CA SystemEDGE

User Guide

Release 5.8

CA Technologies Product References

This document references the following CA Technologies products:

- CA eHealth®
- CA Embedded Entitlements Manager (CA EEM)
- CA IT Asset Manager (CA ITAM)
- CA IT Client Manager (CA ITCM)
- CA Network and Systems Management (CA NSM)
- CA Patch Manager
- CA Server Automation
- CA Service Desk Manager (CA SDM)
- CA Spectrum®
- CA SystemEDGE
- CA Systems Performance for Infrastructure Managers
- CA Virtual Assurance for Infrastructure Managers
- CA Software Delivery

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction 17

CA SystemEDGE 17
- Audience 17
- Related Publications 17
- Conventions 18
- Terminology 20

Chapter 2: Features and Integrations Overview 21

Features Overview 21
- Monitoring Features 22
- State Management 25
- Supported MIBs 25
- MIB Extensions 30
- Corrective Actions 30

Plug-ins and Integrations 31
- Application Insight Modules (AIMs) 31
- CA Virtual Assurance 33
- CA Server Automation 34
- CA Spectrum 35
- CA eHealth 36
- CA NSM 37

Chapter 3: Concepts 39

Architecture Overview 39
- SNMP Versions and Access 40
- SNMP Traps 41
- SNMP Tables 42
- Create Monitor by Instance 43
- Configuration Files 43

Monitoring Architecture 45
- SNMPv2 Row Status 46
- How Stateful Monitoring Works 47
- How Stateless Monitoring Works 48
- Self Monitoring 50
- Process and Service Monitoring 55
- Process Group Monitoring 57

Log File Monitoring 58
Windows Event Monitoring 59
History Collection 60
State Management Model 61
Object Status 62
Object Aggregation 63
Super Aggregation 64
How State Management and Aggregation Works 66
State Management Traps 67
State Management Configuration Options 68
Legacy Mode Operation 69
Legacy Monitors 69
Monitor Examples 69
Integration with CA Virtual Assurance 70
Platform Management Modules 70
Remote Configuration and Deployment 71
Event Engine 72
CA Virtual Assurance Security Options 72
AIM Architecture 73
On-Demand Modules 73
AIM Integration with CA Virtual Assurance 74
Guidelines for Using the SystemEDGE Agent 74

Chapter 4: Installation and Deployment 77

Installation Notes 77
Agent Configuration During Installation 78
Installation with CA Virtual Assurance 78
Installation on Windows Systems 78
Install the Agent on Windows 79
Install the Agent on Windows from the Command Line 84
Installation on UNIX and Linux Systems 90
Install the Agent on UNIX and Linux Systems 90
Install the Agent on UNIX from the Command Line 96
Legacy Support of the $CASYSEDGE Variable 102
Configure and Use a Response File 104
Install the Agent in Legacy Mode 104
Agent Deployment 106
Deploy the Agent with CA Virtual Assurance 107
Uninstalling SystemEDGE 108
Uninstall SystemEDGE and the AIMs on Windows 109
Uninstall SystemEDGE and the AIMs on UNIX Systems 111

Reinstallation....112
Upgrade Managed Nodes and AIM Servers....113
Agent and AIM Upgrades....113
Import SystemEDGE Monitors into a Policy....116

Chapter 5: Agent Configuration 119

Configuration Basics....119
Interactions Between sysedge.cf and sysedge.mon....121
Configuration Using CA Virtual Assurance....121
File-Based Configuration....121
Configure System Information....122
Configure SNMP Information....123
Configure Authentication Failure Traps....130
Configure SNMP Set Notifications....131
Configure SNMP Set Restrictions....131
Configure GetBulk Response Message Size....132
Configure Federal Information Processing Standard Mode....133
Configure Diagnostic Logging Mechanism....134
Configure Maintenance Mode....136
Configure Support for Actions....136
Configure User and Group Permissions for Subprograms (UNIX Only)....137
Configure MIB Table Restrictions....137
Configure Device Status Checking Restrictions....138
Configure Warm Start Discovery....140
Configure Linux Free Memory Calculation....141
Configure Monitor Aggregation in a Multi-tier Hierarchy....141
Monitoring Configuration....144
Load SystemEDGE AIMs....145
Recommendations for Configuring Security....147
Regular Expression Examples for Monitors and Autowatchers....147
Perl Compatible Regular Expression (PCRE) Support....150
Multiline Regular Expressions....151
Configure Text Pattern Exclusion....152
Regular Expression Examples....152
SNMPv3 Configuration....155
Configure the SNMPv3 Engine ID....156
Configure SNMPv3 User Information....156
Configure SNMPv2c and SNMPv3 Trap Destinations....161
Encrypt the SNMPv3 Configuration File....163
Disable SNMPv1 and SNMPv2c....164
SystemEDGE Control Panel for Windows....164

Chapter 6: Starting the Agent 165

Start or Stop the Agent on Windows Systems 165
Start or Stop the Agent on UNIX Systems 166
Service Startup Script for UNIX Systems 166
Command Line Startup Options 167
Automatic Agent Startup at System Boot 168
Start the Agent Automatically on Windows Systems 168
Start the Agent Automatically on UNIX Systems 168
Agent Warm Start 169
Warm Start Exceptions 169
Maintenance Mode 169

Chapter 7: Systems Management Empire MIB 171

Systems Management Empire MIB Overview 172
System Group 173
Device Table 173
Kernel Configuration Group 174
Boot Configuration Group 174
Streams Group 175
User Table 175
Group Table 175
Process Table 175
Change the nice Value of a Process (UNIX only) 176
Send a Signal to a Process 176
Who Table 176
Remote Shell Group 177
Run a Remote Command 177
Performance Group 178
Interprocess Communication Group 178
Delete IPC Mechanisms 179
Buffers Group 179
Message Buffers Group 179
The Streams Buffers Group 179
I/O Buffer Cache Group 180
Directory Name Lookup Cache Group 180
AIX Logical Partition Group 180
Table of Work Load Manager (WLM) Classes 181
Trap Community Table 181
NT System Group 181
NT Thread Table 182
NT Registry Group 182

NT Services Table 182
NT Performance Groups 183
NT Event Monitor Group 184
NT Registry and Performance Extension Group 184
Self Monitor Table 185
Distributed Systems Group 185
RPC Statistics Group 186
NFS Statistics Group 186
History Table 186
Log Monitor Table 187
Disk Statistics Table 188
Configure Disk Performance Statistics Collection for AIX Systems 188
CPU Statistics Table 189
Extension Group 190
Process Monitor Table 190
Process Group Monitor Table 190
Aggregate State Table 191
LPAR Group 191
Mirror Tables 191
Autowatcher Tables 191
Unsupported Systems Management Empire MIB Objects 192

Chapter 8: Self Monitoring 193

Self Monitoring Overview 193
State Management of Self Monitors 194
Managed Object Creation 194
Aggregate State Table 196
Self Monitor Table 196
Self Monitor Table Columns 197
Sample Self Monitor Table Entry 203
Row Creation 204
Self Monitor Table Flags 205
Self Monitor Table Action Parameters 208
View the Self Monitor Table 210
Self Monitoring Configuration 211
monitor Directive--Add Entries to the Self Monitor Table 212
Reverse OID Lookup 215
Self Monitoring Examples 216
edgemon Commands for Self Monitoring 230
edgemon Examples 233
Remove Self Monitoring Entries 235

Chapter 9: Process and Service Monitoring 237

Process and Service Monitoring Overview 237
Windows Service Monitoring 238
State Management of Process Monitors 239
Managed Object Creation 240
Aggregate State Table 241
Process Monitor Table 241
Process Monitor Table Columns 242
Sample Process Monitor Table Entry 248
Process Attributes 249
Row Creation Objects 251
Process Monitor Table Flags 253
Process Monitor Table Action Parameters 259
View the Process Monitor Table 261
Process Monitoring Configuration 261
watch process Directive--Add Entries to the Process Monitor Table 263
watch ntservice Directive--Add Service Monitoring Entries to the Process Monitor Table 266
Process Monitoring Examples 268
edgewatch Commands for Process Monitoring 275
edgewatch Examples 278
Remove Process Monitoring Entries 281
Solaris Zone Process Monitoring 282
Recommendations for Process and Service Monitoring 283

Chapter 10: Autowatchers 285

How Autowatchers Work 285
Generic Autowatchers 287
Process and Service Autowatchers 287
edgewatch Utility--Using Autowatchers 288
edgewatch Commands for Autowatchers 292
edgewatch Autowatcher Examples 296
Specify an Autowatcher in sysedge.cf 297
Generic Autowatcher Examples 300
Service Autowatcher Example 301
Process Autowatcher Examples 301

Chapter 11: Process Group Monitoring 303

Process Group Monitoring Overview 303
Process Group Monitor Table 304
Process Group Monitor Table Columns 304

Row Creation Objects....309
Process Group Monitor Table Flags....310
Process Group Monitor Table Action Parameters....312
View the Process Group Monitor Table....313
Process Group Monitoring Configuration....313
watch procgroup Directive--Add Entries to the Process Group Monitor Table....314
watch procgroupex Directive--Add Entries to the Process Group Monitor Table (UNIX only)....316
Process Group Monitoring Examples....318
edgewatch Utility--Monitor Process Groups....319
edgewatch Commands for Process Group Monitoring....323
edgewatch Examples....324
Remove Process Group Monitoring Entries....324
Solaris Zone Process Group Monitoring....325

Chapter 12: Log File Monitoring 327

Log File Monitoring Overview....327
Directory Monitoring....328
Log Monitor Table....328
Log Monitor Table Columns....328
Row Creation Objects....332
Log Monitor Table Flags....333
Log Monitor Table Action Parameters....336
View the Log Monitor Table....337
Log Monitoring Configuration....337
watch logfile Directive--Add Entries to the Log Monitor Table....338
Log File Monitoring Examples....340
edgewatch Utility--Monitor Log Files....342
edgewatch Commands for Log File Monitoring....345
edgewatch Examples....347
Remove Log Monitoring Entries....348
Recommendations for Log File Monitoring....349
Rotating Log Files....351

Chapter 13: Windows Event Monitoring 353

Windows Event Monitoring Overview....353
Windows Event Search Criteria....354
NT Event Monitor Table....355
NT Event Monitor Table Columns....355
Row Creation Objects....358
NT Event Monitor Table Flags....360
NT Event Monitor Table Action Parameters....362

View the NT Event Monitor Table 363
Windows Event Monitoring Configuration 364
watch ntevent Directive--Add Entries to NT Event Monitor Table 365
Windows Event Monitoring Examples 367
edgewatch Utility--Monitor Windows Events 368
edgewatch Commands for Windows Event Monitoring 372
edgewatch Examples 373
Remove Windows Event Monitoring Entries 375

Chapter 14: History Collection 377

History Collection Overview 377
History Sampling 377
History Control Table and History Data Table 378
History Control Table Columns 378
History Table Columns 380
Row Creation Objects 381
History Sampling Examples 381
View the History Control Table 383
History Control Table Configuration 383
Initial Configuration During Startup 383
Dynamic Configuration During Operation 383
emphistory Directive--Add Entries to History Control Table 384
History Collection Example 385
emphistory Commands for Managing Entries in the History Control Table 385

Chapter 15: Custom MIB Objects 389

Systems Management Empire MIB Extension Group 389
Extension Group Features 390
Extension Variable Configuration 390
extension Keyword--Add Entries to the Extension Group 391
Additional Parameters 392
Extension Examples 392
Extension Scripts 394
Example Script Test 395
Using Extension Variables with Your Management Software 396
How to Edit empire.asn1 for Extension Variables 397
How to Edit a Separate MIB Specification for Extension Variables 397
Recommendations for Using Extensions 398

Chapter 16: Windows Registry and Performance MIB Objects 399

Windows Registry and Performance Functionality 399
Systems Management Empire MIB ntRegPerf Group 400
Registry Data 400
Performance Data 401
Windows Registry and Performance Variable Configuration 403
ntregperf Keyword—Add Windows Registry and Performance MIB Objects 404
Windows Registry and Performance Examples 405
Using Windows Registry and Performance Variables with Your Management Software 406
How to Edit empire.asn1 for ntRegPerf Variables 406
How to Add a Separate MIB Specification for ntRegPerf Variables 407

Chapter 17: SystemEDGE AIMs 409

Core AIMs 409
Top Processes AIM 409
Monitor Maintenance Windows AIM 410
Enable the Monitor Maintenance Windows AIM 410
Limitations of the Monitor Maintenance Windows AIM 411
Performance Cube AIM 411
Enable the Performance Cube AIM 411

Chapter 18: Command Line Utilities 413

SystemEDGE Command Line Utilities 413
SystemEDGE Additional Command Line Utilities 414
Configuring SystemEDGE Command Line Utilities 414

Chapter 19: Examples Using State Management 415

Monitor the Windows Task Scheduler 415
Monitor the Existence of Quarantined Files 417
Monitor File Systems 417
Reconfigure Extension Scripts Through SNMP 419
Monitor Log Files and Send Notification Through Email 423
Monitor Processes 424
Create Monitored Objects 425

Chapter 20: Troubleshooting and Usage Suggestions 427

Using diagsysedge.exe 427
Determine Whether the Agent Is Running with diagsysedge 427
Obtain a Report for Troubleshooting 428

Validate AIM Plugins 429
Common Problems and Questions 429
Diagnostic Logging Information for Troubleshooting 430
Agent Not Responding to SNMP Requests 430
Management System Not Receiving SNMP Trap Messages 433
Bind Failed: Address Already In Use 434
CA SystemEDGE: xtrapmon.exe Utility Replaced by edgetrapmon.exe 435

Appendix A: FIPS 140-2 Encryption 437

FIPS 140-2 Mode 437
FIPS Library Installation 437
Platform Support 438
Supported Encryption Protocols 438
Supported Authentication Protocols 439
Configure FIPS 140-2 Mode 439
FIPS Mode Considerations 440
Key Protection 440

Appendix B: SystemEDGE Advanced Encryption 441

SystemEDGE Advanced Encryption 441
Supported Platforms 441
Supported Encryption Protocols 441
Supported Authentication Protocols 442
FIPS Compatibility 442
Installation Prerequisites 442
Install SystemEDGE Advanced Encryption 442
Install SystemEDGE Advanced Encryption on Windows 443
Uninstall SystemEDGE Advanced Encryption on Windows 443
Install SystemEDGE Advanced Encryption on UNIX 444
Uninstall SystemEDGE Advanced Encryption on UNIX 444
Installed Files 445

Appendix C: Co-existence with Other SNMP Agents 447

Multiple SNMP Agents Support 447
Multiple SystemEDGE Instance Support 448
Coexistence with the Microsoft Windows SNMP Agent 448
Coexistence with the HP SNMP Agent 449
Coexistence with the AIX SNMP Agent 449

Appendix D: Host Resources MIB 451

Host Resources MIB Overview 451
Host Resources System Group 452
Host Resources Storage Group 452
Host Resources Device Group 453
- Device Table 453
- Processor Table 454
- Disk Storage Table 454
- Partition Table 455
- File System Table 455

Host Resources Running Software Group 456
Host Resources Installed Software Group 456
Unsupported Host Resources MIB Objects 457

Appendix E: Private Enterprise Traps 459

Format of Trap PDUs 459
- monitor Trap 460
- monitorEntryNotReady Trap 460
- logMonMatch Trap 461
- logMonNotReady Trap 461
- ntEventMonMatch Trap 462
- ntEventMonNotReady Trap 462
- monitorClear Trap 463
- processStop Trap 463
- processStart Trap 464
- processThreshold Trap 464
- processClear Trap 465
- processNotReady Trap 465
- addrChange Trap 466
- procGroupChange Trap 466
- aggregateState Trap 467
- aggregateAdd Trap 467
- aggregateDelete Trap 468

SNMPv1 Trap Format 468

Index 471

Chapter 1: Introduction

This section contains the following topics:

CA SystemEDGE (see page 17)

CA SystemEDGE

SystemEDGE increases the productivity of system administration staff by enabling them to control all servers on their networks from a single, central location and automates systems management tasks and inventory tracking, increasing productivity and system stability while helping to reduce rising system support costs.

The SystemEDGE agent provides powerful systems management through the industry-standard SNMP. It enables remote management systems to access important information about system configuration, status, performance, users, processes, file systems and much more. The agent also includes intelligent monitoring and state management capabilities that enable reporting and managing of exceptions and calculation of object severity and status information.

Audience

This guide is intended for an administrator who installs, configures, and uses the SystemEDGE agent to manage UNIX, Linux, and Windows servers. It assumes that you have a basic familiarity with operating system environment your system and with Simple Network Management Protocol (SNMP).

Related Publications

The CA Virtual Assurance documentation consists of the following deliverables:

Administration Guide

Explores how to administer and use CA Virtual Assurance to manage virtual resources in your environment.

Installation Guide

Contains brief architecture information, various installation methods, post-installation configuration information, and Getting Started instructions.

Online Help

Provides window details and procedural descriptions for using the CA Virtual Assurance user interface.

Reference Guide

Provides detailed information about AutoShell, CLI commands, and MIB attributes.

Performance Metrics Reference

Describes the performance metrics that are available for monitoring the systems performance of the supported platforms.

Release Notes

Provides information about operating system support, system requirements, published fixes, international support, known issues, and the documentation roadmap.

Service Response Monitoring User Guide

Provides installation and configuration details of SRM.

SystemEDGE User Guide

Provides installation and configuration details of SystemEDGE.

SystemEDGE Release Notes

Provides information about operating system support, system requirements, and features.

Conventions

This guide uses the following conventions:

Case-Sensitivity

All names of classes, commands, directives, environment parameters, functions, and properties mentioned in this guide are case-sensitive and you must spell them exactly as shown. System command and environment variable names *may* be case-sensitive, depending on your operating system's requirements.

Cross-References

References to information in other guides or in other sections in this guide appear in the following format:

Guide Name

Indicates the name of another guide.

"Chapter Name"

Indicates the name of a chapter in this or another guide.

Synonyms

Terms such as attribute, object, object identifier (OID) are synonymous to the term 'variable' in this document.

Terms such as SystemEDGE Agent, CA SystemEDGE are synonymous to SystemEDGE in this document.

Syntax

Syntax and user input use the following form:

Italic

Indicates a variable name or placeholder for which you must supply an actual value.

{a|b}

Indicates a choice of mandatory operands, a or b.

[ ] or [[ ]]

Indicates optional operands.

Syntax Example

The following example uses these conventions:

`modify -t ZONE [-m zoneserver] -p psetname {-min mincpu|-max maxcpu} pset -session ssh`

The operands -min and -max are mandatory, but you can only use one of them depending on what you want to define, the minimum number of CPUs in the processor set or the maximum number. The operand -m is not required for this command to function. All other parts of the command must be entered as shown.

Default Directory

CASYSEDGE used in path statements indicates the directory in which SystemEDGE is installed. **Default:** C:\Program Files\CA\SystemEDGE.

Installation Path

Install_Path used in path statements indicates the directory in which CA Virtual Assurance or components of CA Virtual Assurance are installed.

Defaults:

- Windows x86: C:\Program Files\CA
- Windows x64: C:\CA, C:\Program Files (x86)\CA, or C:\Program Files\CA
- UNIX, Linux: /opt/CA

Terminology

MIB Object, MIB Attribute

These terms are used synonymously.

Chapter 2: Features and Integrations Overview

This section contains the following topics:

Features Overview (see page 21)
Plug-ins and Integrations (see page 31)

Features Overview

SystemEDGE provides the following systems management features in a lightweight, easy-to-deploy package:

- Collects SNMP metrics to be used by any SNMP manager.
- Monitors objects such as metric values, processes and services, process groups, log files, directories, and Windows events. These objects are monitored using configurable MIB tables that trigger notifications through SNMP traps.
- Manages object models state using Self Monitor and Process Monitor tables. This facilitates to aggregate monitored object severity, update object status, and send state change traps.
- Collection of metrics in a history table and in cube files for baselining and trend analysis.
- File-based remote agent configuration and remote deployment (when integrated with CA Virtual Assurance or CA Server Automation).
- An extensible MIB that lets you add custom MIB objects, such as registry entries, for monitoring.
- Ability to trigger actions when monitor entries exceed the specified threshold.
- Flexible architecture that permits Application Insight Modules (AIMs), which are plug-in modules for additional monitoring capabilities.

More Information

Concepts (see page 39)

Monitoring Features

You can configure the SystemEDGE agent to monitor specific Management Information Base (MIB) objects for thresholds or other exceptions. When you manage a large enterprise with hundreds of systems, you can configure the information that the agent monitors. The ability to configure the agent monitored information provides the management by exception that is necessary to maintain large-scale implementations without impacting management overhead.

Note: For information about collecting and monitoring capabilities of each agent, see *Concepts* chapter.

The maximum resolution of the monitor supported is 64 bit. The SystemEDGE agent provides the following types of monitoring:

Self monitoring

Provides monitoring of any integer-based MIB object that the agent supports. You can create entries in the Self Monitor table to specify objects to monitor, comparison operators, threshold values, and severities, and the agent automatically monitors the objects according to the created entries. The agent monitors the objects on a polling interval, maintains a current state according to specified threshold and severity values. If the state change threshold exceeds, the agent sends a state change trap.

Note: For more information about self monitoring, see *Self Monitoring* chapter.

Process and service monitoring

Provides monitoring of any single process instance or Windows service. You can create entries in the Process Monitor table to monitor whether a process or service is running or to monitor process table MIB objects against specified thresholds. The agent monitors the processes, maintains a current state according to specified threshold and severity values. If the state change threshold exceeds or the state of a process (running or stopped) changes, the agent sends a state change trap.

Note: For more information about process and service monitoring, see *Process and Service Monitoring* chapter. To monitor more than a single process instance, use a process group monitor because a process monitor can only monitor a single process instance.

Process group monitoring

Provides the ability to define a group of processes and monitor that group. You can create entries in the Process Group Monitor table defining process groups. The agent monitors the groups and sends a trap when the process group change. The agent monitors also provide process group metrics such as memory or cpu time consumed by all processes in a group. If necessary, a self monitor can monitor the process group metrics.

Note: For more information about process group monitoring, see *Process Group Monitoring* chapter.

Log file and directory monitoring

Provides monitoring of any UTF-8 encoded system or application log file by searching for strings specified as regular expressions. You can create entries in the Log Monitor table, and the agent monitors the specified log file for lines matching user-defined regular expressions and sends a trap when a match occurs. You can associate a severity with the monitor, which is included with the sent trap. If you configure entries in the Log Monitor table accordingly, you can also monitor the size and number of files in a directory. You can monitor log file of size up to 2 GB.

Note: For more information about log file and directory monitoring, see *Log File Monitoring* chapter.

Windows event monitoring

Provides monitoring of Windows event log entries using different filters, such as event source. You can create entries in the NT Event Monitor table, and the agent monitors the event log for events matching user-defined regular expressions and sends a trap when a match occurs.

Note: For more information about Windows event monitoring, see *Windows Event Monitoring* chapter.

History collection

Provides historical data collection for manager-side baselining and trend analysis. You can create entries in the History Control table, and the agent collects metrics over a time interval. The metrics collected is used to provide a picture of average system performance during a specific time interval.

Note: For more information about history collection, see *History Collection* chapter.

Monitor Scale for 64-bit metrics

You can specify a threshold for the monitors to monitor 64-bit metrics. For example, if you specify a 64-bit threshold, such as 9,000,000,000,000,000,001 Bytes (9 * 10^18 plus 1) is a valid 64-bit value. However, you can define a threshold of 9 EB (Exabyte) by defining a scaling factor of 10^18 and a threshold of 9. The scaling factor defines the resolution of the monitor. You can define a scaling factor of 10^16, for a resolution of one percent, the corresponding threshold is 900.

If the scaling factor of a monitor is 2^32, for a 64-bit (2^64) metric, the maximum monitor value is 2^32.

Process Monitoring Enhancement

SystemEDGE provides the process CPU utilization as a percentage of new variable, processTimePermil in the processTable. The Process Monitoring and Process Group Monitoring capabilities are provided using processTimePermil.

processTimePermil is the CPU permil utilization of a process over the last sample interval. The variable value is system-dependent. The value is the difference in the number of CPU ticks divided by the total CPU ticks. As SNMP does not support fractions, the value reported is multiplied by 1000. For example, a CPU permil of 505.833 is returned as 505833. To convert to percent, divide the returned value by 10000.

Object Aggregation

A new Aggregate table in the Systems Management MIB allows for aggregation of objects that have multiple monitors defined with different severities. The Aggregate table uses the new class, instance, and attribute properties in the Self Monitor and Process Monitor tables to determine whether monitors refer to the same object. The overall object state is determined by sending a state change trap for the breached monitor with the highest severity.

Multi-tier Object Instance Hierarchies

To relate an object instance to its hosted hierarchy, // and / are used as delimiters such as *//hierarchy/instance.* The hierarchy can be multi-tier such that object instances are supported for more than two tiers.

The hierarchy can be a host name //host/instance, it can be also of the form *//<any1>/<any2>/.../<anyN>/<instance>*. For object instances on local system, only instance can be specified and *//./* is prepended automatically. You can prepend a namespace to identify virtual object instances such as 'lpar:'. For example, the object instance on an LPAR is represented as *lpar://hmc1/hyper1/lpar1/disk1*.

The super aggregation of object instances is provided only for the lowest two tiers and not all tiers. For example, *//host/** aggregates all instances on a host. The special super aggregate *//** is created (which aggregates all object instances) is dependent on the selected level of super aggregation.

State Management

The SystemEDGE agent computes status information from created self monitors and process monitors. Monitors have a threshold condition and a severity (monitor1: “greater 70% is Warning” monitor2: “greater 90% is Critical”). Several Monitors are combined into a single MonitorObject having the worst state of all associated monitors. The agent sends status information through state change traps, and stateful managers such as CA Virtual Assurance access and interpret this status information to give you a state-centric view of the systems, processes, services, and applications in your enterprise.

Note: For more information about the SystemEDGE state management model, see *Concepts* chapter.

Supported MIBs

A MIB acts as an information base used to store the information of the agent-controlled elements. The MIB is a structured collection of objects, each of which represents an item of management information.

A management system monitors a managed resource by reading the values of its MIB objects. It can also control the resource by modifying (setting) the values of objects in the resource MIB through SNMP SET requests.

MIBs are defined in a MIB specification that describes the management objects relating to a particular resource. The MIB specification also defines how the collection of objects is structured. The MIB module resembles a data-definition document used by both the management system and the agent.

The SystemEDGE agent supports the collection of metrics from the following MIBs:

- CA Systems Management Empire MIB
- MIB-II (RFC 1213)
- Host Resources MIB (RFC 1514)
- IP-MIB (partial) (RFC 4293)
- IF-MIB (partial) (RFC 2233)
- TCP-MIB (partial) (RFC 4022)
- UDP-MIB (partial) (RFC 4113)

The agent can use its flexible monitor concept through SNMP traps to send information collected from these MIBs to any management system that accepts SNMP data.

CA Systems Management Empire MIB

The Systems Management Empire MIB is a private enterprise MIB that includes objects for monitoring the health and performance of the underlying system and its applications. This MIB defines management information for the following objects:

Systems Management Empire MIB

- System details
- Mounted devices
- Kernel and system parameters
- Boot configuration
- Network, streams, and I/O buffer statistics
- Network file system (NFS) and Remote Procedure Call (RPC) statistics
- Kernel performance statistics, such as the number of context switches and page faults
- File systems
- Users and groups
- Processes
- Interprocess communications
- System resources
- Logical partitions
- Disk and CPU statistics
- Windows-specific statistics such as the registry and Windows event logs

The Systems Management Empire MIB contains the following tables to configure autonomous monitoring and data storage capabilities of agent:

- Self Monitor table
- Process Monitor table
- Process Group Monitor table
- Log Monitor table
- NT Event Monitor table
- History Control table

The Systems Management Empire MIB also contains tables that provide information derived from the defined monitors and process monitors about the state of monitored objects.

For more information about how the SystemEDGE agent uses the Systems Management Empire MIB, see the chapters "Concepts" and “Systems Management Empire MIB."

Note: Systems Management Empire MIB object support varies by platform. For a list of unsupported Systems Management Empire MIB objects on each platform, see the *SystemEDGE Release Notes*.

MIB-II

MIB-II is the standard (RFC 1213) that provides information about network interfaces and protocol statistics. This MIB includes information about the following groups:

- System
- Interfaces
- Address Translation
- Internet Protocol (IP)

  Note: SystemEDGE supports only the ifXTable for IF-MIB file.
- Internet Control Message Protocol (ICMP)
- Transfer Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Simple Network Management Protocol (SNMP)

Note: On some Windows releases, SystemEDGE might not be able to retrieve ifXTable data from the system if the Wireless Zero Configuration service is active. This service uses NDISUIO driver in an exclusive mode and SystemEDGE might not be able to access the NDISUIO driver. This condition does not occur on Window 2008, Windows Vista, or Windows 7.

If you require ifXTable data on earlier Windows releases, you must disable the Wireless Zero Configuration service or any services that use ndisuio.sys.

MIB-II object support varies by platform. For a list of unsupported MIB-II objects on each platform, see the *SystemEDGE Release Notes*.

Host Resources MIB

The Host Resources MIB is defined by the Internet Engineering Task Force (IETF) to provide management information for generic host systems. The MIB includes information especially useful for asset management, such as the following:

- Storage areas, such as file systems and disk partitions
- Running and installed software
- System devices, such as keyboards, disks, and network cards

You can use SystemEDGE with the Host Resources MIB to automate asset tracking and provide a current picture of your installed hardware and software. SystemEDGE can determine whether your systems are properly configured and whether they include the current patches and service packs. This information can help simplify systems management, improve performance, and reduce security risks.

For more information about how the SystemEDGE agent uses the Host Resources MIB, see *Host Resources MIB* appendix.

Note: Host Resources MIB object support varies by platform. For a list of unsupported Host Resources MIB objects on each platform, see the *SystemEDGE Release Notes*.

IP-MIB Tables

SystemEDGE supports the following IP MIB tables:

MIB Table	Windows	Solaris	HP-UX	AIX	Linux
ipSystemStatsTable (RFC 4293)	No	Yes	No	Yes	Yes
ipIfStatsTable (RFC 4293)	No	No	Yes	No	No
ipAddressPrefixTable (RFC 4293)	No	No	Yes	No	Yes
ipNetToPhysicalTable (RFC 4293)	No	Yes	Yes	No	No
ipAddressTable (RFC 4293)	No	Yes	Yes	Yes	Yes
ipDefaultRouterTable (RFC 4293)	No	Yes	Yes	Yes	Yes
icmpStatsTable (RFC 4293)	Yes	Yes	No	Yes	Yes
tcpConnectionTable (RFC 4022)	Yes	Yes	Yes	Yes	Yes
udpEndpointTable (RFC 4113)	Yes	Yes	Yes	Yes	Yes

Note: The supported IP MIB objects varies by platform. For a list of unsupported IP MIB objects on each platform, see the *SystemEDGE Release Notes*.

MIB Extensions

The Systems Management Empire MIB is user-extensible. You can add extensions into the MIB to manage objects important to your enterprise that are not covered in the base MIB. You can customize the Systems Management Empire MIB in the following ways:

- Create your own scalar MIB objects for customized management. You can configure SystemEDGE with each customized MIB object number and MIB object type and the name of a script or program to run when the new MIB variable is queried or set.
- Create custom MIB objects to manage Windows registry and performance data. You can configure SystemEDGE to report additional registry parameters and performance data without using external programs or scripts. This feature lets you monitor the health of applications that make performance data available through the performance registry.

Note: For more information about creating custom MIB objects, see *Custom MIB Objects* chapter. For more information about creating objects to manage Windows registry and performance data, see *Windows Registry and Performance MIB Objects* chapter.

Corrective Actions

The SystemEDGE agent can automatically respond to critical situations on a system by invoking actions, which are specific commands or scripts to run when certain criteria are met. For example, you can specify actions that enable the agent to restart a failed process, send email to an administrator in the event of an unauthorized access to the system, and so on.

When using actions, you must specify the full path of the command (with any parameters). Usage of environment variables is supported. The agent runs this command each time the conditions are met for the monitoring table entry.

For more information about specifying actions and detailed examples, see the following chapters:

- "Self Monitoring”
- "Process and Service Monitoring”
- "Process Group Monitoring”
- "Log File Monitoring”
- "Windows Event Monitoring”

Plug-ins and Integrations

The SystemEDGE agent's SNMP architecture enables integration with any SNMP-based manager. The agent is configurable without any specific management product and can send traps to any application that collects SNMP information. Many CA products feature enhanced integrations with the agent to take advantage of its full range of functionality and provide interfaces for interpreting its data and configuring and deploying the agent remotely.

For specific management disciplines, CA provides SystemEDGE plug-in modules called Application Insight Modules (AIMs). AIMs integrate into the agent's existing architecture and extend its functionality by providing the tools to monitor specific applications or services.

This section discusses the CA products that feature integrations with SystemEDGE and the AIMs available for use as agent plug-ins.

Application Insight Modules (AIMs)

The SystemEDGE agent provides a plug-in architecture through which it can load optional Application Insight Modules (AIMs) when it initializes. These AIMs provide an extensible and flexible approach to support application-specific semantic knowledge.

The following AIMs adhere to the SystemEDGE state management model and support deployment and configuration through manager user interfaces:

VMware vCenter Server AIM

Provides management capabilities of systems that are under VMware vCenter Server control.

Remote Monitoring AIM

Provides the management and monitoring capabilities of remote (agent-less) systems.

Service Response Monitor (SRM)

Monitors the service response on managed nodes. The Service Response Monitor (SRM) AIM creates monitors utilizing the SystemEDGE state management model and supports deployment and configuration through the CA Virtual Assurance user interface.

Cisco Unified Computing System (UCS)

Collects device and statistical data from Cisco UCS Manager and provides SNMP support for external applications.

LPAR AIM (AIX PowerVM)

Provides capabilities to manage and monitor the entire system including LPARs.

Microsoft Cluster Server AIM

Monitors Microsoft clusters from a remote system. MSCS monitoring requires remote access to clusters and individual cluster nodes for metric collection such as CPU and memory utilization.

Microsoft Hyper-V AIM

Provides capabilities to manage and monitor Hyper-V Servers.

Exchange Server and Active Directory AIM

Provides capabilities to monitor the Exchange Server and Active Directory on both off-premise and on-premise infrastructure.

Solaris Zones AIM

Provides capabilities to manage and monitor Solaris 10 systems configured to run containers and zones.

KVM AIM

Monitors *kernel-based virtual machine* environments from a remote system. The KVM AIM communicates with one or more RHEV managers.

Citrix XenServer AIM

Monitors Citrix XenServer environments. The AIM communicates with Pool Master only.

CA Virtual Assurance

CA Virtual Assurance is a policy-based product that automatically monitors, reconfigures and provisions virtual resources to dynamically meet the load demands of complex service-oriented data centers. CA Virtual Assurance is built on a Service Oriented Architecture (SOA) and continuously analyzes your data center to help ensure that your servers are optimally provisioned to perform required tasks.

CA Virtual Assurance uses SystemEDGE as the basis for monitoring and managing a broad range of systems.

Specific features are only available when you use the agent with the CA Virtual Assurance user interface. Using SystemEDGE with these products provides the following unique features:

- Visualization of the agent's state management data through Platform Management Modules. You can view the state of all monitors and find and diagnose critical monitors that are causing state propagation.
- Agent deployment to remote systems. This feature includes the ability to configure settings such as SNMP communities and any other installation settings and deploy the agent with those settings to any managed system.
- Agent configuration for remote systems. CA Virtual Assurance features a file-based configuration solution that lets you create monitor entries, configure any aspect of the SystemEDGE agent, and deploy that configuration to any number of managed systems.
- Agent warm start. When you deploy an agent to a remote system, a full agent restart is not required in most cases. The agent performs a warm start and implements configuration changes during runtime.
- Configuration lock down options such as the ability to be notified of SNMP-based changes to any managed agent and the option to overwrite any changes not initiated through CA Virtual Assurance.
- Deployment and configuration of the Service Response Monitor AIM.
- Virtual platform management through a wide selection of virtual environment monitoring AIM plug-ins.

CA Virtual Assurance enhances the functionality of the SystemEDGE agent, and are the recommended products for using SystemEDGE to manage all physical and virtual systems in your environment.

For more information about CA Virtual Assurance, see the *CA Virtual Assurance* documentation.

CA Server Automation

CA Server Automation is a policy-based product that automatically monitors, reconfigures and provisions physical and virtual resources to dynamically meet the load demands of complex service-oriented data centers. CA Server Automation is built on a Service Oriented Architecture (SOA) and continuously analyzes your data center to help ensure that your servers are optimally provisioned to perform required tasks.

CA Server Automation contains embedded CA Virtual Assurance functionality, including the SystemEDGE agent. Using SystemEDGE within CA Server Automation provides a system-centric management view of your data center. Information provided by SystemEDGE gives you the ability to make more informed decisions about automating the physical and virtual systems in your data center. Combining the agent with CA Virtual Assurance functionality provides management for virtual and physical systems through the CA Virtual Assurance AIMs.

Platform Management Modules collect information from the SystemEDGE agent for display on the CA Server Automation user interface. The CA Server Automation user interface provides a topology view of all physical and virtual systems in several different contexts. Selecting a system managed by SystemEDGE in this view displays a monitored object state summary, defined monitors, and various system metrics. The user interface also provides options for performing system actions enabled by the virtual monitoring AIMs. You have access to the full suite of CA Virtual Assurance functionality within CA Server Automation, and you can view SystemEDGE traps in the CA Server Automation event view.

CA Spectrum

CA Spectrum supports managing discovered systems with the SystemEDGE agent. Managing SystemEDGE with CA Spectrum can provide systems management within the context of your distributed network. The agent can alert CA Spectrum to under-performing system metrics and stopped processes or applications that may affect the operation of your network.

You can view SNMP metrics collected by the SystemEDGE agent and the entries in the self-monitoring tables from the Spectrum OneClick interface. CA Spectrum can collect traps sent by the agent and display them as events, so that you can manage exceptions from OneClick. You can configure the CA Spectrum alarm view with rules that generate alarms when specific SystemEDGE traps are collected, so that you can discover and resolve specific system problems in the same way that you do network faults.

CA Spectrum 9.2 supports the SystemEDGE r5.0 Systems Management Empire MIB, which enables the state management functionality. You can add, edit, and delete monitors using the embedded OneClick tables provided by SystemEDGE 5.0. CA Spectrum r9.2 dynamically checks for the release (latest or previous) of SystemEDGE agent and displays release-specific information.

CA Spectrum also supports SystemEDGE AIMs. When receiving AIM traps in CA Spectrum 9.2, the traps include the agent type, agent name, and alarm type.

Note: Several features specific to CA Virtual Assurance are not available when you use the SystemEDGE agent with OneClick. For more information about these features, see *Concepts* chapter.

To enable the full functionality of the agent, you must manage the agent with CA Virtual Assurance. For more information, see the *CA Virtual Assurance* documentation.

CA eHealth

You can run the nhAddSysEdgeMonEntries command to configure the SystemEDGE agent to monitor critical CA eHealth processes and system logs. The command adds entries to the sysedge.cf file, and it stops and restarts the SystemEDGE agent to implement the changes.

Note: For more information about the nhAddSysEdgeMonEntries command, see the *CA eHealth Administration Reference*.

When you use SystemEDGE with CA eHealth, the CA eHealth poller can collect data from SystemEDGE agents and store it in the CA eHealth database so it is available to the CA eHealth reporting and real-time monitoring tools. You can run At-a-Glance (AAG), Trend, Top N, What-If Capacity Trend, System Health, and MyHealth reports for systems to perform capacity planning, accurately document service problems, and troubleshoot potential problems.

Note: For more information about CA eHealth application, see the *CA eHealth System and Application Administration Guide*.

CA eHealth supports running the SystemEDGE agent with legacy monitors (that is, monitors that do not use SystemEDGE r5.0 functionality). Several CA Virtual Assurance-specific features are not available when using the agent with CA eHealth and eHealth AdvantEDGE View. For more information about legacy monitors and legacy mode (running the agent without CA Virtual Assurance), see the chapter "Concepts." To enable the full functionality of the agent, you must manage the agent with CA Virtual Assurance. For more information, see the *CA Virtual Assurance* documentation.

CA NSM

CA NSM can integrate with the SystemEDGE agent to use agent data to manage discovered systems. Combining SystemEDGE agent management capabilities with CA NSM DSM functionality can provide a layered systems monitoring architecture with unique managed objects monitored by each agent.

CA NSM contains DSM classes and policy for the SystemEDGE agent and can discover and poll the agent for system information. A DSM-monitored object is created each time an object table entry appears, and CA NSM can handle traps sent from the object tables. The SystemEDGE Agent View lets you view a summary of the agent and create entries in the Self Monitor and Process Monitor tables within CA NSM.

Starting with CA NSM r11.2 SP1, CA NSM can interpret states created and maintained by the SystemEDGE state management model for the SystemEDGE agent and the SRM AIM. When you integrate CA NSM with CA Virtual Assurance, you can access the CA Virtual Assurance user interface from in-context links in the Management Command Center to access the remote agent deployment and configuration functionality.

Note: For more information about the CA NSM integration with CA Virtual Assurance, see the *CA Virtual Assurance* documentation.

Chapter 3: Concepts

This section contains the following topics:

Architecture Overview (see page 39)
Monitoring Architecture (see page 45)
State Management Model (see page 61)
Integration with CA Virtual Assurance (see page 70)
AIM Architecture (see page 73)
Guidelines for Using the SystemEDGE Agent (see page 74)

Architecture Overview

The SystemEDGE agent uses a lightweight, flexible SNMP architecture to monitor systems. SNMP is a standard for managing TCP/IP-based networks and devices. At its most basic level, SystemEDGE performs the agent role in the SNMP framework: it carries out SNMP-based instructions on managed objects. Instructions may come from a management system or from the agent itself, and these instructions may be requests for information about an object or requests to modify system parameters. The SystemEDGE agent carries out these requests and sends a reply to the requesting application. The agent exchanges information adhering to standard SNMP protocol, so any manager that can communicate through SNMP can integrate with the agent.

The SystemEDGE agent accesses system metrics through MIB attributes. A MIB is a store of management information that adheres to a common attribute format. The agent can collect or modify attributes in all MIBs that it supports and maintains.

Note: For more information about supported MIBs, see the chapter "Introduction."

The SystemEDGE agent communicates through SNMP on UDP port 161 by default (you can change this port during installation). In an SNMPv1 environment, it uses the following standard SNMP message types to exchange management information:

GetRequest

Obtains the value of a specific object from the MIB. A management system sends this message to the agent when requesting specific information about a system.

GetNextRequest

Obtains the next object instance.

GetResponse

Returns the requested MIB object information to the requesting application.

SetRequest

Instructs the agent to change the value of a MIB object

Trap

Notifies a management system of exceptions. Traps allow the agent to send unsolicited messages to a manager. For more information, see SNMP Traps (see page 41).

In an SNMPv2 environment, the standard SNMP message types to exchange management information is:

GetBulkRequest

Obtains values (tabular) for one or more than one MIB objects. This is similar to GetNextRequest.

This common SNMP workflow (get or set request, retrieval, or modification of MIB attribute, SNMP reply) comprises the base of all SystemEDGE functionality. However, the agent can move beyond basic SNMP collection to intelligent monitoring and state management. For more information, see Monitoring Architecture (see page 45) and State Management Model (see page 61).

SNMP Versions and Access

SystemEDGE supports SNMPv1, SNMPv2c, and SNMPv3 protocols. Each protocol contains unique features, with the main difference being message security. The agent can transmit SNMP messages using any of the protocols' security access methods.

SNMPv1 and SNMPv2c Communities

SNMPv1 and SNMPv2c use communities for transmitting messages between systems. A community is a relationship between an agent and a number of management systems that defines authentication and access control permissions for distributed communication. A community is identified by a community name, which is a string of octets that appears in the header portion of every SNMP message.

The agent checks the community name in an SNMP message header to determine if the message is authentic, and then processes the message. If the community name does not match a name accepted by the agent, the agent records an authentication failure and drops the message.

The SystemEDGE agent is installed with only one community defined by default, named public. This community provides read-only access to the agent's MIB object values for any manager system. Before you can modify MIB values, you must configure a read-write community string during or after installation. You can add an IP address list to a community definition to limit access to specific systems.

Note: For more information about configuring SNMP communities, see the chapter "Agent Configuration."

SNMPv3 Users and Security

SNMPv3 uses a combination of authentication and encryption over the network to provide secure device access:

Authentication

Helps ensure that the packet was not altered during transmission.

Encryption

Encrypts package contents to prevent viewing from unauthorized sources.

SNMPv3 uses security models and security levels. A security model is an authentication strategy set up for a user and the group in which that user resides. A security level is the permitted level of security in a security model. A combination of a security model and a security level determines the security mechanism for handling SNMP packets. This combination enables secure data collection from SNMP devices and encryption of confidential information to prevent exposure in network transmissions.

The SystemEDGE agent uses the User-based Security Model (USM) for SNMP-based communication. You can configure the SystemEDGE agent to use specified SNMPv3 users, authentication and encryption protocols, and passwords. For more information about enabling and configuring SNMPv3, see the chapter "Agent Configuration."

SNMP Traps

SystemEDGE can send SNMPv1 traps and SNMPv2c and SNMPv3 traps and notifications. When a configured threshold exceeds aggregate state of a monitored object changes, or another specified condition is met (such as log monitor match), the agent sends a state change trap.

SNMP trap configuration consists of configuring trap communities (including destinations) for SNMPv1 and SNMPv2c traps, and SNMPv3 users for SNMPv3 traps. Trap destinations indicate which management systems receive the trap messages that the agent generates. The monitoring features of the agent reports exception condition by sending trap messages to the management systems in the trap community.

Note: For information about configuring the SystemEDGE agent to send traps, see *Agent Configuration* chapter. For more information about the types of traps that the agent can send, see Monitoring Architecture (see page 45).

The SystemEDGE agent stores trap configuration information in the read only trapCommunity table of Systems Management Empire MIB. For more information about this table, including how to disable if you do not want to expose this access information, see *Agent Configuration* chapter. For more information about viewing this table in management systems, see the *CA Virtual Assurance* and *CA eHealth* documentation.

SNMP Tables

The SystemEDGE agent uses tables to represent 'multi-instance' items. The majority of these tables use an integer value as a table index. An index is important for uniquely identifying an item in a table and is persistent across agent restarts.

For example, the self monitor table uses an integer index. The following command (in sysedge.cf file) creates a monitor for the devTable for space utilization of a disk drive. The index(99) is associated with this particular monitor instance:

```
monitor devTableEntry 'C:'  devCapacity 99 0x100 30 absolute >= 95 'exampl'
```

With index value 99, an SNMP query can obtain the current value of a monitor entry. The OID monCurrVal.99 (symbolic notation) can be queried using the SystemEDGE snmpget command as follows:

```
snmpget -h <host> -c public -o 1.3.6.1.4.1.546.6.1.1.6.99
```

The OID monCurrVal.99 (symbolic notation) can also be queried using the symbolic notation:

```
snmpget -h <host> -c public -o  monCurrVal.99
```

Create Monitor by Instance

When you can create a self monitor on tabular MIB object you need to know the index of the monitored instance. For several important MIB tables, the agent offers a look-up feature (used in above example). This enables the following two options:

- Using sysedge.cf configuration file

  For example, although the devTable has an integer index, the lookup feature allows the monitor for a devTableEntry to be defined using a drive letter. SystemEDGE searches the devTable entries for matches of the drive letter. If devTableEntry has an index entry as 3, the monitor's object identifier is set to this index (devCapacity.3).

- Using SNMP

  SystemEDGE determines the monitor's object identifier when the table name, instance name and attribute name are provided in the monitor's MIB attributes: monObjClass, monObjInstance, monObjAttribute.

  The following example creates a monitor entry with an index of 111 that automatically derives the correct devTable index for the monitored OID using the attributes provided. The monitor is actually created when you set the row status.

```
set monObjClass.111 = "devTableEntry"
set monObjInstance.111 = "C:"
set monObjAttribute.111 = "devCapacity"
set monRowStatus.111       = active(1)
```

Note: You can use the snmpset command provided with the SystemEDGE agent to set SNMP values. For more information, see *Command Line Utilities* chapter.

The only two agent tables that are not indexed using an integer value are the mirrorMonitorTable and the mirrorAggregateTable, which use the object class, instance, and attribute values as indices to allow access for a particular managed object.

Configuration Files

The SystemEDGE agent bases its operation on the contents of the following configuration text files:

- sysedge.cf
- sysedgeV3.cf

The agent installation creates the sysedge.cf file in CASYSEDGE\config containing basic information defined during installation, such as system information, SNMP communities, CA Virtual Assurance manager registration, and the data directory (referred to in this document as CASYSEDGE_DATA).

When the SystemEDGE agent starts, it reads the file created during installation to determine how it should operate and creates a copy of sysedge.cf file in a subdirectory of CASYSEDGE_DATA.

Use this version of the file to make post-installation runtime configuration changes. You can edit configuration settings defined during installation and define other settings.

Note: For more information about the location of the installation and data directory, see the chapter "Agent Configuration." This document refers to the installation directory as CASYSEDGE.

The sysedgeV3.cf file contains SNMPv2c and SNMPv3 configuration information. You cannot configure SNMPv2c and SNMPv3 during installation, but you can modify the sysedgeV3.cf file (copied to a subdirectory of data directory when the agent is started initially) to the data directory to enable SNMPv3 operations and define SNMPv3 users, access lists, and security protocols. We recommend that you hold sysedgeV3.cf in a cental configuration repository, or (to take advantage of an automatic deployment of this file) use the CA Virtual Assurance policy-based configuration capability.

Note: For more information about using sysedge.cf and sysedgeV3.cf to configure basic properties and SNMPv3 settings of the agent, see *Agent Configuration* chapter.

The sysedge.cf file also controls monitoring settings of the agent. Any method of configuring monitoring entries (through a user interface such as CA Virtual Assurance or through direct file manipulation) other than submitting SNMP Set requests ultimately populates the sysedge.cf file in the data directory with the monitoring information. The agent uses this file to gather instructions for performing all monitoring operations.

Note: For more information about the monitoring capabilities of the agent, see Monitoring Architecture (see page 45).

The sysedge.mon file acts as a backup file to store the changes made through SNMP Sets that are not automatically entered in the sysedge.cf file. The file also stores runtime information, such as the current state of a monitor. Entries in sysedge.cf take precedence over entries in sysedge.mon. When the agent performs a full (or cold) restart, it first reads the sysedge.mon file in the data directory for monitoring entries, then reads the sysedge.cf file and overwrites existing monitors and adds additional monitors. When the agent performs a partial (or warm) start, it only reads the sysedge.cf file. Use the sysedge.cf file, either through configuration in the CA Virtual Assurance user interface or direct manipulation, to configure the agent.

Monitoring Architecture

The SystemEDGE agent monitoring functionality extends beyond basic SNMP metric collection to intelligent monitoring and exception reporting. You can control what the agent monitors and when the agent returns a trap to the management system.

The agent maintains its monitoring data in monitor tables in the Systems Management Empire MIB. Creating a monitor entry in one of the monitor tables instructs the agent to read the table and act on the entry. Accordingly, the agent polls the system at intervals, and when the return value of a monitored object matches specific criteria (a threshold, a process state, or a regular expression, for example), the agent sends a trap to all systems defined as trap destinations.

The Systems Management Empire MIB contains the following monitoring tables:

Self Monitor table (monitorTable)

Contains MIB objects to monitor and compare to user-specified thresholds to control severity and resultant object status.

Process Monitor table (processMonTable)

Contains processes to monitor for status (that is, whether they are running) and resource utilization.

Process Group Monitor table (processGroupMonTable)

Contains groups of processes to monitor for status (that is, whether a change occurred).

Log Monitor table (logMonitorTable)

Contains log files and directories to monitor for specified regular expression strings.

NT Event Monitor table (ntEventMonTable)

Contains event logs in which to search for specific events.

History Control table (empireHistoryCtrlTable)

Contains the sample interval and number of samples for MIB objects that the agent monitors and stores in the History table for future retrieval by the management system.

Each table contains MIB attributes whose values you specify when you create an entry. The tables are indexed so that each entry contains unique attribute values. The agent reads the tables, and the entry attributes give the agent instructions about what system attributes to monitor and when to report exceptions.

All monitor table entries are maintained in the local sysedge.cf and sysedge.mon files. The sysedge.mon file is used for compatibility reasons only. Directives let you enter entries directly in the sysedge.cf file, or you can configure entries in a management product that integrates with the agent. When you use the agent with CA Virtual Assurance, you can create monitor entries and deploy a file with those entries to agents on one or more managed systems.

SNMPv2 Row Status

SystemEDGE uses the concept of row status introduced with SNMPv2 to manage the creation, activation, and maintenance of rows (or entries) in any of the monitoring MIB tables. The RowStatus textual convention manages the creation and deletion of rows and is the value of the SYNTAX clause for the status column of a conceptual row:

```
RowStatus ::= TEXTUAL-CONVENTION

STATUS  current
```

Each entry that you define in one of the monitoring tables becomes a row in the table and has a row status value. The row status determines whether the row is active with accessible data. The available values for row status are as follows:

active

Indicates that the row is available for use by the managed device. A monitor entry must have a row status of active for the agent to carry out the monitoring instructions in the entry. By default, when you create an entry with valid information, the row status is automatically set to active.

notInService

Indicates that the row exists in the agent, but is unavailable for use by the managed device.

notReady

Indicates that the row exists in the agent, but is missing the necessary information for use by the managed device.

createAndGo

Supplied by a management station wanting to create a row and to have it available for use by the managed device.

createAndWait

Supplied by a management station wanting to create an instance of a conceptual row but not to have it available for use by the managed device.

destroy

Supplied by a management station wanting to delete a row.

You can supply any of these statuses during row creation except for notReady and destroy. You use the destroy status to delete an existing row. Only the following three values are returned in response to a manager Get request:

active

Indicates that the row is available for use by the managed device.

notInService

Indicates that the row is not available for use by the managed device, though the agent has sufficient information to make it available.

notReady

Indicates that the row is not available for use by the managed device because the agent lacks sufficient information.

Note: For more information about controlling the row status of a monitor entry, see the monitoring chapters in this guide. For more information about the RowStatus textual convention, see RFC 1443: Textual Conventions for SNMPv2.

How Stateful Monitoring Works

Stateful monitoring uses defined severity information and the object model to assign a state to each object represented by a monitor or set of monitors. The following tables support stateful monitoring:

- Self Monitor (see page 50)
- Process Monitor (see page 55)

When you create an entry in the Self Monitor table, you specify a MIB object (OID) to monitor for a specific threshold together with a severity and object information. When you create an entry in the Process Monitor table, you specify a process to monitor for its running state or a process attribute threshold. This process describes how the agent handles stateful monitors with severity information. The agent operates differently when handling legacy monitors, which have no severity.

Note: For specific information about self and process monitoring, see the specific section for each type of monitoring in the chapter "Self Monitoring" and "Process Monitoring."

The SystemEDGE agent carries out stateful monitoring instructions as follows:

1. You create an entry in an appropriate Systems Management Empire MIB table to monitor a process group, log file, or Windows event log for specified group changes, regular expressions, or events.

   The agent requires a restart to recognize changes to the sysedge.cf file unless you use CA Virtual Assurance. If you make an entry and deploy the changes through CA Virtual Assurance, the agent performs a warm start and begins reading the entry while it is running.

2. (Process monitoring only) The agent uses the regular expression you defined in the entry to determine the process ID (or Windows Service index) to monitor.

3. The agent polls specified MIB attributes (OIDs) or process attributes according to a specified poll interval.

4. When an attribute exceeds a threshold, the agent sets the current status of the monitor entry. The current status is set to a value as defined in the severity column of entry.

   Note: The agent aggregates the state of monitors with the same object information, identical values for monObjClass, monObjInstance, and monObjAttribute. For more information about monitor aggregation, see How State Management and Aggregation Works.

5. The agent sends a state change trap to all systems in the trap community. The trap specifies object information, the threshold exceeded, and the current object state (worst state of all aggregated monitors).

6. The agent executes any action that are defined in the table entry (of the worst monitored object). For example, if you defined a process restart action for a stopped process, the agent restarts the process.

7. The agent continues to poll the attribute, and if the threshold condition disappears, the agent updates the current status value of the monitor and sends a state change trap (if the worst status changes).

For more information about operating the agent with legacy monitors, see Legacy Monitors (see page 69).

How Stateless Monitoring Works

Stateless monitors do not derive object status information or use the object model to maintain an overall object state. These monitors do maintain a severity value, but this severity is for tracking the importance of the individual monitor and is not used to calculate object state. The following tables support stateless monitoring:

- Process Group Monitor (see page 57)
- Log File Monitor (see page 58)
- NT Event Monitor (see page 59)

When you create an entry in the Process Group Monitor table, you specify a set of processes to monitor for changes and to calculate group statistics. When you create an entry in the Log File Monitor table, you specify a log file to monitor for regular expressions. In the Log File Monitor table, you can additionally specify a directory to calculate for number of entries and size. When you create an entry in the NT Event Monitor table, you specify regular expressions for monitoring specific Windows event logs for specific types of events.

Note: For specific information about process group, log file, and Windows event monitoring, see the specific section for each type of monitoring.

The SystemEDGE agent carries out stateless monitoring instructions as follows:

1. You create an entry in appropriate Systems Management Empire MIB table to monitor a process group, log file, or Windows event log for specified group changes, regular expressions, or events.

   Note: For more information about creating stateless monitor entries, see the chapters "Process Group Monitoring," "Log File Monitoring," and "Windows Event Monitoring."

   The agent requires a restart to recognize changes to the sysedge.cf file unless you use CA Virtual Assurance. If you make an entry and deploy the changes through CA Virtual Assurance, the agent performs a warm start and begins reading the entry while it is running.

2. According to specified polling intervals, the agent reads the entry and searches the system, the specified log file, or the specified Windows event log for processes, entries, or events that match the specified regular expression.

3. (Process group monitoring only) The agent populates the Process Group Monitor table entry with the number of processes matching the regular expression and the PIDs of the monitored processes. It polls the processes and updates the values of the read-only attributes for the process group monitor entry that contain aggregating group statistics and checks for group membership changes.

4. The agent sends a trap to all systems in the trap community when it detects a process group change, a matching log file message, a matching event.

5. The agent executes any actions that you defined in the table entry.

6. The agent continues to poll the process group, log file, directory, or Windows event log for trap conditions.

Self Monitoring

The SystemEDGE agent can monitor attributes of the listed types (INTEGER, GAUGE, COUNTER, and TIMETICKS) from supported MIBs or AIM implementations. The agent can monitor for user-defined thresholds and can report threshold breaches to the management system with a trap. Monitoring with the Self Monitor table lets you configure what to monitor, the exception conditions to report, and the severities of each condition.

When you create an entry in the Self Monitor table, you set the polling interval, comparison operator, threshold value, and severity, and the agent automatically monitors the MIB objects that you specify. The terms MIB object and MIB attributes are used interchangeably. The agent uses the provided severity value to determine the status of an object, and it can aggregate the status of objects with multiple monitors defined.

For example, you can configure SystemEDGE to monitor the used space on a particular file system. To monitor thresholds with multiple severities, you create several entries in sysedge.cf as follows:

```
monitor devTableEntry 'C:' devCapacity 30 0x0 30 absolute >= 80 'C:' '' 'SysHealth'
'sysdrive' 'FS usage' ‘ok’

monitor devTableEntry 'C:' devCapacity 30 0x0 120 absolute >= 80 'Used space on C:
drive' '' 'SysHealth' 'sysdrive' 'FS usage' 'warning'

monitor devTableEntry 'C:' devCapacity 31 0x0 120 absolute >= 85 'Used space on C:
drive' '' 'SysHealth' 'sysdrive' 'FS usage' 'minor'
monitor devTableEntry 'C:' devCapacity 32 0x0 120 absolute >= 90 'Used space on C:
drive' '' 'SysHealth' 'sysdrive' 'FS usage' 'critical'
```

This example defines multiple entities with different severities for each threshold. The monitored object is defined by the parameters class (SysHealth), instance (sysdrive), and attribute (FS usage), and the status of this object is either ok (if no threshold is crossed) or minor, warning, or critical if the corresponding threshold is breached. Whenever object status changes, the agent sends a state change trap.

Note: For more information and detailed monitoring examples, see Self Monitoring (see page 193) chapter.

More Information:

How Stateful Monitoring Works (see page 47)

Supported MIBs

When you create an entry in the Self Monitor table, you can instruct the agent to monitor attributes in any supported MIB for specified thresholds:

- CA Systems Management Empire MIB
- MIB-II (RFC 1213)
- Host Resources MIB (RFC 1514)
- IP-MIB (partial) (RFC 4293)
- IF-MIB (partial) (RFC 2233)
- TCP-MIB (partial) (RFC 4022)
- UDP-MIB (partial) (RFC 4113)
- Any SystemEDGE Version 5.0 AIM MIB

For more information about the objects in the Host Resources MIB, see the appendix "Host Resources MIB." For more information about the objects in MIB-II, see RFC 1213.

Systems Management Empire MIB Groups

The Systems Management Empire MIB is an enterprise-specific collection of attributes that contains tables and groups of objects for self-monitoring.

The following groups and tables have objects that you can monitor in the Systems Management Empire MIB:

System Group (sysedgeSystem)

Contains basic system information such as host name, CPU type, amount of memory, time zone, fully qualified domain name of the system, list of loaded plug-ins, or SystemEDGE's uptime.

Mounted Devices Table (devTable)

Contains information about devices and file systems mounted on the host such as mount point, block size, free blocks, inodes in use, or maximum file name length. You can also create monitors for values such as file system usage or unmount a mounted device by setting a column value in this table.

Kernel Configuration Group (kernelConfig)

Contains kernel information such as number of CPUs, amount of virtual memory, memory per process, open files per process, swap space, or clock rate. Use this group to monitor kernel properties and configuration.

Boot Configuration Group (bootconf)

Contains information about the root file system, dump file system, and swap space. Monitor this table to track values such as root file system name, file system blocks, or file system type.

Streams Group (streams)

Contains information about the streams I/O subsystem. Use this group to monitor objects such as number of streams in use, number of stream allocation failures, and number of streams in queue to track the health of the subsystem.

User Table (userTable)

Contains information about the user accounts on the system.

Group Table (groupTable)

Contains information about the user groups on the system.

Process Table (processTable)

Contains information about running processes. Monitor this table to track the processes that are currently running, and to set specific attributes to control processes. For example, you can stop a process by setting the value of the processkill column to 9.

Who Table (whoTable)

Contains information about the users currently logged on to the system. Monitor attributes in this table to track who is using a system at any given time.

Remote Shell Group (remoteshell)

Contains attributes for running shell scripts and programs on the remote system. Set the attributes in this table to specify a command, its arguments, and the name of an output file.

Kernel Performance Group (kernelperf)

Contains information about the health and performance of the host operating system. You can monitor attributes such as the number of current processes and open files, the number of active jobs, and the number of jobs in the scheduler's run queue.

Interprocess Communication Tables (msgqueTable, shmemTable, semTable)

Contains information about message queues, shared memory, and semaphores in separate tables. Monitor these tables to coordinate communication between processes.

Message Buffers Allocation Table (mbufAllocTable)

Contains information about how your system is using message buffers. Monitor attributes in this table to track information such as the number of times buffer requests were denied or delayed.

Streams Buffers Allocation Table (strbufAllocTable)

Contains information about buffer allocation and usage statistics for buffers used by the streams subsystem.

