = tmp+1;
    if ( strcmp ( obj, "nlMatrixDSTable" ) != 0 )
    {
        _msg ( "ERROR", "agent %d, nlMatrixDSTable not supported.\n",
            agent.get_agent_no () );
        return -1;
    }

    _debug ( "timefilter.dll: nlMatrixDSTable.\n" );

    // if first time, then seed the table
    if ( !_nlMatrixDSTable_seeded )
    {
        // TODO get following working
        // valuespace.set_value ( "nlMatrixDSPkts", "*", "r", "10" );
        // valuespace.set_value ( "nlMatrixDSPkts", "*", "tu", "1" );
        // valuespace.set_value ( "nlMatrixDSOctets", "*", "r", "1000" );
        // valuespace.set_value ( "nlMatrixDSOctets", "*", "tu", "1" );
        // valuespace.set_value ( "nlMatrixDSCreateTime", "*", "v", "100" );

        for ( int i=1; i<=2; i++ ) // ctl_idx
        {
            for ( int j=1; j<=3; j++ ) // prot_idx
            {
                for ( int k=1; k<=2; k++ ) // src_addr
                {
                    for ( int l=1; l<=1; l++ ) // dest_addr
                    {
                        char instance[256];
                        sprintf ( instance, "%d.0.%d.4.192.9.215.%d.4.10.1.101.%d", i, j, k, l );
                        valuespace.add ( "nlMatrixDSEntry", instance );
                    }
                }
            }
        }
    }
}

```

```

        _debug ( "timefilter.dll:  added %s.\n", instance );
    }
}

    _nlMatrixDSTable_seeded = 1;
    return 0;
}

std::list<std::string> inst_list;
valuespace.get_instances( "nlMatrixDSPkts", inst_list );

valuespace.remove ( "nlMatrixDSEntry", "*" );

for (unsigned int i = 0; i < inst_list.size(); i++)
{
    std::string inst = *(get(inst_list.begin(),i));

    int ctl_idx, dummy;
    char rest[256];

    sscanf ( inst.c_str(), "%d.%d.%s", &ctl_idx, &dummy, rest );

    char instance[256];
    sprintf ( instance, "%d.%d.%s", ctl_idx, timevalue, rest );
    valuespace.add ( "nlMatrixDSEntry", instance );

    _debug ( "timefilter.dll:  added %s.\n", instance );
}

return 0;
}

static int
_on_action_module_run (
    action_module_run_t *data
)
{
    _debug ( "timefilter.dll: run invoked.\n");

    // parse agent list from passed in arguments
    std::list<int> agents;
    char *cp = data->globals->get("argv");
    while ( 1 )
    {
        int agt;
        if ( sscanf ( cp, "%d", &agt ) <= 0 )
            break;
        agents.push_back(agt);
        cp = strstr ( cp, " " );
        if ( cp == NULL )
            break;
    }
}

```



```

    cp++;
}

// compute new timevalue to use
_time_value += 5;

// for each agent, populate the tables with new
// timefilter based values...
for (unsigned int i = 0; i < agents.size(); i++ )
{
    // get hold of agent
    int nagent = *(get(agents.begin(),i));
    Mimic::Agent& agent = data->session->get_agent(nagent);
    _debug ( "timefilter.dll: agent = %d\n", nagent );

    // check if agent is running
    if ( agent.get_state() != 1 )
    {
        _msg ( "WARN", "agent %d, not running. continuing...\n", nagent );
        continue;
    }

    // call updates for various tables
    _update_nlMatrixSDTable ( agent, _time_value );
    _update_nlMatrixDSTable ( agent, _time_value );
}

_debug ( "timefilter.dll: success.\n");
return ACTION_SUCCESS;
}

```

```

#ifdef __cplusplus
extern "C" {
#endif

```

```

//////////
// DLL entry point
//////////

```

```

#ifdef WIN32
__declspec( dllexport )
#endif
int
process_action_message (
    int msg_id,
    void* msg_data
)
{
    switch( msg_id )
    {
        case ACTION_MODULE_REGISTER:
        {
            return _on_action_module_register (

```

```
        (action_module_register_t*)msg_data
    );
}

case ACTION_MODULE_INIT:
{
    return _on_action_module_init (
        (action_module_init_t*)msg_data
    );
}

case ACTION_MODULE_UNINIT:
{
    return _on_action_module_uninit (
        (action_module_uninit_t*)msg_data
    );
}

case ACTION_MODULE_RUN:
{
    return _on_action_module_run (
        (action_module_run_t*)msg_data
    );
}
}

return ACTION_SUCCESS; // default success
}

#ifdef __cplusplus
}
#endif
```