I/O Buffer Cache Group (ioBufferCache)

Contains information about I/O buffer allocation and usage for basic disk I/O. Monitor this table to track information such as peak periods of I/O buffer activity.

Directory Name Lookup Cache Group (dnlc)

Contains information about directory and file name cache performance.

AIX Logical Partition Group (logicalPartition)

Contains information about IBM AIX logical partitions (LPARs). You can monitor attributes such as physical or logical CPU for each partition and the number of CPUs for each partition.

Trap Community Table (trapCommunityTable)

Contains SNMP information such as configured communities, users, and trap destinations.

NT System Group (ntSystem)

Contains information specific to Windows systems. This group contains System, Thread, Registry, Service, System Performance, Cache Performance, Memory Performance, Page File Performance, and Event Monitor groups to monitor attributes for these areas on Windows systems.

RPC Statistics Group (rpc)

Contains information about kernel remote procedure calls. Monitor this table to track attributes such as counters and statistics for detecting peak periods of RPC activity.

NFS Statistics Group (nfs)

Contains information about the kernel's NFS facility. Monitor this table to track attributes such as statistics and counters for detecting peak periods of NFS activity.

Disk Statistics Table (diskStatsTable)

Contains information about disk I/O.

CPU Statistics Table (cpuStatsTable)

Contains performance statistics for each CPU. Monitor this table to track attributes such as time spent in Idle mode and time spent in Wait mode.

The Systems Management Empire MIB also contains the monitoring tables and tables to support object aggregation. For more information about object aggregation, see State Management Model (see page 61).

Self Monitor Traps

The SystemEDGE agent sends the following traps when monitoring entries in the Self Monitor table that contain a severity greater than none(1):

aggregateStateTrap

The agent sends this trap to indicate that the state of a monitor has changed. Monitors with the same object information are aggregated. The corresponding aggregate state is equal to the worst state of these aggregated monitors.

For example, if a monitor with a severity of critical is breached, but the aggregate monitor's current state is fatal (which signifies that a separate monitor for the same attribute with a severity of fatal has breached its threshold), the agent does not send a trap because the aggregate state is not affected.

Note: For more information about monitor aggregation and other aggregation traps, see State Management Model (see page 61).

The SystemEDGE agent sends the following traps when monitoring entries in the Self Monitor table that have a severity of none (legacy monitors):

monitorTrap

The agent sends this trap when a monitored attribute's threshold condition has been breached.

monitorEntryNotReadyTrap

The agent sends this trap when the row status of a Self Monitor table entry is set to notReady(3). A notReady row status may indicate that the object is unreachable.

monitorClearTrap

The agent sends this trap to indicate that a threshold breach that previously existed no longer exists.

The traps for legacy monitors are not sent if you define a severity in the entry. For more information about legacy monitors and using the agent in legacy mode, see Legacy Monitors (see page 69).

For more information about these and other traps, see the appendix "Private Enterprise Traps."

Process and Service Monitoring

The SystemEDGE agent can monitor any process or Windows service to check their running status or the value of a process attribute. You can configure the processes or services to monitor and the attributes to track by creating entries in the Process Monitor table of the Systems Management Empire MIB.

When you create an entry in the Process Monitor table, you set the polling interval, a regular expression indicating the process (or service) to monitor, comparison operator, threshold value, and severity, and the agent automatically determines the process ID and monitors the process attribute that you specify. The agent uses the provided threshold and severity value to determine the status of the process attribute, and it can aggregate the state of multiple monitors defined to derive overall object state.

For example, you can monitor a mission-critical process for system calls, and when they exceed a provided threshold, the agent adjusts the severity of the object and sends a state change trap. You also can monitor whether the process is running, and the agent can automatically restart a stopped process.

Note: For more information, see the chapter "Process and Service Monitoring."

More Information:

How Stateful Monitoring Works (see page 47)

Process Monitor Traps

The SystemEDGE agent sends the following traps when monitoring entries in the Process Monitor table that contain a severity greater than none(1):

aggregateStateTrap

The agent sends this trap to indicate that the state of a process monitor has changed. The agent only sends this trap when a threshold condition breach changes the state of a process monitor. For example, if a monitor with a severity of critical is breached, but the aggregate monitor's current state is fatal (which signifies that a separate monitor for the same attribute with a severity of fatal has breached its threshold), the agent does not send a trap because the aggregate state is not affected.

Note: For more information about monitor aggregation and other aggregation traps, see State Management Model (see page 61).

The SystemEDGE agent sends the following traps when monitoring entries in the Process Monitor table that have a severity of none (legacy monitors):

processThresholdTrap

The agent sends this trap when a monitored process attribute's threshold condition has been breached.

processStopTrap

The agent sends this trap when a monitored process has stopped or is in a state where it cannot run.

processStartTrap

The agent sends this trap to indicate that a stopped monitored process has restarted.

processClearTrap

The agent sends this trap to indicate that a threshold breach that previously existed and caused a trap generation no longer exists.

processNotReadyTrap

The agent sends this trap when the row status of a Process Monitor table entry is set to notReady(3). A notReady row status may indicate that the process or attribute is unreachable.

The traps for legacy monitors are not sent if you define a severity in the entry. For more information about legacy monitors and using the agent in legacy mode, see Legacy Monitors (see page 69).

For more information about these and other traps, see the appendix "Private Enterprise Traps."

Process Group Monitoring

The SystemEDGE agent needs an additional self monitor to monitor a set of processes to sum up the single process attribute values and track the processes in the group. You can configure to monitor the processes as a group by creating entries in the Process Group Monitor table of the Systems Management Empire MIB (needs extra self monitors).

When you create an entry in the Process Group Monitor table, you set the polling interval, a regular expression for populating a process group, and the monitor severity, and the agent monitors all processes that match the regular expression as a group. The agent populates read-only attributes of the Process Group Monitor table with sum group statistics, such as the combined process size, thread count, and real memory usage. If membership in the process group changes (for example, if the agent detects a new process matching the regular expression or if a process in the group becomes unreachable), the agent sends a group change trap. On UNIX, you can also configure the agent to match processes by user name and group name.

For example, you can monitor a group of important processes and be notified when one of the processes goes down. If a process goes down, you can set self monitors on the read-only attributes in the Process Group Monitor table. For example, you can monitor the total CPU time used by a set of processes and set a threshold that corresponds to a service level agreement associated with the corresponding applications.

Note: For more information, see the chapter “Process Group Monitoring."

More Information:

How Stateless Monitoring Works (see page 48)

Process Group Monitor Traps

The SystemEDGE agent sends the following trap when monitoring entries in the Process Group Monitor table:

procGroupChange

The agent sends this trap to indicate that the process group membership has changed, either due to a new process matching the process group monitor entry specifications or a previously monitored entry becoming unreachable. If you defined a severity for the monitor, the trap includes this severity value.

For more information about this trap and other traps, see the appendix "Private Enterprise Traps."

Log File Monitoring

The SystemEDGE agent can monitor any text files in a row-wise for regular expressions. To configure the agent to monitor log files, you create entries in the text (Log) File Monitor table of the Systems Management Empire MIB. Log file monitoring lets you automatically monitor multiple system or application log files for entries that require action.

When you create an entry in the Log File Monitor table, you set the polling interval, the log file(pattern) to monitor, the monitor severity, and a regular expression for which to search. The agent searches the log file for the specified regular expression and sends a trap when the file contains a message that matches the expression.

For example, you can configure the SystemEDGE agent to monitor an application log file for failure messages or messages that indicate unauthorized or incorrect usage.

You can also create entries in the Log File Monitor table to monitor file directories for size and contents (needs extra self monitors).

Note: For more information, see *Log File Monitoring* chapter.

More Information:

How Stateless Monitoring Works (see page 48)

Log File Monitor Traps

The SystemEDGE agent sends the following traps when monitoring entries in the Log Monitor table:

logMonMatchTrap

The agent sends this trap to indicate that the agent has detected a match in a log file against a specified regular expression. If you defined a severity for the monitor, the trap includes this severity value.

logMonNotReadyTrap

The agent sends this trap when the row status of a Log Monitor table entry is set to notReady(3). A notReady row status may indicate that the file is unreachable.

For more information about these and other traps, see the appendix "Private Enterprise Traps."

Windows Event Monitoring

The SystemEDGE agent can monitor Windows event logs for important event types, event identifiers, or events that match specific regular expressions. You configure the agent to monitor Windows event logs by creating entries in the NT Event Monitor table of the Systems Management Empire MIB. Creating Windows event monitoring entries lets you manage Windows event logs for critical events or those that require action. Because Windows events include several identifying characteristics in addition to the textual message, this monitoring capability lets you specify more sophisticated types of matches than basic log file monitoring.

When you create an entry in the NT Event Monitor table, the NT event monitors poll at regular intervals. The event type and source to monitor, the monitor severity, and a regular expression to match the event message text (including event ID). The agent monitors the Windows event logs and sends a trap when an event that matches the expression appears.

For example, you can configure the SystemEDGE agent to monitor the Windows Security event log for authentication failures.

Note: For more information, see the chapter “Windows Event Monitoring."

More Information:

How Stateless Monitoring Works (see page 48)

Windows Event Monitor Traps

The SystemEDGE agent sends the following traps when monitoring entries in the NT Event Monitor table:

ntEventMonMatchTrap

The agent sends this trap to indicate that the agent has detected a match in a Windows event log against a specified regular expression. If you defined a severity for the monitor, the trap includes this severity value.

ntEventMonNotReadyTrap

The agent sends this trap when the row status of an NT Event Monitor table entry is set to notReady(3). A notReady row status may indicate that the Windows event log store is unreachable.

For more information about these and other traps, see the appendix "Private Enterprise Traps."

History Collection

The SystemEDGE agent can sample attributes values (metrics) from any supported MIB, and you can use the collected samples for baselining and trend analysis of your system. The samples populate the History Table, and you can optionally have the samples written into a cube file.

To configure history collection, you create entries in the History Control table of the Systems Management Empire MIB. Configuring history collection lets you analyze metrics over-time without having to constantly poll a manager. Data written to cube files is available for analysis by CA Virtual Assurance Systems Performance.

When you create an entry in the History Control table, you set the polling interval, the MIB attribute to collect, and how many samples to save. SystemEDGE collects the data as configured, and a manager can periodically retrieve the history for a view of an attribute over a specific interval.

For example, you can configure the agent to collect CPU MIB attributes from the CPU Statistics table of the Systems Management Empire MIB every 10 minutes and save the last 288 samples. You can then configure a management system to upload the data every 48 hours to obtain all collected data for analysis.

For more information about configuring history collection, see the chapter “History Collection."

How History Collection Works

The SystemEDGE agent carries out history collection instructions as follows:

1. You create an entry in the History Control table of the Systems Management Empire MIB to monitor a MIB attribute over a specified time interval.

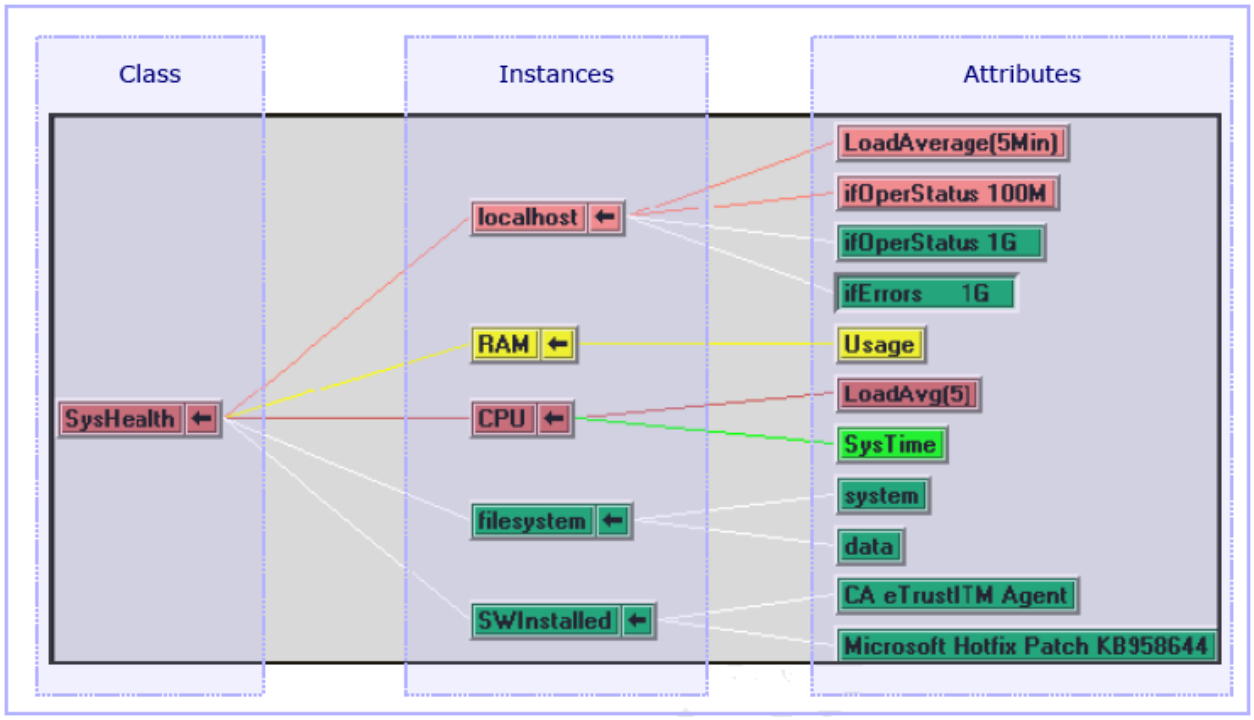
   Note: For more information about creating history collection entries, see the chapter "History Collection."

   The agent requires a restart to recognize changes to the sysedge.cf file unless you use CA Virtual Assurance. If you make an entry and deploy the changes through CA Virtual Assurance, the agent performs a warm start and begins reading the entry while it is running.

2. The agent polls the attribute according to a specified poll interval and collects the attribute information into the History table (sliding window). When the specified number of samples is reached, the oldest sample is dropped.

State Management Model

The SystemEDGE agent supports a state management model for self monitors and for process monitors fully integrated with the overall CA Virtual Assurance Management Model. The agent aggregates multiple monitors of different severities into a single Managed Object, which has a state corresponding to the monitor with the worst severity. The following illustration shows the flexibility and capabilities provided that this new mechanism provides:



The Attributes pane in the illustration, contains several monitored objects (such as SysHealth.localhost.LoadAverage). The states that the SystemEDGE agent can assign to an object are compliant with CIM ManagedSystemElement Management states. The state of each attribute object is defined by the corresponding SystemEDGE monitor. For example, if you assign LoadAverage threshold of 250 with a fatal severity and the current LoadAverage value is 250, the state of the monitored object is fatal.

The flexible object model of SystemEDGE enables you to group disparate objects from different sources. The state of the objects is aggregated across each group. For example, the agent can aggregate the health state of the server's CPU with the response time taken to access a Web page (gathered by the SRM AIM) with the performance of a disk (gathered by the Remote Monitoring AIM).

The Class pane in the illustration shows, how the state management model groups the managed objects into aggregate managed objects. Here, as illustrated, SysHealth object aggregates the state of all objects managed by it.

You can configure the state management functionality from the Self Monitor and Process Monitor tables of the Systems Management Empire MIB, in which columns containing object and severity information guide the calculation and propagation of object status for each monitor entry.

Object Status

The severity values entered in the Self Monitor and Process Monitor tables determine the object status of a monitored object. A specific status is assigned to a monitored object which breaches a specific threshold condition. Severity values entered in the three other monitoring tables do not determine a status for a monitored object, but the importance of the event associated with the monitor (stateless monitoring).

You can enter the following severity values for a monitor:

- none (1)
- ok (2)
- warning (3)
- minor (4)
- major (5)
- critical (6)
- fatal (7)

A severity of none (1) creates a legacy monitor for, which monitors the object without sending state change traps. None is the default severity if you do not specify a value in the objectState column, or if you are upgrading monitors from a previous release. For more information, see Legacy Monitors (see page 69).

You can use ok (2) as a temporary 'do no monitor' feature because with current state as ok(2) and a monitored attribute severity breach, state change trap is not sent.

The resultant object state, calculated by the agent and recorded as a read-only attribute in the monitor entry, can be one of the following:

- unknown (1)
- ok (2)
- warning (3)
- minor (4)
- major (5)
- critical (6)
- fatal (7)
- up (11)
- down (12)

If a monitor has a severity of none, the state toggles between up (11) and down (12).

When the agent shuts down, it saves the current state of any monitor. These saved states are used to initialize monitor states during agent startup.

Object Aggregation

The object class, instance, and attribute values in the Self Monitor and Process Monitor tables describe the monitored (or managed) object. Multiple monitors within same object instance, class, and attribute values are considered to be connected and describe the same managed object. The agent aggregates the worst state of all connected monitors into a single managed object state. The agent creates an entry (aggregate) for each aggregation in the Aggregate table.

However, if two connected monitors A and B have the same severity and use “AND” flag (00800 in monFlags or 20000 in pmonFlags), A and B are excluded from worst state calculation, if A and B do not have the same state. If you do not enter values into the corresponding columns, the agent populates them with meaningful default information. It bases default self monitor values on the monitored OID and maps them to class, instance, and attribute values. It bases default process monitor values on the process regular expression and monitored attribute. For more information about default object information assignment, see the chapters "Self Monitoring" and "Process and Service Monitoring."

All connected monitors must have the same interval which helps ensure that the aggregate state is evaluated only once for all connected monitors and prevents unnecessary state changes.

The Aggregate table contains the following information:

- Object class, instance, and attribute values
- Row status
- Values for the monitor entry with the worst aggregated state, such as threshold, operator, and current value
- Current aggregate state
- Monitor statistics, such as the number of state change traps sent, the time spent in any of the possible states, and the previous status

Note: For more information about the columns in the Aggregate table, see the chapter "Self Monitoring."

The agent updates the current state in the Aggregate table. The agent sends a state change trap only when the threshold evaluation of all connected monitors and the subsequent state aggregation leads to a different worst state than the previous one. For example, assume the agent returns 82% CPU usage, while you monitor CPU usage with three monitors: one for 60% (assigned a warning severity), one for 80% (critical severity), and one for 100% (fatal severity). This value causes a threshold breach for the 60% and 80% monitors, but the agent only sends one state change trap for the 80% monitor and changes the aggregate state to critical.

Super Aggregation

The Aggregate Table contains per default an entry for all self and process monitors with the same object class, instance, and attribute, but different severities (different thresholds). The table aggregates on a severity level (standard aggregates).

Additionally, the table enables super aggregations on a class level, attribute level or instance level. Corresponding table entries use an asterisk (*) as a placeholder.

Attribute level

There is one entry for all monitors with the same object class and instance (aggObjAttribute set to *)

Instance level

There is one entry for all monitors with the same object class and attribute (aggObjInstance set to *)

Class level

There is one entry for all monitors with the same object instance and attribute (aggObjClass set to *)

A combination of the above three levels can be configured in the sysedge.cf file, causing aggregations up to an overall agent state.

The index of super aggregates starts at 100001.

Important! Setting the row status attribute of the *, *, * entry to destroy (6), effectively deletes all monitors with a severity greater none (1).

Aggregation levels can be configured in sysedge.cf file according to thr following format:

`aggregate_level` *`level`*

level

Specifies a bit field allowing to specify the following aggregation levels:

001: Aggregate all monitors with same class and instance in an entry with attribute set to *

002: Aggregate all monitors with same class and attribute in an entry with instance set to *

004: Aggregate all monitors with same instance and attribute in an entry with class set to *

010: Aggregate all monitors with same class in an entry with instance and attribute set to *

020: Aggregate all monitors with same instance in an entry with class and attribute set to *

040: Aggregate all monitors with same attribute in an entry with class and instance set to *

100: Aggregate all monitors in an entry with class, instance and attribute set to *

The entries of the aggregate table (only some selected attributes) is stored in the sysedge.mon file. The main purpose is to store the index of an entry between restarts, because typically a manager uses the index to poll the object and relies that the index does not change frequently.

How State Management and Aggregation Works

The SystemEDGE agent calculates and aggregates object status while carrying out self monitoring and process monitoring instructions as mentioned in this section.

Note: This process represents the default state management and aggregation behavior. For information about configuring the agent to manage status and aggregate object status differently, see State Management Configuration Options (see page 68).

1. You create a self monitor or process monitor entry that contains a severity greater than none in the Self Monitor or Process Monitor table. You should define object instance, class, and attribute information, or the agent will populate these columns with meaningful default information.

   The agent requires a restart to recognize changes to the sysedge.cf file unless you use CA Virtual Assurance. If you make an entry and deploy the changes through CA Virtual Assurance, the agent performs a warm start and begins reading the entry while it is running.

2. The agent monitors the attribute defined in the entry according to a specified polling interval and derives the current state of monitor according to the specified row status, threshold breach condition, and severity, and populates a read-only attribute in the monitor entry with the current state.

3. The agent adds the state information to the Aggregate table in the following way:

   - If an entry in the Aggregate table already exists with the object class, instance, and attribute values of the monitor. The agent (internally) connects the monitor to that entry in the table.
   - If no entry in the Aggregate table exists with the monitor's object class, instance, and attribute values, the agent adds a new entry to the Aggregate table.

4. The agent sends an aggregate add trap to all systems in the trap community when it adds a new entry in the Aggregate table.

5. When a monitor crosses a threshold, the agent changes the current state of the monitor in the Self Monitor or Process Monitor table, aggregates the worst state of all connected monitors and updates the corresponding entry in the Aggregate table.

6. The agent sends a state change trap only if the state of the aggregate changed.

   Note: Aggregation adheres to the specified lag setting. If you set a lag value for monitor entries, individual monitor state changes or aggregate state changes do not occur until the lag time is satisfied. For more information about using table flags to set lag values, see *Self Monitoring* and *Process and Service Monitoring* chapters.

7. The agent executes actions associated with the worst monitor.

8. The agent continues to poll the attribute, and behaves as follows in the following situations:

   - If the threshold breach condition endures, the agent keeps executing the action associated with the monitor (as long as it still represents the worst state monitor) but does not send additional state change traps.
   - If the threshold breach condition disappears, the agent updates the current state in the monitor, updates the current state in the Aggregate table with the worst current state and sends the appropriate state change trap.
   - If a connected monitor with a greater severity crosses its threshold, the agent changes the aggregate state and sends a state change trap.

   Note: The agent continues to manage individual self monitor and process monitor entries in the Self Monitor and Process Monitor tables. The agent continues to poll attributes for each individual monitor and maintains the state of each monitor in the tables. For these reasons, you should set the same poll interval for all connected monitors so that all object monitors are evaluated simultaneously and prevents unnecessary state changes (for example, from 'ok' to 'critical', followed by 'critical' to 'fatal' with staggered poll intervals).

State Management Traps

The SystemEDGE agent sends the following traps when monitoring self monitor and process monitor entries that contain a severity greater than none:

Note: For self monitor and process monitor entries with a severity of none (that is, legacy monitors), the agent sends the threshold breach traps from previous agent releases instead. For more information about these traps, see Self Monitor Traps and Process Monitor Traps.

aggregateStateTrap

The agent sends this trap when the state of an aggregate changes. When a threshold breach of a connected monitor changes the aggregate state. For example, if a monitor with a severity of critical is crossed, but the aggregate's current state is fatal, the agent does not send a trap because the aggregate state is not affected and will remain fatal.

Note: You can configure the agent to send state change traps even if the aggregate state is not affected. For more information, see the chapter "Agent Configuration."

aggregateAddTrap

The agent sends this trap when a row is added to the Aggregate table (that is, when a monitor with a severity greater than none is created and no corresponding aggregate with the monitor's object information already exists). Otherwise, the new monitor is added (internally) as a connected monitor to the existing row in the Aggregate table and does not trigger an aggregateAdd trap.

aggregateDeleteTrap

The agent sends this trap when a row is deleted from the Aggregate table.

State Management Configuration Options

You can configure the following aspects of monitor aggregation:

- You can configure the agent to aggregate identical entries in the Aggregate table into super aggregate objects. This can be done independently at the class level, instance level, and attribute level.

  For example, if you configure the agent to aggregate (super) at the attribute level, the agent creates additional entries in the Aggregate table for all aggregates with the same object class and instance but different attributes. The attribute value of the super aggregates is an asterisk (*). Using this example, you can aggregate all monitored attributes of a single process to calculate one aggregate state for the process. You can use a parameter in the sysedge.cf file to configure any of these aggregation combinations.

  Note: For more information about and examples of configuring aggregation levels, see *Agent Configuration* chapter.

- By default, the agent sends aggregate state change traps only. You can use a parameter in the sysedge.cf file to configure the agent to send legacy traps for threshold breaches of all connected monitors.

- By default, the agent performs actions only for the connected monitor with the worst state. You can use a parameter in the sysedge.cf file to configure the agent to perform actions for all connected monitors.

  Note: For more information about configuring aggregate traps and actions, see *Agent Configuration* chapter.

- The aggRowStatus attribute is the only read-write attribute in the Aggregate table. The table aggregates the row status of connected monitors similar to how it aggregates object status. Manipulating the row status attribute lets you change the aggregate row status of all connected monitors. You can set the row status to active (1), notInService (2), or destroy (6), and all connected monitors inherit the same status. Setting the row status to destroy deletes an entry from the Aggregate table.

  Note: For more information about row status aggregation and deleting entries from the Aggregate table, see *Self Monitoring* and *Process and Service Monitoring* chapters.

Legacy Mode Operation

You can operate SystemEDGE in legacy mode, that is, without a CA Virtual Assurance Manager controlling its configuration. An agent running in legacy mode is not restricted to legacy monitors.

Legacy Monitors

Existing managers can monitor SystemEDGE attributes and processes without incorporating the object state model. All of the state management columns in the Self Monitor and Process Monitor tables are optional; leave these fields blank if you want the agent to monitor attributes and processes without calculating status information.

Self and process monitors that operate without maintaining and calculating state are legacy monitors. These monitors function as did all monitors previous to SystemEDGE Version 5.0. When you do not enter a severity for a self or process monitor, the monitor inherits a severity of none (1) by default. All monitors with a severity of none (1), whether defined manually or by default, are legacy monitors. They can only have a current state of up or down, and they send legacy threshold traps instead of state change traps.

Note: For more information about the legacy threshold traps, see Self Monitor Traps and Process Monitor Traps.

Monitor Examples

The following example display monitors and resulting aggregate state:

Aggregate State = Critical

Monitor	Severity	AND Flag	Breached	State
M1	Warning	No	Yes	Warning
M2	Warning	No	No	Ok
M3	Critical	No	Yes	Critical
M4	Critical	No	No	Ok

Aggregate State = Ok

Monitor	Severity	AND Flag	Breached	State
M1	Warning	Yes	Yes	Warning
M2	Warning	Yes	No	Ok
M3	Critical	Yes	Yes	Critical
M4	Critical	Yes	No	Ok

Aggregate State = Warning

Monitor	Severity	AND Flag	Breached	State
M1	Warning	Yes	Yes	Warning
M2	Warning	Yes	Yes	Warning
M3	Critical	Yes	Yes	Critical
M4	Critical	Yes	No	Ok

Integration with CA Virtual Assurance

CA Virtual Assurance is a management solution designed to manage physical and virtual systems with the SystemEDGE agent. The agent tightly integrates with CA Virtual Assurance to provide functionality that no other manager or standalone agent offers. Integration with CA Virtual Assurance lets you view SystemEDGE data and remotely configure and deploy SystemEDGE to all managed systems from the CA Virtual Assurance user interface.

When you install the agent (through CA Virtual Assurance or standalone), you specify ports and traps which are necessary to make the agent visible and manageable through the CA Virtual Assurance manager. You also specify a data directory and default port during installation. CA Virtual Assurance uses the data directory to deliver configuration changes. The data directory is appended with the default port number. For example, if you use the default port 161, the data directory would be C:\Program Files\CA\SystemEDGE\data\port161 (on Windows).

Platform Management Modules

The CA Virtual Assurance manager component achieves integration with the SystemEDGE agent through Platform Management Modules (PMMs). The SystemEDGE PMM is a web service that polls the SystemEDGE agent on all managed systems using SNMP. It converts all information received from the agent, including MIB object values and traps, into properties that fit into the CA Virtual Assurance architecture and creates corresponding managed object representations in the CA Virtual Assurance database. The SystemEDGE PMM provides visualization of SystemEDGE information through these managed objects in the CA Virtual Assurance user interface. You can view all installed agents, system details, existing monitors, and status information and make configuration changes from the user interface.

All AIMs compatible with visualization and instrumentation in CA Virtual Assurance have separate PMMs. For CA Virtual Assurance, PMMs are included for the Service Response Monitor AIM, Remote Monitoring AIM, and all the virtual monitoring AIMs.

For more information about Platform Management Modules and how they integrate agent information into the CA Virtual Assurance framework, see the CA Virtual Assurance documentation.

Remote Configuration and Deployment

CA Virtual Assurance provides end-to-end management of SystemEDGE from a centralized location through file-based configuration and deployment. File-based configuration does the following:

- Allows for changes that SNMP managers are unable to perform through SNMP Sets, like configuring read-only areas of the Systems Management Empire MIB such as community strings, operational parameters, trap communities, AIM loading, and so on
- Enables verifiable delivery and creates fewer security vulnerabilities than SNMP
- Enables complete remote agent configuration and delivery and minimizes or even eliminates the requirement for local manual file manipulation

The CA Virtual Assurance manager provides installation packages for SystemEDGE for each supported platform. The deployment solution lets you specify credentials and installation parameters, and when you have specified all required information, you can save the deployment package and deploy it to managed systems in your enterprise.

From the CA Virtual Assurance user interface, you can set all agent configuration parameters in the sysedge.cf file and create self-monitoring entries using a graphical representation of the monitor tables. You can save an agent configuration as a profile and apply that profile to a deployment package for deployment to managed systems. You can also make point configuration changes to installed agents and apply those changes instantly.

When you deploy a configuration change, CA Virtual Assurance compiles the sysedge.cf file using the template directives in the file to apply the information you entered in the CA Virtual Assurance user interface for configuration and self monitoring. It sends the compiled sysedge.cf file to the specified systems using CAM (CA Messaging) as the transport mechanism. CAM sends the file to the data directory specified during installation. The SystemEDGE agent listens for CAM messages containing configuration files, and when a file is detected, the agent checks the file for correct syntax and then automatically loads and uses the file. Configuring the agent through CA Virtual Assurance triggers an agent warm start, so that you can make changes to the agent that it can apply automatically without requiring a restart.

Event Engine

The CA Virtual Assurance event engine collects the following events to track all SystemEDGE and CA Virtual Assurance-side activity:

- SystemEDGE state change traps
- Events that track and display the status of CA Virtual Assurance-initiated deployments and configuration changes
- Events notifying the CA Virtual Assurance administrator of SNMP changes to the agent configuration

The event engine provides a reliable location for tracking SystemEDGE exceptions and operations and auditing agent configuration changes.

CA Virtual Assurance Security Options

CA Virtual Assurance provides several SystemEDGE security lockdown options that enhance the agent security and establish CA Virtual Assurance as the single point of configuration through which all other changes must gain approval. You can configure the following security options through CA Virtual Assurance:

SNMP Write Removal

Removes the ability to modify the agent through SNMP, which establishes CA Virtual Assurance or local sysedge.cf file manipulation as the only methods for configuring the agent. CA Virtual Assurance requires SNMP write-access.

Managed Mode

Causes any CA Virtual Assurance-initiated changes to overwrite changes made to the agent by other methods. When you enter a CA Virtual Assurance Manager node during SystemEDGE installation, the agent is in managed mode. Managed mode establishes CA Virtual Assurance as the mode of configuration that supersedes all other changes. For example, if a user directly modifies the syedge.cf file, and a file is later deployed to that system through CA Virtual Assurance, the settings in the CA Virtual Assurance-delivered file override those in the local file.

SNMP Change Notification

Notifies the CA Virtual Assurance Manager when an agent applies SNMP-based changes initiated by an SNMP Set operation. This option is only available for agents in managed mode. By default, this option is enabled when the agent is in managed mode. From the CA Virtual Assurance user interface, you can decide the changes that are acceptable and overwrite any unwanted changes. When this option is enabled, SNMP Set changes are also logged to the sysedge_audit.log file located in the data directory of the agent installation.

For more information about enabling and configuring these security options, see the CA Virtual Assurance documentation.

AIM Architecture

Application Insight Modules (AIMs) extend the SystemEDGE agent capabilities to application-specific monitoring or provide specialized monitoring capabilities that do not exist in the base agent. A wide variety of AIMs are available with different architectures, but the following conventions apply to all AIMs:

- Typically, AIMs are automatically loaded after installation. The AIM installation program creates the enable_req file in the directory SystemEDGE/plugins/plugin-name as the last step of the installation program. This helps ensure that the AIM product files are available when the SystemEDGE agent discovers the enable_req file.
  Alternatively, you can enter the sysedge_plugin <plugin name> parameter in the sysedge.cf file if the SystemEDGE agent is in an unmanaged mode. After the AIMs are loaded successfully, the SystemEDGE agent creates the enable_ack file.

  Note: Write-access must be specified during CA Virtual Assurance installation.
- SystemEDGE can load AIMs with a warm start, but the agent requires a full restart to remove previously loaded AIMs.
- Each SystemEDGE Version 5.0 AIM has its own MIB that the core agent supports.
- Each AIM can have its own configuration file in which you can enter AIM configuration instructions.

For more information about using AIMs, see the documentation specific to AIM.

On-Demand Modules

SystemEDGE provides the following on-demand modules with the base agent:

- Monitoring Windows
- Perfcube

These modules do not require a sysedge_plugin statement in the sysedge.cf file. SystemEDGE automatically loads the on-demand modules when the functionality is required.

Note: The functionality previously included in the Top Processes AIM is now integrated with the base agent. For more information, see the chapter "Process and Service Monitoring."

For more information about configuring the modules and the functionality they provide, see *SystemEDGE AIMs* chapter.

AIM Integration with CA Virtual Assurance

CA Virtual Assurance supports remote deployment for all AIMs shipped with CA Virtual Assurance and policy configuration for SystemEDGE and the SRM AIM. The AIMs support the SystemEDGE state management model by creating stateful monitors, so that you can view status information associated with response time tests through the CA Virtual Assurance user interface.

The following AIMs are supported for use with SystemEDGE Release 5.7.1, but they are not supported for visualization and configuration through CA Virtual Assurance:

- IIS AIM
- Apache AIM
- Voice AIM
- CA Insight AIM
- Exchange Server and Active Directory AIM

You can load and configure these AIMs locally, but CA Virtual Assurance has no Platform Management Modules for interpreting the AIM data and provides no interface for configuration.

Note: For a detailed AIM support matrix, see the *SystemEDGE Release Notes*.

Guidelines for Using the SystemEDGE Agent

You can gain the most value from the SystemEDGE agent by setting up effective management policies for your systems, networks, and applications. In particular, follow these guidelines:

- Manage the agent in managed mode from CA Virtual Assurance and restrict SNMP-based changes or monitor these changes so that you have one consistent point of agent configuration.
- Limit SNMP access to the agent through access control lists and binding to private interfaces.
  - Use read-write communities for SNMP sets and read-only communities for querying and polling.
  - Use either port 161 (the SNMP industry standard port) or 1691 (which is reserved with the Internet Assigned Numbers Authority [IANA] for use with the SystemEDGE agent).

    Note: For more information about specifying community strings and configuring ports, see the chapter “Agent Configuration."

- Create groups in your management software based on the following:
  - Operating systems
  - Database systems
  - Clients
  - File servers
  - Email servers
  - Web servers
  - Physical location

  Note: For more information about creating groups, see the *CA Virtual Assurance Administration Guide* or *CA eHealth Administration Guide.*
- Define configuration policies for groups of systems instead of on an individual system basis. Use CA Virtual Assurance to deploy configuration sets to groups of systems.
- Include meaningful information in the system location and contact field of your sysedge.cf file. For example, include information such as Rack 0, Slot 1, Atlanta, and email: it@yourdomain.com.
- Use a well known range of row indexes across your monitoring tables. When defining these indexes and reserving rows, keep in mind the following:
  - Use large ranges of index numbers for each type of monitoring to enable growth.
  - Use standard index entries for specific types of monitoring entries. For example, always use row 10,000 for monitoring the total amount of CPU available.
  - Use CA Virtual Assurance to apply your self-monitoring entries to groups of systems.

  Note: For more information about using standard index rows, see Row Creation for the Self Monitor Table in the chapter “Self Monitoring”.
- Consider the following as you create entries that you can use across multiple systems:
  - Monitor total CPU utilization states (which enable the configuration to be portable across single- and multi-processor systems).
  - Monitor thresholds and configure SystemEDGE to send a limited number of traps (for example, two or three) to prevent flooding of CA Virtual Assurance.

  Note: For more information about effectively monitoring thresholds and clearing traps, see Monitor Table Flags in the chapter "Self Monitoring."

- When you are managing a large network (hundreds or thousands of systems) and polling each system for granular data (at intervals of less than 15 minutes), do the following:
  - Use history collection to gain highly granular data at the agent level, and let the management system poll the History table. You can use SystemEDGE for short-term history collection and CA eHealth (or another management system) for the long-term historical view.
  - For more information, see the chapter "History Collection."
  - Push the monitoring out to the agent, and configure the agent to send traps based on these self-monitoring entries.
  - Limit the number of potential traps from a single monitored entity by using the stateful monitors.

For more usage tips, see the following sections in this guide:

- Recommendations for Configuring Security in the chapter “Agent Configuration”
- Recommendations for Process and Service Monitoring in the chapter “Process and Service Monitoring”
- Recommendations for Log File Monitoring in the chapter “Log File Monitoring”
- Recommendations for Using Extensions in the chapter “Custom MIB Objects”

Chapter 4: Installation and Deployment

This chapter describes how to perform a standalone installation of the SystemEDGE agent on all supported platforms and introduces the concept of agent deployment using CA Virtual Assurance.

This section contains the following topics:

Installation Notes (see page 77)
Agent Configuration During Installation (see page 78)
Installation with CA Virtual Assurance (see page 78)
Installation on Windows Systems (see page 78)
Installation on UNIX and Linux Systems (see page 90)
Configure and Use a Response File (see page 104)
Install the Agent in Legacy Mode (see page 104)
Agent Deployment (see page 106)
Uninstalling SystemEDGE (see page 108)
Reinstallation (see page 112)
Upgrade Managed Nodes and AIM Servers (see page 113)

Installation Notes

Consider the following before installing the SystemEDGE agent:

- Verify that your system meets the system requirements listed in the *SystemEDGE Release Notes*.
- The CA_SystemEDGE_Core package installs the SystemEDGE agent.
- The CA_SystemEDGE_AdvancedEncryption package provides agent encryption libraries. For more information about installing the advanced encryption package, see *SystemEDGE Advanced Encryption* appendix.
- You can change most values that you define during installation in the CASYSEDGE_DATA\port<*default-port*>\sysedge.cf file. This file also provides additional configuration options. For more information about using this file to configure the agent, see *Agent Configuration* chapter.
- You can configure SNMPv1 and SNMPv2c settings during installation, but you must configure SNMPv3 settings after installation. For more information, see the *Agent Configuration* chapter.
- When using special or reserved characters in any installation operations, consider the *Special Character Limitations* described in the readme and limitations based on the respective operating environment. Operating environment limitations include, but are not limited to, special or reserved characters in a UNIX shell or in the Windows Command Prompt.

Agent Configuration During Installation

The SystemEDGE installation lets you perform some configuration tasks when you install the agent. For example, you can configure system description and location, read-only and read-write communities, and SNMPv1 trap destinations during the installation process. You can later modify any of those settings by editing the sysedge.cf file.

Note: SNMPv3 users and trap destinations are not configured during installation. You must enter these settings by manually editing the sysedgeV3.cf configuration file in the data directory. For more information about configuring SNMPv3 for the SystemEDGE agent, see SNMPv3 Configuration (see page 155).

More Information

File-Based Configuration (see page 121)

Installation with CA Virtual Assurance

This chapter describes how to perform a standalone installation of the SystemEDGE agent. When managing the agent with CA Virtual Assurance, the best practice is to deploy agents using the CA Virtual Assurance user interface. For more information about deployment, see Deploy the Agent with CA Virtual Assurance (see page 107) and the *CA Virtual Assurance Implementation Guide*.

A CA Virtual Assurance manager installation installs SystemEDGE on the manager system by default. Deployment on the local system is not supported.

Installation on Windows Systems

This section describes how to manually install SystemEDGE on Windows systems. You can install using an interactive wizard or silently using the command line. Separate packages are available for all supported hardware architectures. The installer detects your hardware architecture and runs the appropriate installation package.

The following situations require you to install the agent manually, instead of using the recommended CA Virtual Assurance deployment method:

- You are installing the agent on a system, which does not support Remote Deployment.
- You want to install the agent in legacy mode.

Throughout this guide, the term *Windows* encompasses supported versions of Windows. For a list of supported versions, see the *SystemEDGE Release Notes*.

More Information

Install the Agent in Legacy Mode (see page 104)

Install the Agent on Windows

You can use an interactive wizard to install the SystemEDGE agent for Windows manually.

Follow these steps:

1. Log in to the Windows system as an Administrator and do one of the following:
   - Double-click setup.hta from DVD1 of the CA Virtual Assurance installation image and click Install SystemEDGE Agent on the CA Virtual Assurance dialog.
   - Open the DVD1\Installers\Windows\Agent\SysMan\CA_SystemEDGE_Core folder and double-click ca-setup.exe.

   Note: For systems running Windows Vista or later, you can install as a non-administrator. The operating system prompts you to authorize the installation with Administrator credentials.

   The installation wizard opens. The installer detects the hardware architecture of a system and runs the appropriate version of the installation package.
2. Click Next.

   The License Agreement page opens.
3. Read the License agreement and scroll down to the bottom. Select I accept the terms of the License Agreement option and click Next.

   The Installation Type page opens.
4. Select Typical or Custom and click Next.

   Note: The following procedure describes a custom installation. If you select Typical, the Review Settings page opens when you click Next.

   The Destination Location dialog appears.

5. Accept the default or Browse to select the locations for the installation and data directories, click Next, and continue with Step 9. If you want to specify different locations, click Advanced and continue with Step 8.

   Destination Location

   Specifies the location at which to install the agent. By default, the installation directory is *Install_Path*\SystemEDGE and the runtime program data is stored in the config subdirectory. To specify more parameters, click Advanced.

   Note: When installing the agent on a system with a previous version of the agent already installed, the installer selects the installation directory of the existing agent.

   If you skip the Advanced dialog, the Configuration Manager Settings dialog appears.

6. Complete the following fields in the Advanced Destination Locations dialog and click Next.

   SystemEDGE binary path

   Specifies the directory for program binaries and documentation.

   SystemEDGE data path

   Specifies the directory for run-time program data.

   CA Shared Components path

   Specifies the directory for CA Shared Components. Once set by any CA software, this directory cannot be changed and the corresponding field in the user interface is disabled.

   The Configuration Manager Settings dialog appears.

7. Complete the following fields and click Next.

   Configuration Manager Host Name

   Specifies the host name of the configuration manager from which you want to manage this agent. Enter a value for this parameter to configure this agent from the system on which CA Virtual Assurance runs. Enter an asterisk (*) to accept the first manager that discovers the agent system.

   Default Configuration Policy Name

   Specifies the name of the configuration policy file (maintained by the CA Virtual Assurance manager) for the agent to use. Enter a value for this parameter to set the SystemEDGE configuration according to an existing configuration file from the manager.

   If the installer detects the native Microsoft SNMP agent running on the system, the following Native SNMP Options dialog appears.

8. If Simple Network Management Protocol (SNMP) is installed on your system, select one of the following options and click Next.

   Default from existing SNMP agent

   Specifies whether to use the default settings from the native SNMP agent. Leave this check box cleared to use different community strings and trap destinations than the native SNMP agent.

   Disable native SNMP agent

   Specifies whether to stop and disable the native SNMP agent. If you leave the native SNMP agent enabled, run SystemEDGE on a different port.

9. Complete the following fields and click Next.

   SNMP port number

   Specifies the port on which to run the SystemEDGE agent.

   Default: 161

   Important! Dedicate this port number to SystemEDGE agent. The installation fails if other application uses this port number. If the native SNMP agent already uses the default port, specify a different port, for example, 1691 or 6665.

   The SNMP System Information dialog appears.

10. Complete the following fields and click Next.

    System Description

    Specifies information about the system (such as a system name) that populates the sysDescr MIB-II object.

    System Location

    Specifies the system location that populates the sysLocation MIB-II object.

    System Contact

    Specifies system contact information that populates the sysContact MIB-II object.

    The SNMP Community Settings dialog appears.

11. Complete the following fields, click Next, and continue with Step 13. If you want to specify multiple community strings, click Advanced and continue with Step 12.

    Read-only Community

    Specifies the SNMP read-only community string.

    Default: public

    Read-write community

    Specifies the SNMP read-write community string.

    If you skip the Advanced dialog, the SNMP Trap Settings dialog appears.

12. Complete the following fields in the SNMP Community Settings - Advanced dialog and click Next.

    Read-only Community

    Specifies the SNMP read-only community string. You can specify multiple communities by separating each with a semicolon (for example, public1; public2). You can also include an IP address list for each community to restrict access (for example, public 1.2.3.4).

    Default: public

    Read-write community

    Specifies the SNMP read-write community string. You can specify multiple communities by separating each with a semicolon (for example, rwcomm1; rwcomm2). You can also include a space delimited IP address list for each community to restrict access (for example, rwcomm1 1.2.3.4). A read-write community is required for correct operation of some AIMs (for example, RM) and for some remote uses (for example, creating monitors).

    The SNMP Trap Settings dialog appears.

13. Complete the following fields, click Next, and continue with Step 15. If you want to specify multiple trap destinations, click Advanced and continue with Step 14.

    Trap community string

    Specifies the SNMP community encoded in sent trap messages.

    Default: public

    Destination host

    Specifies the destination of the trap messages.

    Default: The *configuration manager host name* set in Configuration Manager Settings dialog.

    Port number

    Specifies the port trap messages are sent to.

    Default: 162.

    If you skip the Advanced dialog, the Miscellaneous Settings dialog appears.

14. Complete the following field in the SNMP Trap Settings - Advanced dialog and click Next.

    Trap Configuration

    Specifies one or more trap destinations. You can specify multiple entries by separating each with a semicolon (for example, public server1; public server2 1162).

    The Miscellaneous Settings dialog appears.

15. Complete the following fields and click Next:

    Start After Install

    Specifies whether the agent is started at the end of installation.

    Install Documentation

    Specifies whether to install the documentation.

    The Review Settings page appears.

16. Review the installation settings and click Install.

    The Installation Completed page appears after the installation finishes.

17. Click Finish.

    The installation is complete.

Install the Agent on Windows from the Command Line

You can install the SystemEDGE Windows package using a command line version of the installer. When installing from the command line, you use parameters to set various installation properties. You can do the following from the command line:

- Start the installation wizard, with or without prefilled installation parameters
- Specify installation parameters and run the installer without the interactive wizard

This procedure covers the second scenario: running an unattended installation from the command line.

Follow these steps:

1. Log in to the Windows system as an Administrator.

   Note: For systems running Windows Vista and later, you can install as a non-administrator, and the operating system prompts you to authorize the installation with Administrator credentials.

2. Open a command prompt, navigate to the DVD1\Installers\Windows\Agent\SysMan\CA_SystemEDGE_Core folder, and enter the following required parameters (do **not** run the command until you complete Step 3):

   ```
   ca-setup CA_SETUP_MODE=UNATTENDED EULA_ACCEPTED="YES" [parameter]
   ```

   Note: For help with installation parameters, enter ca-setup -? at the command prompt.

   CA_SETUP_MODE

   Specifies the installation mode. Set this parameter to UNATTENDED to run the installation in silent mode. If you omit this parameter, the installation wizard opens with specified parameter values prefilled after you run the command.

   EULA_ACCEPTED

   Read the license agreement and specify whether to accept the license agreement. If you omit this parameter or set it to anything other than **YES** when you set the installation mode to UNATTENDED, the installation fails. This parameter is not required in an interactive installation.

3. Add optional parameters as appropriate and run the command.

   Note: The following optional parameters do not require a value if you omit them:

   CA_SETUP_LOG_FILE
   CA_SETUP_VERBOSE
   CASE_INSTALLDIR
   CASE_PUBDATADIR
   CASE_SNMP_PORT
   CASE_SNMP_SYS_DESC
   CASE_SNMP_SYS_LOC
   CASE_SNMP_SYS_CONTACT
   CASE_SNMP_READ_COMMUNITY
   CASE_SNMP_READ_ALLOWED_MANAGERS
   CASE_SNMP_WRITE_COMMUNITY
   CASE_SNMP_WRITE_ALLOWED_MANAGERS
   CASE_SNMP_TRAP_COMMUNITY
   CASE_SNMP_TRAP_DESTINATION
   CASE_SNMP_TRAP_PORT
   CASE_DISABLE_NATIVE_SNMP
   CASE_DEFAULT_FROM_NATIVE_SNMP
   CASE_MANAGER_HOSTNAME
   CASE_MANAGER_POLICY_NAME
   CASE_START_AFTER_INSTALL
   CASE_INSTALL_DOCS
   CASE_LEGACY_MODE

   CA_SETUP_LOG_FILE

   Specifies a location and file name in which to log installation messages.

   CA_SETUP_VERBOSE

   Turns on verbose installation mode when set to "yes". Verbose mode logs more information to the installation log file.

CASE_INSTALLDIR

Specifies the SystemEDGE installation directory. This directory contains everything SystemEDGE-related, such as AIMs and Advanced Encryption, and is never modified after being set by the Core installer.

Note: If you define a non-default installation directory, the installer installs agent files directly to the specified directory without creating a SystemEDGE subfolder.

Default: C:\Program Files\CA\SystemEDGE

CASE_PUBDATADIR

Specifies the SystemEDGE data directory, where all configurations take place and dynamic data is stored (referred to in this document as the variable CASYSEDGE_DATA). The configuration files of the agent are located in a port-specific subdirectory which is based on the SNMP_PORT parameter value.

Defaults: C:\Documents and Settings\All Users\Application Data\CA\SystemEDGE (Windows XP and 2003), C:\Users\Public\CA\SystemEDGE (Windows Vista and 2008)

CASE_SNMP_PORT

Specifies the port used by SystemEDGE. This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a port using the CASE_SNMP_PORT parameter, it overrides the inherited value. This port must be unique, or the installation fails. If the default port of 161 is already in use by the native SNMP agent (and you do not plan on disabling this agent), you must specify a different unique port, for example, 1691 or 6665.

Default: 161

CASE_SNMP_SYS_DESC

Specifies information about the system (such as a system name) that populates the sysDescr MIB-II object. This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a description using the CASE_SNMP_SYS_DESC parameter, it overrides the inherited value.

CASE_SNMP_SYS_LOC

Specifies the system location that populates the sysLocation MIB-II object. This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a location using the CASE_SNMP_SYS_LOC parameter, it overrides the inherited value.

CASE_SNMP_SYS_CONTACT

Specifies system contact information that populates the sysContact MIB-II object. This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a contact using the CASE_SNMP_SYS_CONTACT parameter, it overrides the inherited value.

CASE_SNMP_READ_COMMUNITY

Specifies the name of the SNMP read community that can send GET requests to the agent. You can specify multiple communities by separating them with a semicolon (for example, public1;public2), and you can include a space delimited IP address list for each community to restrict access (for example, public 1.2.3.4). This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a read community using the CASE_SNMP_READ_COMMUNITY parameter, it overrides the inherited value.

Default: snmp_public (valid for a new installation only (no upgrade) and if no read-write community is specified)

CASE_SNMP_READ_ALLOWED_MANAGERS

Specifies the space-separated list of IP addresses/hostnames of SNMP managers allowed to query the agent with SNMP_READ_COMMUNITY. When this is specified, SNMP_READ_COMMUNITY must only contain a single word, the SNMP community.

CASE_SNMP_WRITE_COMMUNITY

Specifies the name of the SNMP write community that can send GET and SET requests to the agent. You can specify multiple communities by separating them with a semicolon (for example, rwcomm1;rwcomm2), and you can include a space delimited IP address list for each community to restrict access (for example, rwcomm1 1.2.3.4). This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a write community using the CASE_SNMP_WRITE_COMMUNITY parameter, it overrides the inherited value.

CASE_SNMP_WRITE_ALLOWED_MANAGERS

Specifies the space-separated list of IP addresses/hostnames of SNMP managers allowed to query the agent with SNMP_WRITE_COMMUNITY. When this is specified, SNMP_WRITE_COMMUNITY must only contain a single word, the SNMP community.

CASE_SNMP_TRAP_COMMUNITY

Specifies the SNMP trap community and the trap destination address. You can specify multiple trap community settings by separating them with a semicolon (;). The value for this parameter can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a trap community using the CASE_SNMP_TRAP_COMMUNITY parameter, it overrides the inherited value. The following values are required for this parameter:

- Community name
- Destination address to which to send traps

The following values are optional for this parameter:

- Port number to which to send traps
- Trap source encoding options
- Trap source host name

Syntax of a trap community setting:

community-string {*ip-address*|*hostname*} [*port* [*encoding* [*source*]]]

Example:

public 1.2.3.4;public 2.3.4.5 1162;trapcom 3.4.5.6 1162 100 4.5.6.7

CASE_SNMP_TRAP_DESTINATION

Specifies the hostname or IP address to which to send traps. When this is specified, SNMP_TRAP_COMMUNITY must only contain a single word, the SNMP community and SNMP_TRAP_PORT must also be specified.

CASE_SNMP_TRAP_PORT

Specifies the destination port number to which to send traps. When this is specified, SNMP_TRAP_COMMUNITY must only contain a single word, the SNMP community and SNMP_TRAP_DESTINATION must also be specified.

CASE_DISABLE_NATIVE_SNMP

Specifies whether to stop and disable the native SNMP agent.

Default: no

CASE_DEFAULT_FROM_NATIVE_SNMP

Specifies whether to use the default SNMP settings from the native SNMP agent.

Default: no

CASE_MANAGER_HOSTNAME

Specifies the host name of the configuration manager from which you want to manage this agent. Entering a value for this parameter lets you configure this agent from the specified manager. Entering an asterisk (*) accepts the first manager that discovers the agent system. This manager will have full control over the agent's configuration. By default, no manager host is entered (used), and the agent runs in unmanaged mode.

CASE_MANAGER_POLICY_NAME

Specifies the name of the configuration manager policy file that the agent should use. Entering a value for this parameter sets the SystemEDGE configuration according to an existing configuration file from the manager. By default, the agent uses the installed policy file.

CASE_START_AFTER_INSTALL

Specifies whether to start the agent automatically after the installation completes.

Default: yes

CASE_INSTALL_DOCS

Specifies whether to install the SystemEDGE documentation with the agent.

Default: yes

CASE_LEGACY_MODE

Specifies whether to install the agent in legacy mode, which installs the base agent only while omitting all materials that facilitate usage with CA Virtual Assurance. Install using legacy mode if you do not want to use the agent with CA Virtual Assurance.

You can turn the agent into managed mode by reinstalling or upgrading and specifying CASE_LEGACY_MODE=no.

Default: no

Enter the command and all required parameters and values in the command line. Press Enter to start the installation. The installer detects the hardware architecture of the operating system and runs the appropriate version of the installer.

Note: If you do not accept the installation agreement, the installation fails.

To verify the installation, confirm the existence of the CA SystemEDGE service in the Windows Services dialog, the SystemEDGE files in the installation directory, or the presence of CA SystemEDGE Core in the Add or Remove Programs dialog. You can also check the installation log file if you specified one to confirm a successful installation.

Note: The SystemEDGE installer automatically installs "Microsoft Visual C++ 2005 Redistributable Package" if you are installing on a system running Windows Vista or later. SystemEDGE will not function without this package installed. To install "Microsoft Visual C++ 2005 Redistributable Package" you must accept the license agreement that is displayed. The license agreement for "Microsoft Visual C++ 2005 Redistributable Package" is suppressed in the interactive version of the SystemEDGE installer.

Installation on UNIX and Linux Systems

This section describes how to manually install the SystemEDGE agent on UNIX and Linux systems. You can install using an interactive wizard or silently using the command line. The interactive wizard displays in text mode on the console or as a graphical application if Xserver is available and the DISPLAY environment correctly set.

The following situations require you to install the agent manually, instead of using the recommended CA Virtual Assurance deployment method:

- You are installing the agent on a system, which does not support Remote Deployment.
- You want to install the agent in legacy mode.

For more information about the supported UNIX and Linux platforms and versions, see the *SystemEDGE Release Notes*.

More Information

Install the Agent in Legacy Mode (see page 104)

Install the Agent on UNIX and Linux Systems

You can use an interactive wizard to install the SystemEDGE agent manually for UNIX and Linux.

Notes:

- This document refers to the installation directory as CASYSEDGE and to the data directory as CASYSEDGE_DATA.
- The installation program does not modify the system environment settings in /etc/profile.

Follow these steps:

1. Log in to the system as the root user and mount DVD2.
2. Open a terminal console and change to Installers/*platform*/Agent/SysMan/CA_SystemEDGE_Core directory, selecting the *platform* directory that corresponds to your operating system.
3. Run the installer from this directory as follows:

   ```
   sh ca-setup.sh
   ```

   The Introduction page of the installer appears.
4. Click Next.

   The License Agreement page appears.
5. Read the license agreement and select I accept the terms of the License Agreement. Click Next.

   The Installation Type page appears.
6. Select Typical or Custom and click Next.

   Note: This procedure describes a custom installation. If you select Typical, the Review Settings page appears when you click Next.

   The Destination Location dialog appears.
7. Accept the default or Browse to select the locations for the installation and data directories, click Next, and continue with Step 9. If you want to specify different locations click Advanced and continue with Step 8.

   Destination Location

   Specifies the location at which to install the agent. By default, the installation directory is /opt/CA/SystemEDGE and the runtime program data is stored in the config subdirectory. To specify more parameters, click Advanced.

   Note: If you are installing the agent on a system with a previous version of the agent already installed, the installer automatically selects the installation directory of the existing agent.

   If you skip the Advanced dialog, the Configuration Manager Settings dialog appears.

8. Complete the following fields in the Advanced Destination Locations dialog and click Next.

   SystemEDGE binary path

   Specifies the directory for program binaries and documentation.

   SystemEDGE data path

   Specifies the directory for run-time program data.

   CA Shared Components path

   Specifies the directory for CA Shared Components. Once set by any CA software, this directory cannot be changed and the corresponding field in the user interface is disabled.

   The Configuration Manager Settings dialog appears.

9. Complete the following fields and click Next.

   Configuration Manager Host Name

   Specifies the host name of the configuration manager from which you want to manage this agent. Enter a value for this parameter to configure this agent from the system on which CA Virtual Assurance runs. Enter an asterisk (*) to accept the first manager that discovers the agent system.

   Default Configuration Policy Name

   Specifies the name of the policy file (maintained by the CA Virtual Assurance manager) for the agent to use. Enter a value for this parameter to set the SystemEDGE configuration according to an existing configuration file from the manager.

   If the installer detects the native SNMP agent running on the system, the Native SNMP Options dialog appears.

10. Complete the following fields and click Next.

    Default from existing SNMP agent

    Specifies whether to use the default settings from the native SNMP agent. Leave this check box cleared to use different community strings and trap destinations than the native SNMP agent.

    Disable native SNMP agent

    Specifies whether to stop and disable the native SNMP agent. If you leave the native SNMP agent enabled, run SystemEDGE on a different port.

11. Complete the following fields and click Next.

    SNMP port number

    Specifies the port on which to run the SystemEDGE agent. This port must be not be used by any other application or the installation fails. If the native SNMP agent already uses the default port, specify a different port, for example, 1691 or 6665.

    Default: 161

    The SNMP System Information dialog appears.

12. Complete the following fields and click Next.

    System Description

    Specifies information about the system (such as a system name) that populates the sysDescr MIB-II object.

    System Location

    Specifies the system location that populates the sysLocation MIB-II object.

    System Contact

    Specifies system contact information that populates the sysContact MIB-II object.

    The SNMP Community Settings dialog appears.

13. Complete the following fields, click Next, and continue with Step 15. If you want to specify multiple community strings, click Advanced and continue with Step 14.

    Read-only Community

    Specifies the SNMP read-only community string.

    Default: public

    Read-write community

    Specifies the SNMP read-write community string.

    If you skip the Advanced dialog, the SNMP Trap Settings dialog appears.

14. Complete the following fields in the SNMP Community Settings - Advanced dialog and click Next.

    Read-only Community

    Specifies the SNMP read-only community string. You can specify multiple communities by separating each with a semicolon (for example, public1;public2). You can also include an IP address list for each community to restrict access (for example, public 1.2.3.4).

    Default: public

    Read-write community

    Specifies the SNMP read-write community string. You can specify multiple communities by separating each with a semicolon (for example, rwcomm1;rwcomm2). You can also include a space delimited IP address list for each community to restrict access (for example, rwcomm1 1.2.3.4). A read-write community is required for correct operation of some AIMs (for example, RM) and for some remote uses (for example, creating monitors).

    The SNMP Trap Settings dialog appears.

15. Complete the following fields, click Next, and continue with Step 17. If you want to specify multiple trap destinations, click Advanced and continue with Step 16.

    Trap community string

    Specifies the SNMP community encoded in sent trap messages.

    Default: public

    Destination host

    Specifies the destination of the trap messages.

    Default: The *configuration manager host name* set in Configuration Manager Settings dialog.

    Port number

    Specifies the port trap messages are sent to.

    Default: 162.

    If you skip the Advanced dialog, the Privilege Separation User dialog appears.

16. Complete the following field in the SNMP Trap Settings - Advanced dialog and click Next.

    Trap Configuration

    Specifies one or more trap destinations. You can specify multiple entries by separating each with a semicolon (for example, public server1;public server2 1162).

    The Privilege Separation User dialog appears.

17. Complete the following field and click Next:

    User Name

    Specifies the user name under which credentials the agent run during SNMP communication.

    This entry instructs the agent (UNIX only) to run SNMP communication under another user account. The agent also uses this user's default group as an effective group.

    Default: The agent operates using root account.

    The Miscellaneous Settings dialog appears.

18. Complete the following fields and click Next:

    Start After Install

    Specifies whether the agent is started at the end of installation.

    Install Documentation

    Specifies whether to install the documentation.

    The Review Settings page appears.

19. Review the installation settings and click Install.

    The Installation Completed page appears after the installation finishes.

20. Click Finish.

    The installation is complete.

SystemEDGE Installation on 64-bit Linux Releases Fails

Symptom:

When I install SystemEDGE on a 64-bit Linux release, the installation fails.

Solution:

To run and install SystemEDGE on a 64-bit Linux release, install the required 32-bit libraries:

- Valid on Redhat or SuSE distributions:

  ```
  yum install glibc.i686
  ```

- Valid on Debian distribution:

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

Install the Agent on UNIX from the Command Line

You can install the SystemEDGE UNIX package using a command line version of the installer. When installing from the command line, you use parameters to set various installation properties. You can do the following from the command line:

- Start the installation wizard, with or without prefilled installation parameters
- Specify installation parameters and run the installer without the interactive wizard

Notes:

- UNIX does not support the automatic creation of a response file to run a silent installation. However, you can create a response file manually and use it for an unattended installation.
- The installation program does not modify the system environment settings in /etc/profile.

This procedure covers the second scenario: running an unattended installation from the command line.

Follow these steps:

1. Log in to the system as root.
2. Navigate to the DVD2/Installers/*platform*/Agent/SysMan/CA_SystemEDGE_Core folder (specifying the platform folder that corresponds to your operating system), and enter the following required parameters (do **not** run the command until you complete Step 3):

   ```
   sh ca-setup.sh CA_SETUP_MODE="UNATTENDED" EULA_ACCEPTED="yes" [parameter]
   ```

   Note: For help with installation parameters, enter ca-setup -? at the command prompt.

   CA_SETUP_MODE

   Specifies the installation mode. Set this parameter to UNATTENDED to run the installation without displaying the installation wizard. If you omit this parameter, the installation wizard opens with specified parameter values prefilled after you run the command.

   EULA_ACCEPTED

   Read the license agreement and specify whether to accept the license agreement. If you omit this parameter or set it to anything other than **YES** when you set the installation mode to UNATTENDED, the installation fails. This parameter is not required in an interactive installation.

3. Add optional parameters as appropriate, and run the command.

   Note: The following optional parameters do not require a value if you omit them:

   CA_SETUP_LOG_FILE
   CA_SETUP_VERBOSE
   CASE_INSTALLDIR
   CASE_PUBDATADIR
   CASE_SNMP_PORT
   CASE_SNMP_SYS_DESC
   CASE_SNMP_SYS_LOC
   CASE_SNMP_SYS_CONTACT
   CASE_SNMP_READ_COMMUNITY
   CASE_SNMP_READ_ALLOWED_MANAGERS
   CASE_SNMP_WRITE_COMMUNITY
   CASE_SNMP_WRITE_ALLOWED_MANAGERS
   CASE_SNMP_TRAP_COMMUNITY
   CASE_SNMP_TRAP_DESTINATION
   CASE_SNMP_TRAP_PORT
   CASE_DISABLE_NATIVE_SNMP
   CASE_DEFAULT_FROM_NATIVE_SNMP
   CASE_MANAGER_HOSTNAME
   CASE_MANAGER_POLICY_NAME
   CASE_PRIVSEP_USER
   CASE_START_AFTER_INSTALL
   CASE_INSTALL_DOCS
   CASE_LEGACY_MODE

   CA_SETUP_LOG_FILE

   Specifies a location and file name in which to log installation messages. By default, messages are logged to /opt/CA/installer/log/CA_SETUP_*PACKAGE_NAME*.log.

   CA_SETUP_VERBOSE

   Turns on verbose installation mode when set to "yes". Verbose mode logs more information to the installation log file.

   CASE_INSTALLDIR

   Specifies the SystemEDGE installation directory. This directory contains everything SystemEDGE-related, such as AIMs and Advanced Encryption, and is never modified after being set by the Core installer.

   Note: If you define a non-default installation directory, the installer installs agent files directly to the specified directory without creating a SystemEDGE subfolder.

   Default: /opt/CA/SystemEDGE

CASE_PUBDATADIR

Specifies the SystemEDGE data directory, where all configurations take place and dynamic data is stored (referred to in this document as the variable CASYSEDGE_DATA). The configuration files of the agent are located in a port-specific subdirectory which is based on the SNMP_PORT parameter value.

Default: /opt/CA/SystemEDGE/config

CASE_SNMP_PORT

Specifies the port used by SystemEDGE. This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a port using the CASE_SNMP_PORT parameter, it overrides the inherited value. This port must be unique, or the installation fails. If the default port of 161 is already in use by the native SNMP agent (and you do not plan on disabling this agent), you must specify a different unique port, for example, 1691 or 6665.

Default: 161

CASE_SNMP_SYS_DESC

Specifies information about the system (such as a system name) that populates the sysDescr MIB-II object. This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a description using the CASE_SNMP_SYS_DESC parameter, it overrides the inherited value.

CASE_SNMP_SYS_LOC

Specifies the system location that populates the sysLocation MIB-II object. This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a location using the CASE_SNMP_SYS_LOC parameter, it overrides the inherited value.

CASE_SNMP_SYS_CONTACT

Specifies system contact information that populates the sysContact MIB-II object. This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a contact using the CASE_SNMP_SYS_CONTACT parameter, it overrides the inherited value.

CASE_SNMP_READ_COMMUNITY

Specifies the name of the SNMP read community that can send GET requests to the agent. You can specify multiple communities by separating them with a semicolon (for example, public1;public2), and you can include a space delimited IP address list for each community to restrict access (for example, public 1.2.3.4). This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a read community using the CASE_SNMP_READ_COMMUNITY parameter, it overrides the inherited value.

Default: snmp_public (valid for a new installation only (no upgrade) and if no read-write community is specified)

CASE_SNMP_READ_ALLOWED_MANAGERS

Specifies the space-separated list of IP addresses/hostnames of SNMP managers allowed to query the Agent with CASE_SNMP_READ_COMMUNITY. When this is specified, CASE_SNMP_READ_COMMUNITY must only contain a single word, the SNMP community.

CASE_SNMP_WRITE_COMMUNITY

Specifies the name of the SNMP write community that can send GET and SET requests to the agent. You can specify multiple communities by separating them with a semicolon (for example, rwcomm1;rwcomm2), and you can include a space delimited IP address list for each community to restrict access (for example, rwcomm1 1.2.3.4). This value can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a write community using the CASE_SNMP_WRITE_COMMUNITY parameter, it overrides the inherited value.

CASE_SNMP_WRITE_ALLOWED_MANAGERS

Specifies the space-separated list of IP addresses/hostnames of SNMP managers allowed to query the agent with CASE_SNMP_WRITE_COMMUNITY. When this is specified, CASE_SNMP_WRITE_COMMUNITY must only contain a single word, the SNMP community.

CASE_SNMP_TRAP_COMMUNITY

Specifies the SNMP trap community and the trap destination address. You can specify multiple trap community settings by separating them with a semicolon (;). The value for this parameter can also be inherited from the native SNMP agent using the CASE_DEFAULT_FROM_NATIVE_SNMP parameter. If you specify a trap community using the CASE_SNMP_TRAP_COMMUNITY parameter, it overrides the inherited value. The following values are required for this parameter:

- Community name
- Destination address to which to send traps

The following values are optional for this parameter:

- Port number to which to send traps
- Trap source encoding options
- Trap source host name

Syntax of a trap community setting:

community-string {*ip-address*|*hostname*} [*port* [*encoding* [*source*]]]

Example:

public 1.2.3.4;public 2.3.4.5 1162;trapcom 3.4.5.6 1162 100 4.5.6.7

CASE_SNMP_TRAP_DESTINATION

Specifies the hostname or IP address to which to send traps. When this is specified, CASE_SNMP_TRAP_COMMUNITY must only contain a single word, the SNMP community and CASE_SNMP_TRAP_PORT must also be specified.

CASE_SNMP_TRAP_PORT

Specifies the destination port number to which to send traps. When this is specified, CASE_SNMP_TRAP_COMMUNITY must only contain a single word, the SNMP community and CASE_SNMP_TRAP_DESTINATION must also be specified.

CASE_DISABLE_NATIVE_SNMP

Specifies whether to stop and disable the native SNMP agent.

Default: no

CASE_DEFAULT_FROM_NATIVE_SNMP

Specifies whether to use the default SNMP settings from the native SNMP agent.

Default: no

CASE_MANAGER_HOSTNAME

Specifies the host name of the configuration manager from which you want to manage this agent. Entering a value for this parameter lets you configure this agent from the specified manager. Entering an asterisk (*) accepts the first manager that discovers the agent system. This manager will have full control over the agent's configuration. By default, no manager host is entered (used), and the agent runs in unmanaged mode.

CASE_MANAGER_POLICY_NAME

Specifies the name of the configuration manager policy file that the agent should use. Entering a value for this parameter sets the SystemEDGE configuration according to an existing configuration file from the manager. By default, the agent uses the installed policy file.

CASE_START_AFTER_INSTALL

Specifies whether to start the agent automatically after the installation completes. Valid values are: Yes, No, PRESERVE. PRESERVE can be used when upgrading to only start the agent when it was running at the time when the installation started.

Default: PRESERVE

CASE_PRIVSEP_USER

Specifies the user name under which credentials the agent run during SNMP communication.

This entry instructs the agent (UNIX only) to run SNMP communication under another user account. The agent also uses this user's default group as an effective group.

Default: The agent operates using root account.

CASE_INSTALL_DOCS

Specifies whether to install the SystemEDGE documentation with the agent.

Default: yes

CASE_LEGACY_MODE

Specifies whether to install the agent in legacy mode, which installs the base agent only while omitting all materials that facilitate usage with CA Virtual Assurance. Install using legacy mode if you do not want to use the agent with CA Virtual Assurance.

You can turn the agent into managed mode by reinstalling or upgrading and specifying CASE_LEGACY_MODE=no.

Default: no

The installation begins. If you did not accept the license agreement, the installation fails.

There is always the lsm installer log file in /opt/CA/installer/log/$CA_SETUP_PACKAGE_NAME.log that you can use to check if the installation was successful.

If you want to verify the installation, confirm the existence of the SystemEDGE files in the installation directory.

Legacy Support of the $CASYSEDGE Variable

The installation program on Linux/UNIX does not modify the system environment settings in /etc/profile. As a result the $CASYSEDGE variable is no longer available.

If you have to support $CASYSEDGE in your environment, do one of the following options:

- Create $CASYSEDGE in a shell.

  Run the following command in a sh, ksh, or bash shell:

  ```
  ./etc/profile.CA
  ```

  If you use csh, run the following command:

  ```
  source /etc/csh_login.CA
  ```

- Force the installation program to modify /etc/profile and create $CASYSEDGE.

  Open a shell and enter the following commands:

  ```
  Update_Profile=1;export Update_Profile
  sh ca-setup.sh
  ```

  If you use csh, run the following commands:

  ```
  setenv Update_Profile 1
  sh ca-setup.sh
  ```

  The installation program installs SystemEDGE and creates $CASYSEDGE in the environment.

During an upgrade to Release 5.7.1, the SystemEDGE installation program does not change the existing environment. The previously created $CASYSEDGE variable remains in the environment.

Configure and Use a Response File

You can create a response file for running a silent installation with no user interaction. Using a response file has the same effect as specifying CA_SETUP_MODE=UNATTENDED in the command line. A response file turns the installation into the silent, unattended mode.

The installer uses the properties in the response file to install the agent without prompting for user input.

Follow these steps:

1. Log in to the computer system as administrator or root.
2. Create a response file based on the parameters specified in the Install the Agent from Command Line sections. A response file is a text file that consists of parameter settings like the following:

   ```
   parameter1=value1
   parameter2=value2
   ...
   ```

3. Navigate to the *DVDdrive*\Installers*OperatingSystem*\Agent\SysMan\CA_SystemEDGE_Core directory, and enter one of the following commands according to your operating system:

   ```
   ca-setup CA_SETUP_RESPONSE_FILE="<name of the response file>" (Windows)
   ```

   ```
   sh ca-setup.sh CA_SETUP_RESPONSE_FILE="<name of the response file>" (UNIX, Linux)
   ```

   CA_SETUP_RESPONSE_FILE

   Specifies the path and name of the response file.

   The installation uses the settings in the response file to run silently.

Install the Agent in Legacy Mode

You can install SystemEDGE in legacy mode to acquire all base agent functionality while omitting the following components that facilitate use with CA Virtual Assurance:

- CAM, which enables remote configuration from CA Virtual Assurance.
- IDPrimer, which enables remote deployment from CA Virtual Assurance.

Install SystemEDGE in legacy mode *only* if you do not plan to manage it with CA Virtual Assurance or a similar management application. If you install an agent in legacy mode and want to manage it with CA Virtual Assurance later, you can upgrade the agent.

Note: When you upgrade a previous version of SystemEDGE, it automatically upgrades to the full agent unless you specify otherwise.

Follow these steps:

1. Copy the CA_SystemEDGE_Core directory from the installation media to the harddisk.
2. Change to the CA_SystemEDGE_Core directory and open ca-setup.dat with an ASCII editor.
3. Edit ca-setup.dat to set CASE_LEGACY_MODE=yes.
4. Save ca-setup.dat.
5. Run the installation as described in Install the Agent on Windows (see page 79) or Install the Agent on UNIX (see page 90).
6. Complete the installation.

To install the agent in legacy mode from the command line, include the following parameter in the ca-setup command:

```
CASE_LEGACY_MODE="yes"
```

Agent Deployment

CA Virtual Assurance provides a comprehensive solution for remotely deploying the SystemEDGE agent to all managed systems. You can create deployment templates that are based on the provided packages that contain customized installation parameters and simultaneously deploy these templates to numerous managed systems. This automated deployment solution provides one location from which to deploy and configure the agents throughout your enterprise.

CA Virtual Assurance provides the following base deployment packages:

- SystemEDGE Agent Core
- SystemEDGE Advanced Encryption
- CA Citrix XenServer AIM
- SystemEDGE Remote Monitoring AIM
- SystemEDGE Service Response Monitor AIM
- CA Systems Performance LiteAgent
- CA IBM LPAR AIM
- CA IBM High Availability Cluster Multiprocessing AIM
- CA KVM AIM (not applicable to CA Server Automation)
- CA Cisco UCS AIM
- CA Microsoft Hyper-V AIM
- CA Microsoft Cluster Service Support AIM
- CA Solaris Zones AIM
- CA VMware vCenter Server AIM
- CA VMware vCloud AIM
- CA Exchange Server and Active Directory AIM

CA Virtual Assurance supports the following SystemEDGE deployment scenarios:

- You can deploy a SystemEDGE Release 5.7.1 agent to systems with no pre-existing SystemEDGE agent.
- You can deploy a SystemEDGE Release 5.7.1 agent to systems with a pre-existing SystemEDGE 4.3 agent. The deployment automatically upgrades the existing agent to Release 5.7.1.
- You can make configuration changes to individual systems or through a configuration template. The latter option allows you to apply changes to an existing managed SystemEDGE agent or a group of managed SystemEDGE agents.

Note: For more information about how deployment works, see the *CA Virtual Assurance Online Help* and the *Administration Guide*.

CA Virtual Assurance does *not* support the following SystemEDGE deployment scenarios:

- You cannot deploy a version of the agent earlier than SystemEDGE 5.0.0.
- You cannot deploy the SystemEDGE agent on the CA Virtual Assurance manager system. The agent is automatically installed through the CA Virtual Assurance manager installation.

Note: For more information about deployment support, see the C*A Virtual Assurance Administration Guide*.

Deploy the Agent with CA Virtual Assurance

The Implementation Guide and Online Help describe the deployment of agent packages from the user interface in detail.

To deploy agents to systems, create a deployment job. Deployment jobs contain the details that are required for CA Virtual Assurance to deliver the deployment packages to the appropriate systems at the appropriate time.

Follow these steps:

1. Select Resources, Deploy.

   The Deployment pane displays the Packages, Templates, and Jobs.

2. Right-click the Jobs folder in the Manage Resource pane and select Create New Job. You can also select the Jobs folder and Click + (New) on the Job Status toolbar.

   The Jobs Setup page appears.

3. Enter a name in the Job Name pane and optionally base the job on an existing template, and click Next.

   The Package Selection page appears.

4. Select a platform and the packages you want to deploy.

5. (Optional) Click the Details tab.

   The Package Wrapper Details dialog appears and lets you edit the package properties in-line. If the package wrappers are in an incomplete or invalid state, and the fields can be modified through in-line editing.

   a. Click Edit and modify the package wrapper properties.

   b. Click Save, and then click OK.

      The package wrapper properties are updated.

6. Click the down arrow to add the package wrappers to the job, and click Next.

   The Machine Selection page appears.

7. Select the systems to deploy to and click Next. If you have many servers in your environment, multiple pages with some entries can be required to list all servers. When you select servers on a page and scroll to the next page, any selections that are made on previous pages remain valid.

   The Machines Selected page appears.

8. Click Set Credentials, set the system credentials that are required to establish a connection and click Next.

   Note: Deployment to Windows target systems using domain credentials must be in the form of DOMAIN\username.

   The Advanced page appears.

9. (Optional) Set the distribution server to manage the deployment. If not set, it is automatically chosen.

10. Select the scheduling options for the job:

    Immediate Delivery

    Starts the job immediately after creating new deployment job. The immediate delivery is the default option.

    Staggered Delivery

    Delivers the packages over a specific time period.

    Scheduled Delivery

    Schedules the deployment for a specific time in the future.

11. (Optional) If a package has previously been successfully deployed to a system using this deployment infrastructure, you can force it to run again.

12. Click Next.

    The Summary page appears.

13. Review the details of the job and click Deploy.

    The deployment job is created.

Note: You can save the job as a template after you create it. A template saves the package and machine selections so that you can easily reuse them for subsequent jobs. See the Online Help and the Implementation Guide for further details.

Uninstalling SystemEDGE

This section explains how to remove the files and subdirectories associated with the SystemEDGE agent.

Uninstall SystemEDGE and the AIMs on Windows

The uninstaller removes SystemEDGE from your system. You can specify whether to remove the configuration data from the data directory. You can uninstall the agent and the AIMs from the Windows Add or Remove Programs window or from the command line.

Consider the following dependencies when you uninstall:

- AIMs for virtual environments, RM AIM and the SRM AIM depend on Advanced Encryption and SystemEDGE.
- Advanced Encryption depends on SystemEDGE.

Based on these dependencies, the uninstallation sequence is as follows:

1. RM AIM, SRM AIM, or AIMs for virtual environments
2. Advanced Encryption
3. SystemEDGE Core

The uninstallation is not possible if you use any other sequence. In case of removing components that were originally installed through Remote Deployment, Idprimer and CAM are not uninstalled.

To uninstall SystemEDGE or an AIM from the Windows Add or Remove Programs Window

1. Select Start, Settings, Control Panel, Add or Remove Programs.

   The Add or Remove Programs window appears and lists the following components:

   - AIMs for virtual environments
   - CA SystemEDGE Core
   - CA SystemEDGE AdvancedEncryption
   - CA SystemEDGE RM
   - CA SystemEDGE SRM

2. According to the uninstallation sequence, right-click the appropriate component and select Uninstall.

   In case of SystemEDGE, a dialog prompts you to specify whether to preserve the configuration files.

3. Click Yes or No.

   A dialog charts the uninstallation process. When the uninstallation completes, the dialog closes.

To uninstall SystemEDGE or an AIM from the command line

1. Open a command prompt and change to the DVD1\Installers\Windows\Agent\SysMan directory. It contains the following subdirectories:
   - CA_SystemEDGE_SRM
   - CA_SystemEDGE_RM
   - CA_SystemEDGE_AdvancedEncryption
   - CA_SystemEDGE_Core
2. Run the following command:

   ```
   ca-setup -x
   ```

   In case of SystemEDGE, a dialog prompts you to specify whether to preserve the configuration files.
3. Click Yes or No.

   A dialog charts the uninstallation process. When the uninstallation completes, the dialog closes.

To uninstall SystemEDGE or an AIM Silently

- Perform the following steps to uninstall SystemEDGE:
  a. Open a Command Prompt window and change to the following directory path:

     DVD1\Installers\Windows\Agent\SysMan\CA_SystemEDGE_Core

  b. Run the following command:

     ```
     ca-setup.exe CA_SETUP_MODE=UNINSTALL CASE_KEEP_DATA=[YES|NO]
     ```

     The SystemEDGE is uninstalled from the computer.

  Note: To verify, go to the control panel and see that the SystemEDGE is deleted from the control panel items.
- Perform the following steps to delete an AIM:
  a. Open a Command Prompt window and change to the appropriate AIM directory path:

     DVD1\Installers\Windows\Agent\SysMan*AIM*

  b. Run the following command:

     ```
     ca-setup.exe CA_SETUP_MODE=UNINSTALL
     ```

     AIM is uninstalled from the computer.

  Note: Navigate to Program Files, CA, SystemEDGE, plugins and verify that the AIM folder is deleted.

Uninstall SystemEDGE and the AIMs on UNIX Systems

The uninstaller removes SystemEDGE or the AIMs from your system. You can specify for SystemEDGE whether to remove the configuration data from the data directory.

Consider the following dependencies when you uninstall:

- The SRM AIM depends on Advanced Encryption and SystemEDGE.
- Advanced Encryption depends on SystemEDGE.

Based on these dependencies, the uninstallation sequence is as follows:

1. SRM AIM
2. Advanced Encryption
3. SystemEDGE Core

The uninstallation is not possible if you use any other sequence. It case of removing components that were originally installed through Remote Deployment, Idprimer and CAM are not uninstalled.

To uninstall the agent or an AIM using ca-setup.sh

1. Open a terminal console and log in as root (su).
2. Change to the DVD2/Installers/<*Platform*>/Agent/SysMan directory. It contains the following subdirectories:
   - CA_SystemEDGE_SRM
   - CA_SystemEDGE_AdvancedEncryption
   - CA_SystemEDGE_Core
3. Change to the appropriate directory and run the following command:

   ```
   sh ca-setup.sh -x
   ```

   A dialog prompts you to specify whether to preserve the configuration files.
4. Click Yes or No.

   A dialog charts the uninstallation process. When the uninstallation completes, the dialog closes.

To uninstall the agent or an AIM using lsm

1. Open a terminal console and log in as root (su).
2. Run the appropriate command from the following list:

   ```
   lsm -e CA_SystemEDGE_SRM
   lsm -e CA_SystemEDGE_AdvancedEncryption
   lsm -e CA_SystemEDGE_Core
   ```

   In case of SystemEDGE, you are prompted to specify whether to preserve the configuration files.
3. Click Yes or No.

   The uninstallation completes.

To uninstall the agent or an AIM Silently

- Perform the following steps to uninstall agent:
  a. Open a terminal console and log in as root (su). Change to the following directory path:

     DVD2/Installers/*Platform*/Agent/SysMan/CA_SystemEDGE_Core

  b. Run the following command:

     ```
     sh ca-setup.sh CA_SETUP_MODE=UNATTENDED_UNINSTALL CASE_KEEP_DATA=[YES|NO]
     ```

     The agent is uninstalled on the computer.
- Perform the following steps to uninstall an AIM:
  a. Open a terminal console and log in as root (su). Change to the appropriate AIM directory path:

     DVD2/Installers/*Platform*/Agent/SysMan/*AIM*

  b. Run the following command:

     ```
     sh ca-setup.sh CA_SETUP_MODE=UNATTENDED_UNINSTALL
     ```

     AIM is uninstalled from the computer.

Reinstallation

When you reinstall SystemEDGE, the installer only uses the following parameters:

CASE_START_AFTER_INSTALL

CASE_SNMP_PORT

CASE_INSTALL_DOCS

Other parameters are ignored.

Upgrade Managed Nodes and AIM Servers

The recommended method is Remote Deployment to upgrade managed nodes and remote AIM servers. Remote Deployment allows you to upgrade SystemEDGE, Advanced Encryption, and AIMs.

If your system does not support Managed Mode for SystemEDGE, you cannot use Remote Deployment. We recommend using *one* of the following options:

- Manual installation from the installation media
- Remote installation of SystemEDGE, Advanced Encryption, and SRM AIM using, for example, SSH

Follow these steps:

Agent and AIM Upgrades (see page 113)
Import SystemEDGE Monitors into a Policy (see page 116)

Agent and AIM Upgrades

You can upgrade to SystemEDGE Release 5.8 from any upgrade-eligible release of SystemEDGE:

- 4.3.4 and above (4.3.x)
- 5.1.0 and above (5.1.x)
- 5.6.0 and above (5.6.x)
- 5.7.0 and above (5.7.x)

The installer creates an upgraded directory at the root of the original installation path and copies configuration files from the previous release to it. When the agent starts for the first time, it migrates the configuration data to the newly created data directory. When upgrading SystemEDGE to Release 5.8, upgrade Advanced Encryption and all the corresponding AIMs to the latest versions.

Note: SystemEDGE Release 5.8 does not load AIMs of previous CA Virtual Assurance releases.

You can upgrade SystemEDGE in either of the following ways:

- Remote Deployment through CA Virtual Assurance to a system with an upgrade-eligible agent installed.

  Note: If you want to upgrade SystemEDGE, Advanced Encryption, and AIMs, use a single Remote Deployment job for the upgrade. Add SystemEDGE, Advanced Encryption, and all required AIMs to the Remote Deployment job. SystemEDGE loads only AIMs which are at Release 5.8 level or which are legacy AIMs like iddmod. That is, AIMs at 5.7.x level or below are quarantined.

  For more information about Remote Deployment, see the Remote Deployment chapter in the *Administration Guide*.

- Custom Manager installation on Windows. This option upgrades all the installed components. Select any additional agents and AIMs you want to install.

- Manual installation on a system with an upgrade-eligible agent installed.

  To perform an upgrade, install the agent as described in Install the Agent on Windows (see page 79) or Install the Agent on UNIX (see page 90).

If an upgrade of an older agent release is not supported, do the following steps:

- For earlier releases of SystemEDGE, uninstall the agent and then install SystemEDGE Release 5.8. You can save all configuration data before uninstalling and then apply this data to SystemEDGE Release 5.8.

More Information:

Agent Upgrade from SystemEDGE 4.3.4 (see page 114)

Agent Upgrade from SystemEDGE 4.3.4

The installer does the following when upgrading configuration files from a previous 4.3.4 release.

- The installer detects and copies previous versions of the following files to the CASYSEDGE\upgraded directory:
  - system32\sysedge.cf (Windows) and /etc/syedge.cf (UNIX)
  - CASYSEDGE\config\sysedgeV3.cf (Windows and UNIX)
  - system32\sysedge.mon (Windows) and /etc/sysedge.mon (UNIX)
  - system32\sysedge.lic (Windows) and /etc/sysedge.lic (UNIX)
  - CASYSEDGE\plugins\monwin\monwin.cf (Windows and UNIX)

- When the agent starts for the first time, it copies the following files directly into the CASYSEDGE_DATADIR directory:
  - CASYSEDGE\config\sysedge.cf (Windows and UNIX)
  - CASYSEDGE\config\sysedgeV3.cf (Windows and UNIX)
- As part of the sysedge.cf copy operation, the agent migrates the configuration data in the sysedge.cf, sysedge.mon, and monwin.cf files in the upgraded directory into the new sysedge.cf file in the data directory.
- As part of the sysedgeV3.cf copy operation, the agent migrates the configuration data in the sysedgeV3.cf file in the upgraded directory into the new sysedgeV3.cf file in the data directory.

Note: The data directory stores the version of the sysedge.cf file used for runtime configuration changes.

The agent automatically makes the following changes to configuration file settings when upgrading from a previous release:

- The procAlive process monitor syntax has changed to adhere to the threshold process monitor syntax. The agent automatically converts any existing procAlive entries into the new format.

  Note: For more information about the syntax for procAlive entries, see *Process and Service Monitoring* chapter.
- Disabling the no_process_sets and no_remoteshell_group parameters would allow critical access to the agent system with only an SNMP write community. Consequently, the agent does not migrate previous settings for those parameters and always enables them.
- The sysedge_memory parameter no longer exists and is removed in the upgraded sysedge.cf file. Outstanding alarms are now handled by storing the current state of any monitor in the sysedge.mon file. Existing entries are migrated to the new alarm handling configuration.
- The tc_publish parameter is renamed no_trapcommunity_table (with reverse logic) and is enabled by default.
- When you upgrade from SystemEDGE Release 4.3.4 to SystemEDGE Release 5.8 using Remote Deployment, the migrated monitors are stored in the sysedge.cf.bak file.
- Depending on your deployment strategy, you can use Policy Configuration to apply migrated monitors in your environment.

  Note: For more information, see the CA Virtual Assurance Bookshelf.
- When you upgrade from SystemEDGE Release 4.3.4 to SystemEDGE Release 5.8 on the local system, the installation program migrates monitors and provides them in the new configuration.

All previous syntax is compatible with the new version of the agent. However, many new options are available that enhance agent capabilities. We recommend that you examine older monitor entries and edit the syntax to contain additional options.

Note: When you upgrade SystemEDGE from Release 4.3.4 or above (4.3.x), the installer uses the following parameters only:

CASE_PUBDATADIR
CASE_MANAGER_HOSTNAME
CASE_MANAGER_POLICY_NAME
CASE_START_AFTER_INSTALL
CASE_LEGACY_MODE
CASE_SNMP_PORT
CASE_INSTALL_DOCS
CASE_SNMP_TRAP_COMMUNITY (1)
CASE_SNMP_TRAP_DESTINATION (1)
CASE_SNMP_TRAP_PORT (1)
CASE_SNMP_READ_COMMUNITY (1)
CASE_SNMP_WRITE_COMMUNITY (1)
CASE_SNMP_READ_ALLOWED_MANAGERS (1)
CASE_SNMP_WRITE_ALLOWED_MANAGERS (1)

Other parameters are ignored.

(1) These parameters are special. Their settings are appended to the existing SystemEDGE 4.x settings allowing both the SystemEDGE 4.x manager and SystemEDGE 5.x manager to function.

Import SystemEDGE Monitors into a Policy

Before you upgrade the agent, verify the SNMP community strings used in the SystemEDGE agent are specified under the Administration tab, SNMP. Community strings can also be specified on a computer by computer basis under Policy, *computer,* Metrics, SNMP. The SystemEDGE installer moves SystemEDGE files to the 'upgraded' directory, which the agent reads after the upgrade. The OIDs are preserved. Policy Configuration only reads the entries from the existing sysedge.cf file. The sysedge.mon file is not imported.

Remote Deployment imports the raw OIDs of the monitors into the instance.

Follow these steps:

1. Verify that the SystemEDGE agent community strings are included under the Administration tab, SNMP for the port of the agent.
2. Upgrade SystemEDGE to Release 5.8.
3. Verify that SystemEDGE Release 5.8 is discovered correctly (Resources tab) and has a policy.

4. Click the Resources tab, open the Configuration pane, expand Policies, and click SystemEDGE.

   The SystemEDGE pane appears.

5. Click + (New) on the Available Policies toolbar.

   The New SystemEDGE Policy dialog appears.

6. Enter a name for the policy and click 'Import...'.

   The SystemEDGE Agent Machine dialog appears.

7. Select the server that was upgraded and click OK.

   The SystemEDGE Agent Machine dialog closes.

8. Click OK on the New SystemEDGE dialog.

   CA Virtual Assurance creates a policy that contains the imported monitors.

You can edit or update the monitors and use the SystemEDGE Release 5.8 state model.

Chapter 5: Agent Configuration

This chapter describes how to set configuration parameters that determine how the SystemEDGE agent operates. These configuration parameters are defined in the configuration files sysedge.cf and sysedgeV3.cf.

This section contains the following topics:

Configuration Basics (see page 119)
Interactions Between sysedge.cf and sysedge.mon (see page 121)
Configuration Using CA Virtual Assurance (see page 121)
File-Based Configuration (see page 121)
Regular Expression Examples for Monitors and Autowatchers (see page 147)
Perl Compatible Regular Expression (PCRE) Support (see page 150)
SNMPv3 Configuration (see page 155)
SystemEDGE Control Panel for Windows (see page 164)

Configuration Basics

All SystemEDGE configuration information is contained in the sysedge.cf and sysedgeV3.cf configuration files. Modify the sysedge.cf file, either through configuration in the CA Virtual Assurance user interface or by direct manipulation, to configure the agent.

The sysedge.cf and sysedgeV3.cf configuration files contain all SystemEDGE agent configuration information.

The sysedge.cf configuration file, a UTF-8 encoded (plain) text file viewable with a text editor, contains local system values such as the following:

- System description, community strings, and trap communities
- Agent behavior for reporting sensitive information and running action scripts
- Entries for the agent's monitoring tables
- AIM loading and configuration parameters

The sysedgeV3.cf configuration file is an unencrypted file that can be encrypted using the se_enc utility. The file contains SNMPv3 information such as the following:

- SNMPv3 user configuration
- SNMPv2c and SNMPv3 trap destinations

Notes:

- When you manually modify the sysedge.cf or sysedgeV3.cf file, you must restart the SystemEDGE agent for the changes to take effect.
- The sysedge.mon file is created during runtime and updated automatically through the SystemEDGE agent.

On UNIX and Windows platforms, the sysedge.cf and sysedgeV3.cf files are installed in the *CASYSEDGE*\config directory. *CASYSEDGE* is a placeholder for the directory in which SystemEDGE is installed on UNIX. On Windows, the location is stored in the registry, HKLM\Software\ComputerAssociates\SystemEDGE\InstallDir. During install, the user is prompted to give information like SNMPv1 communities, sysContact, agent SNMP port and data directory. The information provided by the user is updated in the sysedge.cf file by the installer.

When the agent starts up for the first time, SystemEDGE copies the *.cf files to its data directory. Use the files in the data directory to configure the agent. On Windows systems, you can also access these files through the SystemEDGE control panel. To access the files, select Start, Control Panel, SystemEDGE, and click sysedge.cf or sysedgeV3.cf.

The installer sets up the port for the default agent instance (SystemEDGE service) during installation. When you start the agent from the command line, do not specify the –p parameter for the default instance. The default port is used automatically. Only the default agent instance communicates with the CA Virtual Assurance manager. In consequence, any configuration file (sysedge.cf, sysedgeV3.cf) received from the CA Virtual Assurance manager is placed in the SystemEDGE\data\port*default-port* directory. For example: C:\Program Files\CA\SystemEDGE\data\port161

During an upgrade, the agent additionally merges the settings from the previous versions. For more information about upgrading your configuration file settings, see Upgrade Considerations (see page 114).

Interactions Between sysedge.cf and sysedge.mon

The SystemEDGE agent uses the sysedge.mon file as a backup to store the changes made through SNMP Sets that are not automatically entered in the sysedge.cf file. The file also stores runtime information, such as the current state of a monitor. Entries in sysedge.cf take precedence over entries in sysedge.mon.

When the agent performs a full (or cold) restart, it initally reads the sysedge.mon file in the data, then reads the sysedge.cf file and overwrites existing monitors (except current state information and notInService row status) and adds additional monitors. When the agent is reconfigured using the CA Virtual Assurance manager, it only reads the entries from the delivered sysedge.cf file.

If you manually remove a monitoring entry from sysedge.cf and that entry also exists in sysedge.mon, you must also remove it from sysedge.mon to prevent the agent from using it.

Configuration Using CA Virtual Assurance

You can configure the SystemEDGE agent remotely using the CA Virtual Assurance file-based configuration capabilities. From the CA Virtual Assurance user interface, you can configure any parameter in the sysedge.cf file (including system settings, security settings, monitors, and AIMs) and deploy the configuration to one or many agent instances from one central location.

When you deploy configuration files through CA Virtual Assurance, the agent can apply any changes during runtime, and only a warm start is required. Remote configuration through CA Virtual Assurance is the recommended method for SystemEDGE configuration.

For more information about configuring the agent from the CA Virtual Assurance user interface, see the *CA Virtual Assurance Administration Guide* and *Online Help*. For more information about agent security lockdown options that you can implement with CA Virtual Assurance, see the chapter "Concepts."

File-Based Configuration

You can configure the agent in the following ways:

- Remotely using file-based configuration through CA Virtual Assurance policies
- Remotely using SNMP Sets through CA eHealth AdvantEDGE View, CA Spectrum, CA NSM Agent View, or SystemEDGE command line utilities
- Locally by updating the sysedge.cf file directly

When you configure the agent remotely through CA Virtual Assurance, the CA Virtual Assurance manager sends the configured sysedge.cf file to the CASYSEDGE_DATADIR directory of the specified agent instance or instances. This directory location contains and completes any runtime configuration deployed from CA Virtual Assurance. When the agent detects a new file uploaded to this directory, it completes a warm start and applies the changes. Any modifications made by SNMP Sets are overwritten when a sysedge.cf file is delivered to the agent by a CA Virtual Assurance manager.

When you use SNMP sets to configure the agent, the changes are applied and saved to the sysedge.mon file.

Note: You cannot use SNMP sets to configure some system settings, such as SNMP and security information.

When you configure the agent by directly updating the local sysedge.cf file you must restart the agent for the changes to take effect.

The following topics describe the parameters in the sysedge.cf file and the configuration options available for each parameter. For configuration procedures using CA Virtual Assurance, see the *CA Virtual Assurance Administration Guide* and *Online Help*.

The default entries in the sysedge.cf file are prefixed with a template directive. The template directives control the creation of actual configuration entries when you configure the agent through CA Virtual Assurance. When you configure the agent through CA Virtual Assurance, actual configuration entries are automatically created when you deploy the file.

When you edit the sysedge.cf file manually, you must manually enter configuration entries that do not have the template directive to implement the parameter setting. Entries are described and grouped into relational areas in the file. Enter necessary entries manually beneath the template directives in the appropriate area.

Use the sysedge.cf file in the data directory when manually editing the file.

Configure System Information

You can define system description, location, and contact information during installation. This information populates the system group in MIB-II. You can modify these values after installation in the sysedge.cf file.

To configure system information, enter the following fields in the sysedge.cf file, or update the fields if you entered settings during installation:

```
sysdescr <description>
syscontact <contact>
syslocation <location>
```

Configure SNMP Information

The sysedge.cf file contains SNMP access and configuration information that guides how the SystemEDGE agent communicates through SNMP.

You can configure the following SNMP information in the sysedge.cf file:

Note: Some of this information is configured based on information entered when you install the agent. You can edit this information in the file if changes are required.

- IP method for SNMP communications
- SNMP bind address
- Access communities
- SNMPv1 trap communities
- SNMPv1 trap source
- SNMPv1 trap destinations

Configure IP Family for SNMP User Datagram Protocol Communications

Hosts can have multiple IP family source sockets and multiple family destination addresses, and you can set a preferred method for the agent's SNMP User Datagram Protocol (UDP) communications.

To set a preferred IP family method for UDP communications, add the following line to the sysedge.cf file:

`sysedge_ip_family <method>`

method

Specifies the preferred communication method. If the specified method is not available, the agent tries using IPv4 and then IPv6. The following options are available:

1

Specifies that the agent tries to use only IPv4 as the preferred SNMP UDP communication method.

2

Specifies that the agent tries to use only IPv6 as the preferred SNMP UDP communication method.

3

Specifies that the agent tries to use both IPv4 and IPv6 as the preferred SNMP UDP communication methods. This is the default method.

Examples

Enter the following line to use only IPv4 mode (if available) for SNMP UDP communications:

```
sysedge_ip_family 1
```

Configure the SNMP Bind Address

By default, SystemEDGE binds to all interfaces (*/UDP-161). You can bind the agent to a specific interface by using the bind_address parameter in the sysedge.cf file as follows:

```
bind_address <address>
```

address

Specifies a particular interface for the agent to bind to. This value can be an IPv4 or IPv6 address.

Examples

Entering the following line in the sysedge.cf file binds to the 10.1.0.202 address only:

```
bind_address 10.1.0.202
```

Entering the following line in the sysedge.cf file binds to the ea2f:fe90:abcd:0000:230:a2f:200:ad01 address only:

```
bind_address ea2f:fe90:abcd:0000:230:a2f:200:ad01
```

Setting the Bind Address

Consider the following cases when you set the bind address for SystemEDGE:

- The SystemEDGE installer does not support the setting of bind address at install time.
- Policy Configuration requires the bind address to be defined at the policy level, not the individual machine level.

Policy Configuration requires the bind address to be set within the policy. If a bind address is set prior to the agent being managed/pushing up its configuration, the bind address is pulled from the original configuration and automatically included in the delivered configuration. There are a few ways how this process can be achieved.

The following terminology is used throughout these use cases:

Static sysedge.cf file

Identifies the file laid down by the installer, and is located in the *Installed_Dir*\SystemEDGE\config directory.

Default:

Windows: C:\Program Files\CA\SystemEDGE\config

UNIX/Linux: /opt/CA/SystemEDGE/config

Dynamic sysedge.cf file

Identifies the ongoing SystemEDGE configuration file, and is located under the *Data_Dir*\port<*number*> directory.

Default:

Windows: C:\Users\Public\CA\SystemEDGE\port161

UNIX/Linux: /opt/CA/SystemEDGE/config/port161

Option 1: Setting Bind Address prior to installing and starting the agent

If the agent is yet to be installed, the bind address can be set as follows:

1. Install the agent in unmanaged mode (do not specify a manager name).
2. Clear the 'Start After Install' option so the agent is not started.
3. Once installed, edit the static sysedge.cf file, and set the bind_address and manager_name.
4. Start the agent through the normal mechanism.

Option 2: Setting Bind Address on a running system, prior to it being managed

If SystemEDGE has been installed in unmanaged mode, the bind address can be set as follows:

1. Stop SystemEDGE, using the normal mechanism.
2. Edit the dynamic sysedge.cf file and set the bind_address1.
3. Edit the static sysedge.cf file and set the manager_name as required.
4. Start SystemEDGE, using the normal mechanism.

An alternative here is to delete the dynamic sysedge.cf file, and set the bind_address in the static sysedge.cf file instead.

Option 3: Setting Bind Address on an existing managed system

If SystemEDGE is already managed by Policy Configuration, set the bind address as follows:

1. Stop SystemEDGE, using the normal mechanism.
2. Delete the dynamic sysedge.cf file.
3. Delete the .sysegde.id file located in the same directory as the dynamic sysedge.cf file (this causes the agent to push up its configuration again).
4. Edit the static sysedge.cf file and set bind_address as required.
5. Start SystemEDGE, using the normal mechanism.

Configure Access Communities

You can define read-only and read-write SNMPv1 access communities during installation and modify or define additional communities in the sysedge.cf file.

Note: Define SNMPv2c and SNMPv3 access information in the sysedgeV3.cf file.

To configure SNMPv1 access communities, enter them or edit any entries provided during installation in the sysedge.cf file using the following format:

```
community <name> <access> <ip-address-list>
```

name

Specifies the community granting the defined access to the defined addresses. You can use any ASCII characters for the community name.

access

Specifies what level of permissions to grant, either read-only or read-write.

ip-address-list

Specifies a space-separated list of IP addresses that have access using the given community name. Access lists are not totally secure because systems can still spoof IP addresses. Access lists do, however, provide the ability to restrict legitimate use. You can provide IPv4 or IPv6 addresses as access lists.

After installation, sysedge.cf defines a single read-only community named public by default, which provides read-only access to MIB objects.

Note: Common practice provides read-only access using the community name public.

To modify the values of MIB objects, you must define a community that has read-write access permissions. In the following example, SystemEDGE permits read-write access using the community name private to systems with one of the following IP addresses: 45.0.4.10, 45.0.8.12, 198.130.5.7, or ea2f:fe90:abcd:0000:230:a2f:200:ad01. SystemEDGE treats any other system that attempts to use private as an authentication failure:

```
community private read-write 45.0.4.10 45.0.8.12 198.130.5.7
ea2f:fe90:abcd:0000:230:a2f:200:ad01
```

The community name of private is used here only as an example. Use a more unique community name for security purposes.

Note: The community name is *not* encrypted when it appears in an SNMP message header, and access lists are subject to potential IP spoofing. You should take configuration steps to limit potential security violations if you are using SNMPv1 communities. For more information, see Recommendations for Configuring Security.

Access Lists for SNMPv1 Communities

If the access list is empty for a defined community, SystemEDGE grants access to any system that uses the associated community name. The following restrictions apply to the access list:

- The access lists specified in sysedge.cf are used for SNMPv1 communication only.
- The SNMPv2c and SNMPv3 access list is configured separately in the SNMPv3 configuration file sysedgeV3.cf. For more information about how to define access lists for SNMPv3 users, see SNMPv3 Configuration (see page 155).
- You must separate IP addresses by a space character; you cannot use any other characters, including the newline.
- The maximum length of a community string statement (including any access list) is 1024 characters, which provides enough space for about 60 IPv4 addresses and less for IPv6 addresses. To configure longer access lists, define separate communities.
- The SystemEDGE agent includes the edgewatch, edgemon, and emphistory command-line utilities, which act as manager systems, sending requests to the agent. If you are using any of these utilities on the same system on which the agent is installed, include the IP address of that system in the access list.

Specify a SNMPv1 Trap Source

You can optionally specify a SNMPv1 trap source, which is the IP address as a source of origin in SystemEDGE trap protocol description units (PDUs). By default, SystemEDGE uses the value returned by the gethostbyname function call. Specify the trap_source parameter to override the default behavior. This parameter applies to both SNMPv1 and SNMPv2c/SNMPv3 traps.

The agent has only one trap source value, so you set this value only once in sysedge.cf. All traps in all communities take the specified address. The address you specify must be a valid IPv4 or IPv6 address or host name, but the SystemEDGE agent does not perform error checking to determine whether you have specified a valid address for the specific system you are using as the trap source.

To specify a trap source, add a line to the sysedge.cf file as follows:

```
trap_source <address>
```

address

Specifies the source address to use when sending traps. You can specify an IPv4 or IPv6 address or a host name.

Default: hostname

For example, to set the trap source to a system with an IP address of 10.0.7.73 and a host name of system1.ca.com, you can enter either of the following:

```
trap_source 10.0.7.73
```

```
trap_source system1.ca.com
```

Configure SNMPv1 Trap Destinations

The sysedge.cf file contains definitions for SNMPv1 trap destinations, which tell the SystemEDGE agent where to send SNMPv1 trap messages. You can configure the agent to send traps to any number of management systems. You can define SNMPv1 trap destinations during installation and in the sysedge.cf file.

To configure SNMPv1 trap destinations, add a line to the sysedge.cf file as follows for each system to which you want to send SNMPv1 traps:

```
trap_community <name> <address> [<port> [<encoding> [<source>]]]
```

name

Specifies the community name sent with the traps.

address

Specifies the IPv4 or IPv6 address to which to send traps.

port

(Optional) Specifies the UDP port to which to send traps.

Default: 162

encoding

(Optional) Specifies how to include the source address you defined in the trap_source parameter in traps. This parameter is important if trap_source translates to an IPv6 address. Enter the encoding parameter in a three-digit format XYZ, assuming leading zeros.

Default: 000

X

Controls extending the four byte IPv4 source address field (SNMPv1 traps only). Enter 0 to not extend the source address field to include the 16 byte IPv6 address, and enter 1 to extend the source address field.

Y,Z

Controls the inclusion of source information into varbind (Y) or UDP packet (Z; SNMPv1 traps only) in the trap. Enter one of the following for these digits:

0: Do not modify the trap's varbind or the outer UDP packet.

1: Include the trap_source parameter as is in the varbind or packet (IPv4/IPv6 address or host name).

2: Include the trap_source parameter preferably as an IPv4 address (then IPv6 address, then host name).

3: Include the trap_source parameter preferably as an IPv6 address (then IPv4 address, then host name).

4: Include the trap_source parameter preferably as a host name (then IPv4, then IPv6).

5: Follow the preference for 2 and include the host name.

6: Follow the preference for 3 and include the host name.

7: Follow the preference for 1 and include the host name (if trap_source is an IPv6 address).

source

Specifies IPv4 or IPv6 address or host name to use as a trap source.

Default: trap_source parameter value

For example, add the following line to send traps with a community name of mycommunity to the IP address 10.16.5.26 on port 1692 with the trap_source parameter and host name appended to the UDP packet:

```
trap_community mycommunity 10.16.5.26 1692 7
```

Note: The sysedge.cf file only defines SNMPv1 trap destinations. For information about configuring SNMPv2c and SNMPv3 traps, see SNMPv3 Configuration.

Agent Addresses of Traps from SystemEDGE

The source address of the traps sent from the SystemEDGE agent is the address to which the agent is bound. By default, the SystemEDGE agent binds to all of the network interfaces, so the traps sent from the agent use its first successful IP address.

If SystemEDGE is configured to send the traps to a trap receiver (such as edgetrapmon) running on the same local server as the agent, the source address will most likely be a loop back address (127.0.0.1 (for IPv4) or ::1 (for IPv6)).

Configure Authentication Failure Traps

You can configure the SystemEDGE agent to send an authentication failure trap whenever it receives an SNMP message whose community name does not match one of the communities recognized by the agent.

By default, the agent does not send authentication failure traps. To configure the agent to send authentication failure traps, comment out the no_authen_traps directive in sysedge.cf by putting a pound sign (#) character at the beginning of the line as follows:

```
# no_authen_traps
```

Note: This option applies for authentication failure traps for trap destinations set for SNMPv1 traps (configured in sysedge.cf) and for SNMPv2c/SNMPv3 traps (configured in sysedgeV3.cf).

Configure SNMP Set Notifications

When managing the SystemEDGE agent with CA Virtual Assurance, the CA Virtual Assurance manager receives notifications by default when any SNMP Set modifications are performed on an agent's configuration. These notifications allow the CA Virtual Assurance administrator to remain aware of all changes to agent configuration and give the ability to roll back any unwanted changes.

Note: For more information about configuring these notifications, see the *CA Virtual Assurance Administration Guide*.

By default, this functionality is enabled when the agent is managed through CA Virtual Assurance. To disable SNMP Set notifications to the CA Virtual Assurance manager, add the following line to the sysedge.cf file:

```
no_snmp_set_notifications
```

Configure SNMP Set Restrictions

By default, the SystemEDGE agent does not allow SNMP Set requests on the following tables in the Systems Management Empire MIB and Host Resource MIB:

Remote Shell group

Permits management systems to remotely instruct the agent to run shell scripts and programs on the agent system when enabled. The disclosure of this type of information can post a potential security risk.

Note: For more information about this group, see the chapter "Systems Management Empire MIB."

Process table and Running Software table

Permits access to processes and other software running on agent systems. Consider security risks when allowing SNMP SET requests on these tables.

Note: For more information about the Process table, see the chapter "Systems Management Empire MIB." For more information about the Running Software table, see the appendix "Host Resources MIB."

You can enable SNMP Sets to these tables, assuming that queries use a valid community with read-write permissions for SNMPv1 and SNMPv2c communication or valid SNMPv3 user credentials for SNMPv3 communication.

To enable support for SNMP Sets on the Remote Shell group, comment out the following line in the sysedge.cf file by adding a pound sign character (#) as follows:

```
# no_remoteshell_group
```

To enable support for SNMP Sets on the Process and Running Software tables, comment out the following line in the sysedge.cf file by adding a pound sign (#) character as follows:

```
# no_process_sets
```

You can also remove the lines completely.

Configure GetBulk Response Message Size

You can configure the SystemEDGE agent to limit the response size for the GetBulk request. If you limit the byte size of the GetBulk response message payload, it also limits the memory used by the SystemEDGE agent.

If the response message exceeds the limit, the SystemEDGE agent iterates to service the list of OIDs requested in the form of valid response messages. For example, to avoid the fragmentation of IP packet you can set the soft limit to 60000. As the maximum size of IP packet is 65535, it is still below the overflow limit.

To configure the agent to limit GetBulk response, add the bulk_read_response_soft_limit option in sysedge.cf as follows:

```
bulk_read_response_soft_limit <limit>
```

Here, limit specifies the number of bytes the response message must be limited to. It is an unsigned integer with default value 60000.

Note: This option applies for authentication failure traps for trap destinations set for SNMPv2c/SNMPv3 traps (configured in sysedge.cf and sysedgeV3.cf).

Configure Federal Information Processing Standard Mode

You can configure how the SystemEDGE agent handles encryption using the sysedge_fips_mode parameter. The following three methods of encryption are available:

- An internal solution to ensure minimum security functionality (MD5/SHA and DES only)
- CA eTrust Public Key Infrastructure libraries (libetpki/libcapki)
- RSA BSAFE Crypto-C Micro Edition libraries (libcrytocme2) (FIPS 140-2 compliant)

To configure the encryption mode, add a line with the sysedge_fips_mode parameter to the sysedge.cf file as follows:

```
sysedge_fips_mode <method>
```

method

Specifies the type of encryption to use. The following options are available:

0

Enables the CA eTrust Public Key Infrastructure libraries, and if this method fails, falls back to the internal minimum security solution.

1

Enables FIPS compliant encryption, and if this method fails, falls back to method 0. This is the default if the parameter is not configured.

2

Specifies that the agent operates in FIPS only mode. This enables the RSA BSAFE Crypto-C Micro Edition FIPS compliant libraries and performs no encryption if they fail.

For example, enter the following line to run the agent using only FIPS-certified protocols and FIPS-certified libraries:

```
sysedge_fips_mode 2
```

For detailed information about how to enable FIPS encryption, see the appendix "FIPS 140-2 Encryption."

Configure Diagnostic Logging Mechanism

The SystemEDGE agent contains a common logging mechanism for logging diagnostic messages to a file that is primarily used by CA support. You can configure the following aspects of the logging mechanism:

- Log level
- Log file name, size, and size limit handling mechanism
- Which modules to log messages for

To configure agent logging, enter the following lines in the sysedge.cf file:

```
sysedge_loglevel <level>
sysedge_logfile '<file>' [<size> [<number>]]
sysedge_logpattern <modules>
```

level

Specifies the level of messages to log. You can specify any of the following:

- fatal
- critical
- warning
- info
- debug
- debug[1-3]

The agent logs messages starting at the level you specify and of greater severity. For example, if you specify warning, the agent logs all warning, critical, and fatal messages.

Default: info

'file'

Specifies the log file name. You can include an absolute path name with the file name. If you do not include an absolute path name, the log file is created in the agent's port-specific data directory.

Default: 'sysedge.log'

size

Specifies the maximum file size in KB. If you set this parameter to equal or less than zero, the log file will have no size limits.

Default: 10240

number

Controls the handling of the log file when the size limit is reached. You can specify any of the following options:

0

Logging stops when the size limit is reached.

1

Logging saves the current log file contents to '<*file*>.1' when the size limit is reached and wraps to the start of the log file.

>1

Logging uses a wrap-around set (or ring) of up to the specified number of backup files with '<*file*>.<*number*>' containing the oldest log entries at any one time.

-1

Logging wraps to the start of the log file when the size limit is reached.

-2

Logging saves the current log file contents to '<*file*>.YYYY-MM-DD_hh.mm.ss' when the size limit is reached and wraps to the start of the log file.

Default: 1

modules

Specifies a space-delimited list of the modules for which to log messages. The available modules are as follows:

- **se:** SystemEDGE agent
- **svcrsp:** Service Response Monitor AIM
- **rmonwbem:** Remote Monitoring AIM
- **monwin:** Monitor Maintenance Window AIM
- **perfcube:** Performance Cube AIM
- **util:** Common/shared library
- **regexpr:** Common/shared library
- **shm:** Common/shared library
- **Default:** Log messages from any module

The following example logs all messages of severity info and above for the SystemEDGE agent and SRM AIM to a file named sysedgemsg.log with a limit of 30000 KB and a wrapping size limit mechanism that saves all older files:

```
sysedge_loglevel info
sysedge_logfile sysedgemsg.log 30000 -2
sysedge_logpattern se svcrsp
```

Configure Maintenance Mode

The SystemEDGE agent can run in maintenance mode, where it stops processing all monitor entries and sending traps. Maintenance mode is useful if the agent's system is undergoing a planned outage and you want to avoid receiving false alarm traps.

While in maintenance mode, the agent continues to collect metrics and respond to SNMP requests, but it suspends processing all monitors and history collections. Externally, the agent reports the row status of all monitors as notInService, while it maintains the correct row status internally. The agent saves the current value of all monitors at the beginning of the maintenance window, compares it to the current value at the end of the maintenance window, and sends traps in response to the current value as necessary.

To enable maintenance mode, add the following line to the sysedge.cf file:

```
sysedge_maintenance
```

Managing maintenance mode in the configuration file is not as efficient as using the maintenance mode control in the CA Virtual Assurance manager interface. You can toggle maintenance mode from the interface, and the configuration file automatically updates and deploys when the setting changes. The agent performs a warm start when configured through CA Virtual Assurance instead of a full restart. For more information about enabling maintenance mode in CA Virtual Assurance, see the *CA Virtual Assurance Administration Guide* and *Online Help*.

Configure Support for Actions

By default, the SystemEDGE agent does not permit the execution of action commands with the monitoring tables. The capability to run action commands and scripts can be a potential security issue, because the command and scripts can run commands as the root or administrator users. If you want to execute actions when certain monitoring thresholds are reached, you must enable support for actions.

To enable support for action execution, comment out the following line in the sysedge.cf file by adding a pound sign (#) character as follows:

```
# no_actions
```

Configure User and Group Permissions for Subprograms (UNIX Only)

By default, the SystemEDGE agent runs subprograms (for example, remote shell, action, and extension object invocations) with its effective user and group permissions, normally root. Depending on the local security policies in effect, you may want to set the agent to use actions and extension objects that run with different user and group permissions.

To run subprograms with the effective user and group permissions of a user other than root, add the following lines to the sysedge.cf file:

```
subprogram_user_name <name>
subprogram_group_name <name>
```

name

Specifies the user name or group name to use for subprograms. If the provided name is invalid, all subprogram execution is disabled.

Configure MIB Table Restrictions

Several parameters exist in the sysedge.cf file for preventing the SystemEDGE agent from populating certain tables in the Systems Management Empire MIB. None of the parameters are entered by default, but you can add them to restrict access to tables that contain sensitive information or consume significant resources.

To disable support for certain MIB tables, enter any of the following lines in the sysedge.cf file, depending on the table to restrict:

no_process_table

Restricts populating the Process table. The table can discover the processes running on the underlying system, which could violate the local security policies in effect.

no_topprocs_table

Restricts populating objects in the Top Processes table.

no_who_table

Restricts populating the Who table, which provides information about users who are currently logged in to a system. The disclosure of this type of information can pose a potential security risk.

no_usergroup_table

Restricts populating the User and Group tables, which provide information about the user accounts and user groups configured for the system. You may want to restrict access to these table if your organization considers the disclosure of user and group information a security issue or if you have a distributed system with large numbers of users and groups. The agent periodically caches this information internally, so storing user and group information could consume a significant amount of resources.

no_trapcommunity_table

Restricts populating the Trap Community table, which provides information about the agent's current SNMP configuration, including version, communities, and trap destinations. You may want to restrict access to this table due to the sensitive nature of SNMP settings such as communities.

no_mirror_monitor_table

Restricts populating the Mirror Monitor table, which provides a read-only mirror of the Self Monitor table that enables displaying monitor entries in a special context without searching the entire Self Monitor table. You may want to disable this table if a mirror of the Self Monitor table would consume too many resources.

no_mirror_aggregate_table

Restricts populating the Mirror Aggregate table, which provides a read-only mirror of the Aggregate table that enables displaying aggregate entries in a special context without searching the entire Aggregate table. You may want to disable this table if a mirror of the Aggregate table would consume too many resources.

For more information about any of these tables, see the chapter "Systems Management Empire MIB."

Configure Device Status Checking Restrictions

Several parameters exist in the sysedge.cf file for preventing the SystemEDGE agent from querying the status of certain system devices, typically to avoid potential agent blocking. By default, the SystemEDGE agent does not automatically determine the status of all floppy devices on the system.

To enable floppy drive status checking, comment out the following line in the sysedge.cf file by adding a pound sign (#) character as follows:

```
# no_stat_floppy
```

You can also remove the line completely.

You can disable other device status checking that is enabled by default. To disable device status checking, enter any of the following lines in the sysedge.cf file, depending on the device that you want to restrict:

Note: Enabling any of these restrictions affects the corresponding MIB tables, because not all information will be provided for the respective objects.

no_serial_status

Disables checking the status of all serial devices on the host system. The agent does this by default as part of its support for the hrDeviceTable of the Host Resources MIB. Some serial applications encounter problems with this operation because they cannot handle the opening and closing of serial devices (which are necessary for determining status) by any process other than themselves. When you enter this line, the agent checks only the keyboard and mouse in the hrDeviceTable and returns unknown(1) for the status of serial ports in response to a management system query.

no_discover_disks

Enables the search and find option of disk devices on SystemEDGE. If this option is disabled, the disk devices do not appear in the following SystemEDGE SNMP tables:

- hrStorageTable
- hrDeviceTable
- hrDiskStorageTable
- hrFSTable
- devTable
- diskStatsTable

Note: When the no_discover_disks option is enabled, the no_probe_disks option is automatically enabled.

no_probe_disks

Disables checking the size, capacity, description, and other properties of disks and CD-ROMs installed on the underlying system to avoid potentially lengthy agent blocking on older UNIX systems while the driver waits on status information.

Note: If you add this line, the agent will discover disk devices, but may be unable to provide disk statistics, capacity information, device descriptions, and status information.

no_stat_nfs_filesystems

(UNIX only) Disables exposing local and remote file system data, which the agent does by default through the Systems Management Empire MIB. On some UNIX systems, checking for file system data may cause the agent to block if a file system is mounted from a remote file server that is no longer available. There is no way for the agent to unblock in this situation.

Note: If you add this line, it disables the checking of all remote file systems, not just NFS file systems.

no_hpux_probe_graphics

Disables checking the status of graphics on HP-UX systems to avoid potential agent blocking.

Configure Warm Start Discovery

When managing the SystemEDGE agent with CA Virtual Assurance, the agent performs a warm start when you update its configuration through the CA Virtual Assurance manager. The warm start lets the agent implement changes without requiring a full restart. By default, when an agent warm start occurs, the agent does not re-discover all devices, because this can take a long time if you are managing a system with many devices. However, you can enable this discovery to re-discover all devices after a warm start.

To enable warm start discovery, comment out the following line in the sysedge.cf file by adding a pound sign (#) character as follows:

```
# no_warmstart_discovery
```

The warm start functionality only applies to agents in managed mode through CA Virtual Assurance. For more information, see the *CA Virtual Assurance Administration Guide*.

Configure Linux Free Memory Calculation

You can configure how the SystemEDGE calculates free memory on Linux to include system buffers, cached memory, or both in the calcluation.

To configure Linux free memory calcluation, enter the following line in the sysedge.cf file:

```
linux_freemem_include [buffers] [cached]
```

buffers

Includes system buffers in the free memory calculation.

cached

Includes disk cached memory in the free memory calculation.

For example, to include both system buffers and disk cached memory in the free memory calcuation, add the following line:

```
linux_freemem_include buffers cached
```

Configure Monitor Aggregation in a Multi-tier Hierarchy

SystemEDGE can aggregate self monitor and process monitor entries to correlate multiple monitors for the same attribute to display only the greatest severity. Monitors that are aggregated have the same object class, instance, and attribute. By default, when managing aggregate monitor entries, the agent only sends traps and executes actions for the entry with the worst aggregated state. You can change this default setting in the sysedge.cf file.

To send traps and execute actions for all connected monitors, not just the worst severity, add the following lines to sysedge.cf:

```
aggregate_monitor_traps
aggregate_monitor_actions
```

You can also configure the level of monitor aggregation. By default, the agent aggregates all self monitors and process monitors with the same object class, instance and attribute into *standard aggregates*. All monitors belonging to an aggregate are called *connected monitors*. You can configure higher levels of aggregation in the sysedge.cf file. The higher level of aggregates are called super aggregates (use an asterisk '*' as a placeholder). For example, you can create monitor aggregations like the following in the object Aggregate table:

- Monitors with the same object class and attribute and any instance
- Monitors with the same class and any instance and attribute

Aggregates for virtual object instances support more than two tiers. The supported instances are structured as follows:

```
//<any1>/<any2>/.../<anyN>/<instance>
```

Example

Object instances on an LPAR :

```
//hmc1/hyper1/lpar1/disk1
```

Super aggregates are only provided for two tiers (the two rightmost ones) but not for all tiers. For this example the following super aggregates are created:

//hmc1/hyper1/lpar1/*

//hmc1/hyper1/*/disk1

//hmc1/hyper1/*/*

The special super aggregate //* is created to aggregate all instances

Note: For the object instance '//' and '/' are used as delimiters to relate an instance to its hosted hierarchy: '//hierarchy/instance'. The hierarchy can be multi-tier. For local system instances '//./' is prepended.

Examples

```
//VM-Host/VM-Guest/Disk1
//Host/Disk1
//./Disk1
```

Super aggregates can be enabled for one or both of the rightmost two tiers. The special super aggregate //* is created to aggregate all instances.

Examples for level 0002

```
//VM-Host/VM-Guest/*
//Host/*
//./*
```

Examples for level 0004

```
//VM-Host/*/Disk1
//*/Disk1           (excluding //./Disk1)
```

Examples for level 0008

```
//VM-Host/*/*
//*/*               (excluding //./Disk1)
//*
```

To configure monitor entry aggregation levels, add the following line to the sysedge.cf file:

```
aggregate_level <level>
```

level

Specifies a bit field allowing the aggregation levels of the following table:

Class	Instance	Attribute	Level
class	//hierarchy/instance	attribute	(standard aggregate)
class	//hierarchy/*	attribute	0002
class	//hierarchy-1/*/instance	attribute	0004
class	//hierarchy-1/*/*	attribute	0008
class	//hierarchy/instance	*	0010
class	//hierarchy/*	*	0020
class	//hierarchy-1/*/instance	*	0040
class	//hierarchy-1/*/*	*	0080
*	//hierarchy/instance	attribute	0100
*	//hierarchy/*	attribute	0200
*	//hierarchy-1/*/instance	attribute	0400
*	//hierarchy-1/*/*	attribute	0800
*	//hierarchy/instance	*	1000
*	//hierarchy/*	*	2000
*	//hierarchy-1/*/instance	*	4000
*	//hierarchy-1/*/*	*	8000

For example, add the following line to the sysedge.cf file to apply all aggregation rules, causing aggregations up to an overall agent state:

```
aggregate_level 0xffff
```

For more information about monitor aggregation and the object class, instance, and attribute Self Monitor table columns, see the chapters "Concepts" and "Self Monitoring."

Monitoring Configuration

The sysedge.cf file stores the monitor entries that control the SystemEDGE agent's monitoring operations. You can create monitor entries using any of the following methods:

- Dynamically in the CA Virtual Assurance user interface, where you can deploy the changes to any agent system without requiring a restart
- Remotely or locally using SNMP Set requests
- Directly in the local sysedge.cf file for implementation after a restart

You can add lines to the sysedge.cf file to add monitoring entries to the following MIB tables:

- The Self Monitor table for monitoring MIB object thresholds, including file systems, interface, processors, and so on. The Self Monitor table supports entering monitoring entries based on a MIB attribute's OID or an object's class, instance, and attribute values and setting attribute thresholds.
- The Process Monitor table for monitoring process attributes and Windows services. The Process Monitor table supports entering monitoring entries based on process or service regular expressions and setting process attribute thresholds.
- The Process Group Monitor table for defining and monitoring groups of processes. The Process Group Monitor table supports entering monitoring entries based on process regular expressions to form a group to monitor for changes or group statistics.
- The Log Monitor table for monitoring log files for regular expression pattern matches. The Log Monitor table supports entering entries based on a log file and regular expression patterns within the file.

- The NT Event Monitor table for monitoring Windows event logs for regular expression pattern matches. The NT Event Monitor table supports entering monitoring entries based on regular expression patterns to match in Windows event logs.
- The History Control table for collecting historical data for baselining and trend analysis. The History Control table supports entering entries based on any integer-based MIB attribute's OID.

The agent's state management model derives state and severity information from entries in the Self Monitor and Process Monitor tables and aggregates connected monitors to display and act on only the worst severity status.

For more information about how each of these monitoring tables operates, see the chapter "Concepts." For more information about table columns and creating entries in the sysedge.cf file for each of these tables, see the chapter for each monitoring type. For more information about creating monitor entries from the CA Virtual Assurance user interface, see the *CA Virtual Assurance Administration Guide* and *Online Help*.

Note: You can also define MIB extensions and Windows registry or performance data to monitor in the sysedge.cf file. For more information, see the chapters "Custom MIB Objects" and "Windows Registry and Performance MIB Objects."

Load SystemEDGE AIMs

The SystemEDGE agent provides a plug-in architecture through which you can load optional SystemEDGE AIMs at initialization. These AIMs provide an extensible and flexible approach to supporting application-specific MIB variables.

You can edit the sysedge.cf file to specify which AIMs the agent should load. You can specify an absolute path or only the AIM name for the SystemEDGE agent to find the AIM to load.

The following on-demand modules are included with the SystemEDGE distribution on every platform. These modules, unlike AIMs, are loaded implicitly when the functionality is needed:

- Monitor Maintenance Window
- Performance Cube

Note: The functionality previously included in the Top Processes AIM is now integrated into the base agent. For more information, see the chapter "Process and Service Monitoring."

To load SystemEDGE AIMs, enter the AIM name or the name and full path for the AIM on the following line in the sysedge.cf file:

```
sysedge_plugin
```

To load the SRM AIM, for example, enter the following in the sysedge.cf file:

```
sysedge_plugin svcrsp
```

You do not have to enter the absolute path, although entering the path still creates a valid entry. The SystemEDGE agent automatically derives the platform and file location for AIM loading.

For information about enabling other AIMs, see the documentation for that AIM. For more information about configuring AIMs, see the chapter "SystemEDGE AIMs."

Monitor Maintenance Window Module Configuration

The Monitor Maintenance Window module provides time-based control of SystemEDGE monitoring. The Monitor Maintenance Window module can stop the production of traps from self, process, process group, log file, history, and Windows event monitors during periods of the day when they would not be significant without stopping the monitor itself.

This on-demand module is provided with the SystemEDGE agent, and you can configure its operation from the sysedge.cf file. When you add a line to the file for configuring the module, it is automatically enabled.

For more information about this module, see the chapter "SystemEDGE AIMs."

Perfcube Module Configuration

The Perfcube module collects History table metrics in Systems Performance cubes for performance analysis by CA NSM Systems Performance. The module reads the entries in the History table, performs metric averaging over a defined interval, and writes the average metric value into the cube file.

This on-demand module is provided with the SystemEDGE agent, and you can configure its operation from the sysedge.cf file. When you configure a cube collection interval greater than zero in the file using the following parameter, the AIM is automatically enabled:

```
emphistory cube_interval interval
```

interval

Specifies the interval in minutes for collecting history information into the cube file.

For more information about configuring the Perfcube module, see the chapter "SystemEDGE AIMs."

Recommendations for Configuring Security

We recommend that you implement the following configuration parameters for security purposes:

```
no_who_table
no_usergroup_table
no_remoteshell_group
no_process_sets
```

Additional configuration options that can help implement security are as follows:

```
no_actions
no_process_table
no_trap_community_table
no_mirror_monitor_table
no_mirror_aggregate_table
subprogram_user_name
subprogram_group_name
```

Regular Expression Examples for Monitors and Autowatchers

This section provides regular expression examples in context to each monitor type.

Process Monitors

- Matches processes that start with 'svchost':

  ```
  ^svchost
  ```

- Matches processes that contain 'sql':

  ```
  sql
  ```

- Matches processes that end in 'svc':

  ```
  svc$
  ```

- Matches both 'cmd' and 'CMD' processes (requires PCRE):

  ```
  /cmd/i
  ```

Note: Unless you are monitoring for the existence of multiple instances of a process (with the 0x4000 flag), use a regular expression that matches only one process.

Windows Service Monitors

- Matches the service with the exact display name 'Print Spooler':

  ```
  ^Print Spooler$
  ```

Note: For Windows services, use a regular expression that matches only one service.

Process Group Monitors

- Matches all processes that start with ‘sql’ or ‘svchost’:

  ```
  ^sql|^svchost
  ```

Windows Event Monitors (Source Filter)

- Matches events from any source:

  ```
  .*
  ```

- Matches events from the http server:

  ```
  http
  ```

Windows Event Monitors (Description Filter)

- Matches events with Event ID of 277 (requires the ‘Prepend Event ID’ flag – 0x0100):

  ```
  \[277\]
  ```

- Matches events containing the text ‘FATAL’ anywhere in the line:

  ```
  FATAL
  ```

Log File Monitors (Search Filter)

- Matches on text indicating a failed su attempt:

  ```
  su.*fail
  ```

- Matches on the text strings “WARNING:” and “Invalid login attempt” which is separated by an unknown number of lines in the log file (requires PCRE):

  ```
  /^WARNING:(.*\n)*Invalid login attempt/m
  ```

- Matches on the text ‘error’ regardless of case (requires PCRE):

  ```
  /error/i
  ```

Generic Autowatchers

Instance Criteria RegEx, RegEx 2, RegEx 3:

- Matches on text containing ‘WiFi’:

  ```
  WiFi
  ```

- Matches on text containing ‘C:’, ‘D:’, or ‘E:’:

  ```
  C:|D:|E:
  ```

Process Autowatchers

Instance Criteria RegEx, RegEx 2, RegEx 3:

- Matches processes that start with ‘svchost’:

  ```
  ^svchost
  ```

- Matches processes that contain ‘sql’:

  ```
  sql
  ```

- Matches processes that end in ‘svc':

  ```
  svc$
  ```

- Matches both ‘cmd’ and ‘CMD’ processes (requires PCRE):

  ```
  /cmd/i
  ```

- Matches all processes that start with ‘sql’ or ‘svchost’:

  ```
  ^sql|^svchost
  ```

Service Autowatchers

Instance Criteria RegEx, RegEx 2, RegEx 3:

- Match on all services with ‘SQL’ in the display name:

  ```
  .*SQL.*
  ```

- Match on all services starting with ‘SQL’ in the display name:

  ```
  ^SQL
  ```

- Match on all services starting with ‘SQL’ regardless of case:

  ```
  /^SQL/i
  ```

More information:

Perl Compatible Regular Expression (PCRE) Support (see page 150)
Multiline Regular Expressions (see page 151)
Configure Text Pattern Exclusion (see page 152)
Regular Expression Examples (see page 152)

Perl Compatible Regular Expression (PCRE) Support

Perl Compatible Regular Expressions (PCRE) enables you to specify i18n compatible (UTF-8) regular expressions while defining monitors that support regular expressions. These monitors monitor log files, processes, process groups, Windows services and Windows events. You can also use this option to create more complex regular expressions.

You can choose between two regular expression libraries:

- Standard SystemEDGE regular expression library (default)
- Perl Compatible Regular Expression (PCRE) library

PCRE is not 100 percent compatible with the standard regular expression library due to delimiters and a wider range of special characters.

To enable PCRE library, you can manually edit the configuration file in SystemEDGE legacy mode by adding the following line:

```
use_pcre
```

If you edit the configuration file in managed mode, the configuration is wiped out the next time policy is deployed.

You can enable PCRE library in managed mode using CA Virtual Assurance Policy Configuration user interface. In the user interface, use the “Use Perl Compatible Regular Expressions” option in “Control Settings” tab. Specifying the “<JS>” pattern modifier activates JavaScript compatibility mode for this expression.

PCRE covers POSIX basic and POSIX extended regular expressions and a subset of regular expression functionality provided by Perl. For more information about PCRE, see http://www.pcre.org.

Supported formats for input regular expressions:

```
/regular expression/switch
```

```
regular expression
```

Note: The suggested way to use PCRE is specifying regular expressions with delimiters and optional switches (/regex/i).

List of supported switches

/i

PCRE_CASELESS (Perl compatible /i): Matches both uppercase and lowercase letters.

/m

PCRE_MULTILINE (Perl compatible /m): Matches across multiple lines.

/s

PCRE_DOTALL (Perl compatible /s): A dot metacharacter in the pattern matches a character of any value, including one that indicates a newline.

/x

PCRE_EXTENDED (Perl compatible /x): Whitespace data characters in the pattern are ignored except when escaped or inside a character class.

/E

PCRE_DOLLAR_ENDONLY: $ matches only at the end.

/U

PCRE_UNGREEDY: The greediness of the repetition quantifiers is inverted, that is, by default they are not greedy, but if followed by a question mark they are.

/<JS>

PCRE_JAVASCRIPT_COMPAT: Specifies JavaScript compatibility.

List of supported delimiters

In addition to the default regular expression delimiter (/), SystemEDGE supports alternate delimiters to simplify expressions that specify forward slashes in UNIX and Linux paths. The alternate delimiters are (#), (|), and (:).

For example, the following regular expressions are equivalent:

```
/\/dir1\/dir2\/dir3\/filename/i
#/dir1/dir2/dir/filename3#i
|/dir1/dir2/dir3/filename|i
:/dir1/dir2/dir3/filename:i
```

Multiline Regular Expressions

Perl Compatible Regular Expressions (PCRE) provides the ability to match the text that appears on multiple lines. You can use this functionality by using the PCRE_MULTILINE (/m) or PCRE_DOTALL (/s) switch in the expression.

When either of these switches is detected in a Log File Monitor regular expression, the agent searches for the specified expression within the last 8 KB of the log file being monitored.

Configure Text Pattern Exclusion

You can configure the text pattern exclusion in a regular expression using Perl Compatible Regular Expressions (PCRE). The PCRE library enables the execution of regular expression pattern matching. PCRE library provides the flexibility to specify a pattern to be included or excluded in a regular expression during pattern matching.

In some search criteria, you may need to match a text pattern (in a log file) only if a specific text pattern is excluded. You can exclude a text pattern by creating a unique regular expression and identifying the expression with the excluded pattern. For instance, you can create an expression to identify matches only for substrings with ‘abc’, excluding ‘def’. To exclude 'def' in a regular expression, use negative assertion to not to match if subject contains substring 'def'.

Example: Configure Text Pattern Exclusion

The following example specifies the search option to search for text pattern that matches all strings ‘abc_’, which is not followed by string ‘def’. Here, the brackets () is used to group the strings and ?! represents negation.

```
abc_(?!def)
```

Regular Expression Examples

This section provides examples for using the watch logfile directive in the sysedge.cf file to add regular expression to the Log Monitor table.

Simple Expression

Example: /the quick brown fox/

The following example searches for the regular expression '/the quick brown fox/' in the sentences:

Input

```
The quick brown fox jumps over the lazy dog.

the quick brown fox jumps over the lazy dog.
```

Result

```
the quick brown fox
```

Case-Insensitive Expression

Example: /The quick brown fox/i

The following example searches for the regular expression '/The quick brown fox/i' in the sentences. The search matches for the expression 'The quick brown fox', regardless of the case.

Input

```
The quick brown fox jumps over the lazy dog.

the quick brown fox jumps over the lazy dog.

the quick brown FOX jumps over the lazy dog.
```

Result

```
The quick brown fox jumps over the lazy dog.

the quick brown fox jumps over the lazy dog.

the quick brown FOX jumps over the lazy dog.
```

Multiline Mode - Start of Line and End of Line Metacharacters

Example: /^abc$/m

The following example searches for the regular expression '/^abc$/m' in the sentences. This expression matches only if the pattern 'abc' is on its own line.

Input

```
Lazy dog\nquick horse\nabc
```

Result

abc

Multiline Mode - PCRE_DOTALL

Example: /two.*three.*four/s

The following example searches for the regular expression '/two.*three.*four/s' in the sentences. When the PCRE_DOTALL pattern modifier is specified, the dot metacharacter also matches on newlines.

Input

```
one number\ntwo numbers\nthree numbers\nfour numbers\nfive numbers
```

Result

```
Two numbers\nthree numbers\nfour
```

Extended Expression - Ignoring Whitespace

Example: /^1234 # comment in input string/x

The following example searches for the regular expression '/^1234 #comment in extended re/x' in the sentences. This expression matches on the quoted text, regardless of whitespace encountered in the input.

Input

```
1234 #comment in input string

1234 #  comment in      input   string

1234#commentininputstring
```

Result

```
1234 #comment in input string

1234 #  comment in      input   string

1234#commentininputstring
```

Dollar Sign Matches Only at End

Example: /X$/E

The following example searches for the regular expression '/X$/E' in the sentences. The text in the pattern (“X”) matches only at the end of the subject string.

Input

```
Test X

What can I do with X
```

Result

```
X

X
```

JavaScript Compatible

Example: /a[^]b/<JS>

The following example searches for the regular expression '/a[^]b/<JS>' in the sentences. Specifying the “<JS>” pattern modifier activates JavaScript compatibility mode for this expression.

Input

aXb

Result

aXb

SNMPv3 Configuration

SNMPv3 provides an enhanced level of security from previous versions. SNMPv3 has the following levels of communication:

- noAuthNoPriv is similar to SNMPv1 and SNMPv2. Messages are accompanied by a username, which must be consistent between the sender and the receiver.
- AuthNoPriv uses a consistent username and a password.
- AuthPriv uses a username, a password, and an encryption key. This key encrypts the body of the message.

To maintain the consistent information, both the sender and receiver must be synchronized and the information made secure using a provided encryption tool.

You can configure SystemEDGE to use SNMPv3 based communication. The agent is based on the User-based Security Model (USM) defined for SNMPv3.

The sysedgeV3.cf file contains policy defining how the SNMP administrator handles responsibilities and specifies the level of security expected when accessing a host. The file is located in the CASYSEDGE_DATADIR directory, and it lets you specify the following:

- SNMP engine ID prefix
- SNMPv3 security users
- Level of security
- Authentication protocol and password
- Encryption protocol and its password

There is no user interface for configuring the sysedgeV3.cf file. You must configure SNMPv3 settings directly in the file. However, you can access the file from the CA Virtual Assurance manager and deploy the configured file to systems.

Configure the SNMPv3 Engine ID

You must specify a textual SNMP engine ID prefix, which is concatenated with a process ID and IP address by the agent's SNMP library, to operate the agent in SNMPv3.

To configure the SNMPv3 engine ID, add the following line to the sysedgeV3.cf file:

```
SNMP_V3_ENGINE_ID <engineIdPrefix>
```

The default value for engineIdPrefix is SystemEDGEAdmin. Do not use spaces in the configured string.

For example, the following line in the SNMPv3 configuration file specifies the string CompanySNMPV3ADMIN as a prefix for the SNMP engine ID:

```
SNMP_V3_ENGINE_ID CompanySNMPV3ADMIN
```

Configure SNMPv3 User Information

You specify SNMPv3 USM user and security information in the sysedgeV3.cf file using the SNMP_V3_USER_INFO keyword. All of the arguments for this keyword must be on one line and in the specified order separated by blank spaces.

To configure SNMPv3 user information, add the following line to the sysedgeV3.cf file:

```
SNMP_V3_USER_INFO *|access[addresses] userName securityModel securityLevel
[authProtocol authPassword [privProtocol privPassword]]
```

access

Specifies read or write access. Value "read" or "write" is mandatory.

addresses

Specifies an IP filter to filter the requests originating from a specified IP address or a subnet. This field is not mandatory. If this is not specified, agent information is accessible to all of the hosts. For more information, see Address Filtering for SNMPv3 Users.

userName

Specifies the name of the SNMPv3 secure user to which to allow access.

securityModel

Specifies the SNMPv3 security model in use. The SystemEDGE agent currently only supports the User-based Security Model (USM). Only a value of 3 is supported.

securityLevel

The following values are supported for the supported levels of security:

noAuthNoPriv

Indicates that no authentication and no privacy (encryption) protocols are configured for use for this SNMPv3 user.

AuthNoPriv

Indicates that an authentication protocol is configured and no privacy protocol is configured for this SNMPv3 user.

AuthPriv

Indicates that an authentication and a privacy protocol is configured for use with this SNMPv3 user.

authProtocol

Specifies the authentication protocol to be used. Currently MD5 and SHA protocols are the only used. You should only specify this option if AuthPriv or AuthNoPriv security level is set.

authPassword

Specifies the SNMPv3 user's authentication password (key) used by the authentication protocol. Specifying authPassword is required only if authProtocol (MD5 or SHA) is set.

privProtocol

Specifies the encryption (privacy) protocol used by the SNMPv3 user. DES, 3DES, and AES are the only protocols supported. If you specify an encryption protocol, you must specify authProtocol and authPassword also. If you specify privProtocol, AuthPriv is the only supported securityLevel.

privPassword

Specifies the SNMPv3 user's encryption password (key) used by the encryption protocol. This parameter is required only if you set privProtocol.

You can assign read or write access to different security levels. For example, the security levels of No Authentication and No Privacy (noAuthNoPriv) can be equivalent to the public community string, while Authentication and Privacy (AuthPriv) can be equivalent to the admin community string.

Examples

Examples of valid SNMPv3 user definitions follow:

```
SNMP_V3_USER_INFO *|read  joe1 3 AuthPriv  MD5  apass  AES  ppass
SNMP_V3_USER_INFO *|read  joe2 3 AuthPriv  SHA  apass  DES  ppass
SNMP_V3_USER_INFO *|write|192.168.29.0 joe3 3 AuthPriv SHA  apass  3DES  ppass
SNMP_V3_USER_INFO
*|write|100.10-255.100.101,e000-efff:f0ff:bef0:*,192.168.120.0,*:1  joe4  3
AuthNoPriv  SHA evansar
SNMP_V3_USER_INFO *|read  joe5 3 noAuthNoPriv
```

Address Filtering for SNMPv3 Users

The SNMPv3 configuration file lets you specify a source address filter to restrict access to the information only to those source addresses that match the address filter.

Following are supported syntax and usage guidelines for the address filtering field:

- Full IPv4 addresses are supported.

  Example:

  ```
  130.10.100.101
  ```

  This line specifies that the requests from the source address 130.10.100.101 are allowed access to the agent.

- Full IPv6 addresses are supported. Specify hexadecimal notation for IPv6 addresses. You can specify characters in lowercase or uppercase. Following is an example IPv6 address:

  ```
  Ea2f:fe90:abcd:0000:230:a2f:200:ad01
  ```

  This line specifies that the requests from the source address ea2f:fe90:abcd:0000:230:a2f:200:ad01 are allowed access to the agent.

- Host names are supported. However, we highly recommend using IP addresses as filters instead of host names, because a server might have multiple IP addresses associated with it, and not all of the IP addresses might be associated with the host name in the DNS.

  Example:

  ```
  box1.domain.com
  ```

  This line specifies that the requests from box1.domain.com are allowed access to the agent.

  Note: If you specify a host name, the agent must be able to resolve the IP address received to the specified source host name.

- Wild card character asterisks (*) are supported. Use wild card characters to specify a range of addresses that can access the agent.

Example:

```
130.10.*.101
```

This line specifies that the requests from the source IPv4 addresses starting from 130.10 and ending with 101 (i.e. 130.10.0.101 to 130.10.255.101) are allowed access to the agent.

Example:

```
Ea2f:fe90:*:ad01
```

This line specifies that the requests from the source IPv6 addresses starting with ea2f:fe90 and ending with ad01 are allowed access to the agent.

- If you specify wild card character (*), it should be the only character in that IP field. IP fields are the characters between dot "." (for IPv4 addresses) and colon ":" (for IPv6 addresses).

  For example, an entry of 130.10.3*.200 is invalid because the 3 cannot be within the same dot as the asterisk. An entry of Ea2f:fe*:abcd:0000:230:a2f:200:ad01 is also invalid because the 'fe' cannot be within the same colon as the asterisk.

- In the case of multiple wild cards, only the leftmost wild card is stretched to fill the missing fields. For example, entering 2002:*:12f4:*:1012 is the same as entering 2002:*:*:*:*:12f4:*:1012.

- 0 (zero) is treated as a wild card character.

  Example:

  ```
  130.10.120.0
  ```

  This line specifies that the requests from the IPv4 subnet 130.10.120 are allowed access to the agent. Entering the address this way is the same as specifying 130.10.120.*.

  Example:

  ```
  Ea2f:fe90:0:ad01
  ```

  This line specifies that the request from IPv6 addresses starting with ea2f:fe90 and ending with ad01 are allowed access to the agent. This is the same as entering Ea2f:fe90:*:ad01.

- Use two or more zeros to remove the wild card behavior of a single zero and to treat it as a literal zero.

  Example:

  ```
  130.000.120.00
  ```

  This line specifies that the requests from the IPv4 address 130.0.120.0 are allowed access to the agent.

Example:

```
Ea2f:*:0000:000:ad01
```

This line specifies that requests from the IPv6 addresses starting with ea2f and ending with 0:0:ad01 are allowed access to the agent.

- Partial IP addresses are supported. Partial addresses are treated as partial addresses followed by a wild card (*). Any requests coming from the source address starting with the partial address are allowed.

  For example, 130.10 would be same as specifying 130.10.*. Also, Ea2f:fe90 would be same as specifying Ea2f:fe90:*.

- For IPv6 addresses, two consecutive colons (::) are treated as a wildcard between two colons (:*:).

- A range of addresses using a dash (-) is supported for IPv4 and IPv6 addresses. Any or all the fields can have the range defined.

  Examples:

  ```
  130.10-255.100.101
  ```

  This line specifies that the requests from the source IPv4 address range from 130.10 to 130.255 are allowed access to the agent

  ```
  e000-efff:f0ff:bef0:*
  ```

  This line specifies that the requests from the source IPv6 address range from e000:* to efff:* are allowed access to the agent.

- Multiple filters delimited by a comma (,) are supported.

  Example:

  ```
  130.10-255.100.101, e000-efff:f0ff:bef0:*, 130.10.120.0, box2.domain.com
  ```

- No spaces are allowed.

Note: When a host has multiple interfaces or IP addresses, you must supply all its interfaces and IP addresses to access the agent, because the request (such as snmpget, walktree, snmpset, etc) could be sent through any of the interfaces or IP addresses. You must define all IP addresses assigned to the host in the agent's IP filter, or the host does not have access to the agent. As an alternative to defining all IP addresses and interfaces, you can specify the host name, as long as you are aware of the limitations of using host names.

Configure SNMPv2c and SNMPv3 Trap Destinations

The sysedgeV3.cf file lets you specify the types of traps that the agent should send and trap properties. The following trap types are supported:

- SNMPv2c traps
- SNMPv2c notifications (also referred to as INFORM requests and confirmed traps)
- SNMPv3 traps
- SNMPv3 notifications (also referred to as INFORM requests and confirmed traps)

Note: SNMPv1 trap destinations are configured in sysedge.cf instead of sysedgeV3.cf.

SNMPv2c traps and SNMPv2c notifications are sent using a SNMP community string. This community string must be defined in sysedge.cf (for example, community global read-only).

SNMPv3 traps and SNMPv3 notifications are sent using the SNMPv3 user's credentials. The SNMPv3 user must be defined in the sysedgeV3.cf configuration file before the trap definition record.

To configure the agent to send SNMPv2 traps to a specified destination host, add the following line to the sysedgeV3.cf file:

```
SNMP_V2_TRAP_INFO  <destination_host>|<port>  <trap_context> <community>
<trap-encoding>
```

To configure the agent to send SNMPv2c notification traps, also known as confirmed traps and INFORM requests, to a specified destination host, add the following line to the sysedgeV3.cf file:

```
SNMP_V2_NOTIFICATION_INFO  <destination_host>|<port>  <trap_context> <community>
<timeout> <num_of_retries> <trap-encoding>
```

To configure the agent to send SNMPv3 traps to a specified destination host, add the following line to the sysedgeV3.cf file:

```
SNMP_V3_TRAP_INFO  <destination_host>|<port>  <trap_context> <SNMPv3_user>
<trap-encoding>
```

To configure the agent to send SNMPv3 notification traps, also known as confirmed traps and INFORM requests, to a specified destination host, add the following line to the sysedgeV3.cf file:

```
SNMP_V3_NOTIFICATION_INFO <destination_host>|<port> <trap_context> <SNMPv3_user>
<timeout> <num_of_retries> <trap-encoding>
```

destination_host

Specifies the host to which to send the trap. You can specify a host name or an IP address.

port

Specifies the port number on the destination host that you want to send the trap.

trap_context

* (asterisk) is the only supported value for this field. This value is mandatory.

community

(SNMPv2c only) Specifies the SNMP community string with which to send the trap. This string must also be defined in the sysedge.cf file.

SNMPv3_user

(SNMPv3 only) Specifies the SNMPv3 user with which to send the trap. This user must also be defined in the sysedgeV3.cf file.

timeout

(Optional, notifications only) Specifies how long in seconds to wait for a confirmation of notification delivery before timing out.

num_of_retries

(Optional, notifications only) Specifies the number of times to retry sending a notification after a timeout.

trap-encoding

(Optional) Specifies the type of encoding to use when sending traps. This is similar to configuring trap encoding in SNMPv1 For more information, see Configure SNMPv1 Trap Destinations (see page 128).

Examples

The following example sends a SNMPv3 trap notification to the host sysmanager on port 6666 for the SNMPv3 user osmanager, waits for acknowledgement with a timeout of thirty seconds, and retries up to two times:

```
SNMPV3_NOTIFICATION_INFO sysmanager|6666 * osmanager 30 2
```

The following example sends a SNMPv2c trap to the host localhost on port 162 using the private community:

```
SNMPV2_TRAP_INFO localhost|162 * private
```

The following example sends a SNMPv2c trap notification to the host localhost on port 162 using the public community, waits for acknowledgement with a timeout of thirty seconds, and retries up to three times:

```
SNMPV2_NOTIFICATION_INFO localhost|162 * public 30 3
```

Encrypt the SNMPv3 Configuration File

SNMPv3 encryption is provided using the encryption (privacy) protocol, DES as the authentication protocol, and SHA using a default CA defined key. The encrypted configuration files can be generated in a central location and shared among several hosts.

For additional security, you can encrypt the SNMPv3 configuration file, sysedgeV3.cf. You must create a clear text version of the file, encrypt it, and install the encrypted file in the appropriate directory.

If you use the optional –L switch, the utility detects the current locale of the console and language catalog if available. If a language catalog is not found, the utility falls back to English as a default language.

To encrypt the SNMPv3 configuration file

1. Test and validate the syntax in sysedgeV3.cf.
2. Encrypt sysedgeV3.cf with the following command:

   ```
   se_enc -i sysedgeV3.cf -o sysedgeV3.cf.crypt -m 2 -d 2
   ```
3. Move the clear text sysedgeV3.cf to an archive area.
4. Rename sysedgeV3.cf.crypt to sysedgeV3.cf.
5. Copy sysedgeV3.cf to the agent's data directory or apply the file via remote deployment from CA Virtual Assurance manager.

Note: SNMPv3 configuration files can be shared among several hosts.

Disable SNMPv1 and SNMPv2c

To turn off the agent's SNMPv1 and SNMPv2c communication, comment out or delete the lines containing the following in the sysedge.cf file:

```
community <name> <access> <addresses>
```

SystemEDGE Control Panel for Windows

As a standalone Windows service, SystemEDGE has its own SystemEDGE Control Panel. To view the SystemEDGE Control Panel, open the Control Panel dialog and double-click SystemEDGE.

You can use the SystemEDGE Control Panel to perform the following tasks:

- Start and stop the agent
- View community strings and trap destinations
- Open the configuration files
- View the sysedge.log file
- Run the diagsysedge utility
- View the *User Guide* and *Release Notes*

Chapter 6: Starting the Agent

This chapter explains how to start the SystemEDGE agent. Before you do so, you must configure the agent for your environment. For more information, see the chapter "Agent Configuration."

Note: After you start the agent as described in this chapter, you can use the diagsysedge.exe program to verify that your agent is running. For more information, see the chapter "Troubleshooting and Usage Suggestions."

This section contains the following topics:

Start or Stop the Agent on Windows Systems (see page 165)
Start or Stop the Agent on UNIX Systems (see page 166)
Command Line Startup Options (see page 167)
Automatic Agent Startup at System Boot (see page 168)
Agent Warm Start (see page 169)
Maintenance Mode (see page 169)

Start or Stop the Agent on Windows Systems

For Windows systems, you can manually start SystemEDGE from the command line, the Services Control Panel, or the SystemEDGE Control Panel.

To start or stop the agent from the command line, enter the appropriate command from the following list:

```
net start sysedge
net stop sysedge
```

The agent starts as a service in the background using the port that you specified during the installation and the default configuration files in the data directory of the agent installation.

You can also run the agent in the foreground using nondefault options or nondefault configuration files. For more information, see Command Line Startup Options (see page 167).

To start (stop) the agent from the Services Control Panel

1. Select Start, Administrative Tools, Services.

   The Windows Services dialog appears.

2. Right-click SystemEDGE and select Start (Stop).

   The agent starts (stops).

To start the agent from the SystemEDGE Control Panel

1. Access the Control Panel and double-click SystemEDGE.

   The SystemEDGE Control Panel appears.

2. Click Start (Stop) Agent.

   The agent starts (stops).

Note: In the Windows XP Category View, the SystemEDGE Control Panel is under "Network and Internet Connections." In the Windows XP Classic View, the SystemEDGE Control Panel icon is at the main level of the display.

Start or Stop the Agent on UNIX Systems

On UNIX systems, you should start or stop SystemEDGE using the provided service startup script. In rare occasions (such as when instructed by CA Support), you can manually start the agent directly.

Service Startup Script for UNIX Systems

You can start, stop, or restart the agent automatically on UNIX systems using service startup scripts. It starts the agent on the port number that you specified during the installation.

The service script for all UNIX platforms is as follows:

```
CASYSEDGE/bin/sysedgectl [start|stop|status|restart|probe]
```

To start the agent from the command line on UNIX, run the following command from the *Install_Dir*/bin directory:

```
./sysedgectl start
```

The agent starts using the port that you specified during the installation and the default configuration files in the data directory of the agent installation.

You can also run the agent in the background using non-default options or non-default configuration files. For more information, see Command Line Startup Options (see page 167).

Examples

```
cd $CASYSEDGE/bin

./sysedgectl start
./sysedgectl restart
./sysedgectl stop
```

Command Line Startup Options

This section describes the command line options to start the SystemEDGE agent on Windows and UNIX systems. Using these options, you can start the agent with values other than default values of ports and configuration files. The options enable you to run another instance of the agent from a command shell such as testing configuration changes.

Note: Upon startup of the agent, the existing log file is saved with zero extension as '*filename.0*'. If an existing agent log file at runtime is changed by deploying a new configuration in the managed mode, it is saved with zero extension.

Important! Agent installation and CA Virtual Assurance deployment only support the default agent instance.

The sysedge binary supports the following command line options:

`[-p port] [-f config file] [-e SNMPV3 config file] [-m monitor file] [-h] [-b] [-d]`

-p *port*

Specifies the port on which to listen for the incoming SNMP messages. Specify a different port than you entered during installation to run a separate instance of the agent. Port UDP/1691 is reserved for use as an alternate port for running the SystemEDGE agent. Do not specify this argument if you are using the port specified during installation.

Note that only the default agent instance communicates with the CA Virtual Assurance manager. Therefore, any configuration file received from CA Virtual Assurance is placed in the CASYSEDGE_DATADIR directory corresponding to the default port.

-f *config file*

Specifies the path name to use for the sysedge.cf file instead of the default configuration file in CASYSEDGE_DATADIR.

-e *SNMPV3 config file*

Specifies the path name to use for the sysedgeV3.cf file instead of the default SNMPv3 configuration file in CASYSEDGE_DATADIR.

-m *monitor file*

Specifies the path name to use for the sysedge.mon file instead of the default monitor table configuration file in CASYSEDGE_DATADIR.

-h

Displays help for the command. This option lists the available command line options.

-b

(UNIX only) Runs the agent in the background. The agent runs as a daemon process and disconnects from the controlling terminal. Use this flag when starting the agent from a startup script.

-d

Runs the agent in debug mode. This option sets the agent logging level to debug3. For more information about the logging mechanism and logging levels, see the chapter "Agent Configuration."

Automatic Agent Startup at System Boot

The SystemEDGE installation automatically copies scripts to the system startup area so that they can be used to start the agent automatically whenever the system is booted.

Some systems might require additional steps to make the agent start automatically whenever the system is booted.

This section includes instructions for the following operating systems:

- Windows
- UNIX

Note: Upon startup of the agent, the existing log file is saved with zero extension as '*filename.0*'. If an existing agent log file at runtime is changed by deploying a new configuration in the managed mode, it is saved with zero extension.

Start the Agent Automatically on Windows Systems

The SystemEDGE service starts by default at system boot time.

To verify that the agent is set to start automatically

1. Open the Windows Services dialog.
2. Right-click CA SystemEDGE and select Properties.
3. Verify that Automatic is selected in the Startup type drop-down list on the General tab.

Start the Agent Automatically on UNIX Systems

The SystemEDGE agent starts by default at system boot on UNIX systems using a service startup script. It connects to the default port that you specified during installation, and is aware of the installation path.

Agent Warm Start

When integrated with CA Virtual Assurance, the SystemEDGE agent can implement a deployed configuration file without requiring a cold restart. Instead, receiving configuration changes from CA Virtual Assurance triggers a warm start, where the agent can implement changes while running. When performing a warm start, the agent does not have to drop and rebind to interfaces, reestablish OID variables, or restart AIMs, enabling faster and less resource intensive configuration.

For more information about deploying runtime configuration changes through CA Virtual Assurance, see the *CA Virtual Assurance Administration Guide*.

Warm Start Exceptions

When integrated with CA Virtual Assurance, the SystemEDGE agent performs a warm start when receiving most deployed configuration changes. However, the agent is still fully recycled in any of the following situations:

- Manually through user commands. You can still start and stop the agent at any time using the command line or the SystemEDGE Control Panel on Windows.
- Implicitly if the agent receives a configuration file change that requires a restart. Removing an AIM or changing the sysedge_fips_mode parameter triggers a cold restart. The agent restarts automatically using a process that is triggered by these changes.

  Note: Upgrading CAM also triggers an automatic agent restart.
- Implicitly if CAM is manually stopped, because SystemEDGE registers as a CAM client. The agent does not automatically restart in this situation; you must manually start the agent after CAM is running again.

Maintenance Mode

The SystemEDGE agent features a maintenance mode, in which the agent continues to collect metrics and respond to SNMP requests but does not process monitors or send state change traps. This mode is useful when you know of a planned outage or maintenance and do not want to receive traps during this period of time. You can put the agent in maintenance mode to avoid stopping and restarting the agent in this situation.

For more information about enabling maintenance mode, see the chapter "Agent Configuration."

Chapter 7: Systems Management Empire MIB

This chapter gives examples of management information available through the enterprise-specific Systems Management Empire MIB.

This section contains the following topics:

Systems Management Empire MIB Overview (see page 172)
System Group (see page 173)
User Table (see page 175)
Group Table (see page 175)
Process Table (see page 175)
Who Table (see page 176)
Remote Shell Group (see page 177)
Performance Group (see page 178)
Interprocess Communication Group (see page 178)
Buffers Group (see page 179)
Directory Name Lookup Cache Group (see page 180)
AIX Logical Partition Group (see page 180)
Table of Work Load Manager (WLM) Classes (see page 181)
Trap Community Table (see page 181)
NT System Group (see page 181)
Distributed Systems Group (see page 185)
History Table (see page 186)
Log Monitor Table (see page 187)
Disk Statistics Table (see page 188)
CPU Statistics Table (see page 189)
Extension Group (see page 190)
Process Monitor Table (see page 190)
Process Group Monitor Table (see page 190)
Aggregate State Table (see page 191)
LPAR Group (see page 191)
Mirror Tables (see page 191)
Autowatcher Tables (see page 191)
Unsupported Systems Management Empire MIB Objects (see page 192)

Systems Management Empire MIB Overview

The Systems Management Empire MIB modules define a collection of objects for managing host systems. The MIB is organized into groups and tables for monitoring the following types of information:

- System information
- Mounted devices
- Kernel configuration
- Streams
- Users and groups
- Processes
- Remote command execution
- Performance
- Interprocess communications (IPC)
- Buffer statistics for network buffers
- Streams buffers
- I/O buffer cache
- Directory name lookup cache
- Logical partitions
- NFS and RPC statistics
- Windows-specific statistics
- Disk and CPU statistics

The MIB also includes monitoring and object aggregation tables. You can monitor specific objects from the MIB groups and tables using self monitoring entries.

Note: Based on your local security policies, you can disable support for any table. For more information about disabling support for a table, see the “Agent Configuration” chapter.

You can create the self monitoring entries by specifying an object OID or the object class, instance, and attribute. Use the self-monitoring functionality to monitor any MIB object against a defined threshold. You can maintain object state that is based on the current retrieved value. The agent sends the state change traps when the threshold is breached. For more information, see the chapter "Self Monitoring."

The top-level OID of objects in the Systems Management Empire MIB is 1.3.6.1.4.1.546. Refer the MIB file to derive full OIDs.

This chapter contains the following information:

- List of all Systems Management Empire MIB groups and tables
- Default class values for MIB groups and tables
- Default instance values for MIB tables

To define the monitoring entries, use the default class and instance values for each group and table. The syedge.oid file maintains the default values for each attribute to be used to define a self monitor entry without specifying OID.

This chapter defines the default class for all groups, tables, and the default instance for all tables; the instance value is not required when monitoring an attribute in a MIB group. The default attribute value is the name of the MIB attribute.

Note: For more information about the reverse OID lookup feature, see the chapter "Self Monitoring."

For a detailed description of the MIB objects and individual attribute names, see the empire.asn1 file in the mib subdirectory of the agent installation.

System Group

To monitor the basic information, retrieve object values in the sysedge system group.

Class: sysedgeSystem

Device Table

Device table provides you information about the devices and file systems that are mounted on the host systems.

Class: devTableEntry

Instance: value of devMntPt attribute for the specific device

File System Space Monitoring

The devCapacity column of the Device table shows the percentage of file system space used. Use the devCapacity column to check if the file system space usage is full.

Use the Self Monitor table to instruct the agent to monitor the devCapacity values and send traps to the management system if the devCapacity value breaches the threshold value. For example, use the following directive to configure the agent to monitor the /usr directory and send a critical state change trap if the usage capacity of the /usr directory is more than 90%:

```
monitor 11 0x00 60 absolute > 90 'Monitor device space usage' '' devTableEntry '/usr'
devCapacity
```

For more information about setting thresholds, see the chapter "Self Monitoring."

Unmount a Mounted Device

To unmount a mounted device, issue an SNMP Set command for the devUnmount column of that device to set the column value to delete (1). The agent unmounts the device and removes its entry from the Device table.

Note: For SNMP Set operations, use a community name that grants you read-write access.

Kernel Configuration Group

To identify the kernel version running on the system and determine the kernel configuration details, retrieve the objects in the Kernel Configuration group.

Class: kernelConfig

Boot Configuration Group

Use the MIB objects in the Boot Configuration group to determine which device and partition the system uses for the root file system. You can also determine the dump file system, swap space, and the number of blocks that is contained by each system.

Class: bootconf

Streams Group

The Streams I/O subsystem provides a data transit/processing path between applications in user space and drivers in kernel space in the host. You can monitor the health of the Streams subsystem by retrieving objects in the Streams group.

Class: streams

User Table

Use the User table to retrieve information about the user accounts that are created on the system. Each row in this table represents a user account.

Class: userEntry

Instance: value of userLoginID attribute for the specific user

Group Table

Use the Group table to retrieve information about the user groups that are created on the host system. Each row in the table represents a group defined in the /etc/group file.

Class: groupEntry

Instance: value of groupName attribute for the specific group

Process Table

Use the Process table to determine what processes are currently running on the system and process metrics.

Class: processEntry

Instance: value of processName attribute for the specific process

Change the nice Value of a Process (UNIX only)

To set or change a value in the nice column of a process, issue an SNMP Set command for the processNice column for a specific process. For example, you can increase or decrease the scheduling priority of a process by setting the priority value in the nice column of a process.

For a list of values, see the UNIX nice (1) man page or Windows SetPriorityClass () function description on the MSDN web site.

Send a Signal to a Process

To send a signal to a process, issue an SNMP Set command for the processKill column for a specific process to the appropriate signal value.

Note: On Windows, only signal number SIGKILL (9) is supported.

For example, from a remote network management station console, you can terminate an unauthorized process by setting the value of its processKill column to 9. The number 9 represents the UNIX SIGKILL signal, which can kill a process.

For more information about signals, see the UNIX signal (3V) man page.

Who Table

You can use Who table to get a list of users who are currently logged on to the host system. This table is used to get the user details for a particular system at a given time. Each row in this table represents a current user.

Class: whoEntry

Instance: value of whoName and whoDevice attributes for the specific user and device

Remote Shell Group

Use the Remote Shell Group to run shell scripts and programs on the remote host system.

Class: remoteShell

MIB variables in the Remote Shell group let you specify the following commands:

- Command syntax (shellCmd)
- Output file (shellOutput)

Variable in the Remote Shell group let you view the following command information:

- Exit code from the execution of a shell command (shellExitstat)
- Command output from the output file (shellOutputContents)

Run a Remote Command

To run a remote command, perform an SNMP Set on the shellCmd MIB variable. When you perform SNMP Set operations, use a community name that grants you read-write access.

Enter the following at a command prompt to run a remote command:

```
SET shellOutput.0 = output filename
SET shellCmd.0 = command
```

The agent fork/execs the specified command with standard output (stdout) and standard error (stderr) redirected to the file named by the shellOutput MIB variable. When the command finishes executing, the agent puts the exit status of the command in shellExitStat.

Performance Group

Use the Performance group to track the health and performance of the operating system of the host.

Class: kernelperf

Use the Performance group attributes to monitor the statistics that are sampled over a fixed time period.

Note: The sampling interval is system dependent.

For detailed descriptions of these MIB objects, see the empire.asn1 file in the mib subdirectory of the agent distribution.

Interprocess Communication Group

You can track the Interprocess Communication (IPC) mechanisms that is used on a system by retrieving one of the following tables from the IPC group:

Message Queue table

Provides information about system message queues currently in use.

Class: msgqueEntry

Instance: value of queID attribute for the specific message queue

Shared Memory table

Provides information about system shared memory segments currently in use.

Class: shmemEntry

Instance: value of shmemID attribute for the specific shared memory segment.

Semaphore table

Provides information about system semaphores currently in use. A semaphore is a value in operating system or kernel storage that a process can check and change to coordinate activities in which multiple processes compete for the same operating system resources.

Class: semEntry

Instance: value of semID attribute for the specific semaphore

A row in each of these tables represents one instance of IPC usage. The columns differ for each table.

Delete IPC Mechanisms

Use the IPC group tables to delete message queues, shared memory segments, and semaphores. Each of the IPC group tables contains a column that the agent uses to destroy an instance of the IPC usage.

To delete IPC mechanisms, issue an SNMP Set command for queDel (Message Queue table), shmemDel (Shared Memory table), or semDel (Semaphore table) attribute to set the attribute value to delete (1).

Note: For SNMP Set operations, use a community name that grants you read-write access.

Buffers Group

The Buffers group contains groups and tables for monitoring the following buffer types:

- Network or message buffers
- Streams buffers
- I/O buffer cache

Message Buffers Group

Use the Message Buffers group to monitor how a system uses message buffers.

Class: mbufs

The group also contains a Message Buffers Allocation table to track the allocation for each message buffer type.

Class: mbufAllocEntry

Instance: value of mbufDesc attribute for the specific message buffer type.

The Streams Buffers Group

This table defines the WLM Classes of the node which the agent resides on. Note that the data in this table is only available when the SystemEDGE agent is running on an AIX-based OS and the Work Load Manager in the OS is enabled.

Streams Buffers Allocation Table

Use the Streams Buffers Allocation table to monitor buffer allocation and usage statistics of buffers that the Steams subsystem uses.

Class: strbufAllocEntry

Instance: value of strbufAllocIndex attribute for the specific streams buffer

I/O Buffer Cache Group

Use the I/O Buffer Cache group to track I/O buffer allocation and usage of the basic disk I/O.

Class: ioBufferCache

You can also graph the values of these counters to detect peak periods of an I/O buffer activity.

Directory Name Lookup Cache Group

Use the Directory Name Lookup Cache (DNLC) group to monitor the caching of directory and the mapping of file names to vnode on UNIX. The group contains performance statics of this cache.

Class: dnlc

AIX Logical Partition Group

The AIX Logical Partition group lets you monitor information about individual logical partitions (LPARs) on an AIX system.

Class: logicalPartition

For more information about using the AIX Logical Partition group to monitor an LPAR environment, including descriptions of each MIB attribute, refer the AIX Logical Partition table.

Table of Work Load Manager (WLM) Classes

You can view currently configured access communities and other SNMP-specific configuration information in the Trap Community table. You can edit the settings in this table from the CA Virtual Assurance user interface. The table contains information supporting SNMPv1, SNMPv2, and SNMPv3, based on which version you are running.

Class: trapCommunityEntry

Instance: value of tcDestinationAddr and tcDestinationPort attributes for the specific trap destination and port

Trap Community Table

You can view currently configured access communities and other SNMP-specific configuration information in the Trap Community table. You can edit the settings in this table from the CA Virtual Assurance user interface. The table contains information supporting SNMPv1, SNMPv2, and SNMPv3, based on which version you are running.

Class: trapCommunityEntry

Instance: value of tcDestinationAddr and tcDestinationPort attributes for the specific trap destination and port

NT System Group

The NT System group containing groups and tables is designed for Windows systems.

Class: ntSystem

Note: In most cases, the SystemEDGE agent supports the same MIB objects for Windows and UNIX. However, the underlying Windows operating system does not support some of the System Management MIB objects. For a list of the MIB objects not supported on Windows, see the *SystemEDGE Release Notes*.

NT Thread Table

In the Windows operating system, the kernel schedules threads to run on the processors. A thread is a part of a process that runs program code. A process can contain one or many threads.

Use the NT Thread table to obtain detailed status information for each thread executing on the system.

Class: ntThreadEntry

Instance: value of ntThreadPID and ntThreadNumber attributes for the specific thread

NT Registry Group

The Windows registry acts as a database repository to store the system configurations details such as hardware, user accounts, and applications. Use the NT Registry group to query the Windows registry.

Class: ntRegistry

You can determine if your registry is in danger of growing larger than the size limit set for it by querying the following:

- Current size of the registry in MB (ntRegistryCurrentSize)
- Registry size limit in MB (ntRegistrySizeLimit)

NT Services Table

Windows provides several programs that run as Windows services. Use the NT Services table to monitor the Windows services.

Class: ntServiceEntry

Instance: value of ntServiceName attribute for the specific service

NT Performance Groups

NT performance groups exist for monitoring the following Windows performance metrics:

- System
- Cache
- Memory
- Page File

NT System Performance Group

Use the NT System Performance group that provides statistics to track the health and performance of the Windows operating system of a system.

Class: ntSystemPerf

For example, a heavily loaded system typically has a high level of system activity with many process threads competing for CPU time and jobs queuing up as they wait to run. You can track the level of system activity by monitoring the ntContextSwitches and ntRunQLen MIB objects.

ntContextSwitches counts the number of context switches that have occurred. A context switch occurs each time that a thread gives up the CPU and another takes its place. A high rate of context switching indicates a high system load. ntRunQLen indicates whether there is a backup of threads waiting to run.

NT Cache Performance Group

The NT Cache Performance group contains system-wide buffer and file system cache statistics that are used to determine the effectiveness of the caching mechanisms of a system.

Class: ntCachePerf

When an application requests data, the data is first mapped into the cache and then the data is copied into memory from the cache. Later, the data that the application changes is written from the cache to disk by the Lazy Writer system thread or by a write-through call from the application.

NT Memory Performance Group

The NT Memory Performance group contains memory and paging performance counters and statistics. Use this group to track the memory usage of a system.

Class: ntMemoryPerf

NT Page File Performance Group

Windows uses the Pagefile.sys paging file to handle any extra demands for memory. The paging file is a block of disk space that the operating system reserves so that the memory manager can create space in the memory, if needed.

You can monitor the size of the paging file to determine whether you can add more memory to your system using the NT Page File Performance group.

Class: ntPageFilePerf

NT Event Monitor Group

Windows uses event logs to capture important system and application status messages. The NT Event Monitor group provides access to these event logs.

Class: ntEvents

You can use the NT Eventn Monitor table to instruct the agent to continuously monitor Windows event logs for events that you specify. Whenever a matching event is generated on the system, the agent notifies the management systems with a trap message. The agent can also run an action command to immediately handle the event. The NT Event table provides detailed status information for each monitor entry.

Class: ntEventMonEntry

Instance: value of ntEventMonDescr attribute for the specific monitor entry

For more information about NT Event Monitor, see the chapter “Windows Event Monitoring.”

The NT Event Log Monitoring Table

This table contains NT Event Log monitor entries. Each entry specifies an Event Log and regular expression. The agent periodically states the event logs to see if they have been updated. If the events are updated, the agent scans the new entries for the regular expressions. When an entry matches the regular expressions, a trap PDU is sent to appropriate managers.

NT Registry and Performance Extension Group

The SystemEDGE agent provides a powerful mechanism for extending the Systems Management Empire MIB to include information from the Windows registry and performance counters. This includes both configuration data (typically viewed using regedit) and performance data (typically viewed using perfmon).

Using this feature, you can customize the agent to return additional configuration and performance information for systems and applications. For instance, many applications provide registry entries that specify the configuration of the application, and the agent can make these entries available through SNMP. The SystemEDGE agent can also access statistics that many applications provide.

This support is provided in the NT Registry and Performance group.

Class: ntRegPerf

This group contains 128 unspecified scalar MIB variables that you can configure. In response to an SNMP Get request for one of these variables, the SystemEDGE agent reads the Windows Registry and returns the value.

For more information about using and configuring MIB objects in the NT Registry and Performance group, see the chapter “Windows Registry and Performance MIB Objects.”

Self Monitor Table

Use the Self Monitor table to configure the self-monitoring capability of the agent for monitoring any integer-based MIB variable under its control. The agent automatically monitors the MIB variable and notifies the management system with a state change trap when a threshold is breached. The Self Monitor table supports stateful monitoring, where the table maintains the status for each monitor, and aggregate self monitors of other tables to calculate the object state.

Class: monitorEntry

Instance: value of monDescr attribute for the specific self monitor entry

For more information about the Self Monitor table and monitoring MIB objects, see the chapter "Self Monitoring.”

Distributed Systems Group

The distributed systems group provides the required objects to monitor and manage the following distributed system facilities on all platforms:

- RPC
- NFS

RPC Statistics Group

Use the RPC (Remote Procedure Call) Statistics group to track statistics and counters that relate to the RPC facilities usage of the kernel.

Class: rpc

You can graph the values of these counters to detect peak periods of RPC activity. Statistics and counters are divided based on their applicability for client and server-side protocol operations.

Note: For more information about RPC, see RFC 1057.

NFS Statistics Group

Use the NFS Statistics group to track statistics and counters that are related to the NFS facilities usage of the kernel.

Class: nfs

You can graph the values of these counters to detect peak periods of NFS activity. Statistics and counters are divided based on their applicability for client and server-side protocol operations.

Note: For more information about NFS, see RFC 1094.

History Table

SystemEDGE tracks the values of various integer-based MIB objects (counters, gauges, and so on) over time and store these values for later retrieval. This functionality, commonly called history sampling, reduces the amount of management station polling across the network. The agent provides this functionality through two SNMP MIB tables:

History Control table

Class: empireHistoryCtrlEntry

Instance: value of empireHistoryCtrlDescr attribute for the specific History Control table entry

History table

Class: empireHistoryEntry

Instance: value of empireHistorySampleIndex attribute for the specific History table entry

The History Control table contains parameters that describe the data that is sampled and stored in the History table. Each row in the control table defines a specific data collection function by assigning values to the parameters (column objects) of the table. One or more rows (stored samples) in the History table are associated with that single control row.

Each control table row is assigned a unique index value (empireHistoryCtrlIndex). A row defines the data collection function by specifying the object instance that you want to sample, how often to sample (in multiples of 30 seconds), and the number of samples to keep (buckets). Associated with each data-collection function (row of the control table) is a set of rows of the History table. Each row of the History table, also named bucket, holds the value of the specified MIB object gathered during one sampling interval.

As each sampling interval occurs, the agent adds a row in the History table with the same empireHistoryIndex value as other rows for this data collection function. This new row corresponds to the single row in the History Control table and has an empireHistorySampleIndex value one greater than the SampleIndex of the previous sample.

For more information about the attributes in these tables and using them for history collection, see the chapter "History Collection."

Log Monitor Table

You can use the Log Monitor table to configure the agent to monitor system log files for regular expressions. You can also monitor file directories for size and contents.

Class: logMonitorEntry

Instance: value of logMonitorLogFile attribute for the specific log monitor entry

For example, you can configure the agent to monitor the /var/adm/sulog log file for the regular expression su.*fail. Whenever the agent finds a match, it sends an SNMP trap to the configured management systems.

Each row of the Log Monitor table represents the monitoring of a log file for a particular regular expression.

For more information Log Monitor table attributes and using the table to monitor log files, see the chapter “Log File Monitoring.”

Disk Statistics Table

The Disk Statistics table provides the I/O statistics of a disk.

Class: diskStatsEntry

Instance: value of diskStatsHostmibDevTableIndex attribute for the specific disk drive

Each table entry provides the latest disk statistics for one disk. The agent periodically (every 60 seconds) checks the status of the system data structures for each disk and records the values in the table.

When you use the Disk Statistics table, see the diskStatsLastUpdate column for information about the time (in TimeTicks) this row was last updated. This column provides information whether the statistics have been updated since you last queried the table.

Configure Disk Performance Statistics Collection for AIX Systems

For AIX systems, you must manually configure the systems to collect disk-performance statistics. SystemEDGE can monitor these statistics only if you instruct the operating system to track them.

Note: Enabling the collection of these statistics decreases the disk throughput.

To enable collection of disk statistics for AIX systems

1. Log in to the system you want to monitor with root privileges.
2. Enter the following at the command prompt:

```
chdev -l sys0 -a iostat=true
```

To disable collection of disk statistics for AIX systems

1. Log in to the system you want to monitor with root privileges.
2. Enter the following at the command prompt:

```
chdev -l sys0 -a iostat=false
```

CPU Statistics Table

The CPU Statistics table provides performance statistics for each CPU.

Class: cpuStatsEntry

Instance: value of cpuStatsDescr and cpuStatsIndex attributes for the specific CPU

Each table entry provides the latest statistics for one CPU. The agent periodically (every 60 seconds) checks the status of the system data structures for each CPU and records the values in the table. Use the CPU Statistics table to monitor the following statistics:

- CPU type (cpuStatsDesc)
- Number of ticks spent in Idle, User, System, or Wait mode (cpuStatsIdle, cpuStatsUser, cpuStatsSys, cpuStatsWait)
- Time of last statistics update (cpuStatsLastUpdate)
- Percentage of the physical processors consumed by logical CPU (cpuStatsPC)
- Percentage of entitled capacity consumed by logical CPU (cpuStatsEC)
- Percentage of time in Idle, User, Sys, or Wait mode (cpuStatsIdlePercent, cpuStatsUserPercent, cpuStatsSysPercent, cpuStatsWaitPercent)

When you use the CPU statistics table, see the cpuStatsLastUpdate column for information about the time (in TimeTicks) that this row was last updated. This column provides information whether the statistics have been updated since you last queried the table.

Class: cpuGroup

Each table entry provides the latest statistics for all CPUs in one group. The related CPU Totals group adds the metrics from the table to display a total for all CPUs. It also provides totals for processor and entitled capacity consumed by LPAR.

Extension Group

The Extension group provides a powerful mechanism for extending the Systems Management Empire MIB to include a wide range of information about your systems and applications.

Class: extensionGroup

This group contains 232 unspecified scalar MIB variables that you can configure. In response to an SNMP Get request for one of these variables, the SystemEDGE agent invokes the command that you specify for the variable and returns the value that is returned from the command. Using an SNMP Set operation, you can also pass parameters to a command.

For more information about using the Extension group variables, see the chapter “Custom MIB Objects.”

Process Monitor Table

The Process Monitor table provides a powerful and flexible mechanism for monitoring applications and processes. You can monitor various attributes of key processes and applications and can send SNMP state change traps when certain thresholds have been exceeded. Like the Self Monitor table, the Process Monitor table supports stateful monitoring.

Class: processMonEntry

Instance: value of pmonRegExpr attribute for the specific process

For more information about the Process Monitor table attributes and monitoring processes, see the chapter “Process and Service Monitoring.”

Process Group Monitor Table

The Process Group Monitor table provides a powerful and flexible mechanism for monitoring groups of processes. You can create an entry in the Process Group Monitor table using a regular expression. The table takes the sum of the statistics for the processes that match the expression and monitors the group for group changes.

Class: processGroupMonEntry

Instance: value of pgmonProcRegExpr attribute for the specific process group

For more information about the Process Group Monitor table attributes and monitoring process groups, see the chapter “Process Group Monitoring.”

Aggregate State Table

The Aggregate State table aggregates self and process monitors with the same class, instance, and attribute value into one table entry to maintain an aggregate object state. This table enables the agent to maintain the status information for entries in the Self Monitor and Process Monitor table.

Class: aggregateEntry

Instance: value of aggObjInstance and aggObjAttribute attributes for the specific table entry

Note: You can configure the agent to aggregate on higher levels. For more information, see the chapter "Agent Configuration."

LPAR Group

This group provides the statistics of the Physical and Logical CPUs for the logical partition (LPAR).

Class: totalPhysicalCpuStats

Class: logicalPartition

Mirror Tables

The Systems Management Empire MIB contains mirror tables of the Aggregate and Self Monitor tables. These tables contain the same attributes and entries as the main tables and are used as cross reference to the main tables.

Autowatcher Tables

The autowatcher table describes automatic monitoring activities of the agent. A row of this table causes the agent to automatically create monitor entries for all matching instances, which are based on the specified criteria.

Unsupported Systems Management Empire MIB Objects

SystemEDGE supports the Systems Management Empire MIB on all supported operating systems, but some MIB objects within this MIB module are not supported on the underlying operating systems. In addition, some UNIX-specific MIB objects are not applicable to Windows. Therefore, you cannot implement those objects using the Windows version of the agent.

For a list of unsupported Systems Management Empire MIB objects on all operating systems, see the *SystemEDGE Release Notes*.

Chapter 8: Self Monitoring

This chapter explains how to use the SystemEDGE agent to monitor MIB variables against user-specified thresholds.

This section contains the following topics:

Self Monitoring Overview (see page 193)
State Management of Self Monitors (see page 194)
Self Monitor Table (see page 196)
Self Monitoring Configuration (see page 211)
edgemon Commands for Self Monitoring (see page 230)
Remove Self Monitoring Entries (see page 235)

Self Monitoring Overview

The SystemEDGE agent lets you dynamically monitor any integer-based MIB variable under its control. You use the Self Monitor table to specify the variable to monitor, the polling interval, a comparison operator (greater than, equal to, and so on), a threshold value, a severity, object information, and other values. The SystemEDGE agent automatically monitors that variable and sends a trap to the management system if the condition you specified is met.

Note: If monitors use the state model, SystemEDGE sends traps for the managed object and not for each defined monitor.

The SystemEDGE agent can maintain and update a state value for any self monitor you specify and use object class, instance, and attribute values to aggregate the status of an object with multiple monitors defined.

The agent can also run commands on the managed system to perform management functions to correct the cause of the exception immediately. For example, you can configure the agent to monitor the percent capacity for the root file system and to notify the manager if it becomes too full.

State Management of Self Monitors

The SystemEDGE state management model maintains a state value for all entries in the Self Monitor table and lets you define managed objects for deriving an overall object state. The state management model provides the following features for entries in the Self Monitor table:

- The ability to define a severity for each self monitor entry that the agent uses to maintain a state value for each entry. The severities translate to CIM-compliant states.
- The ability to group one or more monitored resources into one or more managed objects through a tri-level naming scheme using object class, instance, and attribute values. This mechanism is flexible so that you can dynamically group resources from different functional areas.
- The ability to create a hierarchy of super managed objects, where the state of specific objects propagates to objects on higher levels of the hierarchy
- Notification through state change traps when the overall state of a managed object changes

The state management model is integrated into the CA Virtual Assurance object model. Each managed object in the AOM database possesses a CIM-compliant state, and SystemEDGE managed objects and states are reflected in CA Virtual Assurance though the SystemEDGE Platform Management Module. Managing SystemEDGE through CA Virtual Assurance lets you visualize managed object state information in the CA Virtual Assurance user interface and see state change traps through the CA Virtual Assurance Event view.

For more information about visualizing SystemEDGE managed objects in CA Virtual Assurance, see the CA Virtual Assurance documentation.

Further information about configuring self monitors to take advantage of the state management model, including attribute values and syntax examples, is available throughout the rest of this chapter.

Managed Object Creation

You control the creation of managed objects from the Self Monitor table using the object class, instance, and attribute values. By default, SystemEDGE creates a managed object for each unique class, instance, and attribute self monitor. If multiple self monitors contain the same value for these three properties, the agent groups them into a single managed object and derives an overall state for the object.

To illustrate the flexibility of how you can use this mechanism, consider the following scenarios:

- You define three self monitor entries with the same class, instance, and attribute values for monitoring CPU utilization. The monitors define a warning severity for CPU usage of greater than 85%, a major severity for CPU usage of greater than 90%, and a critical severity for CPU usage of greater than 95%. The agent combines these monitors into one managed object and maintains the worst current severity as the object state.
- You define three self monitor entries with the same class, instance, and attribute values for monitoring metrics from different functional areas that are connected in your enterprise. For example, you can combine a basic disk space self monitor, a response time monitor from the SRM AIM, and a filesystem monitor from the Remote Monitoring AIM, assigning a warning severity to each self monitor. In this scenario, the agent sends a warning state change trap only if all three self monitor thresholds (AND flag) are breached.
- You define a super aggregation of self monitor entries with the same value for object class and asterisk values for the object instance or attribute. This scenario groups large number of objects for each one. For example, you can aggregate all objects with a top level System class and 'Filesystem' instance, so that all file system self monitors propagate the worst current state to the top level objects.

  Note: You can also configure the agent to aggregate all objects at higher levels. For more information, see the chapter "Agent Configuration."

For syntax examples of these scenarios, see Self Monitoring Examples (see page 216).

If you do not define object class, instance, and attribute values, the SystemEDGE assigns default values for these properties using the sysedge.oid file located in the config directory of the agent installation. This file defines default classes and attributes for all attributes in every agent-supported MIB. It also defines default instances for each MIB attribute within a MIB table (see page 42). The default assigment ensures that the agent still creates objects for self monitors that are monitoring the same MIB attribute.

For more information about the default assignments for groups, tables, and attributes in the Systems Management Empire MIB, see the "Systems Management Empire MIB."

More Information

Object Status (see page 62)
Object Aggregation (see page 63)
Super Aggregation (see page 64)
How State Management and Aggregation Works (see page 66)

Aggregate State Table

The Aggregate State table does the work necessary to create managed object representations and derive overall object state. The Aggregate State table is a filtered view of SystemEDGE self monitors and process monitors. The table includes the aggregated state information calculated based on the managed object naming and state information supplied by each monitor entry.

The table creates an Aggregate State table entry for all self monitors and process monitors with the same object class, instance, and attribute value (or objects grouped together on a higher level). Self monitors and process monitors exist only as individual entries in the Self Monitor and Process Monitor tables. The Aggregate State table maintains the grouping of these monitors, continuously polls the tables, and sends state change traps when the managed object state changes.

Note: The Aggregate State table also sends traps when a new managed object is created or an existing managed object is deleted.

For more information about how the Aggregate State table maintains managed objects and their states, see the chapter "Concepts." For descriptions of the table columns, see Aggregate State Table.

Self Monitor Table

The Self Monitor table of the Systems Management Empire MIB provides information about each MIB attribute that the agent is currently monitoring. Each row represents a defined self monitor entry.

For each entry, the table provides the following types of information:

- Index
- Description
- Interval at which the agent polls the variable
- Sample type (absolute value or delta value)
- Object ID
- Current value that the agent is monitoring
- Threshold value
- Last sample call
- Number of traps that have been sent already
- Last trap

- Row status
- Minimum and maximum value sampled
- Action to take when a trap is sent
- Flags
- Superseded by index of a monitor
- Object class
- Instance
- Attribute
- Defined severity
- Current monitor status

Self Monitor Table Columns

The following list defines the columns of the Self Monitor table. For a complete description of the Self Monitor table and its fields, see the empire.asn1 file, which is located in the mib subdirectory of the agent installation.

monIndex

Defines the unique row index for this entry (11 to MAXINT). Rows 1 through 10 are reserved for the agent's internal use.

Permissions: Read-only

monDesc

Defines a description (0 through 512 characters in length) of the object being monitored.

Permissions: Read-write

Default: :Default Entry:

monInterval

Defines how often in seconds the agent polls (monitors) the variable. The value must be a minimum of 30 and a multiple of 30 seconds.

Permissions: Read-write

Default: 60

monSampleType

Defines whether the agent samples the object's absolute value (absoluteValue(1)) or takes the difference between successive samples (deltaValue(2)). For example, use deltaValue to monitor counter variables, because it describes the rate of change. Use absoluteValue to monitor gauges, because it describes the exact value of the object.

Permissions: Read-write

Default: absoluteValue(1)

monOID

Defines the complete object identifier that represents the monitored MIB object. The instance portion of the object-identifier (for example, .0 for scalars) is required. The object-instance must exist and must be contained within the SystemEDGE agent supported MIBs. Objects should be of integer type (including counter, gauge, timeticks).

Permissions: Read-write

monCurrVal

Specifies the value that was sampled last for the MIB variable being monitored. Every monInterval seconds, the agent updates this field to reflect the latest value of the variable.

Permissions: Read-only

monOperator

Defines the Boolean operator that is used to evaluate the expression *currval operator value*. The operator can have one of the following values:

- nop(1) (no operation; retrieve the value of the object, but do not evaluate the Boolean expression)
- gt(2) (greater than)
- lt(3) (less than)
- ge(4) (greater than or equal to)
- le(5) (less than or equal to)
- eq(6) (equal)
- ne(7) (not equal)

Permissions: Read-write

Default: nop(1)

monValue

Defines the integer value to which the agent compares using monOperator the current value of the monitored MIB variable during each monitoring cycle. If the comparison evaluates to true, the agent either changes the monitor state (for self monitors with a severity other than none) or the agent sends a threshold breach trap (for self monitors with a severity of none).

For example, to be notified when the value of a gauge exceeds 100, set 100 as the monValue.

Permissions: Read-write

monLastCall

Defines the time (based on sysUpTime) at which the agent last sampled the MIB variable that it is monitoring. 0 indicates that the MIB variable has not yet been sampled.

Permissions: Read-only

monNumTraps

Defines the number of traps that the agent has sent for this entry. This value tracks the monitorTrap, which the agent sends when a threshold breach occurs in a self monitor with a severity of none (legacy monitor). This value is useful for determining the frequency at which the exception condition is occurring for legacy monitors. It also provides a means to detect a missed trap message.

Permissions: Read-only

Note: For information about the number of state change traps that are sent for a stateful self monitor managed object, see the Aggregate State table.

monLastTrap

Defines the time (based on sysUpTime) at which the agent last sent a trap for this entry. This value tracks the monitorTrap, which the agent sends when a threshold breach occurs in a self monitor with a severity of none.

Permissions: Read-only

Note: For information about the last state change trap that is sent for a stateful self monitor's managed object, see the Aggregate State table.

monRowStatus

Defines the row status, which can be one of the following:

- active
- notInService
- notReady
- createAndGo
- createAndWait
- destroy(6)

Typically, a row is either active or notInService. These values are identical in meaning to those defined by the SNMPv2 SMI RowStatus textual convention.

Permissions: Read-write

Default: createAndWait(5)

monMinValue

Defines the lowest (minimum) value that the agent has observed since it began polling the MIB variable.

Permissions: Read-only

monMaxValue

Specifies the highest (maximum) value that the agent has observed since it began polling the MIB variable.

Permissions: Read-only

monAction

Defines a quoted command (0 to 2048 characters in length) with any parameters to run when the expression evaluates to True and the agent sends a threshold breach trap (legacy monitors) or state change trap (stateful monitors). For stateful self monitors, the command does not run if a threshold breach does not result in an object state change. If the string is empty, the agent performs no action for this entry.

Note: Do not use Windows batch files for actions; they impose severe programmatic limitations and often do not work correctly with desktop applications. Instead, use a more powerful and flexible scripting language, such as Perl or Visual Basic.

Permissions: Read-write

monFlags

Defines the integer flags that indicate additional behavioral semantics that the entry follows during the course of its operation. For more information about available flags and setting flags, see Monitor Table Flags (see page 205).

Permissions: Read-write

monSupersededBy

Specifies the index of the monitor that supersedes this entry. A superseding monitor is evaluated before this table entry, and if the superseding monitor's threshold is crossed, this entry will no more be evaluated. With such a sequence of correlated monitors, useless sequences of traps can be omitted. A value of zero (0) indicates that no monitor entry correlation is done.

Note: The usage of the object state model obsoletes (and disables) this functionality.

Permissions: Read-write

monObjClass

Defines the object class of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value. A class name can be used to specify a monitored object table. If SystemEDGE does not determine an existing table with the specified name, SystemEDGE creates a class. The class specifies the monitoring object identifier (SNMP OID) and can be referenced for aggregation purposes.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

monObjInstance

Defines the object instance of the monitored MIB object for use with the object state management model. If a value is not configured and the agent cannot determine a default, the value is 'unknown'. The default for a nontabular object is 'null'. For seamless monitoring of remote objects, instance names start with a prefix //hostname/.

For local objects, the agent adds the prefix //./ to each instance name (including 'null'). This attribute cannot be set to an empty string or to '*'. Any such requests are ignored. If the instance name cannot be matched at the time of parsing to a corresponding instance, SystemEDGE creates a reference to this monitoring entry's OID with the specified instance name.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

monObjAttribute

Defines the object instance of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value for objects that corresponds to the MIB attribute name. If the attribute name is not known during parsing of the monitoring entry, SystemEDGE creates an attribute with the specified name.

Permissions: Read-write

Default: ASCII MIB attribute name of the monitored OID

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

monSeverity

Defines the severity to use for the object state management model. This value defines the state to assign to the entry when a threshold breach occurs. Valid values are as follows:

- none(1)
- ok(2)
- warning(3)
- minor(4)
- major(5)
- critical(6)
- fatal(7)

A value of none creates a legacy monitor, which excludes the entry from the object state management model.

Permissions: Read-write

Default: none

monCurrState

Defines the current derived state of the entry, evaluated using the severity and the current breach condition and respecting any lag value defined in the monFlags property. Valid values are as follows:

- unknown(1)
- ok(2)
- warning(3)
- minor(4)
- major(5)
- critical(6)
- fatal(7)
- up(11)
- down(12)

The up and down values only appear for legacy monitors (with a severity of none). A state change in this column results in a state change trap only if the state is the worst of all monitors for the corresponding managed object in the Aggregate State table.

Permissions: Read-only

Sample Self Monitor Table Entry

This section provides an example of an entry in the Self Monitor table that instructs the SystemEDGE agent to monitor the 1-minute load average on the target system (table separated for viewability):

Index	Desc	Interval	Sample Type	OID	CurrVal
21	"1min load avg"	60	absolute Value	loadAvg 1Min.0	320
Oper	**Value**	**Last Call**	**Num Traps**	**Last Trap**	**Row Status**
gt	300	0d 22:40:00	0	0	active
Min Value	**Max Value**	**Action**	**Flags**	**Superseded By**	**Obj Class**
1	3	""	0x0	0	System
Obj Instance		**Obj Attribute**		**Severity**	**Curr State**
Load		LoadAverage1Min		warning	warning

This entry provides the following information:

- This is the 21st row in the table.
- Its purpose is to monitor the system's 1-minute load average.
- Every 60 seconds, the agent checks the absolute value of the loadAvg1Min MIB variable (whose current value is 320) to see if it is greater than 300 (the specified threshold value). If it is, the agent changes the current state to warning (defined by the Severity value) and sends a state change trap if this monitor represents the worst state for the corresponding managed object in the Aggregate State table.
- With each poll, the agent updates the CurrVal column with the value it has just retrieved, records the time in the LastCall column, and updates the MinValue and MaxValue columns if appropriate (that is, if the value just polled is the lowest or highest value observed thus far).
- The RowStatus column shows that this entry is active. In this example, the action is null, so no command will be run when the trap is sent.

- The Flags column is set to zero, which indicates the default Self Monitor table behavior.
- The class, instance, and attribute values establish this self monitor entry as a part of the managed object with these values in the Aggregate State table. You can define other self monitor entries with these same values to monitor for different severities, or you can aggregate this entry with other related objects.

Row Creation

The monIndex column of the Self Monitor table acts as a key field (or row index) to distinguish rows in the table. Rows 1 through 10 are reserved for internal use by the SystemEDGE agent. You can configure rows in the range 11 to MAXINT.

You can use the following MIB objects with the Self Monitor table to optimize row creation:

monUnusedIndex

Returns an unused index number for the Self Monitor table when you perform an SNMP Get on the object. Query this object to obtain a number that you can use for row creation.

monMatchDescr

Determines the index number that corresponds to a particular entry description in the monDescr attribute. Perform an SNMP Set on this MIB object with a monDescr attribute value to cause the agent to search through entries in the Self Monitor table and put the index value of the last matching entry in the monMatchIndex MIB object.

monMatchIndex

Matches a particular entry description with its index number when used with monMatchDescr.

You may choose, as a matter of local policy, to reserve a block of rows for system administration. You can then define entries within a reserved block of rows without being concerned that the row may already be taken by another user's entry. In compliance with the local policy, all other users should use row indices outside the reserved range when they define user-configured entries.

By reserving a block of rows, you can define a consistent set of conditions (row entries) to be monitored across all computers such that the same condition is defined in the same row number on each computer. For example, you can use row 31 (monIndex = 31) to define an entry for monitoring the swapCapacity variable, and you can distribute this configuration to every system so that row 31 monitors the swapCapacity variable on every system. The indexes 11 - 16 are used for predefined monitors. The Remote Monitoring AIM uses indexes greater than 8000.

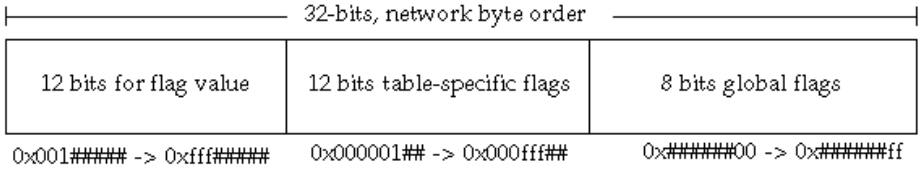
Configuring the agent with CA Virtual Assurance can automate this process. You can configure a block of entries with specific row numbers and deploy that configuration to all managed systems in one operation.

Self Monitor Table Flags

The monFlags column in the Self Monitor table is a 32-bit unsigned integer that can specify additional behavioral semantics for the corresponding Self Monitor table row. By default, the Self Monitor table row does the following:

- Monitors system resources
- Evaluates status
- Sends SNMP traps (legacy monitors only)
- Logs traps to the sysedge.log file
- Invokes actions (if they are configured)

You can set flag bits to alter these defaults. The SystemEDGE agent interprets all flags in hexadecimal (base 16) notation. The following illustration shows the composition of the Self Monitor Table flags field (monFlags).



The flags value consists of three fields:

- Field 1 contains common table flags defined for the monitoring tables of the Systems Management Empire MIB. This portion is the low-order 8 bits of the flag.
- Field 2 contains table-specific flags defined separately for each of the monitoring tables. This field defines the next 12 low-order bits after the common table flags.
- Field 3 contains 12 reserved high-order bits for an integer value for use with table-specific flags.

The following sections explain each flag bit. You can combine these flag values through a logical OR operation.

The following list describes the Self Monitor Table flags:

0x00000001

Disables execution of actions for this entry.

0x00000002

Disables sending of SNMP traps for this entry.

Note: The Self Monitor table only controls sending traps for legacy monitors. For stateful self monitors, state change traps may still occur from the Aggregate State table.

0x00000004

Disables attempts to reinitialize this entry. By default, the agent periodically tries to reinitialize this entry by attempting to query the MIB object it is monitoring. Reinit occurs if query failed

0x00000008

Disables logging of traps for this entry in the sysedge.log file. Setting this bit does not affect sending trap. Disabling event logging is useful when events occur frequently or when a particular entry is used as an agent heartbeat.

0x00000010

Sends continuous monitorEntryNotReady traps for this entry each time the agent attempts to reinitialize monitoring and fails to query the MIB object. The agent's default behavior is to send a single monitorEntryNotReady trap when a MIB object it is monitoring ceases to exist, even when succeeding attempts to reinitialize the entry still fail. Enabling this feature causes the agent to send an additional monitorEntryNotReady trap each time reinitialization fails.

0x00000020

Disables the passing of default arguments to action scripts or programs. SystemEDGE typically passes default action parameters that indicate the trap type, description field, and so on. For more information about action parameters, see Self Monitor Table Action Parameters (see page 208).

0x00000040

Disables sending of notReady traps for this entry.This includes to not log and to not execute actions for notReady trap.

0x00000100

Sends a monitorClear trap for this entry when the self monitor expression transitions from True to False. This includes to log and to execute actions for the monitorClear trap.

0x*XXX*00200

Sends monitor threshold traps or changes the current state only on the *X*th consecutive breach. After the *X*th breach occurs, the agent sends monitor traps for each subsequent True expression evaluation. If the threshold expression transitions from True to False, the agent begins counting subsequent breaches from zero. This flag also applies to action execution and logging. You can specify the value of *X* through the flag value field. For an example of this behavior, see Self Monitoring Examples (see page 216).

0x*XXX*00400

Sends up to *X* consecutive monitor traps, and then sends no more. Enabling this feature puts an upper boundary on the number of consecutive monitor traps and action executions that can occur when a threshold has been exceeded. After the threshold expression transitions from True to False, the agent begins counting subsequent breaches from zero. This flag also applies to action execution and logging. You can specify the value of *X* through the flag value field. For an example of this behavior, see Self Monitoring Examples (see page 216).

0x00000800

Aggregates the state of monitors for the same object information (class, instance, attribute) and with the same severity so that they have an AND relation. All connected monitors with the same severity that have this flag set must have breach conditions for the aggregate state to change accordingly.

0x###00000

Several flag bits use a value *X* for sending traps and executing actions and log. The value *X* is specified as the high-order 12 bits of the flag field. Flag bits utilizing this field are mutually exclusive.

Self Monitor Table Action Parameters

The SystemEDGE agent provides several default parameters to action commands when they are invoked. These parameters are in addition to any parameters you specify in the action string, and they are passed on the command line after those that you specify. The default action parameters are the same as the parameters provided in the SNMP traps that the agent sends for the Self Monitor table.

Note: You can prevent the agent from including these action parameters in the command by setting the 0x00000020 flag in the table entry.

The following list describes the default parameters for Self Monitor table actions:

trapType

Defines the type of trap being sent, such as ntEventMonNotReadyTrap or ntEventMonMatchTrap.

monDescr

Defines the description from the table entry.

monOID

Defines the OID from the table entry.

monCurrVal

Defines the current value from the table entry.

monValue

Defines the value being monitored from the table entry.

monRowStatus

Defines the current row status for the table entry.

monOperator

Defines the operator being used from the table entry. The value passed to the action script is the numeric representation of the actual operator. For example, if the operator is >, the value passed to the action script is 2.

monIndex

Defines the index from the table entry.

monFlags

Defines the flags associated with the table entry. The value passed to the action script is in base 16 (hexadecimal) notation with a leading 0x to indicate that it is a hexadecimal number (for example, 0x00000020).

monObjClass

Defines the object class of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value. A class name can be used to specify a monitored object table. If SystemEDGE does not determine an existing table with the specified name, SystemEDGE creates a class. The class specifies the monitoring object identifier (SNMP OID) and can be referenced for aggregation purposes.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

monObjInstance

Defines the object instance of the monitored MIB object for use with the object state management model. If a value is not configured and the agent cannot determine a default, the value is 'unknown'. The default for a nontabular object is 'null'. For seamless monitoring of remote objects, instance names start with a prefix //hostname/.

For local objects, the agent adds the prefix //./ to each instance name (including 'null'). This attribute cannot be set to an empty string or to '*'. Any such requests are ignored. If the instance name cannot be matched at the time of parsing to a corresponding instance, SystemEDGE creates a reference to this monitoring entry's OID with the specified instance name.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

monObjAttribute

Defines the object instance of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value for objects that corresponds to the MIB attribute name. If the attribute name is not known during parsing of the monitoring entry, SystemEDGE creates an attribute with the specified name.

Permissions: Read-write

Default: ASCII MIB attribute name of the monitored OID

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

monCurrState

Defines the current state of the table entry.

View the Self Monitor Table

You can use CA Virtual Assurance to view the contents of a managed agent's Self Monitor table.

To view the Self Monitor table in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, and click Self Monitors tab.

   The Self Monitor table appears with all existing self monitors on the selected server.

Self Monitoring Configuration

You can control which MIB variables and conditions the SystemEDGE agent monitors by adding, deleting, or modifying the entries in the Self Monitor table.

You can configure the Self Monitor table in the following ways:

Dynamically using file-based configuration from CA Virtual Assurance

CA Virtual Assurance provides the ability to configure agents installed on any system that it manages from the CA Virtual Assurance user interface. This is the recommended method of configuration, because it provides the ability to dynamically define and deliver monitors with a user-friendly and secure mechanism that does not require an agent restart. You can configure monitoring for an individual agent or define monitoring policies to apply on multiple managed systems. When you make an individual configuration change or apply a configuration policy, CA Virtual Assurance delivers a new configuration file to the agent system. CA Virtual Assurance can serve as the central point of agent configuration across your data center, and you can overwrite or deny changes made through other methods.

For more information about configuring the agent using CA Virtual Assurance, see the CA Virtual Assurance documentation.

Dynamically using SNMP Set commands from a management system, such as eHealth AdvantEDGE View, CA Spectrum, or CA NSM

Various management systems support adding monitors to SystemEDGE through SNMP Set commands. eHealth provides this functionality through the AdvantEDGE View interface, CA NSM enables configuration through an Agent View, and CA Spectrum enables configuration through OneClick. All features provided by dynamic configuration through CA Virtual Assurance are not available through these methods.

For more information about SNMP Set configuration, see the documentation for the appropriate management system.

Manually by specifying the entries for the Self Monitor table in the local sysedge.cf file

Manual configuration requires you to add entries to the local sysedge.cf file using specific directives. When you configure the agent directly from the sysedge.cf file, a full restart is required for changes to take effect. Use the sysedge.cf file located in the agent data directory for direct configuration.

This section describes how to configure self monitoring directly in the local sysedge.cf file, but it describes the properties that are required for all forms of configuration.

monitor Directive--Add Entries to the Self Monitor Table

The monitor directive lets you add self monitor entries to the Self Monitor table directly in the sysedge.cf file. The arguments represent columns in the Self Monitor table.

Add a line to the sysedge.cf file in the agent data directory using the syntax described below, save the file, and restart the agent for the change to take effect.

Use the monitor directive to add entries to the Self Monitor table as follows:

(1) monitor oid *oid index flags interval stype oper thresh* ['*descr*' ['*action*' [*supIndex*]]]

(2) monitor oid *oid index flags interval stype oper thresh* ['*descr*' ['*action*' ['*objClass*' '*objInst*' '*objAttr*' *severity*]]]

(3) monitor *table inst attr index flags interval stype oper thresh* ['*descr*' ['*action*' [*supIndex*]]]

(4) monitor *table inst attr index flags interval stype oper thresh* ['*descr*' ['*action*' ['*objClass*' '*objInst*' '*objAttr*' *severity*]]]

oid

Specifies the OID to monitor. You can specify the OID using the complete dotted-decimal value (for example, 1.3.6.1.2.1.25.1.5.0) or the symbolic MIB name (for example, hrSystemNumUsers.0). Either way, you must specify the object instance, which is typically zero for non-tabular MIB variables or corresponds to the specific instance of an attribute that belongs to a MIB table.

Examples:

- 1.3.6.2.1.25.1.5.0
- hrSystemNumUsers.0

table* '*instance*' *attribute

Specifies the tableEntry, instance, and attribute name of the object to monitor. If the attribute has no instance, enter '' for the instance property.

Examples:

- processGroupMonEntry 'iexplore|IEXPLORE' pgmonRSS
- devTableEntry 'C:' devCapacity

For more information, see Reverse OID Lookup (see page 215).

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For more information about available flags, see Self Monitor Table Flags (see page 205).

interval

Specifies how often in seconds the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. The value must be a minimum of 30 and a multiple of 30 seconds.

stype

Specifies whether the agent should sample the attributes's absolute value (absoluteValue) or take the difference between successive samples (deltaValue). Use deltaValue to monitor counter variables for rate of change, and use absoluteValue to monitor gauges for exact value.

oper

Specifies the Boolean operator for evaluating the current entry value against the provided threshold value.

Valid values for the operator are as follows:

- nop (no operation)
- > (greater than)
- < (less than)
- >= (greater than or equal to)
- <= (less than or equal to)
- = (equal)
- != (not equal)

Default: nop

thresh

Specifies the integer threshold value to which the current value of the monitored attribute is compared using the specified operator.

'***descr***'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'*action*'

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the threshold condition is met and the agent sends a threshold breach trap (legacy monitors) or state change trap (stateful monitors). For stateful self monitors, the command does not run if a threshold breach does not result in an object state change. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings when the agent performs actions. For more information, see the chapter "Agent Configuration."

supIndex

Specifies the table index that acts as a parent or overriding partner to this entry. Valid values are existing or future SystemEDGE monitor indexes (a positive integer value). The default is zero (0), indicating default behavior.

Note that usage of the object state model (entering a severity other than none) obsoletes and disables this functionality.

'*objClass*'

Defines the object class of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value. A class name can be used to specify a monitored object table. If SystemEDGE does not determine an existing table with the specified name, SystemEDGE creates a class. The class specifies the monitoring object identifier (SNMP OID) and can be referenced for aggregation purposes.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

'*objInst*'

Specifies the object instance of the monitored object for use with the state management model. Define a value for this property (with objClass and objAttr) for creating a managed object that aggregates all entries with the same values. The agent uses the sysedge.oid file to assign a default value for tabular objects based on a specific attribute value and assigns null as the default for non-tabular objects.

'*objAttr*'

Specifies the object attribute for use with the state management model. Define a value for this property (with objClass and objInst) for creating a manged object that aggregates all entries with the same values. The agent uses the sysedge.oid file to assign a default value that corresponds to the MIB attribute name.

Default: MIB attribute name

severity

Specifies the severity to use for the object state management model. This value defines the state to assign to the entry when a threshold breach occurs. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

A value of none creates a legacy monitor, which excludes the entry from the object state management model.

Default: none

Reverse OID Lookup

The sysedge.oid file located in the config directory of the agent installation maintains default values for class, instance, and attribute properties of every supported SystemEDGE MIB attribute. The file uses the following conventions to assign default class, instance, and attribute values:

Class

Uses the MIB group or table name to which the attribute belongs.

Examples:

- kernelConfig
- userEntry

Instance

Uses the value 'null' for attributes that do not belong to a MIB table. For tabular objects, the agent derives the default instance value based on the value of a specific attribute or attributes from the MIB table.

Examples:

- value of devMntPt attribute (Device table)
- value of processName attribute (Process table)

Attribute

Uses the MIB attribute name.

Examples:

- devDevice
- maxProcs

For examples of a reverse OID lookup, see Self Monitoring Examples (see page 216).

Self Monitoring Examples

This section provides sample entries for the Self Monitor table to monitor thresholds on the target system. Each example shows how to define the entry and describes the condition being monitored. You can add these entries to sysedge.cf.

Example: Monitor the One Minute Load Average

The following examples configure the agent to monitor the system's one minute load average:

```
monitor oid 1.3.6.1.4.1.546.1.1.7.8.26.0 11 0x00 60 absolute > 300 'Monitor 1 minute load average' '' ‘kernelperf’ ‘null’ ‘loadAverage1Min’ warning
```

1.3.6.1.4.1.546.1.1.7.8.26.0

Corresponds to the OID for the loadAverage1Minute variable contained within the Systems Management Empire MIB

11

Indicates that this entry will occupy row 11 (monIndex=11) in the Monitor table.

60

Specifies that the load average should be sampled once every 60 seconds.

absolute

Indicates that the agent should use the object's value, not the difference between successive samples.

300

Indicates the value against which the current load average is compared. If the currently sampled value is greater than (>) 300, an event occurs.

Note: The agent returns load averages as the underlying system's load average multiplied by 100. For example, a load average of 3 is returned as 300.

kernelperf

Indicates that this entry uses kernelperf object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

null

Indicates that this entry uses null object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

loadAverage1Min

Indicates that this entry uses loadAverage1Min object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

warning

Indicates that this entry uses warning severity for the object state model.

Example: Monitor the Five Minute Load Average

The following example configures the agent to monitor the system's five minute load average:

```
monitor oid loadAverage5Min.0 12 0x00500300 300 absolute > 200 'Monitor 5 minute load average' '' ‘kernelperf’ ‘null’ ‘loadAverage5min’ fatal
```

loadAverage5Min.0

Corresponds to the OID for the loadAverage5Minute variable contained within the Systems Management Empire MIBis entry will occupy row 12 (monIndex=12) in the Monitor table.

0x00500300

- Configures the agent to send monitorClear traps when the threshold expression transitions from True to False.
- Configures the agent to begin sending traps (and run actions if they are configured) only at the *X*th consecutive occurrence of the event.

Specifies that *X*=5. Therefore, the agent begins to send traps at the fifth occurrence of the event.

300

Specifies that the load average should be sampled once every 300 seconds.

absolute

Indicates that the agent should use the object's value, not the difference between successive samples.

200

Indicates the value against which the current load average is compared. If the currently sampled value is greater than (>) 200, the agent sends a trap to all configured managers.

''

Indicates that no action is specified.

kernelperf

Indicates that this entry uses kernelperf object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

null

Indicates that this entry uses null object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

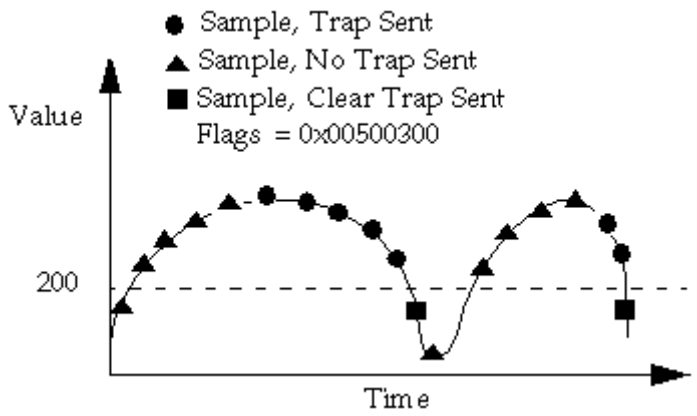
loadAverage5Min

Indicates that this entry uses loadAverage5Min object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

fatal

Indicates that this entry uses fatal severity for the object state model.

The following illustration shows how the agent would send and clear traps based on this monitor directive:



Example: Monitor the Fifteen Minute Load Average

The following example configures the agent to monitor the system's fifteen minute load average:

```
monitor oid loadAverage15Min.0 13 0x0 900 absolute > 200 'Monitor 15 minute load
average' '' ‘’ ‘kernelperf’ ‘null’ ‘loadAverage15Min’ critical
```

loadAverage15Min.0

Corresponds to the OID for the loadAverage15Min variable contained within the Systems Management Empire MIB entry will occupy row 13 (monIndex=13) in the Monitor table.

900

Specifies that the load average should be sampled once every 900 seconds.

absolute

Indicates that the agent should use the object's value, not the difference between successive samples.

200

Indicates the value against which the current load average is compared. If the currently sampled value is greater than (>) 200, the agent sends a trap to all configured managers.

kernelperf

Indicates that this entry uses kernelperf object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

null

Indicates that this entry uses null object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

loadAverage15Min

Indicates that this entry uses loadAverage15Min object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

critical

Indicates that this entry uses critical severity for the object state model.

Example: Monitor the System's Interrupt Rate

The following example configures the agent to monitor the rate at which hardware interrupts are occurring on the local system:

```
monitor oid numInterrupts.0 1003 0x00500400 60 delta > 1000 'Monitor Interrupt Rate'
'' 'kernelperf' 'null' 'numInterrupts' warning
```

numInterrupts.0

Corresponds to the OID for the numInterrupts counter object contained within the Systems Management Empire MIB entry is index 14 in the Monitor table.

0x00500400

- Configures the agent to send a maximum of *X* consecutive traps when this monitor expression evaluates to True.
- Specifies that *X*=5. Therefore, the agent sends 5 consecutive monitorEvent traps, and then stops sending traps until the entry resets itself. The entry resets itself when the expression transitions from True to False.

Does not specify that the agent should send monitorClear traps; consequently, a monitorClear trap is not sent when the expression transitions from True to False.

60

Indicates that the interrupt rate should be sampled every 60 seconds.

delta

Tells the agent to measure the rate at which the number of interrupts has changed. Because this object is a counter, delta is an appropriate sample type.

1000

Indicates the value against which the current number of interrupts is compared.

kernelperf

Indicates that this entry uses kernelperf object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

null

Indicates that this entry uses null object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

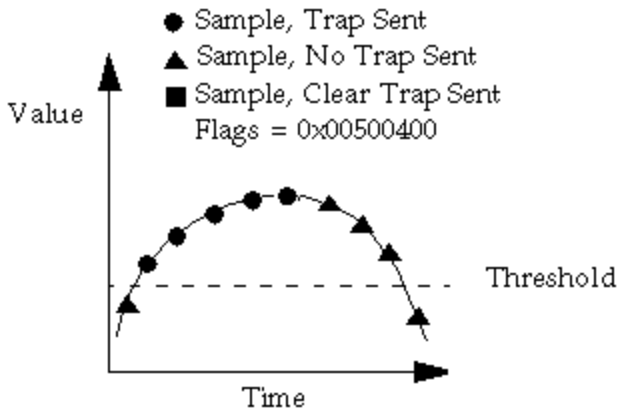
numInterrupts

Indicates that this entry uses numInterrupts object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

warning

Indicates that this entry uses warning severity for the object state model.

The following illustration shows how the agent would send and clear traps based on this monitor directive.



Example: Monitor the System's Page Fault Rate

The following example configures the agent to monitor the rate at which hardware page interrupts are occurring on the local system:

```
monitor oid numPageFaults.0 15 0x00500500 60 delta > 1000 'Monitor Page-fault Rate'
'' ‘kernelperf’ ‘null’ ‘numPageFaults’ major
```

numPageFaults.0

Corresponds to the OID for the numPageFaults counter object contained within the Systems Management Empire MIB entry is index 15 in the Monitor table.

0x00500500

- Configures the agent to send monitorClear Traps when the expression transitions from True to False.
- Specifies that the agent should send at most *X* consecutive monitorEvent traps, and then send no more until the entry resets itself.
- Specifies that *X=5*. Therefore, the agent sends 5 consecutive monitorEvent traps, and then stops sending traps until the entry resets itself.

60

Indicates that the interrupt rate should be sampled every 60 seconds.

delta

Tells the agent to measure the rate at which the number of interrupts has changed. Because this object is a counter, delta is an appropriate value for this entry's sample type.

1000

Indicates the value against which the current number of interrupts is compared.

kernelperf

Indicates that this entry uses kernelperf object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

null

Indicates that this entry uses null object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

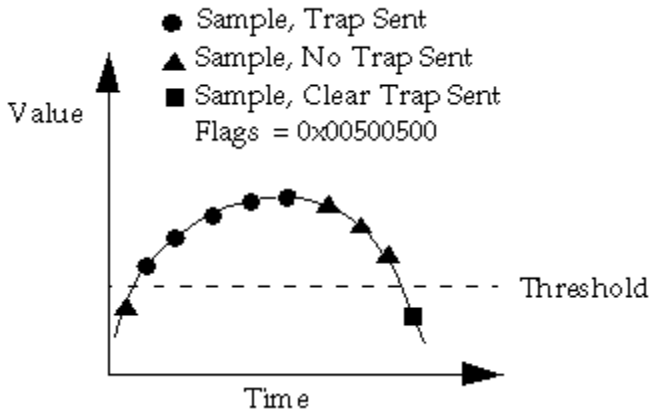
numPageFaults

Indicates that this entry uses numPageFaults object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

major

Indicates that this entry uses major severity for the object state model.

The following illustration shows how the agent would send and clear traps based on this monitor directive:



Example: Monitor Number of Incoming Packets on the Interface

The following example configures the agent to monitor the rate at which packets are received by the first ethernet interface (which is le0 for Sun systems):

```
monitor oid ifInUcastPkts.2 1004 0x0 60 delta > 1000 'Monitor le0 Incoming Packets'
'' 'ifEntry' 'WAN MiniPort' 'ifInUcastPkts' minor
```

ifInUcastPkts.2

Indicates the particular MIB object-instance to sample.

16

Indicates that the entry is index 16 in the Monitor Table.

60

Indicates that the agent should calculate the rate every 60 seconds.

delta

Tells the agent to measure the rate at which the number of incoming packets (ifInUcastPkts) is changing.

1000

Specifies the value to use in the comparison. If the change in rate is greater than (>) 1000, the agent sends a trap.

ifEntry

Indicates that this entry uses ifEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

WAN MiniPort

Indicates that this entry uses WAN MiniPort object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

ifInUcastPkts

Indicates that this entry uses ifInUcastPkts object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

minor

Indicates that this entry uses minor severity for the object state model.

Example: Monitor Number of Outgoing Packets on the Interface

The following example configures the agent to monitor the rate at which packets are transmitted by the first ethernet interface (which is le0 for Sun systems):

```
monitor oid ifOutUcastPkts.2 1005 0x0 60 delta > 1000 'Monitor le0 Outgoing Packets'
'' 'ifEntry' 'WAN MiniPort' 'ifOutUcastPkts ' ok
```

ifOutUcastPkts.2

Indicates the particular MIB object-instance to sample.

60

Indicates that the agent should calculate the rate every 60 seconds.

delta

Indicates the sample type because the object being monitored is a MIB-II ifEntry counter.

1000

Specifies the value to use in the comparison. If the change in rate is greater than (>) 1000, the agent sends a trap message to all configured managers.

ifEntry

Indicates that this entry uses ifEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

WAN MiniPort

Indicates that this entry uses WAN MiniPort object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

ifOutUcastPkts

Indicates that this entry uses ifOutUcastPkts object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

ok

Indicates that this entry uses ok severity for the object state model.

Example: Monitor Number of SNMP Packets Received

The following example configures the agent to monitor the rate at which the agent receives SNMP requests:

```
monitor oid snmpInPkts.0 18 0x0 30 delta > 4000 'Monitor SNMP Packets' '' ‘snmp’ ‘null’ ‘snmpInPkts’ minor
```

snmpInPkts.0

Indicates the MIB II object instance to sample.

18

Specifies that the entry is index 18 in the Monitor table.

30

Indicates that the agent should calculate the rate every 30 seconds.

delta

Indicates the sample type because the object is a counter.

4000

Specifies the value to use in the comparison. If the change in rate is greater than (>) 4000, the agent sends a trap message to all configured managers.

snmp

Indicates that this entry uses snmp object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

null

Indicates that this entry uses null object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

snmpInPkts

Indicates that this entry uses snmpInPkts object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

minor

Indicates that this entry uses minor severity for the object state model.

Example: Monitor Space on the Root File System

The following example configures the agent to monitor the root (/) file system and to send a trap message when it becomes more than 95% full:

```
monitor filesystem / devCapacity 19 0x0 120 absolute > 95 'Monitor / Filesystem' ''
'devTableEntry' '/' 'devCapacity' minor
```

devCapacity

Indicates the particular MIB object-instance to monitor; in this case, the object instance is devTableEntry.devCapacity from the Systems Management Empire MIB devTable. The object instance is not specified because it is determined automatically based on the name of the file system.

19

Indicates that this entry is row 19 in the Monitor table.

120

Indicates that the agent should sample every 120 seconds.

absolute

Indicates the appropriate sample type because the agent is sampling an integer (not counter) value that represents how full the file system is.

95

Specifies the value to use in the comparison. If the file system becomes greater than (>) 95% full, the agent sends a trap message to all configured managers.

devTableEntry

Indicates that this entry uses devTableEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

'/'

Indicates that this entry uses '/' object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

devCapacity

Indicates that this entry uses devCapacity object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

minor

Indicates that this entry uses minor severity for the object state model.

Example: Monitor Space on the /usr File System

The following example configures the agent to monitor the /usr file system and to send a trap message when it becomes more than 95% full:

```
monitor filesystem /usr devCapacity 20 0x00100500 120 absolute > 95 'Monitor /usr
Filesystem' '' ‘devTableEntry’ ‘/usr’ ‘devCapacity’ fatal
```

devCapacity

Indicates the particular MIB object-instance to monitor; in this case, the object instance is devTableEntry.devCapacity from the Systems Management Empire MIB devTable. The object instance is not specified because it is determined automatically based on the name of the file system.

20

Indicates that this entry is row 20 of the Monitor table.

0x00100500

- Configures the agent to send monitorClear traps when the expression transitions from True to False.
- Specifies that the agent should send at most *X* consecutive monitorEvent traps and then send no more until the entry resets itself.
- Specifies *X*=1, which indicates that the agent should send only one monitorEvent trap before waiting for the entry to reset itself.

120

Indicates that the agent should sample every 120 seconds.

absolute

Indicates the appropriate sample type because the agent is sampling an integer value that represents how full the file system is.

95

Specifies the value to use in the comparison. If the file system becomes greater than (>) 95% full, the agent sends a trap message to all configured managers.

devTableEntry

Indicates that this entry uses devTableEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

/usr

Indicates that this entry uses /usr object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

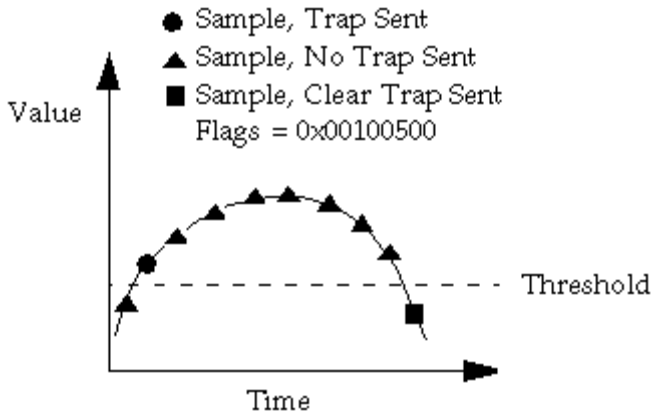
devCapacity

Indicates that this entry uses devCapacity object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

fatal

Indicates that this entry uses fatal severity for the object state model.

The following illustration shows how the agent would send and clear traps based on this monitor directive:



Example: Monitor the Number of Processes

The following example configures the agent to monitor the number of processes currently executing on the system and to send a trap when that number is greater than 120:

```
monitor oid hrSystemProcesses.0 21 0x0 60 absolute >= 120 'Monitor Number of Processes'
'' 'hrSystem' 'null' 'hrSystemProcesses' warning
```

hrSystemProcesses

Indicates the variable to be monitored.

21

Indicates that this entry will be index 21 in the Monitor table.

60

Indicates that the agent should sample the number of processes every 60 seconds.

absolute

Indicates the sample type because the object is a gauge.

>=

Instructs the agent to send a trap whenever the number of processes is greater than or equal to 120.

hrSystem

Indicates that this entry uses hrSystem object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

null

Indicates that this entry uses null object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

hrSystemProcesses

Indicates that this entry uses hrSystemProcesses object attribute for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

warning

Indicates that this entry uses warning severity for the object state model.

edgemon Commands for Self Monitoring

The edgemon command and associated arguments are as follows:

`oid` *`objectOID index flags interval sampleType oper value "descr" "action"`* [*`supersededBy | objClass objInst objAttr severity`*]

`filesystem|interface|cpu|disk` *`name variable-name index flags interval sampleType oper value "descr" "action"`* [*`supersededBy | objClass objInst objAttr severity`*]

`setstatus index status`

`delete` *`index`*

`list`

List of arguments deliminted by | means the list on the left side or the list on the right side could be used.

objectOID

Specifies the OID to monitor. You can specify the OID using the complete dotted-decimal value (for example, 1.3.6.1.2.1.25.1.5.0) or the symbolic MIB name (for example, hrSystemNumUsers.0). Either way, you must specify the object instance, which is typically zero for non-tabular MIB variables and corresponds to the specific instance of an attribute that belongs to a MIB table.

Examples:

- 1.3.6.2.1.25.1.5.0
- hrSystemNumUsers.0

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For more information about available flags, see Self Monitor Table Flags (see page 205).

interval

Specifies how often in seconds the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. The value must be a minimum of 30 and a multiple of 30 seconds.

sampleType

Specifies a sample type of either absolute or delta. This value indicates whether the agent should sample the object's absolute value or take the difference between successive samples.

oper

Specifies the Boolean operator for evaluating the current entry value against the provided threshold value.

Valid values for the operator are as follows:

- nop (no operation)
- > (greater than)
- < (less than)
- >= (greater than or equal to)
- <= (less than or equal to)
- = (equal)
- != (not equal)

Default: nop

value

Specifies the integer threshold value to which the current value of the monitored attribute is compared using the specified operator.

"descr"

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

"action"

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the threshold condition is met and the agent sends a threshold breach trap (legacy monitors) or state change trap (stateful monitors). For stateful self monitors, the command does not run if a threshold breach does not result in an object state change. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

supersededBy

Specifies the table index that acts as a parent or overriding partner to this entry. Valid values are existing or future SystemEDGE monitor indexes (a positive integer value). The default is zero (0), indicating default behavior.

Note that usage of the object state model (entering a severity other than none) obsoletes and disables this functionality.

objClass

Specifies the object class of the monitored object for use with the state management model. Define a value for this property (along with objInst and objAttr) for creating a managed object that aggregates all entries with the same values. The agent uses the sysedge.oid file to assign a default value.

Default: MIB group or table name (see sysedge.oid)

objInst

Specifies the object instance of the monitored object for use with the state management model. Define a value for this property (along with objClass and objAttr) for creating a managed object that aggregates all entries with the same values. The agent uses the sysedge.oid file to assign a default value for tabular objects based on a specific attribute value and assigns null as the default for non-tabular objects.

objAttr

Specifies the object attribute for use with the state management model. Define a value for this property (along with objClass and objInst) for creating a manged object that aggregates all entries with the same values. The agent uses the sysedge.oid file to assign a default value that corresponds to the MIB attribute name.

Default: MIB attribute name

severity

Specifies the severity to use for the object state management model. This value defines the state to assign to the entry when a threshold breach occurs. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

A value of none creates a legacy monitor, which excludes the entry from the object state management model.

Default: none

name

Specifies the name of the file system, network interface, CPU, or disk to monitor, depending on type of command you entered (filesystem, interface, cpu, or disk). For example, C: for filesystem.

variable-name

Specifies the variable from the Systems Management Empire MIB table to monitor corresponding to the command you entered (filesystem, interface, cpu, or disk). For example, you can monitor a file system's capacity by specifying devCapacity as the variable-name.

status

Specifies the entry's status, which can be one of the following:

- active (activate a row)
- notInService (deactivate but preserve a row)
- destroy (delete a Self Monitor table row)

edgemon Examples

This section provides examples of how to use the edgemon command with SNMP versions 1, 2c, and 3.

Example: Monitor One Minute Load Average with edgemon

The following example creates an entry at index 11 in the agent's Self Monitor table that monitors the system's one minute load average for a threshold of 3. The examples below show IPv4 addresses, but you can specify IPv6 addresses as well.

```
edgemon -h fe80::901:dc19 -c private -v 1 -o oid loadAverage1Min.0 11 0x00 60 absolute
'>' 300 "Monitor 1 minute load average" "" “” “” “” fatal
```

```
edgemon -h fe80::901:dc19 -c private -v 2c -o oid loadAverage1Min.0 11 0x00 60 absolute
'>' 300 "Monitor 1 minute load average" "" “” “” “” minor
```

```
edgemon -h 143.45.0.12 -v 3 -u userName -s 3 -a authPassword -A MD5 -x encryptPassword
-X DES -o oid loadAverage1Min.0 11 0x00 60 absolute '>' 300 "Monitor 1 minute load
average" ""
```

Example: Monitor Hardware Interrupts with edgemon

The following example creates a Self Monitor table entry to monitor the number of hardware interrupts on the underlying system against the threshold 1000. The examples below show IPv4 addresses, but you can specify IPv6 addresses as well.

```
edgemon -h fe80::901:dc19 -c private -v 1 -o oid numInterrupts.0 14 0x00500400 60 delta
'>' 1000 "Monitor Interrupt Rate" ""
```

```
edgemon -h fe80::901:dc19 -c private -v 2c -o oid numInterrupts.0 14 0x00500400 60
delta '>' 1000 "Monitor Interrupt Rate" "" “” “” “” minor
```

```
edgemon -h 143.45.0.12 -v 3 -u userName -s 3 -a authPassword -A MD5 -x encryptPassword
-X DES -o oid numInterrupts.0 14 0x00500400 60 delta '>' 1000 "Monitor Interrupt Rate"
""
```

The agent creates this entry at index 14 and samples the numInterrupts variable every 60 seconds. The flags field of 0x00500400 instructs the agent to modify the default Self Monitor table behavior as follows:

0x00000400

Instructs the agent to send up to *X* consecutive monitorEvent traps and then send no more.

0x00500000

Contains the flag value *X*=5 for use with the directive.

Example: Monitor the /usr File System with edgemon

The following example creates a Self Monitor table entry at index 20 to monitor the system's /usr file system for a capacity greater than or equal to 95%, checking the file system every two minutes (120 seconds). The examples below show IPv4 addresses, but you can specify IPv6 addresses as well.

```
edgemon -h 143.45.0.12 -c private -v 1 -o filesystem /usr devCapacity 20 0x00100500
120 absolute '>=' 95 "Monitor /usr Filesystem" "" “” “” “” critical
```

```
edgemon -h 143.45.0.12 -c private -v 2c -o filesystem /usr devCapacity 20 0x00100500
120 absolute '>=' 95 "Monitor /usr Filesystem" ""
```

```
edgemon -h fe80::901:dc19 -v 3 -u userName -s 3 -a authPassword -A MD5 -x
encryptPassword -X DES -o filesystem /usr devCapacity 20 0x00100500 120 absolute '>='
95 "Monitor /usr Filesystem" "" “” “” “” critical
```

Remove Self Monitoring Entries

To stop monitoring a MIB variable, you must remove the appropriate entry from the Self Monitor table and the sysedge.cf file. Removing an entry using point configuration from the CA Virtual Assurance user interface accomplishes both. Manual removal requires that you remove the entry from both places in separate operations.

To remove self monitor entries in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click Self Monitors.

   The Self Monitor table appears with all existing self monitors.
5. Select the monitor to delete and click Actions, Delete.

   The entry is deleted from the table. CA Virtual Assurance loads the changed sysedge.cf file to the agent. The agent performs a warm start to apply the change.

To remove self monitor entries manually

1. Run the edgemon utility using one of the following commands for the -o parameter:

   `edgemon -o delete` *`index`*

   `edgemon -o setstatus` *`index`* `6`

   index

   Specifies the index value of the self monitor entry to remove.

   When using the setstatus command, a value of 6 sets the row status to destroy and deletes the entry.

   Note: If the sysedge.cf is edited, you must restart the agent.
2. Delete the monitor directive for the entry from the local sysedge.cf file and save the file.
3. Restart the agent.

Chapter 9: Process and Service Monitoring

This chapter explains how to use the SystemEDGE agent to monitor processes and services. The agent can monitor many attributes of a process, including whether it is running, its size, CPU and memory and usage, and the number of disk and network I/O operations.

This section contains the following topics:

Process and Service Monitoring Overview (see page 237)
Windows Service Monitoring (see page 238)
State Management of Process Monitors (see page 239)
Process Monitor Table (see page 241)
Process Monitoring Configuration (see page 261)
edgewatch Commands for Process Monitoring (see page 275)
Remove Process Monitoring Entries (see page 281)
Solaris Zone Process Monitoring (see page 282)
Recommendations for Process and Service Monitoring (see page 283)

Process and Service Monitoring Overview

The SystemEDGE agent lets you dynamically monitor specific attributes of important processes, services, and applications that are running on the underlying system. You use the Process Monitor table to specify the process, attribute, threshold value, interval, severity, object information, and other values. The SystemEDGE agent automatically monitors the defined process and sends a trap to the management system if the condition you specified is met.

The SystemEDGE agent can maintain and update a state value for any process monitor you specify and use object class, instance, and attribute values to aggregate the status of an object with multiple monitors defined.

The agent can also run commands on the managed system to perform management functions to correct the cause of the threshold breach immediately.

For information about monitoring process groups, see the chapter “Process Group Monitoring.”

Windows Service Monitoring

On UNIX systems, services or daemons are processes, which you can monitor just like any other process. On Windows systems, however, services are special kinds of processes that start and stop using a graphical interface (for example, through the Services Control Panel). Windows services run within processes, but the mapping between them is not always one to one. For example, multiple Windows services may run within a single process. Consequently, you can monitor Windows services in two ways:

- Monitor Windows services by monitoring their underlying processes. It is sometimes difficult to figure out the underlying process within which a service is running.
- Instruct the SystemEDGE agent to monitor the Windows service, instead of the underlying process, by setting a flag or by using the configuration keyword watch ntservice. For more information, see Process Monitor Table Flags (see page 253).

Note: Because Windows does not track process attributes for Windows services, the SystemEDGE agent can monitor only the procAlive attribute for Windows services. To monitor other attributes (for example, procRSS) for a particular Windows service, you must monitor the underlying process that implements that Windows service.

State Management of Process Monitors

The SystemEDGE state management model maintains a state value for all entries in the Process Monitor table and lets you define managed objects for deriving an overall object state. The state management model provides the following features for entries in the Process Monitor table:

- The ability to define a severity for each process monitor entry that the agent uses to maintain a state value for each entry. The severities translate to CIM-compliant states.
- The ability to group one or more monitored resources into one or more managed objects through a tri-level naming scheme using object class, instance, and attribute values. This mechanism is flexible so that you can dynamically group resources from different functional areas.
- The ability to create a hierarchy of super managed objects, where the state of specific objects propagates to objects on higher levels of the hierarchy
- Notification through state change traps when the overall state of a managed object changes

The state management model is integrated into the CA Virtual Assurance object model. Each managed object in the AOM database possesses a CIM-compliant state, and SystemEDGE managed objects and states are reflected in CA Virtual Assurance though the SystemEDGE Platform Management Module. Managing SystemEDGE through CA Virtual Assurance lets you visualize managed object state information in the CA Virtual Assurance user interface and see state change traps through the CA Virtual Assurance Event view.

For more information about visualizing SystemEDGE managed objects in CA Virtual Assurance, see the CA Virtual Assurance documentation.

Further information about configuring process monitors to take advantage of the state management model, including attribute values and syntax examples, is available throughout the rest of this chapter.

Managed Object Creation

You control the creation of managed objects from the Process Monitor table using the object class, instance, and attribute values. By default, SystemEDGE creates a managed object for each unique class, instance, and attribute process monitor. If multiple process monitors contain the same value for these three properties, the agent groups them into a single managed object and derives an overall state for the object.

To illustrate the flexibility of how you can use this mechanism, consider the following scenarios:

- You define three process monitor entries with the same class, instance, and attribute values for monitoring the real memory usage of a process. The monitors define a warning severity for real memoy usage of greater than 85%, a major severity for real memory usage of greater than 90%, and a critical severity for real memory usage of greater than 95%. The agent combines these monitors into one managed object and maintains the worst current severity as the object state.
- You define three process monitor entries with the same class, instance, and attribute values for monitoring processes from different functional areas that are connected in your enterprise. For example, you can combine a basic operating system process monitor, an application process monitor, and a database process monitor, assigning a warning severity to each process monitor. In this scenario, the agent sends a warning state change trap only if all three process monitor thresholds are breached.
- You define a super aggregation of process monitor entries with the same value for object class and asterisk values for the object instance or attribute. This scenario groups large number of objects without specifically having to define an aggregation rule for each one. For example, you can aggregate all objects that monitor different attributes of the same process, so that all process monitors for that process propagate the worst current state to the top level process object.

  Note: You can also configure the agent to aggregate all objects at higher levels. For more information, see the chapter "Agent Configuration."

For syntax examples of these scenarios, see Process Monitoring Examples (see page 268).

If you do not define object class, instance, and attribute values, the SystemEDGE assigns default values for these properties according to the following convention:

- Class: processEntry
- Instance: value of the pmonRegExpr attribute in the process monitor that defines the regular expression to determine the process to monitor
- Attribute: value of the pmonAttribute attribute in the process monitor that defines the process attribute to monitor

More Information

Object Status (see page 62)
Object Aggregation (see page 63)
Super Aggregation (see page 64)
How State Management and Aggregation Works (see page 66)

Aggregate State Table

The Aggregate State table does the work necessary to create managed object representations and derive overall object state. The Aggregate State table is a filtered view of SystemEDGE self monitors and process monitors. The table includes the aggregated state information calculated based on the managed object naming and state information supplied by each monitor entry.

The table creates an Aggregate State table entry for all self monitors and process monitors with the same object class, instance, and attribute value (or objects grouped together on a higher level). Self monitors and process monitors exist only as individual entries in the Self Monitor and Process Monitor tables. The Aggregate State table maintains the grouping of these monitors, continuously polls the tables, and sends state change traps when the managed object state changes.

Note: The Aggregate State table also sends traps when a new managed object is created or an existing managed object is deleted.

For more information about how the Aggregate State table maintains managed objects and their states, see the chapter "Concepts." For descriptions of the table columns, see Aggregate State Table.

Process Monitor Table

The Process Monitor table provides information about each of the process (or Windows service) that the agent is currently monitoring. Each row represents a defined process monitor entry.

For each entry, the table provides the following types of information:

- Index
- Description
- Interval at which the agent polls the variable
- Sample type (absolute value or delta value)
- Attribute being monitored

- Current value that the agent is monitoring
- Threshold value
- Last sample call
- Number of traps that have been sent already
- Last trap
- Flags
- Action to take when a trap is sent
- Regular expression
- Minimum and maximum value sampled
- Current process ID
- Row status
- Number of events
- Object class
- Instance
- Attribute
- Defined severity
- Current monitor status
- Zone regular expression
- Process name

Process Monitor Table Columns

The following list describes the columns of the Process Monitor table. For a complete description of the Process Monitor Table, see the Systems Management Empire MIB specification (empire.asn1 in the mib subdirectory of the agent installation).

pmonIndex

Defines the unique row index for this entry (1 to MAXINT).

Permissions: Read-only

pmonDescr

Defines a description (0 to 512 characters in length) of the process and attribute being monitored.

Permissions: Read-write

pmonInterval

Defines how often in seconds the agent should monitor the process. The value must be a minimum of 30 and a multiple of 30 seconds.

Permissions: Read-write

Default: 60

pmonSampleType

Defines whether the agent should sample the attribute's absolute value (absoluteValue(1)) or take the difference between successive samples (deltaValue(2)). For example, use deltaValue to monitor counter attributes, because it provides the rate of change. Use absoluteValue to monitor gauges, because it provides the object's exact value.

Permissions: Read-write

Default: absoluteValue(1)

pmonAttribute

Specifies the process attribute being monitored. For a complete list of supported attributes, see Process Attributes (see page 249). For example, to monitor a process to verify that it is alive, specify the procAlive attribute. To track the number of packets received by the particular application or process, specify procMsgsSent.

Permissions: Read-write

pmonCurrVal

Specifies the attribute value that was last recorded for the process being monitored. Every pmonInterval seconds, the agent updates this field to reflect the latest reading for the attribute.

When monitoring procAlive, this value is mapped from the hrSWRunStatus variable (in hrSWRunTable) or ntServiceState (in ntServiceTable) for Windows services. Possible values for hrSWRunStatus:

1, 2, or 3

Indicates that the process is running.

4

Indicates that the process is not running (in this case, it is typically a zombie process)

Possible values for ntServiceState:

1

Indicates that the service is running

2,3,4

Indicates that the service is not running

Permissions: Read-only

pmonOperator

Specifies the Boolean operator used to evaluate the expression *currval operator value*. The operator can be one of the following:

- nop(1) (no operation; monitor the object's value, but do not evaluate the Boolean expression)
- gt(2) (greater than)
- lt(3) (less than)
- ge(4) (greater than or equal to)
- le(5) (less than or equal to)
- eq(6) (equal)
- ne(7) (not equal)

Permissions: Read-write

Default: nop(1)

pmonValue

Defines the integer value to which the agent compares the current value of the monitored process attribute during each monitoring cycle. If the comparison evaluates to True, the agent either changes the monitor state (for process monitors with a severity other than none) or sends a threshold breach trap (for process monitors with a severity of none). For example, if you want to be notified if the real memory usage of a process exceeds 75 percent, set 75 as the pmonValue to which the agent compares the current value of the attribute.

Permissions: Read-write

pmonLastCall

Defines the time (based on sysUpTime) at which the agent last sampled (called) the process attribute it is monitoring. 0 indicates that the process has not yet been sampled.

Permissions: Read-only

pmonNumTraps

Defines the number of traps that have been sent for this entry. This value tracks the processStopTrap, processStartTrap, processClearTrap, processNotReadyTrap, and processThresholdTrap, which the agent sends when a process stops or a threshold breach occurs in a process monitor with a severity of none. This value is useful for determining the frequency at which the exception condition is occurring for legacy monitors. It also provides a means to detect a missed trap message.

Permissions: Read-only

Note: For information about the number of state change traps sent for a stateful process monitor's managed object, see the Aggregate State table.

pmonLastTrap

Defines the time (based on sysUpTime) at which the agent last sent a trap for this entry. 0 indicates that no traps have been sent. This value tracks the processStopTrap, processStartTrap, processClearTrap, processNotReadyTrap, and processThresholdTrap, which the agent sends when a process stops or a threshold breach occurs in a process monitor with a severity of none.

Permissions: Read-only

Note: For information about the last state change trap sent for a stateful process monitor's managed object, see the Aggregate State table.

pmonFlags

Defines the positive integer flags that indicate additional behavioral semantics that the entry follows during the course of its operation. For more information about available flags and setting flags, see Process Monitor Table Flags (see page 253).

Permissions: Read-write

pmonAction

Defines a quoted command (0 to 2048 characters in length) with any parameters to run when sending a legacy trap (legacy monitors) or state change trap (stateful monitors). For stateful process monitors, the command does not run if a threshold breach does not result in an object state change. If the string is empty, the agent performs no action for this entry.

Note: Do not use Windows batch files for actions; they impose severe programmatic limitations and often do not work correctly with desktop applications. Instead, use a more powerful and flexible scripting language, such as Perl or Visual Basic.

Permissions: Read-write

pmonRegExpr

Specifies the regular expression to apply when the agent is attempting to acquire the process ID of an application or process to monitor. For Windows service monitoring, this regular expression matches the name of the Windows service to monitor. Instead of requiring users to specify process IDs (PIDs) or service indexes, which may change, users specify a regular expression for process name or service name. The agent uses this user-specified name to find the process (or service) to monitor. By default, the Process Monitor table keeps attempting to apply the regular expression until a new PID or service is found if the process or service stops running.

Permissions: Read-write

pmonMinValue

Defines the lowest (minimum) value that the agent has observed since it began polling the process attribute.

Permissions: Read-only

pmonMaxValue

Defines the highest (maximum) value that the agent has observed since it began polling the process attribute.

Permissions: Read-only

pmonCurrentPID

Specifies the PID of the process or NT service index currently being monitored.

Permissions: Read-write

pmonRowStatus

Defines the row status, which can be one of the following:

- active
- notInService
- notReady
- createAndGo
- createAndWait
- destroy

Typically, a row is either active or notInService. These values are identical in meaning to those defined by the SNMPv2 SMI RowStatus textual convention.

Permissions: Read-write

Default: createAndWait(5)

pmonNumEvents

Specifies the number of events for this monitor entry and depends on pmonFlags. The events do not always imply traps.

Permissions: Read-write

pmonObjClass

Defines the object class of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value. A class name can be used to specify a monitored object table. If SystemEDGE does not determine an existing table with the specified name, SystemEDGE creates a class. The class specifies the monitoring object identifier (SNMP OID) and can be referenced for aggregation purposes.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

pmonObjInstance

Defines the object instance of the monitored MIB object for use with the object state management model. If a value is not configured and the agent cannot determine a default, the value is 'unknown'. The default for a nontabular object is 'null'. For seamless monitoring of remote objects, instance names start with a prefix //hostname/.

For local objects, the agent adds the prefix //./ to each instance name (including 'null'). This attribute cannot be set to an empty string or to '*'. Any such requests are ignored. If the instance name cannot be matched at the time of parsing to a corresponding instance, SystemEDGE creates a reference to this monitoring entry's OID with the specified instance name.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

pmonObjAttribute

Defines the object instance of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value for objects that corresponds to the MIB attribute name. If the attribute name is not known during parsing of the monitoring entry, SystemEDGE creates an attribute with the specified name.

Permissions: Read-write

Default: ASCII MIB attribute name of the monitored OID

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

pmonSeverity

Defines the severity to use for the object state management model. This value defines the state to assign to the entry when a threshold breach occurs. Valid values are as follows:

- none(1)
- ok(2)
- warning(3)
- minor(4)
- major(5)
- critical(6)
- fatal(7)

A value of none creates a legacy monitor, which excludes the entry from the object state management model.

Default: none

pmonCurrState

Defines the current derived state of the entry, evaluated using the severity and the current breach condition and respecting any lag value defined in the pmonFlags property. Valid values are as follows:

- unknown(1)
- ok(2)
- warning(3)
- minor(4)
- major(5)
- critical(6)
- fatal(7)
- up(11)
- down(12)

The up and down values only appear for legacy monitors (with a severity of none). A state change in this column results in a state change trap only if the state is the worst of all monitors for the corresponding managed object in the Aggregate State table.

Permissions: Read-only

pmonProcName

Defines the process name that was matched by regular expression. On Solaris 10 and newer the name has the format *MatchedZoneName/MatchedProcName* if zone regular expression is used.

Sample Process Monitor Table Entry

This section provides a sample Process Monitor table entry that instructs the agent to monitor the httpd process to make sure it is up and running.

Index	**Desc**	**Interval**	**SampleType**	**Attribute**
21	"Monitor httpd"	60	absoluteValue	procAlive
NumTraps	**LastCall**	**Value**	**Oper**	**CurrVal**
0	0d 22:40:00	4	=	3
LastTrap	**RowStatus**	**Action**	**Flags**	**RegExpr**
0	active	""	0x00000100	httpd

ObjClass	ObjInstance	ObjAttribute	Severity	CurrState
Process	httpd	procAlive	warning	ok

This sample entry provides the following information:

- This entry in this example is the 21st row in the Process Monitor table.
- It instructs the agent to monitor the Web server daemon (httpd) every 60 seconds.
- The attribute being monitored is the status of the process (procAlive).
- The current value of 3 indicates the process is running normally (as far as the operating system is concerned). If the process status goes to 4, or if the process stops running, the agent sets the current state to warning and sends a state change trap if this monitor represents the worst state for the corresponding managed object in the Aggregate State table.
- The RowStatus column shows that this entry is active.
- The Action column is null (''), which indicates that the agent invokes no action when the process stops running or starts again.
- The Flags value (0x00000100) instructs the agent to monitor the parent httpd daemon, instead of child processes. For more information about flags, see Process Monitor Table Flags (see page 253).
- The class, instance, and attribute values establish this process monitor entry as a part of the managed object with these values in the Aggregate State table. You can define other process monitor entries with these same values to monitor for different severities, or you can aggregate this entry with other related objects and even with self monitors (such as monitoring other attributes of the same process).

Process Attributes

The following list describes the attributes that the SystemEDGE agent can monitor for a particular process or service procAlive and the SNMP type for each attribute. You can use the SNMP type to select the sample type and operator. For example, absoluteValue sampling is usually most appropriate for integer or gauge attributes, while deltaValue sampling is usually most appropriate for counter attributes.

Note: The agent's ability to monitor a particular process attribute is dependent on the underlying operating system's ability to track the associated parameter or metric.

procAlive (All OS)

Defines whether the process or service is running.

Type: Boolean

procMemory (All OS)

Defines the percentage (0 to 100) of real memory used by the process.

Type: Gauge

procSize (All OS)

Defines the size of text, data, and stack segments (KB).

Type: Gauge

procRSS (All OS)

Defines the real memory (resident set) size of the process (KB).

Type: Gauge

procTime (All OS)

Defines the accumulated CPU time in seconds for the process.

Type: Integer

procInBlks (AIX, Solaris)

Defines the number of blocks of data input by the process.

Type: Counter

procOutBlks (AIX, Solaris)

Defines the number of blocks of data output by the process.

Type: Counter

procMsgsSent (AIX, HP-UX, Solaris)

Defines the number of messages received by the process.

Type: Counter

procMsgsRecv (AIX, HP-UX, Solaris)

Defines the number of messages sent by the process.

Type: Counter

procNice (All OS)

Defines the priority (nice value) of the process.

Type: Integer

procNumThreads (Windows, AIX, Solaris, Linux)

Defines the number of threads that are running within the process.

Type: Integer

procNumSwaps (AIX, HP-UX, Solaris)

Defines the number of times the process has been swapped.

Type: Counter

procSysCalls (HP-UX, Solaris)

Defines the number of system calls invoked by the process.

Type: Counter

procMinorPgFlts (AIX, HP-UX, Linux, Solaris)

Defines the number of minor page faults incurred by the process.

Type: Counter

procMajorPgFlts (All OS)

Defines the number of major page faults incurred by the process.

Type: Counter

procVolCtx (AIX, Solaris)

Defines the number of voluntary context switches incurred by the process.

Type: Counter

procInvolCtx (AIX, Solaris)

Defines the number of involuntary context switches incurred by the process.

Type: Counter

procTimePermil (All OS)

Defines the CPU utilization in permil multiplied by factor 1000.

Type: Integer

Row Creation Objects

The pmonIndex column of the Process Monitor table acts as a row index to distinguish rows in the table. Rows 1 through 10 are reserved for internal use by the SystemEDGE agent. You can configure rows in the range of 11 to MAXINT.

You can use the following MIB objects with the Process Monitor table to optimize row creation.

pmonUnusedIndex

Returns an unused index number for the Process Monitor table when you perform an SNMP Get on the object. Query this object to obtain a number that you can use for row creation.

pmonMatchDescr

Determines the index number that corresponds to a particular entry description in the pmonDescr attribute. Perform an SNMP Set on this MIB object with a pmonDescr attribute value to cause the agent to search through entries in the Process Monitor table and put the index value of the last matching entry in the monMatchIndex MIB object.

pmonMatchIndex

Matches a particular entry description with its index number when used with pmonMatchDescr.

You may choose, as a matter of local policy, to reserve a block of rows for system administration. You can then define entries within a reserved block of rows without being concerned that the row may already be taken by another user's entry. In compliance with the local policy, all other users should use row indices outside the reserved range when they define user-configured entries.

By reserving a block of rows, you can define a consistent set of conditions (row entries) to be monitored across all computers such that the same condition is defined in the same row number on each computer. For example, you can use row 11 (monIndex = 11) to define an entry for monitoring the swapCapacity variable, and you can distribute this configuration to every system so that row 11 monitors the swapCapacity variable on every system.

Configuring the agent with CA Virtual Assurance can automate this process. You can configure a block of entries with specific row numbers and deploy that configuration to all managed systems in one operation. Also, CA Virtual Assurance detects if you define a row that is already in use and disallows the operation.

For more information about defining rows in CA Virtual Assurance, see the CA Virtual Assurance documentation.

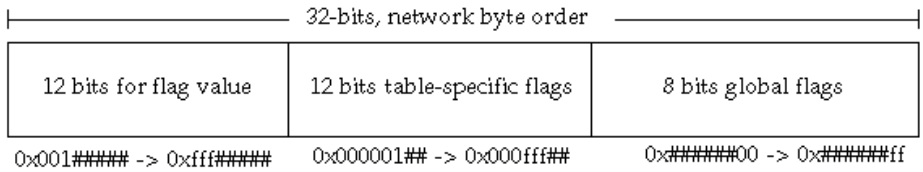
Process Monitor Table Flags

The pmonFlags column in the Process Monitor table is a 32-bit unsigned integer field that can specify additional behavioral semantics for the corresponding row.

By default, the Process Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps (legacy monitors only)
- Logs events to the common log file
- Invokes actions (if they are configured)

You can set different flag bits to alter these defaults. The agent interprets all flags in hexadecimal (base 16) notation. The following illustration shows the composition of the Process Monitor table flags (pmonFlags) field:



The flags value consists of three fields:

- Field 1 contains common table flags defined for the monitoring tables of the Systems Management Empire MIB. This portion is the low-order 8 bits of the flag.
- Field 2 contains table-specific flags defined separately for each of the monitoring tables. This field defines the next 12 low-order bits after the common table flags.
- Field 3 contains 12 reserved high-order bits for an integer value for use with table-specific flags. This field defines the flags that are specific to the Process Monitor table.

The following sections explain each flag bit. You can combine flag values through a logical OR operation.

The following list describes the Process Monitor table flags:

0x00000001

Disables execution of actions for this entry.

0x00000002

Disables sending of SNMP traps for this entry.

Note: The Process Monitor table only controls sending traps for legacy monitors. For stateful process monitors, state change traps may still occur from the Aggregate State table.

0x00000004

Disables attempts to reinitialize this entry. By default, the agent periodically tries to reinitialize this entry by scanning the process table to determine the new process ID if the target process has been restarted.

0x00000008

Disables logging of traps for this entry in the sysedge.log file. Setting this bit does not affect sending trap. Disabling event logging is useful when events occur frequently or when a particular entry is used as an agent heartbeat.

0x00000010

Sends continuous processStop traps for this entry each time the agent attempts to reinitialize process monitoring and fails to match a process. The agent's default behavior is to send a single processStop trap when a process dies and attempt to periodically reinitialize the entry. Enabling this feature causes the agent to send an additional processStop trap each time reinitialization fails. In all cases, the agent does not send processStart and processStop traps unless the corresponding entry is monitoring the procAlive process attribute.

Note: This flag is valid only when the agent is monitoring the procAlive attribute. When you are monitoring Windows services, an entry does not enter the notReady state when the service is not running. Setting this flag causes SystemEDGE to generate processStop traps and run any associated actions for the entry when the service is not running, even though it remains in the ready state.

0x00000020

Disables the passing of default arguments to action scripts or programs. SystemEDGE typically passes default action parameters that indicate the trap type, description field, and so on. For more information about action parameters, see Process Monitor Table Actions (see page 259).

0x00000040

Disables sending of notReady traps for this entry.This includes to not log and to not execute actions for notReady trap.

0x00000100

Monitors the parent process in the process group. Many applications and services (for example, Web server httpd daemons) are designed such that an initial daemon spawns child processes to handle actual service requests. These child processes often service several requests and then exit. In these cases, it is preferable to monitor the main parent daemon instead of the child processes. Enabling this feature causes the agent to search for and monitor the parent daemon instead of the first process it finds that matches the process regular expression.

The agent performs this search by scanning the Process Monitor table for processes that match the name of the process regular expression (pmonRegExpr). If the matching parent process of the process also matches, the agent returns the parent process. This searching algorithm only accommodates parent and child relationships and cannot handle daemons forking daemons. This feature is not available for monitoring of Windows services.

0x00000200

Disables sending of processStart traps for this entry (no log and no action).

0x00000400

Sends processClear traps for this entry (log and action) when a process monitor expression transitions from True to False. This feature is only applicable when the attribute being monitored is not procAlive.

0x00000800

Matches process name and arguments when targeting a process for monitoring. By default, the agent matches only against a process name. Enabling this option causes the agent to apply the pmonRegExpr to both the process name and process arguments, which is sometimes necessary to distinguish between similar processes or multiple invocations of the same application or binary.

0x*XXX*01000

Sends processThreshold traps or changes the current state only after the *X*th consecutive event. Enabling this feature instructs the agent to wait until the *X*th consecutive occurrence of an event before sending processThreshold traps or changing the state of the process monitor. After the *X*th event has occurred, the agent will send processThreshold traps for each subsequent, consecutive True expression evaluation. If the threshold expression transitions from True back to False, the row resets itself, and the agent begins counting events from zero. This flag also applies to action execution. You can specify the value of *X* through the flag value field. Event logging is unaffected by this flag bit. For an example, see Process Monitoring Examples (see page 268).

0x*XXX*02000

Sends up to *X* consecutive processThreshold traps, and then sends no more. Enabling this feature puts an upper boundary on the number of consecutive processThreshold traps and action executions that can occur when a process has exceeded a threshold. After the threshold expression transitions from True to False, the row resets itself, and the agent begins counting events from zero. This flag also applies to action execution. You can specify the value of *X* through the flag value field. Event logging is unaffected by this flag bit. For an example, see Process Monitoring Examples (see page 268).

0x*XXX*04000

Monitors a process group for *X* processes to be alive (procAlive monitoring only).

0x00008000

Monitors the Windows service that matches the corresponding regular expression. Setting this flag instructs SystemEDGE to monitor the procAlive attribute of the matching Windows service within the Windows service table.

Monitors on Solaris 10 a process in a zone. The pmonProcRegExpr attribute is expected in format ZoneRegExp/ProcRegExp.

0x00010000

Disables any processStart processing. If this flag is enabled, SystemEDGE does not invoke actions, log events, or send traps when processStart events occur (see also 000 002 00 for legacy support).

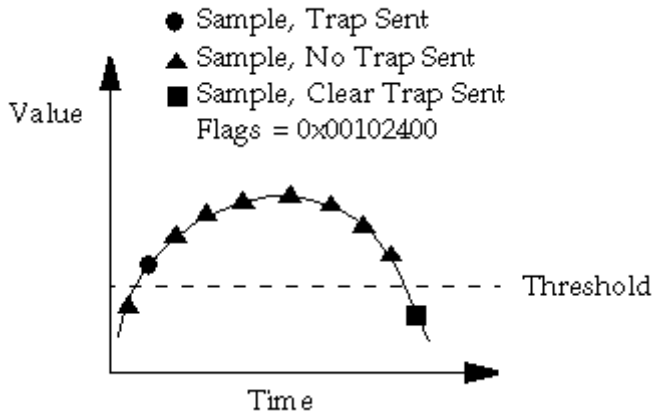
0x00020000

Aggregates the state of monitors for the same object with the same severity so that they have an AND relation. All connected monitors with the same severity that have this flag set must have breach conditions for the aggregate state to change accordingly.

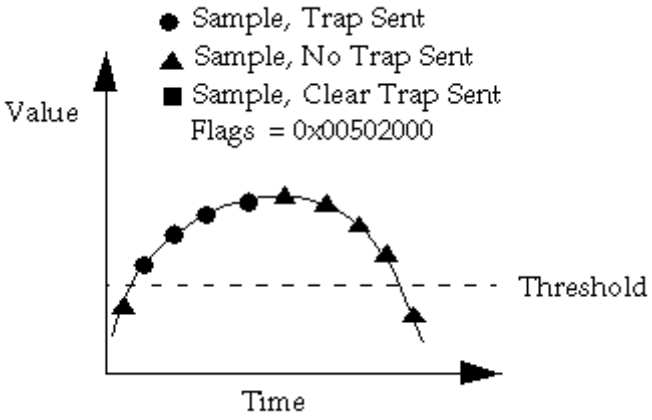
0x###00000

Several flag bits use a value *X* for sending traps and executing actions. The value *X* is specified as the high-order 12 bits of the flag field. Flag bits utilizing this field are mutually exclusive.

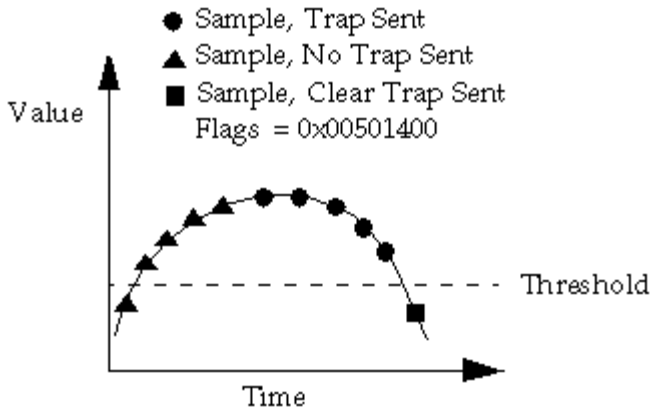
The following illustration shows the agent sending one trap to indicate that the monitored object crossed the threshold, and then sending a processClear trap when value drops below the threshold.



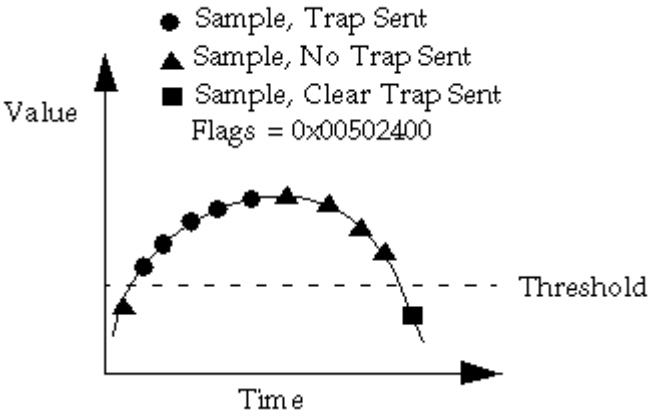
The following illustration shows the agent sending four traps to indicate that the value of the monitored object is above the threshold. It does not send a processClear trap when the value falls below the threshold.



The following illustration shows the agent waiting until an event has occurred several times before it begins sending traps. It then sends traps until the value of the monitored object falls below the threshold, at which time it sends a processClear trap.



The following illustration shows the agent sending traps only a specified number of times. When the value of the monitored object falls below the threshold, the agent sends a processClear trap.



Process Monitor Table Action Parameters

The SystemEDGE agent provides several default parameters to the action commands when they are invoked. These parameters are in addition to any parameters that you specify in the action string and are passed on the command line after those that you specify. The default parameters are the same as the parameters provided in the SNMP traps that are sent for the Process Monitor table.

The following list describes the default parameters for Process Monitor table actions:

Note: You can prevent the agent from including these action parameters in the command by setting the 0x00000020 flag in the table entry.

trapType

Defines the type of trap being sent, which will be one of the following:

- processStop
- processStart
- processThreshold
- processClear

pmonIndex

Defines the index assigned to the table entry.

pmonDescr

Defines the table entry description.

pmonAttribute

Defines the process attribute being monitored by the entry.

pmonCurrVal

Defines the current value obtained for the table entry.

pmonOperator

Defines the operator being used by the table entry.

pmonValue

Defines the comparison value or threshold applied by the table entry.

pmonFlags

Defines flags that are associated with the table entry.

pmonRegExpr

Defines the regular expression that the agent uses to find the process ID of the process to monitor.

pmonCurrentPID

Defines the process ID (PID) of the current process being monitored. If you are monitoring a Windows service (through the watch ntservice directive or by setting flag 0x8000), this column represents the index in the NT Service MIB table that this process or service monitoring entry has acquired.

pmonObjClass

Defines the object class of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value. A class name can be used to specify a monitored object table. If SystemEDGE does not determine an existing table with the specified name, SystemEDGE creates a class. The class specifies the monitoring object identifier (SNMP OID) and can be referenced for aggregation purposes.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

pmonObjInstance

Defines the object instance of the monitored MIB object for use with the object state management model. If a value is not configured and the agent cannot determine a default, the value is 'unknown'. The default for a nontabular object is 'null'. For seamless monitoring of remote objects, instance names start with a prefix //hostname/.

For local objects, the agent adds the prefix //./ to each instance name (including 'null'). This attribute cannot be set to an empty string or to '*'. Any such requests are ignored. If the instance name cannot be matched at the time of parsing to a corresponding instance, SystemEDGE creates a reference to this monitoring entry's OID with the specified instance name.

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

pmonObjAttribute

Defines the object instance of the monitored MIB object for use with the object state management model. The agent uses the sysedge.oid file to assign a default value for objects that corresponds to the MIB attribute name. If the attribute name is not known during parsing of the monitoring entry, SystemEDGE creates an attribute with the specified name.

Permissions: Read-write

Default: ASCII MIB attribute name of the monitored OID

Important! The newly created reference is valid only if the monitoring entry resolves to an existing SNMP OID.

pmonCurrState

Defines the current state of the table entry.

For more information about traps sent by the SystemEDGE agent, see the chapter "Concepts" and the appendix “Private Enterprise Traps.”

View the Process Monitor Table

You can use CA Virtual Assurance to view the contents of a managed agent's Process Monitor table.

To view the Process Monitor table in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click Process Monitors.

   The Process Monitor table appears with all existing process monitors.

Process Monitoring Configuration

You can control the processes and process attributes that the SystemEDGE agent monitors by adding, deleting, or modifying the entries in the Process Monitor table.

You can configure the Process Monitor table in the following ways:

Dynamically using file-based configuration from CA Virtual Assurance

CA Virtual Assurance provides the ability to configure agents installed on any system that it manages from the CA Virtual Assurance user interface. This is the recommended method of configuration, because it provides the ability to dynamically define and deliver monitors with a user-friendly and secure mechanism that does not require an agent restart. You can configure monitoring for an individual agent or define monitoring policies to apply on multiple managed systems. When you make an individual configuration change or apply a configuration policy, CA Virtual Assurance delivers a new configuration file to the agent system. CA Virtual Assurance can serve as the central point of agent configuration across your data center, and you can overwrite or deny changes made through other methods.

For more information about configuring the agent using CA Virtual Assurance, see the CA Virtual Assurance documentation.

Dynamically using SNMP Set commands from a management system, such as eHealth AdvantEDGE View, CA Spectrum, or CA NSM

Various management systems support adding monitors to SystemEDGE through SNMP Set commands. eHealth provides this functionality through the AdvantEDGE View interface, CA NSM enables configuration through an Agent View, and CA Spectrum enables configuration through OneClick. All features provided by dynamic configuration through CA Virtual Assurance are not available through these methods.

For more information about SNMP Set configuration, see the documentation for the appropriate management system.

Manually by specifying the entries for the Self Monitor table in the local sysedge.cf file

Manual configuration requires you to add entries to the local sysedge.cf file using specific directives. When you configure the agent directly from the sysedge.cf file, a full restart is required for changes to take effect. Use the sysedge.cf file located in the agent data directory for direct configuration.

This section describes how to configure self monitoring directly in the local sysedge.cf file, but it describes the properties that are required for all forms of configuration.

You can also dynamically add, delete, or modify Process Monitor Table entries through the edgewatch utility. For more information, see edgewatch Utility--Monitor Processes.

watch process Directive--Add Entries to the Process Monitor Table

The watch process directive lets you add process monitor entries to the Process Monitor table directly in the sysedge.cf file. The arguments represent columns in the Process Monitor table.

Add a line to the sysedge.cf file in the agent data directory using the syntax described below, save the file, and restart the agent for the change to take effect.

Use this syntax to monitor any available process attribute, including procAlive.

When you enter procAlive as the attribute to monitor whether the process is running, certain values are ignored. Monitors for the procAlive attribute monitor based on the expression 'hrSWRunStatus = 4'. If the process stops running or becomes invalid, the agent sends a processStop (legacy monitors) or state change trap (stateful monitors) and invokes any configured action.

When you specify to monitor a Windows service using the 0x00008000 flag, the same values are ignored, and the agent monitors the service procAlive attribute based on the expression 'ntServiceState ? 1'. When the service stops running or becomes invalid, the agent sends a processStop (legacy monitors) or state change trap (stateful monitors) and invokes any configured action.

Use the watch process directive to add entries to the Process Monitor table as follows:

```
watch process attribute 'regexpr' index flags interval stype oper thresh ['descr'
['action' ['objClass' 'objInst' 'objAttr' severity]]]
```

attribute

Specifies the process attribute that the SystemEDGE agent monitors for the specified threshold. You can specify any process attribute, including procAlive, from Process Attributes (see page 249).

'regexpr'

Specifies the regular expression to use when attempting to acquire the process ID of an application or process to monitor. Specify a regular expression for the process name to monitor.

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For more information about available flags, see Process Monitor Table Flags (see page 253).

interval

Specifies how often in seconds the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. The value must be a minimum of 30 and a multiple of 30 seconds.

stype

Specifies whether the agent should sample the attributes's absolute value (absoluteValue) or take the difference between successive samples (deltaValue). Use deltaValue to monitor counter variables for rate of change, and use absoluteValue to monitor gauges for exact value.

Note: The agent ignores this value if you are monitoring the procAlive attribute.

oper

Specifies the Boolean operator for evaluating the current entry value against the provided threshold value.

Valid values for the operator are as follows:

- nop (no operation)
- > (greater than)
- < (less than)
- >= (greater than or equal to)
- <= (less than or equal to)
- = (equal)
- != (not equal)

Default: nop

Note: The agent ignores this value if you are monitoring the procAlive attribute.

thresh

Specifies the integer threshold value to which the current value of the monitored attribute is compared using the specified operator.

Note: The agent ignores this value if you are monitoring the procAlive attribute.

'*descr*'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'*action'*

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the threshold condition is met and the agent sends a threshold breach trap (legacy monitors) or state change trap (stateful monitors). For stateful process monitors, the command does not run if a threshold breach does not result in an object state change. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings when the agent performs actions. For more information, see the chapter "Agent Configuration."

'*objClass*'

Specifies the object class of the monitored object for use with the state management model. Define a value for this property (with objInst and objAttr) for creating a managed object that aggregates all entries with the same values.

Default: MIB group or table name (see sysedge.oid)

'*objInst*'

Specifies the object instance of the monitored object for use with the state management model. Define a value for this property (with objClass and objAttr) for creating a managed object that aggregates all entries with the same values. The agent uses the value of the specified regExpr property as a default value.

'*objAttr*'

Specifies the object attribute for use with the state management model. Define a value for this property (with objClass and objInst) for creating a manged object that aggregates all entries with the same values.

Default: process attribute name

severity

Specifies the severity to use for the object state management model. This value defines the state to assign to the entry when a threshold breach occurs. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

A value of none creates a legacy monitor, which excludes the entry from the object state management model.

Default: none

watch ntservice Directive--Add Service Monitoring Entries to the Process Monitor Table

The watch ntservice directive lets you add entries to the Process Monitor table for monitoring Windows services based on the procAlive attribute to track their running status.

Add a line to the sysedge.cf file in the agent data directory using the syntax described below, save the file, and restart the agent for the change to take effect.

Use the watch ntservice directive to add Windows service monitoring entries to the Process Monitor table as follows:

```
watch ntservice 'regexpr' index flags interval ['descr' ['action' ['objClass'
'objInst' 'objAttr' severity]]]
```

'regexpr'

Specifies the regular expression to use when attempting to acquire the ID of a service to monitor. Specify a regular expression for the service name to monitor.

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001).

interval

Specifies how often in seconds the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. The value must be a minimum of 30 and a multiple of 30 seconds.

'***descr***'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'***action'***

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the threshold condition is met and the agent sends a threshold breach trap (legacy monitors) or state change trap (stateful monitors). For stateful process monitors, the command does not run if a threshold breach does not result in an object state change. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

'***objClass***'

Specifies the object class of the monitored object for use with the state management model. Define a value for this property (with objInst and objAttr) for creating a managed object that aggregates all entries with the same values.

Default: processEntry

'***objInst***'

Specifies the object instance of the monitored object for use with the state management model. Define a value for this property (with objClass and objAttr) for creating a managed object that aggregates all entries with the same values. The agent uses the value of the specified regExpr property as a default value.

'***objAttr***'

Specifies the object attribute for use with the state management model. Define a value for this property (with objClass and objInst) for creating a manged object that aggregates all entries with the same values.

Default: procAlive

severity

Specifies the severity to use for the object state management model. This value defines the state to assign to the entry when a threshold breach occurs. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

A value of none creates a legacy monitor, which excludes the entry from the object state management model.

Default: none

Process Monitoring Examples

This section contains sample configuration file directives for process monitoring. Each example shows how to define an instance of process monitoring and explains the attribute or threshold being monitored.

Example: Monitor Sendmail to Ensure It Is Running

The following example configures the agent to monitor the sendmail daemon on a UNIX email server on the underlying system:

```
watch process procAlive 'sendmail' 11 0x00000100 60 absolute=4 'Monitor sendmail' ''
'processEntry' 'sendmail' 'procAlive' fatal
```

11

Indicates that this entry will occupy row 11 (pmonIndex=11) in the Process Monitor table.

0x00000100

Indicates that the agent should monitor the parent sendmail process if more than one sendmail daemon is present and running.

60

Indicates that the agent should check the sendmail process every 60 seconds.

processEntry

Indicates that this entry uses processEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

sendmail

Indicates that this entry uses sendmail object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

procAlive

Indicates that this entry uses procAlive object attribute for the object state model. The attribute is replaced with default value. Here, '*' is an invalid value.

fatal

Indicates that this entry uses fatal severity for the object state model.

No action is specified, so the agent will not invoke a command when it sends a trap.

Example: Monitor the Simple TCP/IP Services Process To Ensure It Is Running

The following example configures the agent to monitor the TCPSVCS process that makes up the Simple TCP/IP Services service:

```
watch process procAlive 'TCPSVCS' 12 0x00000000 30 absolute=4 'Monitor NT TCP
services' '' 'processEntry' 'TCPSVCS' 'procAlive' warning
```

12

Indicates that this entry will occupy row 12 (pmonIndex=12) in the Process Monitor table.

0x00000000

Indicates that the agent should provide the default process monitoring behavior.

30

Indicates that the agent should check the TCPSVCS process every 30 seconds.

processEntry

Indicates that this entry uses processEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

TCPSVCS

Indicates that this entry uses TCPSVCS object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

procAlive

Indicates that this entry uses procAlive object attribute for the object state model. The attribute is replaced with default value. Here, '*' is an invalid value.

warning

Indicates that this entry uses warning severity for the object state model.

No action is specified, so the agent will not invoke a command when it sends a trap.

Note: This example illustrates how to monitor the underlying process that provides the Windows Simple TCP/IP Services service. The following example illustrates how to monitor the Windows service itself instead of its underlying process.

Example: Monitor the Simple TCP/IP Services Service

Both of the following examples configure the agent to monitor the TCPSVCS service itself instead of the underlying process:

```
watch process procAlive 'Simple TCP/IP Services' 13 0x08000 30 absolute=4 'Monitor
NT TCP/IP Services' '' 'processEntry' 'Simple TCP/IP Services' 'procAlive' minor
```

13

Indicates that this entry will occupy row 13 (pmonIndex=13) in the Process Monitor table.

0x08000

Indicates that the agent should monitor the Windows service, instead of the underlying process.

30

Indicates that the agent should check the Simple TCP/IP Services service every 30 seconds.

processEntry

Indicates that this entry uses processEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

Simple TCP/IP Services

Indicates that this entry uses Simple TCP/IP Services object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

procAlive

Indicates that this entry uses procAlive object attribute for the object state model. The attribute is replaced with default value. Here, '*' is an invalid value.

minor

Indicates that this entry uses minor severity for the object state model.

```
watch ntservice 'Simple TCP/IP Services' 14 0x0 30 'Monitor NT TCP/IP Services' ''
'processEntry' 'Simple TCP/IP Services' 'procAlive' major
```

14

Indicates that this entry will occupy row 14 (pmonIndex=14) in the Process Monitor table.

30

Indicates that the agent should check the Simple TCP/IP Services service every 30 seconds.

processEntry

Indicates that this entry uses processEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

Simple TCP/IP Services

Indicates that this entry uses Simple TCP/IP Services object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

procAlive

Indicates that this entry uses procAlive object attribute for the object state model. The attribute is replaced with default value. Here, '*' is an invalid value.

major

Indicates that this entry uses major severity for the object state model.

No action is specified, so the agent will not invoke a command when it sends a trap.

Example: Monitor ypbind To Ensure It Is Running

The following example configures the agent to monitor the UNIX ypbind daemon on the underlying system:

```
watch process procAlive 'ypbind' 15 0x00000000 60 absolute=4 'Monitor ypbind'
'/example/pager.sh' 'processEntry' 'ypbind' 'procAlive' critical
```

15

Indicates that this entry will occupy row 15 (pmonIndex=15) in the Process Monitor table.

60

Indicates that the agent should check the ypbind process every 60 seconds.

processEntry

Indicates that this entry uses processEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

ypbind

Indicates that this entry uses ypbind object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

procAlive

Indicates that this entry uses procAlive object attribute for the object state model. The attribute is replaced with default value. Here, '*' is an invalid value.

critical

Indicates that this entry uses critical severity for the object state model.

The agent invokes the specified action script /example/pager.sh each time it sends a trap. In this case, it invokes the script each time a processStop or a processStart trap is sent. The script should examine its arguments to determine which trap is being sent and send the appropriate message to the target pager.

Example: Monitor the Size of a Process

The following example configures the agent to monitor the overall size of a particular process:

```
watch process procSize 'netscape' 20 0x00a02400 60 absolute '>' 35000 'Monitor
netscape size' '' 'processEntry' 'netscape' 'procSize' none
```

procSize

Indicates the attribute that the agent is monitoring. It returns the size of text, data, and stack segments of the corresponding process. You can determine if this attribute is leaking memory by monitoring it for a given threshold.

netscape

Indicates the name of the process that the agent will monitor.

20

Indicates that this entry will occupy row 20 (pmonIndex=20) of the Process Monitor table.

0x00a02400

Instructs the agent to modify the default Process Monitor table behavior as follows:

0x00000400

Instructs the agent to send processClear traps.

0x00002000

Instructs the agent to send up to 10 consecutive traps and then send no more.

0x00a00000

Contains the flag value 10 for use with the directive in this example.

processEntry

Indicates that this entry uses processEntry object class for the object state model. The object class is replaced with default value. Here, '*' is an invalid value.

netscape

Indicates that this entry uses netscape object instance for the object state model. The object instance is replaced with default value. Here, '*' is an invalid value.

Note: '//' and '/' can be used as delimiters to relate an instance to a system: '//hostname/instance'. For local system instances '//./' is always prepended to relate the instance to the local system. '//hostname/*', '//*/instance' and '//*/*' are invalid values.

procAlive

Indicates that this entry uses procAlive object attribute for the object state model. The attribute is replaced with default value. Here, '*' is an invalid value.

none

Indicates that this entry uses none severity for the object state model.

absolute

Instructs the agent to compare each sampled value to the threshold instead of measuring the difference (delta) between successive samples.

>

Instructs the agent to compare the sampled procSize attribute against the value 35000 (35,000 KB or 35 MB), and to send a processThreshold trap when that threshold is exceeded.

More Information

Perl Compatible Regular Expression (PCRE) Support (see page 150)
Configure Text Pattern Exclusion (see page 152)
Regular Expression Examples (see page 152)

edgewatch Commands for Process Monitoring

The edgewatch process monitoring commands and associated arguments are as follows:

```
add procAlive processname index flags interval 'descr' 'action' [objClass objInst
objAttr severity]
```

```
add attribute processname index flags interval sampleType oper value 'description'
'action' [objClass objInst objAttr severity]
```

```
setstatus index status
```

```
delete index
```

```
list
```

```
dump
```

processname

Specifies the regular expression used to find the PID of the process to monitor. Enclose this value in quotation marks if it contains spaces or other special characters.

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001).

interval

Specifies how often in seconds the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. The value must be a minimum of 30 and a multiple of 30 seconds.

sampleType

Specifies a sample type of either absolute or delta. This value indicates whether the agent should sample the object's absolute value or take the difference between successive samples.

oper

Specifies the Boolean operator for evaluating the current entry value against the provided threshold value.

Valid values for the operator are as follows:

- nop (no operation)
- > (greater than)
- < (less than)
- >= (greater than or equal to)
- <= (less than or equal to)
- = (equal)
- != (not equal)
- **Default:** nop

value

Specifies the integer threshold value to which the current value of the monitored attribute is compared using the specified operator.

'***descr***'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'***action***'

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the threshold condition is met and the agent sends a threshold breach trap (legacy monitors) or state change trap (stateful monitors). For stateful self monitors, the command does not run if a threshold breach does not result in an object state change. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

objClass

Specifies the object class of the monitored object for use with the state management model. Define a value for this property (along with objInst and objAttr) for creating a managed object that aggregates all entries with the same values.

Default: processEntry

objInst

Specifies the object instance of the monitored object for use with the state management model. Define a value for this property (along with objClass and objAttr) for creating a managed object that aggregates all entries with the same values. The agent uses the value of the specified regExpr property as a default value.

objAttr

Specifies the object attribute for use with the state management model. Define a value for this property (along with objClass and objInst) for creating a manged object that aggregates all entries with the same values.

Default: process attribute name

severity

Specifies the severity to use for the object state management model. This value defines the state to assign to the entry when a threshold breach occurs. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

A value of none creates a legacy monitor, which excludes the entry from the object state management model.

Default: none

status

Specifies the status of the entry, which can be one of the following:

- active (activate a row)
- notInService (deactivate but preserve a row)
- destroy (delete a Self Monitor table row)

attribute

Specifies the process attribute that the agent monitors for the given threshold. You can select any process attribute from the list in Process Attributes (see page 249).

edgewatch Examples

This section includes examples for using the edgewatch command with SNMP versions 1, 2c, and 3.

Example: Monitor the ypbind Process

The following example creates a Process Monitor table entry at index 16 to monitor the ypbind process running on the target system for SNMPv1:

```
edgewatch -h 143.45.0.12 -c private -v 1 -o process add procAlive "ypbind" 16 0x00
60 "Monitor ypbind" "/example/pager.sh" “” “” “” minor
```

The following example creates the same configuration for SNMPv2c:

```
edgewatch -h 143.45.0.12 -c private -v 2c -o process add procAlive "ypbind" 16 0x00
60 "Monitor ypbind" "/example/pager.sh" “” “” “” fatal
```

The following example creates the same configuration for SNMPv3:

```
edgewatch -h 143.45.0.12 -v 3 -u userName -s 3 -a authPassword -A MD5 -x encryptPassword
-X DES -o process add procAlive "ypbind" 16 0x00 60 "Monitor ypbind"
"/example/pager.sh" “” “” “” major
```

procAlive

Indicates the process attribute being monitored. It instructs the agent to monitor the process to make sure it is running. ypbind is the process that the agent is monitoring. It is responsible for client directory lookups and is necessary for computers running Network Information Services (NIS).

0x00

Instructs the agent to provide the default behavior for this table entry.

If the process dies, the agent sends a processStop trap and runs the action script /example/pager.sh.

Example: Monitor the firefox Process

The following example creates a Process Monitor table entry at index 20 to monitor the firefox process running on the target computer for SNMPv1:

```
edgewatch -h 143.45.0.12 -c private -v 1 -o process add procSize "firefox" 20
0x00a02400 60 absolute ">" 35000 "Monitor firefox size" "" “” “” “” critical
```

The following example creates the same configuration for SNMPv2c:

```
edgewatch -h 143.45.0.12 -c private -v 2c -o process add procSize "firefox" 20
0x00a02400 60 absolute ">" 35000 "Monitor firefox size" "" “” “” “” none
```

The following example creates the same configuration for SNMPv3:

```
edgewatch -h 143.45.0.12 -v 3 -u userName -s 3 -a authPassword -A MD5 -x encryptPassword
-X DES -o process add procSize "firefox" 20 0x00a02400 60 absolute ">" 35000 "Monitor
firefox size" "" “” “” “” ok
```

procSize

Indicates the process attribute being monitored. It instructs the agent to monitor the overall size of the program's text, data, and stack segments.

netscape

Indicates the application that the agent is monitoring.

0x00a02400

Instructs the agent to modify the default Process Monitor table behavior as follows:

0x00000400

Instructs the agent to send processClear traps.

0x00002000

Instructs the agent to send up to 10 consecutive traps and then send no more.

0x00a00000

Contains the flag value 10 for use with this directive.

>

Indicates that an event should occur when the process size of netscape exceeds the threshold (35 MB).

35,000 KB or 35 MB

Indicates the threshold.

Example: Monitor the Windows TCPSVCS Process

The following example creates a Process Monitor table entry at index 15 to monitor the Windows TCPSVCS process (or service) running on the target system for SNMPv1:

```
edgewatch -h 143.45.0.12 -c private -v 1 -o process add procAlive "TCPSVCS" 15 0x00
30 "Monitor NT TCP services" "" “” “” “” fatal
```

The following example creates the same configuration for SNMPv2c:

```
edgewatch -h 143.45.0.12 -c private -v 2c -o process add procAlive "TCPSVCS" 15 0x00
30 "Monitor NT TCP services" "" “” “” “” ok
```

The following example creates the same configuration for SNMPv3:

```
edgewatch -h 143.45.0.12 -v 3 -u userName -s 3 -a authPassword -A MD5 -x encryptPassword
-X DES -o process add procAlive "TCPSVCS" 15 0x00 30 "Monitor NT TCP services" "" “”
“” “” warning
```

procAlive

Indicates the process attribute being monitored. It instructs the agent to scan the Process Monitor table periodically (every 30 seconds) to verify that this process is running.

TCPSVCS

Indicates the Windows service responsible for TCP-related services on Windows systems.

0x00

Instructs the agent to provide the default behavior for this table entry.

Example: Display all of the processes on a system

The following example displays (dumps) all processes on the local host that is running sysedge on port 1691:

```
edgewatch -c private -p 1691 -v 1 -o process dump
```

Remove Process Monitoring Entries

To stop monitoring a process, you must remove the appropriate entry from the Process Monitor table and the sysedge.cf file. Removing an entry using point configuration from the CA Virtual Assurance user interface accomplishes both. Manual removal requires that you remove the entry from both places in separate operations.

To remove process monitor entries in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click Process Monitors.

   The Process Monitor table appears with all existing self monitors.
5. Select the monitor to delete and click Actions, Delete.

   The entry is deleted from the table. The agent performs a warm start to apply the change.

To remove process monitor entries manually

1. Delete the watch process directive for the entry from the local sysedge.cf file and save the file.
2. Run the edgewatch utility using one of the following commands for the -o parameter:

   `-o delete` *`index`*

   `-o setstatus` *`index`* `6`

   index

   Specifies the index value of the process monitor entry to remove.

   When using the setstatus command, a value of 6 sets the row status to destroy and deletes the entry.
3. Restart the agent.

   The entry is deleted.

Solaris Zone Process Monitoring

SystemEDGE monitors a process in a Solaris Zone, if the flag 0x8000 (Solaris Zone process) and the variable pmonRegExpr are set accordingly. The variable pmonRegExpr specifies a regular expression with format *ZoneRegExpr/ProcRegExpr*. For compatibility reasons the variable pmonZoneRegExpr is still available, but allows modification only if the monitor was not initialized from sysedge.cf with the 0x8000 flag.

The variable pmonProcName is available on all platforms. If a Solaris Zone is monitored, this variable has the format *MatchedZoneName/MatchedProcessName*. If Solaris Zones monitoring is not required or not possible (non-Solaris 10 OS), the variable contains *MatchedProcessName* only.

When you want to monitor a process in a Solaris Zone, use the watch process directive with flag 0x8000 and a regular expression with format *ZoneRegExpr/ProcRegExp*r.

During upgrade procedure all watch zone process directives are translated to the new format.

SystemEDGE sends the value of pmonProcName in all process monitor related traps.

Recommendations for Process and Service Monitoring

When you are configuring process and service monitoring, consider the following:

- Monitor the following:
  - Status of the processes:
    - Use the processState (operating system dependent) or processStateStr (operating system independent) OIDs to return the process state. For more information, see empire.asn1.
    - For processes that run through an interpreter such as Perl, set the 0x00000800 flag to match the process name and arguments (UNIX only) when creating process monitoring entries. For more information, see Process Monitor Table Flags (see page 253).
  - CPU time over interval (if this value is not incrementing, the process might be in the zombie state.)
  - Total CPU time (a high value can indicate problems)
  - Leaking memory (RSS over time; use alarm thresholds to notify you of problems)
- Configure SystemEDGE to automatically restart failed processes.
- Set your UNIX application shutdown files to disable process and service monitoring to prevent race conditions.
- Use the Process Group Monitor table to monitor multiple processes with the same name. For more information, see the chapter “Process Group Monitoring.”

Chapter 10: Autowatchers

This section contains the following topics:

How Autowatchers Work (see page 285)
edgewatch Utility--Using Autowatchers (see page 288)
Specify an Autowatcher in sysedge.cf (see page 297)

How Autowatchers Work

Autowatchers run periodic discovery processes using regular expressions as patterns to match the names of resources for which the Autowatchers create monitors. Autowatchers enable SystemEDGE to create automatically monitors for new resources when they come online. Autowatchers create monitors in a reserved range of indexes (1000000 - 1999999).

SystemEDGE sends traps to CA Virtual Assurance when resources disappear and applies the ‘Loss Actions’ that you have configured in the Autowatcher. In case of the loss of the monitored resource, the 'Loss Action' can remove the monitor or can set the status of the resource to a specific status:

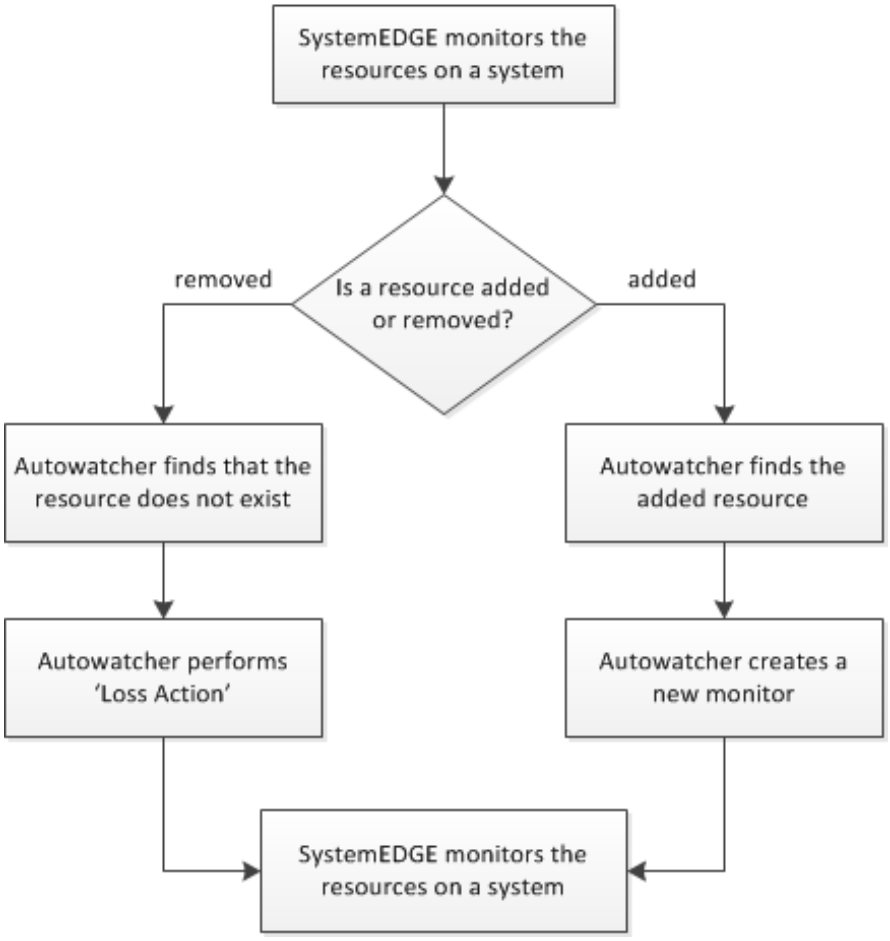
OK, Warning, Minor, Major, Critical, Fatal, Up, or Down

Autowatchers enable you to create flexible Policies or Layered Templates without knowing what resources exist on a managed system. A resource can be a device, a service, or a process running on a managed system.

You can use the following Autowatcher Types:

- Generic Autowatchers – Creates monitors for various resources on a managed system, for example, for devices, interfaces, filesystems, or files.
- Process and Service Autowatchers – Creates monitors for processes and services running on a managed system.

Process Workflow of an Autowatcher



You can use the following guidelines when you configure Loss Actions:

- If the lost resource affects the health of a system, you can configure the Loss Action to change the status of the corresponding resource to a critical state.
- If the lost resource does not affect the health of a system, you can configure the Loss Action to remove the corresponding monitor.

More information:

Generic Autowatchers (see page 287)
Process and Service Autowatchers (see page 287)

Generic Autowatchers

Generic Autowatchers can create monitors for various resources on a managed system, for example, for devices, interfaces, filesystems, or files.

The following list provides you some examples of Generic Autowatchers:

- Capacity of all discovered devices
- Disk service time on all discovered disks
- Resident set size on all cmd processes
- Operating status of all tunneled network interfaces
- Device status of all devices

More information:

How Autowatchers Work (see page 285)

Process and Service Autowatchers

Use Process and Service Autowatchers to create process and service monitors dynamically.

A Service Autowatcher creates multiple service monitors in the process table whenever a service matches an Autowatcher criteria (service name, start type, and so on). For example, you can monitor all installed the SQL services with a start type as ‘automatic’.

A Process Autowatchers create process monitors in two ways:

- Using a process name (default) - When a process name matches the Autowatcher criteria.

  For example, a process monitor is created when a process matches a criteria of process name of 'sql' or 'svchost'. Autowatcher-created process monitors track a matching process that currently runs on a managed system, regardless of PID.

  - Autowatcher-created process monitors have the same semantics as manually created process monitors.
  - You can individually monitor a set of processes with the same name, but different arguments. For example, "java.exe".
  - You can create monitors for a set of related processes.

- Using PID - When a PID matches the Autowatcher Criteria. The Autowatcher enables the Monitor Process using the PID flag in the user interface or specifies the watch flag 0x1000 in the sysedge.cf file.

  Each Autowatcher-created monitor tracks all matching instances of a process.

  - Creates monitors for particular instances of processes.
  - Monitors multiple instances of a process with no distinguishing arguments.

More information:

How Autowatchers Work (see page 285)

edgewatch Utility--Using Autowatchers

The edgewatch command line utility automatically configures the SystemEDGE agent to use Autowatchers for resource monitoring. After you specify the particular autowatcher and the associated arguments, the edgewatch utility issues an SNMP Set request to create the appropriate entry in the MIB table of the target agent. The edgewatch utility is located in the bin subdirectory of the agent installation.

Use the edgewatch utility for Autowatchers as follows:

```
edgewatch [-h hostname | ip _addr] [-p port] [-c community]
    [-v 1 | 2c | 3] [-u secName] [-s secLevel] [-n contextName]
    [-a authPassword] [-A MD5 | SHA]
    [-x privPassword] [-X DES | AES | 3DES]
    [-m FIPS_mode]
    [-r retries]
    [-t timeout] [-d logLevel] [-f logFile]
    -o autowatch command -attribute=value [-attribute=value ...]
```

-h *hostname | ipaddr*

Specifies the host name or IP address of the system on which the agent is running. Accepts IPv4 and IPv6 addresses.

Default: localhost

-p *port*

Specifies the UDP port that the agent is running on (for example, 1691).

Default: 161

-c *community*

Specifies a community string that the agent uses. Valid for SNMPv1 and SNMPv2c only.

Note: Specify a read/write community string for snmpset.

Default: public

-v {1 | 2c | 3}

Indicates the version of SNMP that the agent is running. Specify 1 for SNMPv1, 2c for SNMPv2c, or 3 for SNMPv3.

Default: 1

-u *secName*

Specifies the User-based Security Model (USM) user name that is used for SNMPv3 security.

Default: none

-s *secLevel*

Specifies one of the following security levels for SNMPv3 communication:

- 1 - noAuthNoPriv
- 2 - AuthNoPriv

3 - AuthPriv (SNMPv3 only)

-n *contextName*

Specifies the instance name for the MIBMuxed agent.

Default: none

-a *authPassword*

Specifies the authentication password if the agent is configured for SNMPv3 with secLevel 2 (AuthNoPriv) or 3 (AuthPriv).

Note: This option is not required for SNMPv3 communication.

Default: none

-A {MD5 | SHA}

Specifies the authentication protocol to use if the agent is configured for SNMPv3 with secLevel 2 (AuthNoPriv) or 3 (AuthPriv). Currently only MD5 (Message Digest Algorithm) and SHA (Secure Hash Algorithm) are used.

Default: MD5

-x *privPassword*

Specifies the privacy (encryption) password if the agent is configured for SNMPv3 with secLevel 3 (AuthPriv).

Default: none

-X {DES | AES | 3DES}

Specifies the privacy protocol if the SNMPv3 user is configured with secLevel 3 (AuthPriv). Specify DES for Data Encryption Standard, AES for Advanced Encryption Standard using cryptographic keys of 128 bits (AES128), and 3DES for Triple Data Encryption Standard.

Default: none

-m *FIPS_mode*

Controls the FIPS mode of operation. Accepted values are 0, 1, and 2.

0

Indicates Non-FIPS mode.

1

Indicates FIPS co-existence mode.

2

Indicates FIPS only mode.

Default: 1

-r *retries*

Specifies the number of retries.

Default: 10

-t *timeout*

Specifies the duration before the SNMP receiver considers the request as timed out.

Default: 10 seconds

-d *logLevel*

Specifies the log level of the SNMP messages. Accepted values are 0 to 5.

0

Logs fatal messages.

1

Logs critical messages.

2

Logs warning messages.

3

Logs informational messages.

4

Logs all of the messages.

5

Logs all of the messages including debugging messages.

Default: 0

-f *logfile*

Specifies the name of the log file that contains error and debug information.

Default: sysedge_utility.log

command

Specifies the command and associated attributes. Supported commands are the following:

- add -index=[watchIndex] [additional attributes ...]
- setstatus -index=[watchIndex] -rowstatus=[active | notInService | destroy]
- delete -index=[watchIndex]
- list

Note: For more information about the commands, see edgewatch Commands for Autowatchers (see page 292).

edgewatch Commands for Autowatchers

The edgewatch autowatch commands and associated arguments are as follows:

```
add -index=[watchIndex] -attribute=value [-attribute=value ...]
setstatus -index=[watchIndex] -rowstatus=[active | notInService | destroy]
delete -index=[watchIndex]
list
```

All of the attributes are specified in '-attribute=value' format and are not positionally dependent.

Mandatory attributes for the add command:

-name

Specifies the name for the Autowatcher entry.

-watchtype

Specifies one of the following types: generic, process, or service.

-criteria

Specifies a regular expression for the primary table criteria for selecting instances to monitor in single quotes.

-severity

Specifies one of the following severities to use for the object state model: none, ok, warning, minor, major, critical, or fatal.

Additional mandatory attributes for add -watchtype=generic:

-table

Specifies the table entry that is monitored (for example, devTableEntry).

-attribute

Specifies the table column attribute that is monitored (for example, devCapacity).

-op

Specifies the comparison operator: nop, gt, lt, ge, le, eq, or ne.

-value

Specifies the threshold for comparison.

Additional mandatory attributes for add -watchtype=process:

-attribute

Specifies the one of the following values: procAlive, procMem, procSize, procRSS, procTime, procInBlks, procOutBlks, procMsgsSent, procMsgsRecv, procNice, procNumThreads, procNumSwaps, procSysCalls, procMinorPgFlts, procMajorPgFlts, procVolCtx, procInvolCtx, or procTimePermil.

-op

Specifies the comparison operator: nop, gt, lt, ge, le, eq, or ne. Not required for procAlive.

-value

Specifies the threshold for comparison.

Optional attributes:

-interval

Specifies the evaluation interval for generated monitors (default: 60)

-desc

Specifies the description that is applied to all generated monitors in single quotes.

-monflags

Specifies the hex integer (for example, 0x0000) applied to all generated monitors (default: 0x0000):

0x0001 - Do not execute actions.

0x0002 - Do not send traps.

0x0004 - Do not reinitialize entry.

0x0008 - Do not send syslog events.

0x0010 - Send continuous notReady traps.

0x0020 - Do not send SystemEDGE arguments to action scripts.

0x0040 - Do not send notReady traps for this entry.

-watchtype = generic:

0x0100 - Send clear traps.

0x0200 - Send threshold traps after X breaches.

0x0400 - Send at most X consecutive threshold traps 0xfff00000 = value of X.

-watchtype = process or service:

0x0200 - Do not send start traps.

0x0400 - Send clear traps.

0x0800 - Match process name and arguments.

0x1000 - Send threshold/stop traps after X breaches.

0x2000 - Send at most X consecutive threshold/stop traps.

0x8000 - Monitor process in a Solaris zone.

0xfff00000 - Value of X.

-matchtype

Specifies one of the following match types for the primary selection criteria: positive, negative (default: positive).

-criteria2

Specifies the column in the monitored table for secondary selection criteria.

-criteria2regex

Specifies the regular expression for the secondary selection criteria in single quotes. (default: '.*')

-matchtype2

Specifies one of the following match types for the secondary selection criteria: positive, negative (default: positive).

-criteria3

Specifies the column in the monitored table for tertiary selection criteria.

-criteria3regex

Specifies the regular expression for the tertiary selection criteria in single quotes. (default: '.*')

-matchtype3

Specifies one of the following match types for the tertiary selection criteria: positive, negative (default: positive).

-stype

Specifies the one of the following sample types to apply to all generated monitors: absolute, delta (default: absolute).

-scale

Specifies the scale value to apply to all generated monitors (default: 1).

-action

Specifies the full path of the command (action) in single quotes (default: no action is configured).

-objclass

Specifies the name of the object class for the object state model.

-objattr

Specifies the name of the object attribute for the object state model.

-watchflags

Specifies the hex integer (for example, 0x0000) to modify the behaviour of the autowatcher (default: 0x0000):

0x0002 - Do not send autowatcher traps.

0x0004 - Do not reinitialize entry.

0x0008 - Do not send syslog events.

0x0010 - Send continuous notReady autowatcher traps.

0x0040 - Do not send notReady traps for this entry.

0x1000 - Monitor processes by PID (only for -watchtype=process).

-scaninterval

Specifies the interval between discovery scans of the monitored table (default: 300).

-limit

Specifies the maximum number of generated monitors for this autowatcher entry (default: 100).

-loss

Specifies one of the following loss actions when a generated monitor can no longer monitor its object: remove, ok, warning, minor, major, critical, fatal, up, or down (default: fatal).

-starttype

Specifies the start type of the Windows Service. This attribute only applies to -watchtype=service. Use one of the following values: automatic, manual, disabled (default: all types).

setstatus

Specifies the status textual convention value to use in setting the status of a row in the MIB table when used with the setstatus operation. Valid values are as follows. Values can be either the assigned integer values or the actual spelled out status text:

- active(1)
- notInService(2)
- destroy(6)

edgewatch Autowatcher Examples

This section provides examples for using the edgewatch utility with SNMP versions 1, 2c, and 3 to create Autowatchers.

Examples:

Add a generic autowatcher entry (at table index 13) which generates devCapacity monitors for all discovered filesystems with a threshold of 90:

```
edgewatch -c private -o autowatch add -index=13 -watchtype=generic
          -name='All FileSystems - Minor' -table=devTableEntry
          -attribute=devCapacity -interval=120 -criteria='.*' -op=ge
          -value=95 -severity=minor -desc='FileSystem Usage (Percentage)'
          -objclass=FileSystem -objattr=PercentUsed
```

Add a process autowatcher entry (at table index 15) which generates procRSS monitors for all discovered SQL processes and automtically removes these monitors when the processes closes:

```
edgewatch -c private -o autowatch add -index=15 -watchtype=process
          -name='SQL Processes RSS' -criteria='.*sql.*'
          -attribute=procRSS -op=ge -value=100000 -severity=major
          -interval=60 -desc='SQL processes' -objclass='Process'
          -objattr='ResidentSet' -loss=remove
```

Add a service autowatcher entry (at table index 17) which generates a service procAlive watcher for all services which are configured for automatic start:

```
edgewatch -c private -o autowatch add -index 17 -watchtype=service
          -name='All Automatic Services' -criteria='.*' -severity=major
          -interval=300 -limit=200 -desc='Automatic Services'
          -objclass='Service' -objattr='Running' -starttype=automatic
```

List all defined autowatch entries:

```
edgewatch -c private -o autowatch list
```

Delete the previously specified service autowatcher entry:

```
edgewatch -c private -o autowatch delete -index=17
```

Set the row status of the previously specified process autowatcher to notInService:

```
edgewatch -c private -o autowatch setstatus -index=15 -rowstatus=notInService
```

Specify an Autowatcher in sysedge.cf

Autowatchers create entries in the monitor and process monitor MIB tables for resources that match the specified autowatcher criteria. The autowatcher directives specify the criteria for the automatically generated monitors.

The syntax for autowatchers is switch-based and does not rely on positional parameters. All parameters have the form:

```
-parameter=value
-parameter='value'
```

Optional parameters can be omitted. Use single quotes if the value contains spaces.

The autowatcher directive has the following format:

```
autowatcher -index=index -watchtype=type -table=table -attribute=attr ...
```

Mandatory parameters:

-index

Specifies the table index that is used to store the autowatcher.

-name

Specifies the name of the autowatcher.

-watchtype

Specifies the type of autowatcher to create (generic, service, or process).

-table

Specifies the table entry that is monitored (for example, devTableEntry). Not required for service or process autowatchers.

-attribute

Specifies the attribute to monitor (for example, devCapacity). Not required for service autowatchers.

-criteria

Specifies a regular expression for the primary criteria for selecting instances to monitor.

-op

Specifies the comparison operator (nop, gt, lt, ge, le, eq, ne).

-value

Specifies the threshold (not required for process procAlive, or service autowatchers).

-severity

Specifies the severity to use for the object state model (none, ok, warning, minor, major, critical, fatal).

Optional parameters:

-interval

Specifies the evaluation interval for all monitors that this autowatcher generates.

Default: 60

-desc

Specifies the description to apply to all monitors that this autowatcher generates.

-monflags

Specifies the set of self-monitor flags to apply to all monitors that this autowatcher generates.

-matchtype

Specifies the match type for the primary selection criteria (positive or negative).

-criteria2

Specifies a column in the monitored table for secondary selection criteria.

-criteria2regex

Specifies a regular expression for the secondary instance selection criteria.

-matchtype2

Specifies the match type for the secondary selection criteria (positive or negative).

-criteria3

Specifies a column in the monitored table for tertiary selection criteria.

-criteria3regex

Specifies a regular expression for the secondary instance selection criteria.

-matchtype3

Specifies the match type for the secondary selection criteria (positive or negative).

-stype

Specifies the sample type to apply to each monitor (absolute, delta).

-scale

Specifies the scale factor for gathered monitor values. A scale factor allows the monitoring of variables in a different scale of the unit in which the variable is originally measured.

With a scale factor of 1024, you monitor the values of variables in Kilobytes, which are originally measured in bytes.

This scaling factor monitors the variables monCurrVal, monVal, monMinValue, and monMaxValue in Kilobytes instead of bytes.

Default: 1

-action

Specifies the full path (including parameters) of a command to execute concurrent to sending a trap for each monitor.

-objclass

Specifies the object class to use for the object state model ('' is replaced by default value; '*' is invalid).

-objattr

Specifies the object attribute to use for the object state model ('' is replaced by default value; '*' is invalid).

-watchflags

Specifies the flags that modify the behavior of the autowatcher.

-scaninterval

Specifies the interval in seconds between scans of the monitored table.

-limit

Specifies the maximum number of monitors that this autowatcher can generate.

-loss

Specifies the loss action to take when a monitor can no longer reach the monitored object (remove, ok, warning, minor, major, critical, fatal, up, down).

Flags valid for all autowatchers

000 000 02 - Do not send any traps, but log traps.

000 000 04 - Do not reinitialize a notReady autowatcher.

000 000 08 - Do not log any traps, but send traps.

000 000 10 - Keep sending notReady traps.

000 000 40 - Do not send notReady traps (including log).

000 000 80 - An AIM plugin has created this entry. This flag is read-only.

More information:

Generic Autowatcher Examples (see page 300)
Service Autowatcher Example (see page 301)
Process Autowatcher Examples (see page 301)

Generic Autowatcher Examples

The following examples specify generic autowatchers for the sysedge.cf file:

- Monitor capacity on all discovered devices.
- Monitor disk service time on all discovered disks.
- Monitor the resident set size on all cmd processes.
- Monitor the operating status of all tunneled network interfaces.
- Monitor the device status of all devices.

```
# Device Table - Monitor capacity on all discovered devices
# (minor at 50%, major at 75%)
autowatcher -watchtype=generic -index=11 -table=devTableEntry
-name='Device Table Autowatcher' -criteria='.*' -attribute=devCapacity -op=ge
-value=50 -severity=minor -desc='Device capacity (percentage)' -monflags=0x0
-limit=10 -interval=60 -objclass='Disk' -objattr='Capacity'
autowatcher -watchtype=generic -index=12 -table=devTableEntry
-name='Device Table Autowatcher' -criteria='.*' -attribute=devCapacity -op=ge
-value=75 -severity=major -desc='Device capacity (percentage)' -monflags=0x0
-limit=10 -interval=60 -objclass='Disk' -objattr='Capacity'
# Disk Statistics Table - Monitor disk service time on all discovered disks
autowatcher -watchtype=generic -index=101 -table=diskStatsEntry
-name='Disk Stats Autowatcher' -criteria='.*' -attribute=diskStatsServiceTime
-op=ge -value=100 -severity=major -desc='Disk service time' -interval=60
-objclass='DiskStats' -objattr='ServiceTime'
```

```
# Process Table - Monitor the resident set size on all cmd processes
# lossAction example
autowatcher -watchtype=generic -index=202 -table=processEntry
-name='Process Table Autowatcher' -criteria='cmd' -attribute=processRSS
-op=ge -value=10000 -severity=major -desc='cmd process RSS (KB)' -interval=30
-objclass='Process' -objattr='process RSS' -scaninterval=30
# MIB-2 Interface Table - Monitor the operating status of all tunneled network
# interfaces - multiple criteria example
autowatcher -watchtype=generic -index=303 -table=ifEntry
-name='WiFi Interface Autowatcher' -criteria='.*' -criteria2=ifType
-criteria2regex='131' -attribute=ifOperStatus -op=eq -value=2
-desc='tunnel interface operating status' -severity=critical -objclass=Network
-objattr=OperatingStatus
# Host Resource Device Table - Monitor the device status of all devices
# dual-column instance example
autowatcher -watchtype=generic -index=404 -table=hrDeviceEntry
-name='HR Device Autowatcher' -criteria='.*' -attribute=hrDeviceStatus
-op=ne -value=2 -desc='HR device status' -severity=fatal -objclass=Hardware
-objattr=OperatingStatus
```

Service Autowatcher Example

The following example specifies a service autowatcher for the sysedge.cf file. The autowatcher creates multiple service monitors in the process table that match the specified criteria (service name, start type, and so on).

- Monitor all installed SQL services with start type of ‘automatic’.

```
autowatch -index=12 -watchtype=service -name='All Automatic SQL Services'
-criteria='.*SQL.*' -starttype=automatic -interval=60 -severity=critical
-desc='SQL services' -objclass='Service' -objattr='Running'
```

Process Autowatcher Examples

The following examples specify process autowatchers for the sysedge.cf file:

- Ensure that none of the eTrust processes exceed 5 percent processor utilization.

```
autowatch -index=22 -watchtype=process -name='eTrust Processes'
-criteria='^Ino.*' -attribute=procTimePermil -op=ge -value=50000 -severity=major
-interval=60 -desc='eTrust processes' -objclass='Process' -objattr='CPU_Usage'
```

- Ensure that all SVCHOST processes use less than 10-MB real memory.

```
autowatch -index=23 -watchtype=process -name='Svchost Processes'
-criteria='^svchost' -attribute=procRSS -op=ge -value=10000 -severity=major
-interval=60 -monflags=0x0800 -desc='Svchost processes' -objclass='Process'
-objattr='ResidentSet'
```

Chapter 11: Process Group Monitoring

This chapter explains how to use the SystemEDGE agent to monitor groups of processes. The agent uses process groups to aggregate the per-process information into a single, easy to poll, easy to monitor value.

This section contains the following topics:

Process Group Monitoring Overview (see page 303)
Process Group Monitor Table (see page 304)
Process Group Monitoring Configuration (see page 313)
edgewatch Utility--Monitor Process Groups (see page 319)
Remove Process Group Monitoring Entries (see page 324)
Solaris Zone Process Group Monitoring (see page 325)

Process Group Monitoring Overview

The SystemEDGE agent lets you dynamically monitor groups of processes running on the underlying system. You use the Process Group Monitor table to specify the process group, regular expression, interval, and severity, and the agent uses that information to monitor the group of processes that match the regular expression.

The SystemEDGE agent monitors process groups to determine what processes exist in each group and whether the group membership changes. If components of an application start or fail, or if members leave a group or are added to a group, the SystemEDGE agent can automatically notify a management system with a trap.

The agent also populates read-only columns of the Process Group Monitor table with combined metric values for the process group. You can create self monitor entries to monitor these attributes based on defined thresholds.

Note: For information about monitoring individual processes and Windows services, see the chapter “Process and Service Monitoring.”

Process Group Monitor Table

The Process Group Monitor table provides information about each group of processes that the agent is currently monitoring. Each row represents a defined process group monitor entry.

For each entry, the table provides the following types of information:

- List of processes in a group
- Interval at which the agent checks the group
- Values for process attributes using the group sum
- Monitor severity

Process Group Monitor Table Columns

The following list describes the columns of the Process Group Monitor table. For a complete description of the Process Group Monitor table and its fields, see the Systems Management Empire MIB specification (empire.asn1 in the mib subdirectory of the agent installation).

Note: SystemEDGE maintains a history of group membership and tracks process statistics even after an individual process has left the group.

pgmonIndex

Defines the unique row index for this entry (1 to MAXINT). Rows 1 through 10 are reserved for the agent's internal use; the index for additional rows must fall in the range of 11 to MAXINT.

Permissions: Read-only

pgmonDescr

Defines a description (0 to 512 characters in length) of the entry.

Permissions: Read-write

pgmonInterval

Defines how often in seconds the agent should monitor the group. The value must be a minimum of 30 and a multiple of 30 seconds.

Permissions: Read-write

Default: 60

pgmonProcRegExpr

Defines the regular expression to use when the agent is attempting to match processes by name.

Permissions: Read-write

pgmonFlags

Defines the positive integer flags that indicate additional behavioral semantics that the entry follows during the course of its operation. For more information about available flags and setting flags, see Process Group Monitor Table Flags (see page 310).

Permissions: Read-write

pgmonNumProcs

Defines the current number of processes in the process group that the entry is tracking. A process belongs to the process group if its name (and possibly, its arguments) matches the regular expression configured for this entry.

Permissions: Read-only

pgmonPIDList

Defines the numeric PIDs in the process group. Each PID is separated from the next by a space character.

Permissions: Read-only

pgmonStatusList

Defines the process status for each process in the process group. Each process state is separated from the next with a space character. Entries in the status list have a one-to-one correspondence with entries in the PID list. For more information about states of a process, see the processStateStr MIB variable.

Permissions: Read-only

pgmonAction

Defines a quoted command (0 to 2048 characters in length) with any parameters to run when the expression evaluates to True and the agent sends a process group change trap. If the string is empty, the agent invokes no action for this entry.

Note: Do not use Windows batch files for actions; they impose severe programmatic limitations and often do not work correctly with desktop applications. Instead, use a more powerful and flexible scripting language, such as Perl or Visual Basic.

Permissions: Read-write

pgmonNumEvents

Defines the number of events generated for the entry. Events do not necessarily imply traps; traps can be turned off for the row through a flags setting.

Permissions: Read-only

pgmonNumTraps

Defines the number of traps (procGroupChangeTrap) that the agent has sent for this entry.

Permissions: Read-only

pgmonLastTrap

Defines the time (based on sysUpTime) at which the agent last sent a trap (procGroupChangeTrap) for this entry.

Permissions: Read-only

pgmonRowStatus

Defines the row status, which can be one of the following:

- active
- notInService
- notReady
- createAndGo
- createAndWait

Typically, a row is either active or notInService. These values are identical in meaning to those defined by the SNMPv2 SMI RowStatus textual convention.

Permissions: Read-write

Default: createAndWait(5)

pgmonRSS

Defines the combined resident set size (RSS) of the group of processes. The RSS for each process in the group is summed at each interval and stored in this variable. Because RSS also includes shared memory, the total RSS for a group could exceed total possible physical memory for the underlying system. For more information, see the processRSS variable of the Systems Management MIBSystems Management Empire MIB**missions:** Read-only

pgmonSize

Defines the combined size of the text, data, and stack segments of the group of processes. The size of each process in the group is summed at each interval and stored in this variable. Size includes shared memory, so the total size for a group could exceed the total virtual memory for the underlying system. For more information, see the processSize variable of the Systems Management MIBSystems Management Empire MIB**missions:** Read-only

pgmonThreadCount

Defines the total number of threads for the group of processes. The number of threads running in each process in the group is summed at each interval and stored in this MIB variable. For more information, see the processNumThreads variable of the Systems Management Empire MIB (in empire.asn1).

Permissions: Read-only

pgmonMEM

Defines the total percentage of real memory being used by the processes in this group. The percentage of memory being used is summed at each interval and stored in this MIB variable. Memory usage includes shared memory (shared libraries and DLLs), so the total percentage may exceed 100.

Permissions: Read-only

pgmonInBlks

Defines the number of blocks of data input by processes in this group.

Permissions: Read-only

pgmonOutBlks

Defines the number of blocks of data output by processes in this group.

Permissions: Read-only

pgmonMsgsSent

Defines the number of messages sent by processes in this group.

Permissions: Read-only

pgmonMsgsRecv

Defines the number of messages received by processes in this group.

Permissions: Read-only

pgmonSysCalls

Defines the number of system calls invoked by processes in this group.

Permissions: Read-only

pgmonMinorPgFlts

Defines the number of minor page faults incurred by processes in this group.

Permissions: Read-only

pgmonMajorPgFlts

Defines the number of major page faults incurred by processes in this group.

Permissions: Read-only

pgmonNumSwaps

Defines the number of times processes in this group have been swapped.

Permissions: Read-only

pgmonVolCtx

Defines the number of voluntary context switches incurred by processes in this group.

Permissions: Read-only

pgmonInvolCtx

Defines the number of involuntary context switches incurred by processes in this group.

Permissions: Read-only

pgmonCPUSecs

Defines the number of seconds of CPU time used by processes in this group.

Permissions: Read-only

pgmonMatchUser

Matches running processes by user name and any process name regular expression when set to a valid user name. This variable is valid only on UNIX systems.

Permissions: Read-write

pgmonMatchGroup

Matches running processes by group name, process name regular expression, and user name when set to a valid group name. This variable is valid only on UNIX systems.

Permissions: Read-write

pgmonProcNameList

The list of process names of the monitored processes. On Solaris 10 or newer the fromat of the name is ZoneName/ProcName if Zone regular expression has been used.

Permissions: Read-only

Row Creation Objects

The pgmonIndex column of the Process Group Monitor table acts as a key field (or row index) to distinguish rows in the table. Rows 1 through 10 are reserved for internal use by the SystemEDGE agent. You can configure rows in the range of 11 to MAXINT.

You can use the following MIB objects with the Process Group Monitor table to optimize row creation:

pgmonUnusedIndex

Returns an unused index number for the Process Group Monitor table when you perform an SNMP Get on the object. Query this object to obtain a number that you can use for row creation.

pgmonMatchDescr

Determines the index number that corresponds to a particular entry description in the pgmonDescr attribute. Perform an SNMP Set on this MIB object with a pgmonDescr attribute value to cause the agent to search through entries in the Process Group Monitor table and put the index value of the last matching entry in the monMatchIndex MIB object.

pgmonMatchIndex

Matches a particular entry description with its index number when used with pgmonMatchDescr.

You may choose, as a matter of local policy, to reserve a block of rows for system administration. You can then define entries within a reserved block of rows without being concerned that the row may already be taken by another user's entry. In compliance with the local policy, all other users should use row indices outside the reserved range when they define user-configured entries.

By reserving a block of rows, you can define a consistent set of conditions (row entries) to be monitored across all computers such that the same condition is defined in the same row number on each computer. For example, you can use row 11 (monIndex = 11) to define an entry for monitoring the swapCapacity variable, and you can distribute this configuration to every system so that row 11 monitors the swapCapacity variable on every system.

Configuring the agent with CA Virtual Assurance can automate this process. You can configure a block of entries with specific row numbers and deploy that configuration to all managed systems in one operation. Also, CA Virtual Assurance detects if you define a row that is already in use and disallows the operation.

For more information about defining rows in CA Virtual Assurance, see the CA Virtual Assurance documentation.

Process Group Monitor Table Flags

The pgmonFlags column in the Process Group Monitor table is a 32-bit unsigned integer field that can specify additional behavioral semantics for the corresponding row.

By default, the Process Group Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps
- Logs events to the common log file
- Invokes actions (if they are configured)

You can set different flag bits to alter these defaults. The agent interprets all flags in hexadecimal (base 16) notation.

The Process Group Monitor Table flags (pgmonFlags) are as follows:

The flags value consists of three fields:

- Field 1 contains common table flags defined for the monitoring tables of the Systems Management Empire MIB. This portion is the low-order 8 bits of the flag.
- Field 2 contains table-specific flags defined separately for each of the monitoring tables. This field defines the next 12 low-order bits after the common table flags.
- Field 3 contains 12 reserved high-order bits for an integer value for use with table-specific flags. This field defines the flags that are specific to the Process Group Monitor table.

The following sections explain each flag bit. You can combine flag values through a logical OR operation. One flag is specific to the Process Group Monitor table: 0x00100. This flag instructs the SystemEDGE agent to match the process name and arguments for this entry.

The following list describes the Process Monitor table flags:

0x00000001

Disables execution of actions for this entry.

0x00000002

Disables sending of SNMP traps for this entry.

0x00000004

Disables attempts to reinitialize this entry. By default, the agent periodically tries to reinitialize this entry by scanning the process table to determine the new process ID if the target process has been restarted.

0x00000008

Disables logging of traps for this entry in the sysedge.log file. Setting this bit does not affect sending trap. Disabling event logging is useful when events occur frequently or when a particular entry is used as an agent heartbeat.

0x00000020

Disables the passing of default arguments to action scripts or programs. SystemEDGE typically passes default action parameters that indicate the trap type, description field, and so on. For more information about action parameters, see Process Group Monitor Table Action Parameters (see page 312).

0x00000040

Disables sending of notReady traps for this entry.This includes to not log and to not execute actions for notReady trap.

0x00000100

Matches the process name and arguments for this entry.

0x00008000

Matches processes in Solaris Zone. The pgmonProcRegExpr attribute is expected in the form ZoneRegExpr/ProcRegExpr.

Process Group Monitor Table Action Parameters

The SystemEDGE agent provides several default parameters to the action commands when they are invoked. These parameters are in addition to any parameters that you specify in the action string and are passed on the command line after those that you specify. The default parameters are the same as the parameters provided in the SNMP traps that are sent for the Process Group Monitor table.

Note: You can prevent the agent from including these action parameters in the command by setting the 0x00000020 flag in the table entry.

The following list describes the default parameters for Process Group Monitor table actions:

pgmonIndex

Defines the index from the table entry.

pgmonDescr

Defines the description from the table entry.

pgmonFlags

Defines the flags associated with the table entry. The value passed to the action script is in base 16 (hexadecimal) notation with a leading 0x to indicate that it is a hexadecimal number (for example, 0x00000020).

pgmonNumProcs

Defines the number of processes that the entry is monitoring.

pgmonRegExpr

Defines the regular expression that the agent uses to find the process ID of the process to monitor as a group.

pgmonRowStatus

Defines the table entry row status.

pgmonPIDList

Defines the list of process IDs (PIDs) of the processes currently being monitored.

pgmonStatusList

Defines the status of each process currently being monitored.

pgmonSeverity

Defines the severity assigned to the table entry.

For more information about traps sent by the SystemEDGE agent, see the chapter "Concepts" and the appendix “Private Enterprise Traps.”

View the Process Group Monitor Table

You can use CA Virtual Assurance to view the contents of a managed agent's Process Group Monitor table.

To view the Process Group Monitor table in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click Process Group Monitors.

   The Process Group Monitor table appears with all existing process group monitors.

Process Group Monitoring Configuration

You can control which processes and process attributes the SystemEDGE agent monitors by adding, deleting, or modifying the entries in the Process Group Monitor table.

You can configure the Process Group Monitor table in one of these ways:

Dynamically using file-based configuration from CA Virtual Assurance

CA Virtual Assurance provides the ability to configure agents installed on any system that it manages from the CA Virtual Assurance user interface. This is the recommended method of configuration, because it provides the ability to dynamically define and deliver monitors with a user-friendly and secure mechanism that does not require an agent restart. You can configure monitoring for an individual agent or define monitoring policies to apply on multiple managed systems. When you make an individual configuration change or apply a configuration policy, CA Virtual Assurance delivers a new configuration file to the agent system. CA Virtual Assurance can serve as the central point of agent configuration across your data center, and you can overwrite or deny changes made through other methods.

For more information about configuring the agent using CA Virtual Assurance, see the CA Virtual Assurance documentation.

Dynamically using SNMP Set commands from a management system, such as eHealth AdvantEDGE View, CA Spectrum, or CA NSM

Various management systems support adding monitors to SystemEDGE through SNMP Set commands. eHealth provides this functionality through the AdvantEDGE View interface, CA NSM enables configuration through an Agent View, and CA Spectrum enables configuration through OneClick. All features provided by dynamic configuration through CA Virtual Assurance are not available through these methods.

For more information about SNMP Set configuration, see the documentation for the appropriate management system.

Manually by specifying the entries for the Self Monitor table in the local sysedge.cf file

Manual configuration requires you to add entries to the local sysedge.cf file using specific directives. When you configure the agent directly from the sysedge.cf file, a full restart is required for changes to take effect. Use the sysedge.cf file located in the agent data directory for direct configuration.

This section describes how to configure self monitoring directly in the local sysedge.cf file, but it describes the properties that are required for all forms of configuration.

watch procgroup Directive--Add Entries to the Process Group Monitor Table

The watch procgroup directive lets you add process group monitor entries to the Process Group Monitor table directly in the sysedge.cf file. The arguments represent columns in the Proces Group Monitor table.

Add a line to the sysedge.cf file in the agent data directory using the syntax described below, save the file, and restart the agent for the change to take effect.

Use the watch procgroup directive to add entries to the Process Group Monitor table as follows:

```
watch procgroup 'regexpr' index flags interval ['descr'] ['action'] [severity]
```

'regexpr'

Specifies the regular expression to use when attempting to acquire process IDs for the group to monitor.

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For more information about available flags, see Process Group Monitor Table Flags (see page 310).

interval

Specifies how often in seconds the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. The value must be a minimum of 30 and a multiple of 30 seconds.

'*descr*'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'action'

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when a match is found for the process group. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

severity

Specifies the severity to assign to the entry when a group changes and a trap is sent. The severity is included with the trap. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

Note that the severity designation only specifies the importance of the monitor and is not used to calculate status.

Default: none

watch procgroupex Directive--Add Entries to the Process Group Monitor Table (UNIX only)

The watch procgroupex directive lets you add process group monitor entries to the Process Group Monitor table directly in the sysedge.cf file. The arguments represent columns in the Proces Group Monitor table.

Add a line to the sysedge.cf file in the agent data directory using the syntax described below, save the file, and restart the agent for the change to take effect.

Use the watch procgroupex directive to add entries to the Process Group Monitor table as follows:

```
watch procgroupex 'regexpr' user group index flags interval ['descr'] ['action']
[severity]
```

'regexpr'

Specifies the regular expression to use when attempting to acquire process IDs for the group to monitor.

user

Specifies the user name to match in addition to any process name regular expression (UNIX specific).

group

Specifies the group name to match in addition to any process name regular expression (UNIX specific).

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For more information about available flags, see Process Group Monitor Table Flags (see page 310).

interval

Specifies how often in seconds the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. The value must be a minimum of 30 and a multiple of 30 seconds.

'*descr*'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'action'

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when a match is found for the process group. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

severity

Specifies the severity to assign to the entry when a group changes and a trap is sent. The severity is included with the trap. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

Note that the severity designation only specifies the importance of the monitor and is not used to calculate status.

Default: none

Process Group Monitoring Examples

This section contains sample configuration file directives for process group monitoring. You can add these entries using watch procgroup in the sysedge.cf file.

Example: Monitor the xterm Process Group

The following example configures the agent to monitor the xterm process group:

```
watch procgroup 'xterm.*' 101 0x00 60 'Watch xterms' '' warning
```

101

Indicates that this entry will occupy row 101 (pgmonIndex=101) in the Process Group Monitor table.

0x00

Indicates the default behavior.

60

Indicates that the agent should check the xterm process group every 60 seconds.

warning

Indicates that a process group change trap will contain a warning severity.

No action is specified, so the agent invokes no command when it sends a trap.

Example: Monitor the emacs Process Group

The following example configures the agent to monitor the emacs process group:

```
watch procgroup 'emacs.*|xmibmgr' 12 0x00 60 'Watch emacs' '' major
```

12

Indicates that this entry will occupy row 12 (pgmonIndex=12) in the Process Group Monitor table.

0x00

Indicates the default behavior.

60

Indicates that the agent should check the emacs process group every 60 seconds.

major

Indicates that a process group change trap will contain a major severity.

No action is specified, so the agent invokes no command when it sends a trap.

Example: Monitor the DT Process Group

The following example configures the agent to monitor the DT process group:

```
watch procgroup 'dt/bin' 13 0x00 60 'Watch DT stuff' ''
```

13

Indicates that this entry will occupy row 13 (pgmonIndex=13) in the Process Group Monitor Table.

0x00

Indicates the default behavior.

60

Indicates that the agent should check the DT process group every 60 seconds.

No action is specified, so the agent invokes no command when it sends a trap.

More Information

Perl Compatible Regular Expression (PCRE) Support (see page 150)
Configure Text Pattern Exclusion (see page 152)
Regular Expression Examples (see page 152)

edgewatch Utility--Monitor Process Groups

The edgewatch command line utility automatically configures the SystemEDGE agent to monitor processes, process groups, log files, and Windows event logs. After you specify the particular process, process group, log file or Windows event log and the associated arguments, the edgewatch utility issues an SNMP Set request to create the appropriate entry in the target agent's appropriate monitoring table. The edgewatch utility is located in the bin subdirectory of the agent installation.

Use the edgewatch utility for process group monitoring as follows:

```
edgewatch [-h hostname | ip _addr] [-p port] [-c community]
    [-v 1 | 2c | 3] [-u secName] [-s secLevel] [-n contextName]
    [-a authPassword] [-A MD5 | SHA]
    [-x privPassword] [-X DES | AES | 3DES]
    [-m FIPS_mode]
    [-r retries]
    [-t timeout] [-d logLevel] [-f logFile]
    -o procgroup command
```

-h *hostname | ipaddr*

Specifies the host name or IP address of the system on which the agent is running. Accepts IPv4 and IPv6 addresses.

Default: localhost

-p *port*

Specifies the UDP port that the agent is running on (for example, 1691).

Default: 161

-c *community*

Specifies the community string that edgewatch uses in its SNMP requests to the agent. Valid on SNMPv1 and SNMPv2c.

Default: public

-v {1 | 2c | 3}

Indicates the version of SNMP that the agent is running. Specify 1 for SNMPv1, 2c for SNMPv2c, or 3 for SNMPv3.

Default: 1

-u *secName*

Specifies the User-based Security Model (USM) user name that is used for SNMPv3 security.

Default: none

-s *secLevel*

Specifies one of the following security levels for SNMPv3 communication:

- 1 - noAuthNoPriv
- 2 - AuthNoPriv

3 - AuthPriv (SNMPv3 only)

-n *contextName*

Specifies the instance name for the MIBMuxed agent.

Default: none

-a *authPassword*

Specifies the authentication password if the agent is configured for SNMPv3 with secLevel 2 (AuthNoPriv) or 3 (AuthPriv).

Note: This option is not required for SNMPv3 communication.

Default: none

-A {MD5 | SHA}

Specifies the authentication protocol to use if the agent is configured for SNMPv3 with secLevel 2 (AuthNoPriv) or 3 (AuthPriv). Currently only MD5 (Message Digest Algorithm) and SHA (Secure Hash Algorithm) are used.

Default: MD5

-x *privPassword*

Specifies the privacy (encryption) password if the agent is configured for SNMPv3 with secLevel 3 (AuthPriv).

Default: none

-X {DES | AES | 3DES}

Specifies the privacy protocol if the SNMPv3 user is configured with secLevel 3 (AuthPriv). Specify DES for Data Encryption Standard, AES for Advanced Encryption Standard using cryptographic keys of 128 bits (AES128), and 3DES for Triple Data Encryption Standard.

Default: none

-m *FIPS_mode*

Controls the FIPS mode of operation. Accepted values are 0, 1, and 2.

0

Indicates Non-FIPS mode.

1

Indicates FIPS co-existence mode.

2

Indicates FIPS only mode.

Default: 1

-r *retries*

Specifies the number of retries.

Default: 10

-t *timeout*

Specifies the duration before the SNMP receiver considers the request as timed out.

Default: 10 seconds

-d *logLevel*

Specifies the log level of the SNMP messages. Accepted values are 0 to 5.

0

Logs fatal messages.

1

Logs critical messages.

2

Logs warning messages.

3

Logs informational messages.

4

Logs all of the messages.

5

Logs all of the messages including debugging messages.

Default: 0

-f *logfile*

Specifies the name of the log file that contains error and debug information.

Default: sysedge_utility.log

command

Specifies the command and associated arguments. Supported commands include the following:

- add
- setstatus
- delete
- list

Note: For more information about these commands, see edgewatch Commands for Process Group Monitoring.

edgewatch Commands for Process Group Monitoring

The edgewatch process group monitoring commands and associated arguments are as follows:

```
add attribute regexpr index flags interval 'description' 'action'

setstatus index status

delete index

list
```

Note: The arguments in bold are string literals that you must type exactly as shown. Others arguments are values that you must enter.

regexpr

Specifies the regular expression used to find the PID of the process to monitor. Enclose this value in quotation marks if it contains spaces or other special characters.

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For a list of available flags, see Process Group Monitor Table Flags (see page 310).

interval

Specifies how often in seconds the monitoring should occur. For example, the value 30 instructs the agent to monitor this entry every 30 seconds. The value must be a minimum of 30 and a multiple of 30 seconds.

'***descr***'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'***action***'

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when a match is found for the process group. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

status

Specifies the status of the entry, which can be one of the following:

- active (activate a row)
- notInService (deactivate but preserve a row)
- destroy (delete a row)

edgewatch Examples

The following commands define four monitors with indexes 10, 11, 12 and 13. Monitors with indexes 10 and 11 do not have object information specified. Monitors with indexes 12 and 13 have object information (class, instance, attribute) specified.

```
edgewatch -c admin -o process add procAlive "lsold1" 10 0x00 60 "monitor oldls2" ""
edgewatch -c admin -o process add procSize "lsold2"  11 0x00 60 absolute "!=" 100
"monitor oldls1" ""
edgewatch -c admin -o process add procAlive "ls"   12 0x00 60 "monitor ls1" "" testClass
testInst testAttr none
edgewatch -c admin -o process add procSize "lsold2"  13 0x00 60 absolute "!=" 100
"monitor oldls1" "" testClass testInst testAttr none
edgewatch -c admin -o process list
```

Remove Process Group Monitoring Entries

To stop monitoring a process group, you must remove the appropriate entry from the Process Group Monitor table and the sysedge.cf file. Removing an entry using point configuration from the CA Virtual Assurance user interface accomplishes both. Manual removal requires that you remove the entry from both places in separate operations.

To remove process group monitor entries in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click Process Group Monitors.

   The Process Group Monitor table appears with all existing self monitors.
5. Select the monitor to delete and click Actions, Delete.

   The entry is deleted from the table. The agent performs a warm start to apply the change.

To remove process group monitor entries manually

1. Delete the watch procgroup directive for the entry from the local sysedge.cf file and save the file.
2. Run the edgewatch utility using one of the following commands for the -o parameter:

   `-o delete` *`index`*

   `-o setstatus` *`index`* `6`

   index

   Specifies the index value of the process group monitor entry to remove.

   When using the setstatus command, a value of 6 sets the row status to destroy and deletes the entry.
3. Restart the agent.

   The entry is deleted.

Solaris Zone Process Group Monitoring

SystemEDGE monitors a process group in a Solaris Zone, if the flag 0x8000 (Solaris Zone process) and the variable pgmonRegExpr are set accordingly. The variable pgmonRegExpr specifies a regular expression with format *ZoneRegExpr/ProcRegExpr*. For compatibility reasons the variable pgmonZoneRegExpr is still available, but allows modification only if the monitor was not initialized from sysedge.cf with the 0x8000 flag.

The variable pgmonProcNameList is available on all platforms. If Solaris Zones are monitored, this variable contains a space delimited list of values with the format *MatchedZoneName/MatchedProcessName*. If Solaris Zones monitoring is not required or not possible (non-Solaris 10 OS), the variable contains a space delimited list of matched process names.

When you want to monitor a process group in Solaris Zones, use the watch process directive with flag 0x8000 and a regular expression with format *ZoneRegExpr/ProcRegExp*r.

During the upgrade procedure all watch zone procgroup directives are translated to the new format.

SystemEDGE sends the value of pgmonProcNameList in all process monitor related traps.

Chapter 12: Log File Monitoring

This chapter explains how to use the SystemEDGE agent to monitor log files for regular expressions and directories for size and contents.

This section contains the following topics:

Log File Monitoring Overview (see page 327)
Directory Monitoring (see page 328)
Log Monitor Table (see page 328)
Log Monitoring Configuration (see page 337)
edgewatch Utility--Monitor Log Files (see page 342)
Remove Log Monitoring Entries (see page 348)
Recommendations for Log File Monitoring (see page 349)

Log File Monitoring Overview

The SystemEDGE agent lets you monitor UTF-8 encoded text files continuously for the appearance of user-specified regular expressions. Log file monitoring provides a flexible solution for monitoring applications by monitoring the messages that the applications log. This feature is also useful for security management; for example, you can configure the agent to monitor system log files for messages to notify you of possible security violations. You use the Log Monitor table to specify the file to monitor, regular expression to match, interval, action, severity, and other values. The agent automatically monitors the defined log file and sends a trap to the management system when it detects a regular expression match.

The log file specification may be a wildcard expression, which causes the agent to monitor the single, most recently updated log file matching this expression.

When the agent starts (or after rows have been added to the Log Monitor table), it evaluates the log file expression, identifying the most recently updated log file for its current length and last access time. Thereafter, the agent periodically stats each log file that matches the log file expression for additions or modifications since the last status check. This behavior allows a single monitor entry to follow log files that change names (with perhaps date or revision information) without having to manually modify the given entry.

If the monitored log file has changed, the agent scans only the changes--not the entire log file--to see if there is a match for the specified regular expression. If the agent finds a match, you can configure it to send the enterprise-specific logMonMatch SNMP trap and run the specified action for the row, as long as the action field is not null.

Directory Monitoring

You can use the Log Monitor table to monitor file directories for contents and size. You specify a directory path and name, and the agent tracks the size of the directory and the number of files in the directory. You can then define self monitors on these attributes to put thresholds on directory size and contents.

For more information, see Log Monitor Table Columns (see page 328) and watch logfile Directive (see page 338).

Log Monitor Table

The Log Monitor table provides information about each of the log file or directories that the SystemEDGE agent is currently monitoring. Each row represents a defined log monitor entry.

For each entry, the Log Monitor table provides the following types of information:

- Name of the log file that the agent is monitoring for a particular regular expression
- The regular expression for log file matching
- Number of times a trap has been sent because a match was found
- Time at which the last trap was sent
- Log entry that caused the last match
- Severity

Log Monitor Table Columns

The following list describes the columns of the Log Monitor table. For more information about the Log Monitor table, see the Systems Management Empire MIB empire.asn1 in the mib subdirectory of the agent installation.

logMonitorIndex

Defines the unique row index for this entry (1 to MAXINT). Rows 1 through 10 are reserved for the agent's internal use; the index for additional rows must fall in the range of 11 to MAXINT.

Permissions: Read-only

logMonitorLogFile

Defines the complete path and file name of the log file pr directory to be monitored. This value can be a wildcard expression, and it is evaluated by the agent on each scan to identify the most recently updated log file.

Permissions: Read-write

logMonitorRegularExpression

Defines the regular expression to search for when scanning the log files for matches. SystemEDGE supports regular expressions of up to 512 characters.

Permissions: Read-write

logMonitorNumberTraps

Defines the number of times that the agent sent a trap (logMonMatchTrap) because a string matching the regular expression was logged to the file.

Permissions: Read-only

logMonitorLastTrap

Defines the time (based on sysUpTime) at which the agent last sent a trap (logMonMatchTrap) for this entry.

Permissions: Read-only

logMonitorLastMatch

Defines the last log file entry that matched the regular expression. This variable is updated each time a match occurs.

Permissions: Read-write

logMonitorStatus

Defines the row status, which can be one of the following:

- active
- notInService
- notReady
- createAndGo
- createAndWait

Typically, a row is either active or notInService. These values are identical in meaning to those defined by the SNMPv2 SMI RowStatus textual convention.

Permissions: Read-write

Default: createAndWait(5)

logMonitorLogFileSize

Defines the current size in bytes of the file being monitored or the current size in kilobytes of the directory being monitored.

Permissions: Read-only

logMonitorLogFileLastUpdate

Defines the time that the file or directory was last updated.

Permissions: Read-only

logMonitorDescr

Defines a description (0 to 512 characters in length) of the file or directory being monitored.

Permissions: Read-write

logMonitorAction

Defines a quoted command (0 to 2048 characters in length) with any parameters to run when the regular expression is matched and the agent sends a trap. If the string is empty, the agent invokes no action for this entry.

Note: Do not use Windows batch files for actions; they impose severe programmatic limitations and often do not work correctly with desktop applications. Instead, use a more powerful and flexible scripting language, such as Perl or Visual Basic.

Permissions: Read-write

logMonitorFlags

Defines the positive integer flags that indicate additional behavioral semantics that the entry follows during the course of its operation. For more information about available flags and setting flags, see Log Monitor Table Flags (see page 333).

Permissions: Read-write

logMonitorMatches

Defines the number of logfile entries matching the regular expression. This value increments regardless of whether traps are sent for this entry or not. Polling this variable lets you determine the rate or number of matches over time.

Permissions: Read-only

logMonitorInterval

Defines the best-effort interval, in minutes, between successive scans of the log file or directory.

Permissions: Read-write

logMonitorLogFileName

Defines the actual file or directory name being monitored, in case the logMonitorLogFile attribute contains wildcard characters.

Permissions: Read-only

logMonitorLogFileCount

Defines the number of files and subdirectories in a monitored directory. When monitoring a log file, this value is either 1 or 0 depending on the existence of the file.

Permissions: Read-only

logMonitorSeverity

Defines the severity to assign to the entry when a trap is sent. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

Default: none

Row Creation Objects

The logMonIndex column of the Log Monitor table acts as a key field (or row index) to distinguish rows in the table. Rows 1 through 10 are reserved for internal use by the SystemEDGE agent. You can configure rows in the range 11 to MAXINT.You can use the following MIB objects with the Log Monitor table to optimize row creation:

logmonUnusedIndex

Returns an unused index number for the Log Monitor table when you perform an SNMP Get on the object. Query this object to obtain a number that you can use for row creation.

logmonMatchDescr

Determines the index number that corresponds to a particular entry description in the logmonDescr attribute. Perform an SNMP Set on this MIB object with a logmonDescr attribute value to cause the agent to search through entries in the Log Monitor table and put the index value of the last matching entry in the monMatchIndex MIB object.

logmonMatchIndex

Matches a particular entry description with its index number when used with logmonMatchDescr.

You may choose, as a matter of local policy, to reserve a block of rows for system administration. You can then define entries within a reserved block of rows without being concerned that the row may already be taken by another user's entry. In compliance with the local policy, all other users should use row indices outside the reserved range when they define user-configured entries.

By reserving a block of rows, you can define a consistent set of conditions (row entries) to be monitored across all computers such that the same condition is defined in the same row number on each computer.

Configuring the agent with CA Virtual Assurance can automate this process. You can configure a block of entries with specific row numbers and deploy that configuration to all managed systems in one operation. Also, CA Virtual Assurance detects if you define a row that is already in use and disallows the operation.

For more information about defining rows in CA Virtual Assurance, see the CA Virtual Assurance documentation.

Log Monitor Table Flags

The logMonitorFlags column in the Log Monitor table is a 32-bit unsigned integer that can specify additional behavior for the corresponding Log Monitor table row. By default, the Log Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps
- Logs matches the common log file
- Invokes actions (if they are configured)

You can specify different flag bits to alter these defaults. The SystemEDGE agent interprets all flags in hexadecimal (base 16) notation.

The Log Monitor Table flags (logMonitorFlags) are as follows:

The flags value consists of three fields:

- Field 1 contains common table flags defined for the monitoring tables of the Systems Management Empire MIB. This portion is the low-order 8 bits of the flag.
- Field 2 contains table-specific flags defined separately for each of the monitoring tables. This field defines the next 12 low-order bits after the common table flags.

  For more information about how the 12 bits are defined for the Log Monitor table, see the illustration in Log Monitor Table (see page 328).
- Field 3 contains 12 reserved high-order bits for an integer value for use with table-specific flags. This field includes flags specific to the Log Monitor table.

The following sections define each flag bit. You can combine flag values through a logical OR operation.

The following list describes the Log Monitor table flags:

0x00000001

Disables execution of actions for this entry.

0x00000002

Disables sending of SNMP traps for this entry.

0x00000004

Disables attempts to reinitialize this entry. By default, if the monitored log file is ever unavailable, the agent periodically tries to reinitialize this table entry.

0x00000008

Disables logging of traps for this entry in the sysedge.log file. Setting this bit does not affect sending trap. Disabling event logging is useful when events occur frequently or when a particular entry is used as an agent heartbeat.

0x00000010

Sends continuous logMonEntryNotReady traps for this entry every time the agent attempts to reinitialize log file monitoring and fails.

The default behavior sends a single logMonEntryNotReady trap when the log file being monitored ceases to exist, or when an error accessing that log file occurs.

The agent periodically attempts to reinitialize the entry. Enabling this feature causes the agent to send an additional logMonEntryNotReady trap each time reinitialization fails.

0x00000020

Disables the passing of default arguments to action scripts or programs. SystemEDGE typically passes default action parameters that indicate the trap type, description field, and so on. For more information about action parameters, see Log Monitor Table Action Parameters (see page 336).

0x00000040

Disables sending of notReady traps for this entry.This includes to not log and to not execute actions for notReady trap.

0x00000100

Applies the logical NOT operator to the regular-expression evaluation. When you enable this flag, matched regular expressions evaluate to false and result in no trap or action. All entries that do not match the regular expression evaluate to true and result in a trap and any specified action.

0x00000200

Tracks the log file size, but does not parse through the file. This flag is useful for tracking log file existence and file size with self monitoring.

0x*XXX*00400

Specifies that the monitor does not trigger an event until after X matches have occurred. This value is calculated on a per-scan basis. If this flag is set, traps, event logging, and actions are suppressed until the number of matches exceeds the X value stored in 0x###00000; at that point, a single event is triggered. This flag helps to minimize trap traffic with events that occur often.

0x00000800

Specifies to monitor a directory instead of a specific log file.

0x00001000

Monitors a directory recursively.

0x00002000

Specifies to not follow symbolic links during directory monitoring. Only use this flag with the 0x00000800 flag (directory monitoring only).

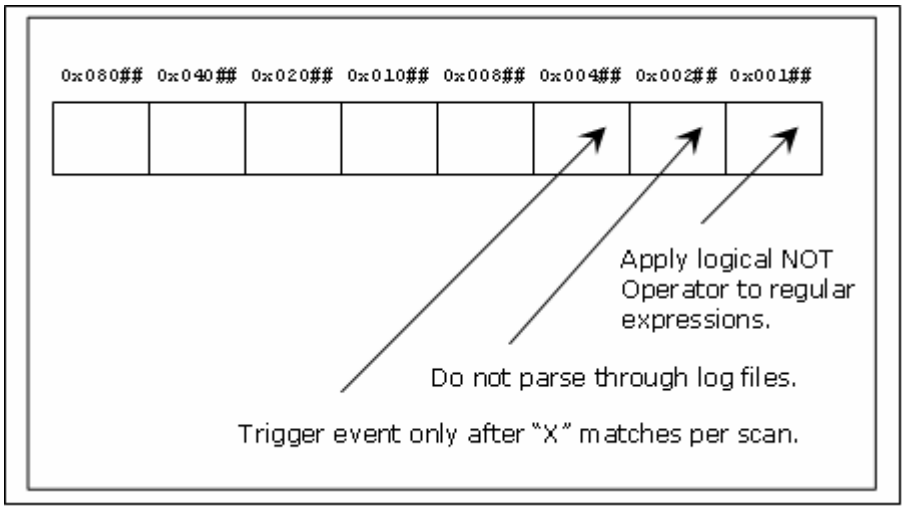
0x00004000

Reports logMonitorLogFileSize variable in kilobytes (instead of bytes).

0x###00000

This X value is used with 0x00000400 to specify the number of events required before triggering an event.

The following illustration shows the flag bits specific to the Log Monitor Table:



Log Monitor Table Action Parameters

The SystemEDGE agent provides several default parameters to the action commands when they are invoked. These parameters are in addition to any parameters you specify in the action string and are passed on the command line after the parameters that you specify. The default parameters are the same as the parameters provided in the SNMP traps sent for the Log Monitor table.

Note: You can prevent the agent from including these action parameters in the command by setting the 0x00000020 flag in the table entry.

The following list describes the default parameters for Log Monitor table actions:

trapType

Defines the type of trap being sent, such as logMonMatchEvent or logMonNotReadyEvent.

logMonitorLogFile

Defines the name of the file that the agent is monitoring.

logMonitorRegularExpression

Defines the regular expression that the agent is attempting to match for the entry.

logMonitorLastTrap

Defines the time that this trap was sent.

logMonitorLastMatch

Defines the line from the log file that triggered this trap.

logMonitorDescr

Defines the table entry description.

logMonitorIndex

Defines the index of the table entry.

logMonitorFlags

Defines the flags field, in hexadecimal notation (for example, 0x0000), for the table entry.

logMonitorLogFileName

Defines the actual file name being monitored, in case the logMonitorLogFile parameter uses wild card characters.

logMonitorSeverity

Defines the severity assigned to the table entry.

View the Log Monitor Table

You can use CA Virtual Assurance to view the contents of a managed agent's Log Monitor table.

To view the Log Monitor table in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Explore pane and Data Center page appear.
3. Expand Managed in the Explore pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click Log Monitors.

   The Log Monitor table appears with all existing log monitors.

Log Monitoring Configuration

You can control which log files the SystemEDGE agent monitors by adding, deleting, or modifying entries in the Log Monitor table.

You can configure the Log Monitor table in the following ways:

Dynamically using file-based configuration from CA Virtual Assurance

CA Virtual Assurance provides the ability to configure agents installed on any system that it manages from the CA Virtual Assurance user interface. This is the recommended method of configuration, because it provides the ability to dynamically define and deliver monitors with a user-friendly and secure mechanism that does not require an agent restart. You can configure monitoring for an individual agent or define monitoring policies to apply on multiple managed systems. When you make an individual configuration change or apply a configuration policy, CA Virtual Assurance delivers a new configuration file to the agent system. CA Virtual Assurance can serve as the central point of agent configuration across your data center, and you can overwrite or deny changes made through other methods.

For more information about configuring the agent using CA Virtual Assurance, see the CA Virtual Assurance documentation.

Dynamically using SNMP Set commands from a management system, such as eHealth AdvantEDGE View, CA Spectrum, or CA NSM

Various management systems support adding monitors to SystemEDGE through SNMP Set commands. eHealth provides this functionality through the AdvantEDGE View interface, CA NSM enables configuration through an Agent View, and CA Spectrum enables configuration through OneClick. All features provided by dynamic configuration through CA Virtual Assurance are not available through these methods.

For more information about SNMP Set configuration, see the documentation for the appropriate management system.

Manually by specifying the entries for the Self Monitor table in the local sysedge.cf file

Manual configuration requires you to add entries to the local sysedge.cf file using specific directives. When you configure the agent directly from the sysedge.cf file, a full restart is required for changes to take effect. Use the sysedge.cf file located in the agent data directory for direct configuration.

This section describes how to configure self monitoring directly in the local sysedge.cf file, but it describes the properties that are required for all forms of configuration.

watch logfile Directive--Add Entries to the Log Monitor Table

The watch logfile directive lets you add log monitor entries to the Log Monitor table directly in the sysedge.cf file. The arguments represent columns in the Log Monitor table.

Add a line to the sysedge.cf file in the agent data directory using the syntax described below, save the file, and restart the agent for the change to take effect.

Use the watch logfile directive to add entries to the Log Monitor table as follows:

```
watch logfile index flags 'file' 'regexpr' ['descr'['action'[interval[severity]]]]
```

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For more information about available flags, see Log Monitor Table Flags (see page 333).

'*file*'

Specifies the complete path and file name of the log file to monitor, or the complete path of the directory to monitor. You can specify this parameter including wildcard characters such as * and ?.

The file you monitor must be an ASCII-based text file. SystemEDGE does not support monitoring of other character sets, such as Unicode.

You can determine a file's encoding by opening it in a text editor and selecting Save As. The encoding is listed in the Save as type field.

'*regexpr*'

Specifies the regular expression to use when scanning the log file for matches. SystemEDGE supports log file regular expressions of up to 512 characters. This parameter is ignored when you are monitoring a directory.

'*descr*'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'*action*'

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the regular expression is matched and a trap is sent. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

interval

Specifies how often to monitor this entry in minutes. Valid interval range is 1 -1440 minutes.

Default: 1

severity

Specifies the severity to assign to the entry when a match occurs and a trap is sent. The severity is included with the trap. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

Note that the severity designation only specifies the importance of the monitor and is not used to calculate status.

Default: none

Log File Monitoring Examples

This section provides examples for using the watch logfile directive in the sysedge.cf file to add log monitor entries to the Log Monitor table.

Example: Search for pop Connection Attempts

The following example instructs the agent to add an entry to the Log Monitor table at table index 15 to search for pop connection attempts on a system and send a trap with a warning severity when a match occurs:

```
watch logfile 15 0x00 /var/log/syslog 'popper' 'NOTICE - pop connection' '' 1 warning
```

Example: Search for su Attempts

The following example instructs the agent to add an entry to the Log Monitor table at table index 16 to search for su attempts on a system and send a trap with a critical severity when a match occurs:

```
watch logfile 16 0x02 /var/adm/messages 'su.*fail' 'WARNING - su attempt'
'/local/bin/mail2admin' 5 critical
```

0x02

Specifies that the agent should not send traps. Instead, the agent invokes the specified action command.

Examples: Monitor Directory Size and Contents

The following example instructs the agent to add an entry to the Log Monitor table at table index 17 to monitor the /var/tmp directory for size and number of files.

```
watch logfile 17 0x00 /var/tmp 'Temporary directory' '' 5
```

You can create a self monitor entry to put thresholds on the size (logMonitorLogFileSize) and contents (logMonitorLogFileCount) attributes.

The following examples instructs the agent to add an entry to the Log Monitor table at table index 232 to monitor a directory for size and number of files.

```
watch logfile 232 0x1800 'C:\testdir23' '' Monitor for dir testdir23' '' 1 warning
```

Flag 0x0800 can be used for monitoring directory non-recursively. In addition to this, create a self monitor entry to put thresholds on the size (logMonitorLogFileSize) and contents (logMonitorLogFileCount) attributes.

```
monitor oid 1.3.6.1.4.1.546.11.1.1.8.232 2323232 0x8 30 absolute >= 10 'Directory Size
is more than 10 bytes' '' 0 (or)
monitor oid logMonitorLogFileSize.232 2323232 0x8 30 absolute >= 10 'Directory Size
is more than 10 bytes' '' 0
monitor oid logMonitorLogFileCount.232 2323231 0x8 30 absolute >= 10 'FileCount is
more than 10' '' 0 (or)
monitor oid 1.3.6.1.4.1.546.11.1.1.16.232 2323231 0x8 30 absolute >= 10 'File Count
is more than 10' '' 0
```

Example: Search for a Pattern on Multiple Lines

The following example specifies a regular expression that instructs an agent to add an entry in the Log Monitor table at table index 20. The entry is used to search for text that appears across multiple lines in the monitored log file. In this example, the text "WARNING:" appears as an unspecified number of lines before the text "Invalid login attempt".

```
watch logfile 20 0x00 /var/log/syslog '/^WARNING:(.*\n)*Invalid login attempt/m'
'NOTICE – Invalid login attempt' '' 1 warning
```

Note: This example requires Perl Compatible Regular Expressions(PCRE) to be activated on the agent using the Policy Configuration Control Settings, or using the SystemEDGE configuration file (if in Legacy Mode).

More Information

Perl Compatible Regular Expression (PCRE) Support (see page 150)
Configure Text Pattern Exclusion (see page 152)
Regular Expression Examples (see page 152)

edgewatch Utility--Monitor Log Files

The edgewatch command line utility automatically configures the SystemEDGE agent to monitor processes, process groups, log files, and Windows event logs. After you specify the particular process, process group, log file or Windows event log and the associated arguments, the edgewatch utility issues an SNMP Set request to create the appropriate entry in the target agent's appropriate monitoring table. The edgewatch utility is located in the bin subdirectory of the agent installation.

Use the edgewatch utility for log file monitoring as follows:

```
edgewatch [-h hostname | ip _addr] [-p port] [-c community]
    [-v 1 | 2c | 3] [-u secName] [-s secLevel] [-n contextName]
    [-a authPassword] [-A MD5 | SHA]
    [-x privPassword] [-X DES | AES | 3DES]
    [-m FIPS mode]
    [-r retries]
    [-t timeout] [-d logLevel] [-f logFile]
    -o logfile command
```

-h *hostname | ipaddr*

Specifies the host name or IP address of the system on which the agent is running. Accepts IPv4 and IPv6 addresses.

Default: localhost

-p *port*

Specifies the UDP port that the agent is running on (for example, 1691).

Default: 161

-c *community*

Specifies a community string that the agent uses. Valid for SNMPv1 and SNMPv2c only.

Note: Specify a read/write community string for snmpset.

Default: public

-v {1 | 2c | 3}

Indicates the version of SNMP that the agent is running. Specify 1 for SNMPv1, 2c for SNMPv2c, or 3 for SNMPv3.

Default: 1

-u *secName*

Specifies the User-based Security Model (USM) user name that is used for SNMPv3 security.

Default: none

-s *secLevel*

Specifies one of the following security levels for SNMPv3 communication:

- 1 - noAuthNoPriv
- 2 - AuthNoPriv

3 - AuthPriv (SNMPv3 only)

-n *contextName*

Specifies the instance name for the MIBMuxed agent.

Default: none

-a *authPassword*

Specifies the authentication password if the agent is configured for SNMPv3 with secLevel 2 (AuthNoPriv) or 3 (AuthPriv).

Note: This option is not required for SNMPv3 communication.

Default: none

-A {MD5 | SHA}

Specifies the authentication protocol to use if the agent is configured for SNMPv3 with secLevel 2 (AuthNoPriv) or 3 (AuthPriv). Currently only MD5 (Message Digest Algorithm) and SHA (Secure Hash Algorithm) are used.

Default: MD5

-x *privPassword*

Specifies the privacy (encryption) password if the agent is configured for SNMPv3 with secLevel 3 (AuthPriv).

Default: none

-X {DES | AES | 3DES}

Specifies the privacy protocol if the SNMPv3 user is configured with secLevel 3 (AuthPriv). Specify DES for Data Encryption Standard, AES for Advanced Encryption Standard using cryptographic keys of 128 bits (AES128), and 3DES for Triple Data Encryption Standard.

Default: none

-m *FIPS_mode*

Controls the FIPS mode of operation. Accepted values are 0, 1, and 2.

0

Indicates Non-FIPS mode.

1

Indicates FIPS co-existence mode.

2

Indicates FIPS only mode.

Default: 1

-r *retries*

Specifies the number of retries.

Default: 10

-t *timeout*

Specifies the duration before the SNMP receiver considers the request as timed out.

Default: 10 seconds

-d *logLevel*

Specifies the log level of the SNMP messages. Accepted values are 0 to 5.

0

Logs fatal messages.

1

Logs critical messages.

2

Logs warning messages.

3

Logs informational messages.

4

Logs all of the messages.

5

Logs all of the messages including debugging messages.

Default: 0

-f logfile

Specifies the name of the log file that contains error and debug information.

Default: sysedge_utility.log

command

Specifies the command and associated arguments. Supported commands are the following:

- add
- setstatus
- delete
- list

Note: For more information about the commands, see edgewatch Commands for Log File Monitoring (see page 345).

edgewatch Commands for Log File Monitoring

The edgewatch commands and associated arguments for log file monitoring are as follows:

```
add index flags "file" "regexpr" "descr" "action" interval

setstatus index status

delete index

list
```

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For more information about available flags, see Log Monitor Table Flags (see page 333).

"file"

Specifies the complete path and file name of the log file to monitor, or the complete path of the directory to monitor. You can specify this parameter including wildcard characters such as * and ?.

The file you monitor must be an ASCII-based text file. SystemEDGE does not support monitoring of other character sets, such as Unicode.

You can determine a file's encoding by opening it in a text editor and selecting Save As. The encoding is listed in the Save as type field.

'*regexpr*'

Specifies the regular expression to use when scanning the log file for matches. SystemEDGE supports log file regular expressions of up to 512 characters. This parameter is ignored when you are monitoring a directory.

“*descr*”

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

“*action*”

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the regular expression is matched and a trap is sent. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

interval

Specifies how often to monitor this entry in minutes.

Default: 1

status

Specifies the RowStatus textual convention value to use in setting the status of a row in the Log Monitor table. Use this parameter with the setstatus operation.

Valid values are as follows. The values can be either assigned integer values or the actual spelled out status text:

- active(1)
- notInService(2)
- destroy(6)

edgewatch Examples

This section provides examples for using the edgewatch utility with SNMP versions 1, 2c, and 3.

Example: List Entries in Log Monitor Table

The following example displays the contents of the Log Monitor table:

```
edgewatch -v 1 -h 127.0.0.1 -c public -o logfile list
```

```
edgewatch -v 2c -h fe80::2367:1 -c public -o logfile list
```

```
edgewatch -v 3 -s 3 -u userName -x privPassword -X encryptProtocol -A authProtocol
-a authPassword -o logfile list
```

Add a Log Monitor Entry

The following example instructs the agent to add an entry to the Log Monitor table at table index 5 to search for su failures on an HP-UX system. The agent runs the script /local/bin/mail2admin when it finds a match.

```
edgewatch -v 1 -h 127.0.0.1 -c private -o logfile add 5 0x00 /usr/adm/sulog "SU.* -"
"su attempt - WARNING" "/local/bin/mail2admin" 1
```

```
edgewatch -v 2c -h fe80::2367:1 -c private -o logfile add 5 0x00 /usr/adm/sulog "SU.*
-" "su attempt - WARNING" "/local/bin/mail2admin" 1
```

```
edgewatch -v 3 -h fe80::2367:1 -s 3 -u userName -A authProtocol -a authPassword -X
encryptProtocol -x privPassword -o logfile add 5 0x00 /usr/adm/sulog "SU.* -" "su
attempt - WARNING" "/local/bin/mail2admin" 1
```

Example: Delete a Log Monitor Entry

The following example deletes an entry from the Log Monitor table at table index 5:

```
edgewatch -v 1 -h 127.0.0.1 -c private -o logfile delete 5
```

```
edgewatch -v 2c -h fe80::2367:1 -c private -o logfile delete 5
```

```
edgewatch -v 3 -h 127.0.0.1 -s 3 -u userName -A authProtocol -a authPassword -X
encryptProtocol -x privPassword -o logfile delete 5
```

Example: Disable a Log Monitor Entry

The following example disables the Log Monitor table entry at table index 5 by setting that entry's status to notInService(2). The entry will remain in the table, but the agent will not scan the log file for the regular expression unless the status returns to active (1).

```
edgewatch -v 1 -h 127.0.0.1 -c private -o logfile setstatus 5 notInService
```

```
edgewatch -v 2c -h fe80::2367:1 -c private -o logfile setstatus 5 notInService
```

```
edgewatch -v 3 -h 127.0.0.1 -s 3 -u userName -A authProtocol -a authPassword -X
encryptProtocol -x privPassword -o logfile setstatus 5 notInService
```

Remove Log Monitoring Entries

To stop monitoring a log file or directory, you must remove the appropriate entry from the Log Monitor table and the sysedge.cf file. Removing an entry using point configuration from the CA Virtual Assurance user interface accomplishes both. Manual removal requires that you remove the entry from both places in separate operations.

To remove log monitor entries in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click Log Monitors.

   The Log Monitor table appears with all existing self monitors.
5. Select the monitor to delete and click Actions, Delete.

   The entry is deleted from the table. The agent performs a warm start to apply the change.

To remove log monitor entries manually

1. Delete the watch logfile directive for the entry from the local sysedge.cf file and save the file.
2. Run the edgewatch utility using one of the following commands for the -o parameter:

   `-o delete` *`index`*

   `-o setstatus` *`index`* `6`

   index

   Specifies the index value of the log monitor entry to remove.

   When using the setstatus command, a value of 6 sets the row status to destroy and deletes the entry.
3. Restart the agent.

   The entry is deleted.

Recommendations for Log File Monitoring

You can monitor system and application logs to obtain in-depth information about user, system, and application behavior.

The following tables describe recommendations for log files you can monitor and the regular expressions for which you can search for monitoring security, device failures, system capacity, Windows security, and applications and systems:

Description	Log File to Monitor	Regular Expression
WARNING - daemon core	/var/log/daemon-log	core dumped
WARNING - daemon core	/var/log/syslog	core dumped
Monitor SU attempts	/var/adm/messages	su.*fail
Monitor rlogins	/var/log/syslog	in.rlogin
Monitor telnets	/var/log/syslog	in.telnet
Monitor rsh	/var/log/syslog	in.rsh
WARNING - Illegal Instruction, Daemon	/var/log/daemon-log	.*llegal.*nstruction
Spam Relay Attempt	/var/log/syslog	Relaying denied
Monitor DENY packets from a Linux firewall	/var/log/messages	DENY

Note: The log files described in this section are not the same for all operating systems. These examples are provided for reference. You should alter them for your operating system.

The following table describes the recommendations for regular expressions for which you can search in the syslog (/var/adm/messages) to monitor for device failures:

Description	Regular Expression
Critical: Badtrap Error	.*BAD TRAP.*
Error: SCSI error	.*SCSI.*[E,e]rror.*
Error: SCSI error	.*SCSI.*failed.*
Error: SCSI error	.*SCSI.*hung.*
Critical: badsimms error	.*SIMM.*
Critical: badsimms error	.*BAD.*SIMM.*
Critical: memory error	.*[M,m]emory [E,e]rror.*
Error: disk error	.*disk not responding.*
Error: disk error	.*[D,d]isk.*[E,e]rror.*
'Warning: disk fragmentation error'	.*optimization changed.*
Error: disk error	.*corrupt label.*
Error: I/O error	.*I/O.*[E,e]rror.*
Error: disk read/write errors	.*Error for Command:.*[read,write].*
Error: media error	.*Media Error.*
Info: serialport error	.*zs[0,1,2]: silo overflow.*
Warning: carrier error	.*no carrier.*
Warning: link is down	.*Link Down - cable problem?.*
Error: SDS error	.*[NOTICE,WARNING,PANIC]: md:.*

The following table describes recommendations for regular expressions for which you can search in /var/adm/messages to monitor system capacity:

Description	Regular Expression
Critical: memory error	*[O,o]ut of [M,m]emory.*
Critical: memory error	.*[F,f]ile system full.*
Error: diskspace error	.*No space left on device.*

The following table describes recommendations for which logs and expressions you can monitor in the Windows event logs to monitor Windows security:

Description	Event Log	Security Type	Regular Expression
Random Password Hack	Security	All	.*bad
Misuse of Privileges	Security	All	.*[user rights,group management,security change, restart,shutdown]
Improper File Access	Security	Failure	.*[read,write]
Improper Printer Access	Security	Failure	.*print
Virus Outbreak Warning: program files updated	Security	All	.*write.*[exe,dll,com]
Security Policies Change	Application	Information	.*[S,s]ecurity policy

The following table describes recommendations for which logs and expressions you can monitor in the Windows event logs to monitor applications and systems:

Description	Event Log	Security Type	Regular Expression
Application Error or Failure	Application	All	.*[F,f]ail|.*[E,e]rror
Application Load Problems	Application	All	.*[L,l]oad.*[P,p]roblem
New Software Installed	Application	All	.*[INSTALL,Install,install]
Server Process failed during Initialization	Application	All	.*4131
Disk Failures and errors	All	All	.*[D,d]isk
Network Adapter Errors	All	Error	.*[N,n]etwork [A,a]dapter

Rotating Log Files

You can monitor rotating log files using an appropriate wildcard entry to specify the log file entries. The wildcard syntax is platform dependent and lets you use the asterisk (*) for any number of characters and the question mark (?) to indicate any single character. The agent monitors the most recent matching log file and automatically switches to the next rolling log file as it is created and updated.

Chapter 13: Windows Event Monitoring

This chapter explains how to use the SystemEDGE agent to monitor Windows event logs for regular expressions.

Important! This chapter is relevant only for Windows versions of the SystemEDGE agent.

This section contains the following topics:

Windows Event Monitoring Overview (see page 353)
NT Event Monitor Table (see page 355)
Windows Event Monitoring Configuration (see page 364)
edgewatch Utility--Monitor Windows Events (see page 368)
Remove Windows Event Monitoring Entries (see page 375)

Windows Event Monitoring Overview

The SystemEDGE agent lets you continuously monitor Windows event logs for regular expressions. This capability is similar to the standard log file monitoring described in the chapter “Log File Monitoring.” You use the NT Event Monitor table to specify the event log, event type, regular expressions to match for source and description, severity, and other values. The agent automatically monitors the defined log and sends a trap to the management system when it detects a regular expression match.

The agent can also run action commands to handle the event immediately. Because Windows events include several identifying characteristics in addition to the text message, this monitoring capability is somewhat more sophisticated than the standard log file monitoring in the types of matches that you can specify.

When the SystemEDGE agent starts (or after the addition of rows to the NT Event Monitor table), it checks the status (stats) of each Windows event log for its current length and the time that it was last updated. Thereafter, the agent periodically scans each event log for additions or modifications since the last update. If the event log file has changed, the agent scans only the changes--not the entire event log--to see if a match exists for the specified filters.

Windows Event Search Criteria

Each Windows Event Monitor table entry instructs the agent to search for matches based on the criteria described in the following table:

Event Log

Specifies the name of the event log. This value can be any of the following:

- Application
- System
- Security
- DirService (for Directory Service)
- DnsServer (for DNS Service)
- FileRepService (for File Replication Service)

Event Type

Specifies the type of event. Types 1 through 5 are defined by Windows as the following:

- error(1)
- warning(2)
- information(3)
- success(4)
- failure(5)

Type all(6) indicates that the agent should match all event types.

Event Source

Specifies the name of the program or module that generated the event. The agent uses regular expressions to match this field.

Event Description

Describes the event. The agent uses regular expressions to match this field.

The SystemEDGE agent generates an SNMP trap message when it finds a match based on all four criteria. This matching is similar to a Boolean AND operation.

NT Event Monitor Table

The NT Event Monitor table provides information about each of the event logs that the agent is currently monitoring for specific regular expression patterns. Each row represents a defined Windows event monitor entry.

For each entry in the NT Event Monitor table, the table provides the following types of information:

- Event log that the agent is monitoring
- Regular expression for which the log is being monitored
- Number of times that a trap has been sent because a match was found
- Time at which the last trap was sent
- Log entry that caused the last match
- Severity

NT Event Monitor Table Columns

The following list describes the columns of the NT Event Monitor table. For more information about the NT Event Monitor table, see the Systems Management Empire MIB empire.asn1 in the mib subdirectory of the agent installation.

ntEventMonIndex

Defines the unique row index for this entry (1 to MAXINT). Rows 1 through 10 are reserved for the agent's internal use; the index for additional rows must fall in the range of 11 to MAXINT.

Permissions: Read-only

ntEventMonLog

Defines the event log to monitor. Valid values are as follows:

- Application(1)
- Security(2)
- System(3)
- Directory Service(4)
- DNS Service(5)
- File Replication Service(6)

Permissions: Read-write

ntEventMonTime

Defines the time, based on sysUpTime, at which the event occurred.

Permissions: Read-only

ntEventMonTraps

Defines the number of times that a match occurred for this entry and the agent sent a trap (ntEventMonMatchTrap).

Permissions: Read-only

ntEventMonTypeLastMatch

Defines the event type of the last event that matched the search criteria. Types 1 through 5 are defined by Windows as follows:

- error(1)
- warning(2)
- information(3)
- success(4)
- failure(5)

Type noMatch(6) indicates that there has not yet been a matching event for this monitoring entry.

Permissions: Read-only

ntEventMonTypeFilter

Defines the event type to match for this entry. Types 1 through 5 are defined by Windows as follows:

- error(1)
- warning(2)
- information(3)
- success(4)
- failure(5)

Type all(6) indicates that the agent should match all event types.

Permissions: Read-write

ntEventMonSrcLastMatch

Defines the Event Source of the last event log entry that matched this monitor entry. The Event Source is usually the name of the program that generated the event. Each time a match occurs, this variable is updated.

Permissions: Read-only

ntEventMonSrcFilter

Defines the regular expression to apply to the Event Source when scanning the events for matches.

Permissions: Read-write

ntEventMonDescLastMatch

Defines the last event log entry that matched this monitor entry. Each time a match occurs, this variable is updated.

Permissions: Read-only

ntEventMonDescFilter

Defines the regular expression to apply to the Event Description when scanning the events for matches.

Permissions: Read-write

ntEventMonStatus

Defines the row status, which can be one of the following:

- active
- notInService
- notReady
- createAndGo
- createAndWait

Typically, a row is either active or notInService. These values are identical in meaning to those defined by the SNMPv2 SMI RowStatus textual convention.

Permissions: Read-write

Default: createAndWait(5)

ntEventMonDescr

Defines a description (0 to 512 characters in length) of the events being monitored.

Permissions: Read-write

ntEventMonAction

Defines a quoted command (0 to 2048 characters in length) with any parameters to run when a match is found and a trap is sent. If the string is empty, the agent invokes no action for this entry.

Note: Do not use Windows batch files for actions; they impose severe programmatic limitations and often do not work correctly with desktop applications. Instead, use a more powerful and flexible scripting language, such as Perl or Visual Basic.

Permissions: Read-write

ntEventMonFlags

Defines the positive integer flags that indicate additional behavioral semantics that the entry follows during the course of its operation. For more information about available flags and setting flags, see NT Event Monitor Table Flags (see page 360).

ntEventMonSeverity

Defines the severity to assign to the entry when a trap is sent. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

Default: none

Row Creation Objects

The ntEventMonIndex column of the NT Event Monitor table acts as a key field (or row index) to distinguish rows in the table. Rows 1 through 10 are reserved for internal use by the SystemEDGE agent. You can configure rows in the range 11 to MAXINT.

You can use the following MIB objects with the NT Event Monitor table to optimize row creation:

ntEventUnusedIndex

Returns an unused index number for the NT Event Monitor table when you perform an SNMP Get on the object. Query this object to obtain a number that you can use for row creation.

ntEventMatchDescr

Determines the index number that corresponds to a particular entry description in the ntEventMonDescr attribute. Perform an SNMP Set on this MIB object with a ntEventMonDescr attribute value to cause the agent to search through entries in the Self Monitor table and put the index value of the last matching entry in the monMatchIndex MIB object.

ntEventMatchIndex

Matches a particular entry description with its index number when used with ntEventMatchDescr.

You may choose, as a matter of local policy, to reserve a block of rows for system administration. You can then define entries within a reserved block of rows without being concerned that the row may already be taken by another user's entry. In compliance with the local policy, all other users should use row indices outside the reserved range when they define user-configured entries.

By reserving a block of rows, you can define a consistent set of conditions (row entries) to be monitored across all computers such that the same condition is defined in the same row number on each computer. For example, you can use row 11 (monIndex = 11) to define an entry for monitoring the swapCapacity variable, and you can distribute this configuration to every system so that row 11 monitors the swapCapacity variable on every system.

Configuring the agent with CA Virtual Assurance can automate this process. You can configure a block of entries with specific row numbers and deploy that configuration to all managed systems in one operation. Also, CA Virtual Assurance detects if you define a row that is already in use and disallows the operation.

For more information about defining rows in CA Virtual Assurance, see the CA Virtual Assurance documentation.

NT Event Monitor Table Flags

You can use the ntEventMonFlags column in the NT Event Monitor Table to specify additional behavioral semantics for the corresponding NT Event Monitor table row. By default, the NT Event Monitor table row does the following:

- Attempts to reinitialize itself
- Sends SNMP traps
- Logs events to the common log file
- Invokes actions (if they are configured)

You can set different flag bits to alter these defaults. The SystemEDGE agent interprets all flags in hexadecimal (base 16) notation.

The NT Event Monitor table flags field (ntEventmonFlags) are as follows:

The flags value consists of three fields:

- Field 1 contains common table flags defined for the monitoring tables of the Systems Management Empire MIB. This portion is the low-order 8 bits of the flag.
- Field 2 contains table-specific flags defined separately for each of the monitoring tables. This field defines the next 12 low-order bits after the common table flags. For Windows event monitoring, there are currently no table-specific flags defined.
- Field 3 contains 12 reserved high-order bits for an integer value for use with table-specific flags. For Windows event monitoring, there are currently no table-specific flags defined.

The following sections define each flag bit. You can combine flag values through logical OR operations.

The following list describes the common table flags for the monitoring tables:

0x00000001

Disables execution of actions for this entry.

0x00000002

Disables sending of SNMP traps for this entry.

0x00000004

Disables attempts to reinitialize this entry. By default, if the monitored event log is unavailable, the agent periodically tries to reinitialize this table entry.

0x00000008

Disables logging of traps for this entry in the sysedge.log file. Setting this bit does not affect sending trap. Disabling event logging is useful when events occur frequently or when a particular entry is used as an agent heartbeat.

0x00000010

Sends continuous ntEventMonEntryNotReady traps for this entry every time the agent attempts to reinitialize event log monitoring and fails. The agent's default behavior is to send a single ntEventMonEntryNotReady trap when the event log that it is monitoring ceases to exist or when an error accessing that event log occurs. The agent periodically attempts to reinitialize the entry. Enabling this feature causes the agent to send an additional ntEventMonEntryNotReady trap each time reinitialization fails.

0x00000020

Disables the passing of default arguments to action scripts or programs. SystemEDGE typically passes default action parameters that indicate the trap type, description field, and so on. For more information about action parameters, see NT Event Monitor Table Action Parameters (see page 362).

0x00000040

Disables sending of notReady traps for this entry.This includes to not log and to not execute actions for notReady trap.

0x00000100

Pre-appends the Event ID number to the event description field. The identifier is added using the form *[#]* where *#* is the identifier. The purpose of this flag is to facilitate searches for specific Event IDs. For example, to specify that only Event ID 528 should match, specify this flag with the ntEventMonDescFilter set to [528].

0x00000200

Applies the logical NOT operator to the event source, type, and description evaluation. When you enable this flag, events that match the specified source, type, and description evaluate to false and result in no trap or action. Events that do not match one or more of the event source, type, and description evaluate to true and result in a trap and the specified action.

Note: Use caution when setting this flag.

0x####0000

The NT Event Monitor Table does not use this flag value field; setting it has no effect on the NT Event Monitor Table operation or on any of the supported flag bits.

NT Event Monitor Table Action Parameters

The SystemEDGE agent provides several default parameters to the action commands. These parameters are in addition to any parameters you specify in the action string and are passed on the command line after those that you specify. The default parameters are the same as the parameters provided in the SNMP traps sent for the NT Event Monitor table.

Note: You can prevent the agent from including these action parameters in the command by setting the 0x00000020 flag in the table entry.

The following list describes the default parameters for NT Event Monitor table actions:

trapType

Defines the type of trap sent, such as a ntEventMonMatchEvent or ntEventMonNotReadyEvent.

ntEventMonLog

Defines the event log that the agent is monitoring.

ntEventMonTypeLastMatch

Defines the type of event that generated this trap.

ntEventMonTime

Defines the time that this event was generated.

ntEventMonSrcLastMatch

Defines the source of the event that generated this trap.

ntEventMonDescLastMatch

Defines a description of the event that generated this trap.

ntEventMonDesc

Defines a description of the table entry.

ntEventMonIndex

Defines the index for the table entry.

ntEventMonFlags

Defines the flags field, in hexadecimal notation (for example, 0x0000), for the table entry.

ntEventMonSeverity

Defines the severity assigned to the table entry.

For more information about traps sent by the SystemEDGE agent, see the chapter "Concepts" and the appendix “Private Enterprise Traps.”

View the NT Event Monitor Table

You can use CA Virtual Assurance to view the contents of a managed agent's NT Event Monitor table.

To view the NT Event Monitor table in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click Windows Event Monitors.

   The Windows Event Monitor table appears with all existing windows event monitors.

Windows Event Monitoring Configuration

You can control which event logs the SystemEDGE agent monitors by adding, deleting, or modifying entries in the NT Event Monitor table.

You can configure the NT Event Monitor table in the following ways:

Dynamically using file-based configuration from CA Virtual Assurance

CA Virtual Assurance provides the ability to configure agents installed on any system that it manages from the CA Virtual Assurance user interface. This is the recommended method of configuration, because it provides the ability to dynamically define and deliver monitors with a user-friendly and secure mechanism that does not require an agent restart. You can configure monitoring for an individual agent or define monitoring policies to apply on multiple managed systems. When you make an individual configuration change or apply a configuration policy, CA Virtual Assurance delivers a new configuration file to the agent system. CA Virtual Assurance can serve as the central point of agent configuration across your data center, and you can overwrite or deny changes made through other methods.

For more information about configuring the agent using CA Virtual Assurance, see the CA Virtual Assurance documentation.

Dynamically using SNMP Set commands from a management system, such as eHealth AdvantEDGE View, CA Spectrum, or CA NSM

Various management systems support adding monitors to SystemEDGE through SNMP Set commands. eHealth provides this functionality through the AdvantEDGE View interface, CA NSM enables configuration through an Agent View, and CA Spectrum enables configuration through OneClick. All features provided by dynamic configuration through CA Virtual Assurance are not available through these methods.

For more information about SNMP Set configuration, see the documentation for the appropriate management system.

Manually by specifying the entries for the Self Monitor table in the local sysedge.cf file

Manual configuration requires you to add entries to the local sysedge.cf file using specific directives. When you configure the agent directly from the sysedge.cf file, a full restart is required for changes to take effect. Use the sysedge.cf file located in the agent data directory for direct configuration.

This section describes how to configure self monitoring directly in the local sysedge.cf file, but it describes the properties that are required for all forms of configuration.

watch ntevent Directive--Add Entries to NT Event Monitor Table

The watch ntevent directive lets you add Windows event monitor entries to the NT Event Monitor table directly in the sysedge.cf file. The arguments represent columns in the NT Event Monitor table.

Add a line to the sysedge.cf file in the agent data directory using the syntax described below, save the file, and restart the agent for the change to take effect.

Use the watch ntevent directive to add entries to the NT Event Monitor table as follows:

```
watch ntevent index flags 'evLog' 'evType' 'evSrc' 'evDescr' ['descr'] ['action']
[severity]
```

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001). For more information about available flags, see NT Event Monitor Table Flags (see page 360).

'evLog'

Specifies the event log to monitor. This value can be one of the following:

- application
- system
- security
- dnsServer
- dirService
- fileRepService

'evType'

Specifies the event type to match for this entry. The following are valid types:

- error
- warning
- information
- success
- failure
- all

Entering 'all' indicates that all event types should match.

'*evSrc*'

Specifies the regular expression to use when scanning the Event Source attribute in each event.

'*evDescr*'

Specifies the regular expression to use when scanning the Event Description attribute in each event.

'*descr*'

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

'*action*'

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the entry is matched and a trap is sent. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

severity

Specifies the severity to assign to the entry when a match occurs and a trap is sent. The severity is included with the trap. Valid values are as follows:

- none
- ok
- warning
- minor
- major
- critical
- fatal

Note that the severity designation only specifies the importance of the monitor and is not used to calculate status.

Default: none

Windows Event Monitoring Examples

This section provides example entries for the NT Event Monitor table to monitor Windows event logs using the watch ntevent directive. You can add these entries to the sysedge.cf file.

Example: Search the Application Log for Web Server Messages

The following example adds a new entry to the agent's NT Event Monitor table at table index 11 to search the Application log for messages from the http Web server application and send a trap with a warning severity when a match occurs:

```
watch ntevent 11 0x00 Application All 'http' '.*' 'Web Server messages' " warning
```

Example: Search the Security Log for Failure Events

The following example adds a new entry to the agent's NT Event Monitor table at table index 12 to search the Security log for Failure events that indicate login failures and send a trap with a critical severity when a match occurs:

```
watch ntevent 12 0x00 Security Failure '.*' '.*' 'Access Failure - WARNING' " critical
```

Example: Search the Application Log for Specific Events

The following example adds a new entry to the agent's NT Event Monitor Table at table index 3 to search the Application log for events with Event ID 277:

```
watch ntevent 3 0x0100 Application All '.*' '\[277\]' 'Event ID 277' "
```

0x0100

Adds the Event ID to the description. [277\] is the description field that the agent will attempt to match.

The backslash character (\) is required because brackets ([]) are special characters for regular expression matching.

More Information

Perl Compatible Regular Expression (PCRE) Support (see page 150)
Configure Text Pattern Exclusion (see page 152)
Regular Expression Examples (see page 152)

edgewatch Utility--Monitor Windows Events

The edgewatch command line utility automatically configures the SystemEDGE agent to monitor processes, process groups, log files, and Windows event logs. After you specify the particular process, process group, log file or Windows event log and the associated arguments, the edgewatch utility issues an SNMP Set request to create the appropriate entry in the target agent's appropriate monitoring table. The edgewatch utility is located in the bin subdirectory of the agent installation.

Although the Windows event monitoring capability is provided on Windows only, you can configure it from any supported platform using the edgewatch utility.

Use the edgewatch utility for Windows event monitoring as follows:

```
edgewatch [-h hostname | ip _addr] [-p port] [-c community]
    [-v 1 | 2c | 3] [-u secName] [-s secLevel] [-n contextName]
    [-a authPassword] [-A MD5 | SHA]
    [-x privPassword] [-X DES | AES | 3DES]
    [-m FIPS_mode]
    [-r retries]
    [-t timeout] [-d logLevel] [-f logFile]
    -o ntevent command
```

-h *hostname | ipaddr*

Specifies the host name or IP address of the system on which the agent is running. Accepts IPv4 and IPv6 addresses.

Default: localhost

-p *port*

Specifies the UDP port that the agent is running on (for example, 1691).

Default: 161

-c *community*

Specifies a community string that the agent uses. Valid for SNMPv1 and SNMPv2c only.

Note: Specify a read/write community string for snmpset.

Default: public

-v {1 | 2c | 3}

Indicates the version of SNMP that the agent is running. Specify 1 for SNMPv1, 2c for SNMPv2c, or 3 for SNMPv3.

Default: 1

-u *secName*

Specifies the User-based Security Model (USM) user name that is used for SNMPv3 security.

Default: none

-s *secLevel*

Specifies one of the following security levels for SNMPv3 communication:

- 1 - noAuthNoPriv
- 2 - AuthNoPriv

3 - AuthPriv (SNMPv3 only)

-n *contextName*

Specifies the instance name for the MIBMuxed agent.

Default: none

-a *authPassword*

Specifies the authentication password if the agent is configured for SNMPv3 with secLevel 2 (AuthNoPriv) or 3 (AuthPriv).

Note: This option is not required for SNMPv3 communication.

Default: none

-A {MD5 | SHA}

Specifies the authentication protocol to use if the agent is configured for SNMPv3 with secLevel 2 (AuthNoPriv) or 3 (AuthPriv). Currently only MD5 (Message Digest Algorithm) and SHA (Secure Hash Algorithm) are used.

Default: MD5

-x *privPassword*

Specifies the privacy (encryption) password if the agent is configured for SNMPv3 with secLevel 3 (AuthPriv).

Default: none

-X {DES | AES | 3DES}

Specifies the privacy protocol if the SNMPv3 user is configured with secLevel 3 (AuthPriv). Specify DES for Data Encryption Standard, AES for Advanced Encryption Standard using cryptographic keys of 128 bits (AES128), and 3DES for Triple Data Encryption Standard.

Default: none

-m *FIPS_mode*

Controls the FIPS mode of operation. Accepted values are 0, 1, and 2.

0

Indicates Non-FIPS mode.

1

Indicates FIPS co-existence mode.

2

Indicates FIPS only mode.

Default: 1

-r *retries*

Specifies the number of retries.

Default: 10

-t *timeout*

Specifies the duration before the SNMP receiver considers the request as timed out.

Default: 10 seconds

-d *logLevel*

Specifies the log level of the SNMP messages. Accepted values are 0 to 5.

0

Logs fatal messages.

1

Logs critical messages.

2

Logs warning messages.

3

Logs informational messages.

4

Logs all of the messages.

5

Logs all of the messages including debugging messages.

Default: 0

-f *logfile*

Specifies the name of the log file that contains error and debug information.

Default: sysedge_utility.log

command

Specifies the command and associated arguments. Supported commands are the following:

- add
- setstatus
- delete
- list

Note: For more information about the commands, see edgewatch Commands for Windows Event Monitoring (see page 372).

edgewatch Commands for Windows Event Monitoring

The edgewatch Windows event monitoring commands and associated arguments are as follows:

```
add index flags evLog evType "evSrc" "evDesc" "descr" "action"

setstatus index status

delete index

list
```

index

Specifies the row (index) of the monitor table to use for this entry. Each row in the table is uniquely identified by an index number. Rows 1 through 10 are reserved for internal use by the agent, so the index value must be greater than 10 and unique across the table.

flags

Specifies any additional behavioral instructions for this entry using a hexadecimal flags value (for example, 0x00000001).

evLog

Specifies the event log to monitor. This value can be one of the following:

- application
- system
- security
- dnsServer
- dirService
- fileRepService

evType

Specifies the event type to match for this entry. The following are valid types:

- error
- warning
- information
- success
- failure
- all

Entering 'all' indicates that all event types should match.

"*evSrc*"

Specifies the regular expression to use when scanning the Event Source attribute in each event.

"*evDescr*"

Specifies the regular expression to use when scanning the Event Description attribute in each event.

"descr"

Specifies an arbitrary description (0 to 512 characters in length) of the table entry.

"*action*"

Specifies a command (0 to 2048 characters in length), including the full path and any parameters, to run when the entry is matched and a trap is sent. If the string is empty, the agent performs no action for this entry.

Note: You can change the default settings for when the agent performs actions. For more information, see the chapter "Agent Configuration."

status

Specifies the Status textual convention value to use in setting the status of a row in the NT Event Monitor table when used with the setstatus operation. Valid values are as follows. Values can be either the assigned integer values or the actual spelled out status text:

- active(1)
- notInService(2)
- destroy(6)

edgewatch Examples

This section provides examples for using the edgewatch utility with SNMP versions 1, 2c, and 3 to monitor Windows events.

Example: List Entries in the NT Event Monitor Table

The following example lists the contents of the agent's NT Event Monitor table:

```
edgewatch -v 1 -h fe80:ab01::901:bdef -c public -o ntevent list
```

```
edgewatch -v 2c -h 127.0.0.1 -c public -o ntevent list
```

```
edgewatch -v 3 -h fe80:ab01::901:bdef -s 3 -u userName -A authProtocol -a authPassword
-X encryptProtocol -x privPassword -o ntevent list
```

Example: Add an NT Event Monitor Entry

The following example adds a new entry to an agent's NT Event Monitor table at table index 5 to search for login failures on a Windows system.

```
edgewatch -v 1 -h 127.0.0.1 -c private -o ntevent add 5 0x0 Security Failure ".*" ".*"
"Failed login attempt - WARNING" "\local\bin\mail2admin.exe"
```

```
edgewatch -v 2c -h fe80:ab01::901:bdef -c private -o ntevent add 5 0x0 Security Failure
".*" ".*" "Failed login attempt - WARNING" "\local\bin\mail2admin.exe"
```

```
edgewatch -v 3 -h fe80:ab01::901:bdef -s 3 -u userName -A authProtocol -a authPassword
-X encryptProtocol -x privPassword -o ntevent add 5 0x0 Security Failure ".*" ".*"
"Failed login attempt - WARNING" "\local\bin\mail2admin.exe"
```

This example also instructs the agent to run the \local\bin\mail2admin.exe script when the agent finds a match.

Example: Delete an NT Event Monitor Entry

The following example deletes an entry from an agent's NT Event Monitor table at table index 5:

```
edgewatch -v 1 -h 127.0.0.1 -c private -o ntevent delete 5
```

```
edgewatch -v 2c -h fe80:ab01::901:bdef -c private -o ntevent delete 5
```

```
edgewatch -v 3 -h 127.0.0.1 -s 3 -u userName -A authProtocol -a authPassword -X
encryptProtocol -x privPassword -o ntevent delete 5
```

Example: Disable an NT Event Monitor Entry

The following example disables the NT Event Monitor table entry at table index 5 by setting that entry's status to notInService(2). The entry will remain in the table, but the agent will not scan the event log for matches unless the entry's status is returned to active(1):

```
edgewatch -v 1 -h 127.0.0.1 -c private ntevent setstatus 5 2
```

```
edgewatch -v 2c -h fe80:ab01::901:bdef -c private ntevent setstatus 5 2
```

```
edgewatch -v 3 -h 127.0.0.1 -s 3 -u userName -A authProtocol -a authPassword -X
encryptProtocol -x privPassword -o ntevent setstatus 5 2
```

2

Corresponds to the Row Status textual convention value notInService(2).

Remove Windows Event Monitoring Entries

To stop monitoring Windows events, you must remove the appropriate entry from the NT Event Monitor table and the sysedge.cf file. Removing an entry using point configuration from the CA Virtual Assurance user interface accomplishes both. Manual removal requires that you remove the entry from both places in separate operations.

To remove Windows event monitor entries in CA Virtual Assurance

1. Access the CA Virtual Assurance user interface.
2. Click the Resources tab.

   The Managed Resource pane and Data Center page appear.
3. Expand Managed in the Managed Resource pane and select the server on which the agent resides.

   A page appears with server details.
4. Click Configuration, then click NT Event Monitors.

   The NT Event Monitor table appears with all existing self monitors.
5. Select the monitor to delete and click Actions, Delete.

   The entry is deleted from the table. The agent performs a warm start to apply the change.

To remove Windows event monitor entries manually

1. Delete the watch ntevent directive for the entry from the local sysedge.cf file and save the file.
2. Run the edgewatch utility using one of the following commands for the -o parameter:

   `-o delete` *`index`*

   `-o setstatus` *`index`* `6`

   index

   Specifies the index value of the Windows event monitor entry to remove.

   When using the setstatus command, a value of 6 sets the row status to destroy and deletes the entry.
3. Restart the agent.

   The entry is deleted.

Chapter 14: History Collection

This chapter describes the SystemEDGE agent history sampling capability and explains how you can instruct the agent to monitor and store the values of MIB variables over time for future retrieval by a manager.

This section contains the following topics:

History Collection Overview (see page 377)
History Control Table and History Data Table (see page 378)
History Sampling Examples (see page 381)
View the History Control Table (see page 383)
History Control Table Configuration (see page 383)

History Collection Overview

SystemEDGE can track the values of various integer-based MIB objects (counters, gauges, and so on) over time and store them for later retrieval. This functionality, commonly referred to as history sampling, can greatly reduce the amount of management station polling across the network.

Instead of having to continuously poll to collect the value of a MIB variable over time, the manager can instruct the agent to sample and store the values. The management system can contact the agent periodically to upload the complete history of samples. The agent continues to sample and store the specified MIB values even during periods of network outage when the management system cannot communicate with the agent.

History Sampling

The agent uses two SNMP MIB tables to provide the history capability:

- The History Control table for defining the data collection functions.
- The History table for storing the actual data samples.

The control table lets you dynamically configure the agent to sample and store the values of any integer-based MIB variable under its control. The data table stores the values for future retrieval.

To perform baselining and trend analysis, you can configure SystemEDGE to monitor and store the values for swapCapacity, for example, by configuring the agent to sample the value every 5 minutes, and to store the most recent 144 samples. Then, once every 12 hours, the management system can upload the entire 144 samples to obtain the values for swapCapacity that were collected during the preceding 12 hour period.

History Control Table and History Data Table

The History Control table contains parameters that describe the data that the agent will sample and store in the History table. Each row of the History Control table assigns values to the parameters (columnar objects) of the table and thereby defines a specific data collection function. One or more rows (stored samples) in the History table are associated with that single control row.

Each control table row is assigned a unique value of empireHistoryCtrlIndex. A row defines the data collection function by specifying the object instance to be sampled, how often to sample (in multiples of 30 seconds), and the number of samples to keep (buckets). Associated with each data collection function (row of the control table) is a set of rows of the History table. Each row of the History table, which is also named a bucket, holds the value of the specified MIB object that was gathered during one sampling interval.

As each sampling interval occurs, the agent adds a new row to the History table with the same empireHistoryIndex as the other rows for this data collection function. Each new row corresponds to the single row in the History Control table, and has an empireHistorySampleIndex that is one greater than the SampleIndex of the previous sample.

History Control Table Columns

The following list describes the columns of the History Control table:

empireHistoryCtrlIndex

Specifies an integer (1 to MAXINT) that uniquely identifies the entry in the table.

empireHistoryCtrlDescr

Describes the data collection function defined by this entry.

empirehistoryCtrlInterval

Specifies an integer value indicating how often (in seconds) the agent should sample the MIB variable.

Note: The interval must be a multiple of 30 seconds.

empireHistoryCtrlObjID

Specifies the complete object instance identifier of the MIB variable to be sampled.

Note: You must include the instance portion of the object identifier (for example, .0 for scalars). The object instance must exist and must be contained within the Systems Management Empire MIB.

For example, any supported (INTEGER-based) object in MIB-II, the Host Resources MIB, or the Systems Management Empire MIB is valid. Objects should be of integer type including counter, gauge, integer, or enumerated integer.

empireHistoryCtrlObjType

Specifies the ASN1/SNMP type of the MIB variable that the agent is sampling.

empireHistoryCtrlBucketsReq

Specifies the requested number of discrete samples to be saved in the History table. Depending on available resources, the agent sets the empireHistoryBucketsGrant column as close to this value as possible.

empireHistoryCtrlBucketsGrant

Specifies the actual number of discrete samples that the agent will save in the History table for the data collection function defined by this entry. The agent keeps the most recent BucketsGrant number of samples.

empireHistoryCtrlLastCall

Specifies the last time, based on sysUptime, that a sample was taken on behalf of this entry.

empireHistoryCtrlCreateTime

Specifies the time, based on sysUptime, at which this history sampling function was created.

empireHistoryCtrlStatus

Specifies the status of the entry, which can be one of the following:

- active
- notInService
- notReady
- createAndGo
- createAndWait
- destroy

These values are defined in the SNMPv2 SMI RowStatus textual convention.

Setting the status to destroy(6) causes the agent to discontinue history sampling for this entry, and to delete both this row and the corresponding data sample rows in the History table.

empireHistoryCtrlFlags

Represents a bit field. Setting one or more bits changes the behavior of this entry.

000 000 80 - Entry is created by an AIM plugin (Read-only)

000 001 00 - Collects performance cube data

Note: For more information about the History Control table, see the specification empire.asn1 in the mib subdirectory of the agent installation and the chapter "Systems Management Empire MIB."

History Table Columns

Following are the columns of the History table:

Note: For more information about the History table columns, see the MIB specification (empire.asn1 in the mib subdirectory of the agent installation) and the chapter "Systems Management Empire MIB."

empireHistoryIndex

Identifies the row in the History Control table of which this sample is a part. It has the same value as the empireHistoryCtrlIndex for the corresponding entry in the control table.

empireHistorySampleIndex

Specifies the index that uniquely identifies the particular sample this represents among all samples associated with the same history control entry. This index starts at 1 and increases by one as each new sample is stored.

empireHistoryStartTime

Specifies the time, based on sysUptime, at which the first sample was taken.

empireHistorySampleTime

Specifies the time, based on sysUptime, at which this particular sample was taken.

empireHistoryValue

Specifies the current value of the MIB variable taken at this sample.

Row Creation Objects

You can use the following MIB objects with the History table to optimize row creation:

histCtrlUnusedIndex

Returns an unused index number for the History table when you perform an SNMP Get on the variable.

Permissions: Read-only

histCtrlMatchDescr

Determines the index number that corresponds to a particular entry description. Perform an SNMP Set of this MIB object to cause the agent to search through entries in the History table and put the index value of the last entry whose description matches in the histCtrlMatchIndex MIB object.

Permissions: Read-write

histCtrlMatchIndex

Matches a particular entry description with its index number when used with histCtrlMatchDescr.

Permissions: Read-only

History Sampling Examples

The following table shows sample entries in the History Control table:

Index	Desc	Int	ObjID	Type	Bckts-Req	Bckts-Grant	Last-Call	Create-Time	Status
1	1 Minute CPU Load Avg.	60	loadAverage1Min.0	2	5	5	0d 6:30:00	0d 0:0:0	active
2	Memory In Use	30	memInUse.0	2	10	10	0d 6:30:30	0d 0:0:0	active
3	Swap Capacity	900	swapCapacity.0	2	48	48	0d 6:30:00	0d 2:00:00	active

The History Control table uses the SNMPv2 Structure of Management Information (SMI) Row Status Textual Convention for adding and deleting rows.

In this table, the rows are defined as follows:

- Control row 1 causes the agent to poll the 1-minute load average every minute and to store the most recent 5 samples (5 minutes).
- Control row 2 causes the agent to poll the memoryInUse MIB variable every 30 seconds and to store the most recent 10 samples (5 minutes).
- Control row 3 causes the agent to poll the swapCapacity variable (percentage of swap in use) every 15 minutes and to store the most recent 48 samples. By storing 48 samples, a management system needs to upload these samples only once every 12 hours (that is, 48 samples at 15-minute intervals provide 12 hours of coverage).

The following table shows the history samples that were stored in the History table as a result of the sampling configuration defined in the sample History Control table:

Index	SampleIndex	StartTime	SampleTime	Value
1	387	0d 0:0:0	0d 6:26:00	2
1	388	0d 0:0:0	0d 6:27:00	0
1	389	0d 0:0:0	0d 6:28:00	1
1	390	0d 0:0:0	0d 6:29:00	2
1	391	0d 0:0:0	0d 6:30:00	2
2	772	0d 0:0:0	0d 6:26:00	32204
2	773	0d 0:0:0	0d 6:26:30	32213
:	:	:	:	:
:	:	:	:	:
2	781	0d 0:0:0	0d 6:30:30	32224
3	1	0d 2:0:0	0d 2:00:00	70
3	2	0d 2:0:0	0d 2:15:00	64
3	3	0d 2:0:0	0d 2:30:00	66
:	:	:	:	:
:	:	:	:	:
3	19	0d 2:0:0	0d 6:30:00	80

For more information about using the History Control table, see the chapter “History Collection.”

View the History Control Table

If you are using CA eHealth AdvantEDGE View, you can query a system for History information by selecting the system you want to monitor from the System list, selecting History Collection from the Configuration list, and clicking the Configuration icon.

Note: For more information, see the *CA eHealth AdvantEDGE View Web Help*.

History Control Table Configuration

You can control which MIB objects SystemEDGE stores in the History table by adding, deleting, or modifying entries in the History Control table.

You can configure the History Control table in one of these ways:

- Dynamically using one of the following two methods:
  - Use SNMP commands from a management system, such as CA eHealth AdvantEDGE View, to modify the table.
  - Use file-based configuration in CA Virtual Assurance.

  Note: For more information, see Dynamic Configuration During Operation.

  At start-up initialization by specifying the entries for the History Control table in the SystemEDGE configuration file sysedge.cf.

  Note: For more information, see Initial Configuration During Startup.

Initial Configuration During Startup

On startup, the agent reads the sysedge.cf configuration file. You can use this file to specify MIB objects for the agent to monitor by adding appropriate entries to the History Control table using the emphistory directive.

Dynamic Configuration During Operation

You can use the emphistory command line utility to configure entries in the History Control table and to retrieve data samples from the History Control table dynamically. The emphistory utility is located in the bin subdirectory of the agent installation. You can also add history table entries to the agent configuration file manually.

Note: For more information about how to add history table entries to the agent configuration file, see emphistory Utility--Manage Entries in History Control Table.

emphistory Directive--Add Entries to History Control Table

The emphistory directive lets you add entries to the History Control table. You use this directive and its arguments, which represent columns in the History Control table, in the sysedge.cf file to define MIB objects to collect for history sampling.

Use the emphistory directory in the sysedge.cf file to add entries in the History Control table as follows:

`emphistory` *`index interval objid buckets 'description' flags`*

index

Specifies the row number in which the agent will create the entry.

interval

Specifies the interval (in seconds) at which the agent will sample the object's value. The interval must be a multiple of 30 seconds.

objid

Specifies the object instance within the agent's MIB whose value should be sampled. You can specify the OID using the complete dotted-decimal value (for example, 1.3.6.1.2.1.25.1.5.0) or the symbolic MIB name (for example, hrSystemNumUsers.0). In both cases, you must specify the object instance, which is typically zero for non-tabular MIB variables.

buckets

Specifies the number of buckets, or samples, that the agent will store internally. The agent stores the last buckets number of samples. As the agent takes each new sample, it deletes the oldest sample.

'description'

Specifies a quoted string (0 to 128 characters in length) that indicates the description field for this entry.

empireHistoryCtrlFlags

Represents a bit field. Setting one or more bits changes the behavior of this entry.

000 000 80 - Entry is created by an AIM plugin (Read-only)

000 001 00 - Collects performance cube data

flags

Represents a bit field. Setting one or more bits changes the behavior of this entry.

000 000 80 - Entry is created by an AIM plugin (Read-only)

000 001 00 - Collects performance cube data

History Collection Example

The following entry instructs the agent to sample the value of the object MemInUse through table index 15 every 120 seconds and to store the last 60 samples:

```
emphistory 15 120 memInUse.0 60 'MemInUse history'
```

emphistory Commands for Managing Entries in the History Control Table

The emphistory command specifies the type of command to be carried out from the available list of commands: add, setstatus, delete, dump, and list.

Depending on the type of command, you may be required to specify additional parameters to complete the command. The syntax for the commands is as follows:

```
add index objid interval buckets "description" flags
```

```
setstatus index status
```

```
delete index
```

```
dump index
```

```
list
```

index

Specifies the row number for this entry in the agent's History Control table.

Rows are indexed starting at 1. When you specify the dump operation, the agent retrieves all data samples from the History table that correspond to the entry in the History Control table identified by index. If you specify an index value of -1 for dump operations, the agent retrieves all of the contents of the History table.

You must specify an index value for add operations to specify the particular MIB table index of an unused row to use for row creation.

objid

Specifies the complete object instance identifier of the MIB variable that the agent will sample and store in the History Control table. You can specify the OID using the complete dotted-decimal value (for example, 1.3.6.1.2.1.25.1.5.0) or the symbolic MIB name (for example, hrSystemNumUsers.0). You must provide the instance portion of the OID (that is, .0 for scalars). The object instance must exist and must be of integer type (which includes counter, gauge, integer, or enumerated integer).

interval

Specifies the interval in seconds between successive samples; this value must be a multiple of 30 seconds.

buckets

Specifies the number of discrete data samples to be saved in the History table on behalf of this control entry. Each sample, named a bucket, contains the snapshot value of the MIB variable, the time at which the sample was created, the sample index, and an index that corresponds to the entry in the History Control table that defines the data collection function. The History table stores the most recent buckets number of samples.

status

Specifies the RowStatus textual convention value to use in setting the status of a row in the History Control table when used with the setstatus operation. The Status parameter can be one of the following values:

- (1)active
- (2)notInService
- (6)destroy

Setting the row status to destroy(6) causes the agent to discontinue history sampling for that entry, and to delete the row in the History Control table and the corresponding data sample rows in the History table.

'***description***'

(Optional). Specifies a description for the row. If you specify a value, emphistory uses the supplied string instead of the default History Control entry description string. If you include a description, you must enclose it within single quotation marks (' '). The description must be less than 128 characters, not including the quotation marks. Longer strings are truncated.

Flags

Represents a bit field. Setting one or more bits changes the behavior of this entry.

000 000 80 - Entry is created by an AIM plugin (Read-only)

000 001 00 - Collects performance cube data

Important! The following usage of the emphistory utility is deprecated. We strongly encourage using the new argument, because the following format will not be supported in the future.

`emphistory add` *`ipaddr[:port]`* *`commstr ctrlIndex objid interval buckets ['description']`*

emphistory Examples

This section provides examples for using the emphistory utility with SNMP versions 1 and 3. The agent is assumed to be running with default port 161.

Example: List Entries in the History Control Table

The following examples list the contents of the agent's History Control table:

```
emphistory -h 127.0.0.1 -v 1 -c public -o list
```

```
emphistory -h 127.0.0.1 -v 3  -u username  -s 3 -A MD5 -a authPassword -X DES -x
encryptPassword -o list
```

Example: Add a History Control Entry

The following examples add a new control entry at table index 5 to the agent's History Control table. This control entry instructs the agent to sample the ifInOctets.1 MIB object-instance every 60 seconds and to store the most recent 10 samples:

```
emphistory -h 127.0.0.1 -v 1 -c private -o add 5 ifInOctets.1 60 10
```

```
emphistory -h 127.0.0.1 -v 3  -u username  -s 3 -A MD5 -a authPassword -X DES -x
encryptPassword -o add 5 ifInOctets.1 60 10
```

Example: Delete a History Control Entry

The following entries delete the History Control table entry at table index 3:

```
emphistory -h 127.0.0.1 -v 1 -c private -o delete 3
```

```
emphistory -h 127.0.0.1 -v 3  -u username  -s 3 -A MD5 -a authPassword -X DES -x
encryptPassword -o setstatus 3 destroy
```

Note: These entries also instruct the agent to delete the stored data samples in the History table that correspond to this control entry.

Example: Set the Row Status of a Control Entry

The following examples disable the control entry in the History Control table at table index 5, but save the corresponding stored samples in History table:

```
emphistory -h 127.0.0.1 -v 1 -c private -o setstatus 5 2
```

```
emphistory -h 127.0.0.1 -v 3  -u username  -s 3 -A MD5 -a authPassword -X DES -x
encryptPassword -o setstatus 5 2
```

2

Corresponds to the RowStatus textual convention value notInService(2).

Example: Retrieve Stored Data Samples

The following examples retrieve all the stored data samples that correspond to the data-collection function defined in row 5 of the History Control table:

```
emphistory -h 127.0.0.1 -v 1 -c private -o dump 5
```

```
emphistory -h 127.0.0.1 -v 3  -u username  -s 3 -A MD5 -a authPassword -X DES -x
encryptPassword -o dump 5
```

The following examples retrieve all the stored samples for all control entries using -1. This command retrieves the entire contents of the History table:

```
emphistory -h 127.0.0.1 -v 1 -c private -o dump -1
```

```
emphistory -h 127.0.0.1 -v 3  -u username  -s 3 -A MD5 -a authPassword -X DES -x
encryptPassword -o dump -1
```

Chapter 15: Custom MIB Objects

This chapter explains how to extend the Systems Management Empire MIB by adding custom MIB objects.

This section contains the following topics:

Systems Management Empire MIB Extension Group (see page 389)
Extension Group Features (see page 390)
Extension Variable Configuration (see page 390)
Extension Examples (see page 392)
Extension Scripts (see page 394)
Using Extension Variables with Your Management Software (see page 396)
Recommendations for Using Extensions (see page 398)

Systems Management Empire MIB Extension Group

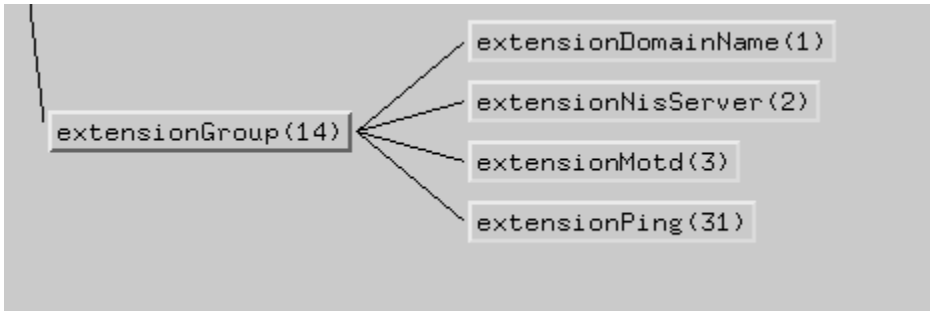
The SystemEDGE agent provides a mechanism for extending the Systems Management MIB to include information about your systems and applications. Using this feature, you can extend the agent to manage your system environment more effectively. You can also design application-specific MIB variables to manage your applications using SNMP without implementing SNMP support within the application source.

The SystemEDGE agent provides this extension support through the Extension group (extensionGroup) of the Systems Management MIB. This group contains unspecified scalar MIB variables that you can configure. In response to a SNMP GetRequest for one of these variables, the SystemEDGE agent invokes the command that you specify for the variable and returns the value that the command returns. Using the SNMP Set operation, you can also pass parameters to a command.

The Extension group is supported on UNIX and Windows systems. On Windows, you may also configure the agent to report on performance and configuration data available in the Windows registry. To support this reporting, the SystemEDGE agent also provides a Windows registry extension group.

Note: For more information about this group, see the chapter “Windows Registry and Performance MIB Objects.”

The following illustration shows four sample extension variables distributed with the agent:



Extension Group Features

The Extension Group of the Systems Management MIB is located at OID 1.3.6.1.4.1.546.14. This group provides space for 2^{32} user-defined scalar MIB variables. These new variables are numbered 1 through 2^{32} and are referenced as any other scalar MIB object. For example, extension object 1 is referred to in Get and Set request messages as 1.3.6.1.4.1.546.14.1.0. In all cases, extension variables use the object instance identifier of .0.

You can specify variables as read-only or read-write. An extension variable can be any valid base SNMP type, including the following:

- Integer
- Counter
- Gauge
- Octetstring
- TimeTicks
- ObjectId
- IpAddress

Note: For more information about starting the agent with its debugging options turned on, see the chapter “Starting the Agent.”

Extension Variable Configuration

You can configure extension variables in the SystemEDGE agent configuration file, sysedge.cf. On startup, the agent configures these values and verifies whether the program or command associated with the variable is executable. Within the configuration file, the keyword extension defines an extension variable. The next section describes how to use this keyword.

extension Keyword--Add Entries to the Extension Group

You can use the extension keyword to add entries in the Systems Management Empire MIB Extension group. Use the keyword in the sysedge.cf file to specify custom MIB objects to make available for monitoring.

Use the extension keyword to add entries to the extension group as follows:

`extension` *`LeafNumber Type Access 'Command'`*

LeafNumber

Specifies an extension variable number in the range of 1 through 2^{32} defined by this entry.

Type

Specifies the SNMP type for this entry. The supported types are as follows:

- Integer
- Counter
- Gauge
- Octetstring
- TimeTicks
- Objectid
- IPAddress

Access

Specifies the access type, which can be either read-only or read-write.

'Command'

Specifies a quoted string, 0 to 256 characters in length, that specifies the full path of the command that should be run (with any parameters). The command can be a platform independent extension name or full path of the command that runs when this variable is accessed through either a Get, GetNext or Set request. In case of platform independent extension the executable file named <ExtensionName> is searched in *CASYSEDGE*/extension/*<ExtensionName>*. If the command file is not currently accessible, the configuration of this variable fails.

Additional Parameters

In addition to the parameters that you specify in the configuration file as part of the extension command, the SystemEDGE agent passes some additional parameters to help you decide how to treat the request. These parameters are passed after any parameters that you specified in quotation marks as part of the extension statement.

The following list describes the additional parameters that you can pass to the extension command:

Leafnumber

Specifies an integer value that represents the leaf number of this variable (1 through 232). If you have a single script that supports multiple extension values, you can use this parameter to determine which variable is being requested.

Request-Type

Indicates the type of SNMP request, which can be one of the following:

- Get
- GetNext
- Set

The request type is passed with all letters capitalized.

Set-Value

Specifies a string that contains the value that passed in a Set request. Use this string in your extension program to modify the current value of the variable in such a way that a future Get or GetNext can return this value.

Extension Examples

The SystemEDGE agent includes several sample extension variables. These examples are defined in the sample sysedge.cf file. The scripts that implement these examples are included in the contrib subdirectory of the SystemEDGE agent's installation. These variables are also defined in the Systems Management Empire MIB (empire.asn1 in the mib subdirectory of the agent installation).

Important! Before you add your own extensions, carefully review the examples in this chapter and in the Systems Management Empire MIB. For clarity, these examples include the appropriate configuration file extension commands.

You can add these extensions to your sysedge.cf file to make them available to the SystemEDGE agent.

Example: Platform Independent Extension

The following extension object returns on Unix the output of command /opt/CA/SystemEDGE/extension/myextension/myextension or on Windows output of "C:\Program Files\CA\SystemEDGE\extesnion\myextension\myextension.<EXT>" where <EXT> is one of the executable extensions from environment variable %PATHEXT%:

```
extension 1 OctetString ReadOnly myextension
```

The instance identifier of this object is 1.3.6.1.4.1.546.14.1.0.

Example: Remote Ping (UNIX and Windows)

Use the following extension objects to instruct the SystemEDGE agent to ping a remote host from the host where the agent is running.

For UNIX systems, enter the following in the sysedge.cf file:

```
extension 31 OctetString ReadOnly "/bin/ping www.ca.com"
```

For Windows systems, enter the following in the %SystemRoot%\System32\sysedge.cf file:

```
extension 31 OctetString ReadOnly "%WINDIR%\system32\ping.exe www.ca.com"
```

The instance identifier of this object in both examples is 1.3.6.1.4.1.546.14.31.0.

Extension Scripts

The SystemEDGE agent places very few constraints on the operation of extension scripts. It does, however, require the following:

- All operations (Set, Get, or GetNext) must have output. The output of Set invocations should echo the value that was actually set, while the output of Get and GetNext should be the object's value. If there is no output, the SNMP query fails.
- Output from extension scripts is only parsed up through the first newline character. If the first character is a newline, the output is considered NULL, causing the SNMP query to fail and returning an error.

Note: Because the SystemEDGE agent runs as root or administrator, make sure that all commands and scripts use absolute pathnames, are fully debugged, and contain no ambiguous code or unnecessary options.

On Windows systems, you can use batch files for writing extension scripts. The agent can directly run those batch files. However, batch file functionality is severely limited. Use Perl and other scripting languages instead for Windows.

Example Script Test

After you create an extension script, test it at the command line. The following example creates an OctetString extension on a UNIX system, tests its output, and then uses SNMP to return the value from the script.

To test the script

1. Create a file executable file named /opt/CA/SystemEDGE/debugext.sh with the following information:

```
#!/bin/sh
Script=$0
LeafNumber=$1
Operation=$2
SetValue=$3

if [ "$Operation" = "SET" ]; then
echo $Script     > /tmp/debugext_set.txt
echo `date`      >> /tmp/debugext_set.txt
echo $LeafNumber >> /tmp/debugext_set.txt
echo $Operation  >> /tmp/debugext_set.txt
echo $SetValue   >> /tmp/debugext_set.txt
echo $SetValue   >  /tmp/debugext_setvalue.txt #SET VALUE HERE
echo $SetValue   #You must echo back what was set
fi

# Override GETNEXT and just do GET
if [ "$Operation" = "GETNEXT" ];  then
Operation="GET"
fi

if [ "$Operation" = "GET" ]; then
echo $Script     > /tmp/debugext_get.txt
echo `date`      >> /tmp/debugext_get.txt
echo $LeafNumber >> /tmp/debugext_get.txt
echo $Operation  >> /tmp/debugext_get.txt
if [ ! -f /tmp/debugext_setvalue.txt ]; then
echo "ERROR: do set first... Script:$Script
LeafNumber:$LeafNumber Operation:$Operation"
exit 0
fi
cat /tmp/debugext_setvalue.txt  #return SET VALUE
fi
```

   Note: This example tests an extension script that is an OctetString. It is valid for UNIX operating systems. On Windows systems, you must call the interpreter in your action script command. For example, enter perl.exe myscript.pl.

2. Add the following line to sysedge.cf:

```
extension 1 OctetString ReadWrite /opt/CA/SystemEDGE/debugext.sh
```

Note: This example tests an extension script that is an OctetString. It is valid for UNIX operating systems. On Windows systems, you must call the interpreter in your action script command. For example, enter perl.exe myscript.pl.

3. Enter the following at the command line to create a file named myset.txt:

   ```
   echo "1.3.6.1.4.1.546.14.1.0 04 debugSetString" > myset.txt
   ```

   You now have a file named myset.txt. You are setting the OctetString(04) “debugSetString” to the OID 1.3.6.1.4.1.546.14.1.0.

4. Issue the SNMP Set by entering the following:

   ```
   ./snmpset -c private -h 127.0.0.1 < myset.txt
   ```

5. Retrieve the value of your debugSetString by entering the following:

   ```
   ./snmpget -c public -h 127.0.0.1 -o 1.3.6.1.4.1.546.14.1.0
   ```

6. Test the setup by entering the following at the command line:

   ```
   ./debugext.sh 1 SET debugSetString2
   ```

   This command should return the value as being set as output.
   ```
   ./debugext.sh 1 GET
   ```

   This command should return the value as set in the earlier SET call.

Using Extension Variables with Your Management Software

All methods for incorporating extension variables into your management system software require you to edit and import MIB specification files. This guide does not discuss details of importing MIB specification files into management system software, but it does describe the two overall strategies that exist for incorporating extension variables:

- Edit empire.asn1 to include the extension variables defined for your site.
- Edit a separate MIB file to include the extension variables defined for your site.

How to Edit empire.asn1 for Extension Variables

Follow these steps to add your own extension variables to the Systems Management MIB (empire.asn1):

1. Create and debug the relevant extension scripts, and then configure them in the agent's configuration file, sysedge.cf, to include them.
2. Edit empire.asn1 to include new extension MIB variables that exist under the extensionGroup.
3. Perform a test compile of empire.asn1 to make sure that there are no syntax errors. This procedure is specific to your management system and MIB compiler.
4. Import the new empire.asn1 file into your management system software. This procedure is specific to your management system and MIB compiler.

   Note: If you do not reimport the MIB file, your management system software will not be able to access the new extension MIB objects.

How to Edit a Separate MIB Specification for Extension Variables

You can put extension variables in a separate MIB specification file for ease of updating. You can save time and effort by making changes to a MIB specification file, which is much smaller than empire.asn1, thereby avoiding recompiling and reloading the entire empire.asn1 file.

To add your extension variables to a separate MIB specification file

1. Create and debug the relevant extension scripts, and configure them in the agent's configuration file, sysedge.cf, to include them.
2. Edit your own MIB specification file (for example, extensions.asn1) to include new extension MIB variables that exist under the Systems Management Extension group.
3. Perform a test compile of extensions.asn1 to check that there are no syntax errors. This procedure is specific to your management system and MIB compiler.
4. Import the extensions.asn1 file into your management software. This procedure is specific to your management system and MIB compiler.

   Note: If you do not reimport the MIB file, your management system software will not be able to access the new extension MIB objects.

Recommendations for Using Extensions

When you create SystemEDGE extensions, follow these guidelines:

- Do not edit empire.asn1. Use a separate MIB specification.
- Use SystemEDGE debugging (log file monitoring) to find output, plug-in, and extension problems.
- Use 128 characters or less for extension entries to meet the Windows limitation on command line length. Remember that the length includes paths, commands, and arguments.
- Do not use batch files. Instead use Windows scripting, VBScript, Perl, C, Shell, and so on.
- Watch the fork/exec time for new extensions. Extensions are synchronous and can block other actions.

Chapter 16: Windows Registry and Performance MIB Objects

This chapter explains how to add objects to the Systems Management Empire MIB for Windows registry and performance data.

This section contains the following topics:

Windows Registry and Performance Functionality (see page 399)
Systems Management Empire MIB ntRegPerf Group (see page 400)
Windows Registry and Performance Variable Configuration (see page 403)
Windows Registry and Performance Examples (see page 405)
Using Windows Registry and Performance Variables with Your Management Software (see page 406)

Windows Registry and Performance Functionality

SystemEDGE provides a powerful mechanism for extending the Systems Management MIBSystems Management Empire MIBn from the Windows registry and performance counters. This information includes configuration data (which is typically viewed using the Windows program regedit) and performance data (which is typically viewed using the Windows program perfmon).

Using this feature, you can customize the agent to return additional configuration and performance data for your systems and applications. For example, you can use the agent to make many application registry entries available through SNMP. Also, you can access performance statistics for many applications.

Systems Management Empire MIB ntRegPerf Group

The SystemEDGE agent provides the support for additional Windows registry and performance parameters in the ntRegPerf group of the Systems Management Empire MIB. The ntRegPerf group of the Systems Management Empire MIB is located at OID 1.3.6.1.4.1.546.5.7. This group provides a space for up to 128 user-defined scalar MIB variables. These new variables are numbered 1 through 128 and are referenced just like any other scalar MIB object. For example, ntRegPerf object 1 is referred to in Get request messages as 1.3.6.1.4.1.546.5.7.1.0. In all cases, ntRegPerf variables use the object-instance identifier of .0. In response to a SNMP Get request for one of these variables, the agent will read the Windows registry and return the value obtained.

An ntRegPerf variable can be any valid base SNMP type, including the following:

- Integer
- Counter
- Gauge
- OctetString
- TimeTicks
- ObjectId
- IpAddress

Registry Data

SystemEDGE provides data from the standard configuration registry. This data is indexed by both a key name and a value. For example, the key SYSTEM\CurrentControlSet\Control\CrashControl and the value DumpFile identify a text string that describes the location of the system dump file. Only LOCAL_MACHINE registry data is supported.

The following list matches the registry data types that SystemEDGE supports with the preferred SNMP type:

REG_DWORD

Integer

REG_SZ

OctetString

REG_EXPAND_SZ

OctetString

REG_MULT_SZ

OctetString; only the first string is returned

Performance Data

SystemEDGE provides access to performance counters in the performance registry by specifying the object and counter names. For instance, the object NWLink NetBIOS and counter Bytes Total/sec identifies the NetBIOS byte-counter statistic. When using performance counters, you must interpret the data carefully. There are at least 27 different counter types in Windows. Most counters do not actually report the data in the format that you assume based on the name of the counter.

Familiarize yourself with Windows Performance counters before using this feature. For more information, refer to the section on optimizing Windows in the Windows Resource Kit.

The following list matches the 4-byte performance data types that SystemEDGE supports with the preferred SNMP types:

PERF_COUNTER_COUNTER

Counter or integer

PERF_COUNTER_RAWCOUNT

Counter, integer, or gauge

PERF_COUNTER_RAWCOUNT_HEX

Counter, integer, or gauge

PERF_SAMPLE_COUNTER

Counter, integer, or gauge

The following list matches the 8-byte performance data types that SystemEDGE supports with the preferred SNMP types. Because SNMPv1 supports only 4-byte values, agent returns only the least significant 4 bytes of data.

PERF_COUNTER_TIMER

Counter, integer, or gauge

Note: Returns data as percentage multiplied by 100.

PERF_COUNTER_BULK_COUNT

Counter, integer, or gauge

PERF_COUNTER_LARGE_RAWCOUNT

Counter, integer, or gauge

PERF_COUNTER_LARGE_RAWCOUNT_HEX

Counter, integer, or gauge

PERF_COUNTER_TIMER_INV

Counter, integer, or gauge

Note: Returns data as percentage multiplied by 100.

PERF_AVERAGE_BULK

Counter, integer, or gauge

PERF_100NSEC_TIMER

Counter or integer

Note: Returns data as percentage multiplied by 100.

PERF_100NSEC_TIMER_INV

Counter or integer

Note: Returns data as percentage multiplied by 100.

PERF_COUNTER_MULTI_TIMER

Counter or integer

Note: Returns data as percentage multiplied by 100.

PERF_COUNTER_MULTI_TIMER_INV

Counter or integer

Note: Returns data as percentage multiplied by 100.

PERF_100NSEC_MULTI_TIMER

Counter or integer

Note: Returns data as percentage multiplied by 100.

PERF_100NSEC_MULTI_TIMER_INV

Counter, integer, or gauge

Note: Returns data as percentage multiplied by 100.

PERF_ELAPSED_TIME

Integer

PERF_SAMPLE_FRACTION

Integer or gauge

Note: Returns data as percentage multiplied by 100.

PERF_RAW_FRACTION

Integer or gauge

Note: Returns data as percentage multiplied by 100.

PERF_LARGE_RAW_FRACTION

Integer or gauge

Note: Returns data as percentage multiplied by 100.

Unsupported Performance Data Types

Several counters with multiple samples and internal data are difficult to present in a single value and require significant post-processing. For that reason, CA SystemEDGE does not support the following counter types:

- PERF_COUNTER_QUEUELEN_TYPE
- PERF_COUNTER_TEXT
- PERF_COUNTER_NODATA
- PERF_SAMPLE_BASE
- PERF_AVERAGE_TIMER
- PERF_AVERAGE_BASE
- PERF_COUNTER_MULTI_BASE
- PERF_RAW_BASE
- PERF_COUNTER_HISTOGRAM_TYPE

Windows Registry and Performance Variable Configuration

Configure Windows registry and performance variables in the sysedge.cf file. When SystemEDGE starts, it configures these values and verifies that the value associated with each variable is accessible. If not, it prints an error message to sysedge.log and does not create the MIB object.

ntregperf Keyword—Add Windows Registry and Performance MIB Objects

Use the ntregperf keyword to define the Windows registry and performance MIB objects in the sysedge.cf file for the ntRegPerf group.

You can use the ntregperf keyword to add entries in the ntRegPerf group as follows:

Registry data:

```
ntregperf LeafNumber Type Registry 'Key' 'Value'
```

Performance data:

```
ntregperf LeafNumber Type Performance 'Object' 'Counter' 'PerfInstance'
```

LeafNumber

Defines the ntRegPerf variable number, in the range of 1 through 128.

Type

Specifies the SNMP type for this entry, which can be one of the following:

- Integer
- Counter
- Gauge
- Octetstring
- TimeTicks
- Objectid
- IPAddress

Registry

Selects a configuration registry entry.

Performance

Selects a configuration performance entry.

'Key'

Specifies a quoted string, 0 to 512 characters in length, that specifies the registry key to be accessed.

'Value'

Specifies a quoted string, 0 to 128 characters in length, that specifies the registry key' to be accessed.

'Object'

Specifies a quoted string, 0 to 512 characters in length, that specifies the performance object to be accessed.

'Counter'

Specifies a quoted string, 0 to 128 characters in length, that specifies the object's performance counter value to be accessed.

'PerfInstance'

Specifies the performance counter instance to be accessed, which should be equivalent to that listed in the Windows perfmon utility.

Windows Registry and Performance Examples

The SystemEDGE agent includes several sample ntRegPerf variables. These examples are defined in the sample sysedge.cf file. Before you add your own ntRegPerf extension, study these examples and their definitions in the Systems Management Empire MIB (empire.asn1 in the mib subdirectory of the agent installation). For clarity, these examples include the appropriate configuration file ntRegPerf commands.

Example: CrashControl DumpFile

The following ntRegPerf object returns the path to the dump file:

```
ntregperf 1 OctetString Registry 'SYSTEM\CurrentControlSet\Control\CrashControl'
'DumpFile'
```

The object instance identifier of this object is 1.3.6.1.4.1.546.5.7.1.0.

Example: Total Number of Threads

The following ntRegPerf object returns the total number of threads currently available in the system:

```
ntregperf 2 Gauge Performance 'Objects' 'Threads' '1'
```

The object instance identifier of this object is 1.3.6.1.4.1.546.5.7.2.0.

Example: TCP Segments Sent/Sec

The following ntRegPerf object returns the total number of TCP segments that were transmitted by the system:

```
ntregperf 3 Counter Performance 'TCP' 'Segments Sent/sec' '1'
```

The object instance identifier of this object is 1.3.6.1.4.1.546.5.7.3.0.

Using Windows Registry and Performance Variables with Your Management Software

There are several methods for incorporating Windows registry and performance variables into your management system software, all of which require editing and importing MIB specification files. While the details of importing MIB specification files into management system software are beyond the scope of this guide, two overall strategies exist for incorporating Windows registry and performance variables:

- Edit empire.asn1 to include the variables defined for your site.
- Edit a separate MIB file to include the variables defined for your site.

How to Edit empire.asn1 for ntRegPerf Variables

You can add your own ntRegPerf variables to the Systems Management Empire MIB using the following process:

1. Create and debug the relevant ntRegPerf entries in the sysedge.cf file to include them.
2. Edit empire.asn1 to include new ntRegPerf MIB variables.
3. Perform a test compile of empire.asn1 to make sure there are no syntax errors. This procedure is specific to your management system and its corresponding MIB compiler.
4. Import the new empire.asn1 file into your management system software. This procedure is specific to your management system and MIB compiler.

   Note: If you do not reimport the MIB file, your management system software will not be able to access the new MIB objects.

How to Add a Separate MIB Specification for ntRegPerf Variables

You can put ntRegPerf variables in a separate MIB specification file to avoid directly manipulating the MIB file. You can save time and effort by making changes to a MIB specification file, which is much smaller than empire.asn1. This avoids recompiling and reloading the entire empire.asn1 file.

To add a separate MIB specifications for ntRegPerf variables, complete the following process:

1. Edit your own MIB specification file (for example, ntregperf.asn1) to include the new ntRegPerf MIB variables that exist under the Systems Management Empire MIB ntregperfGroup.
2. Perform a test compile of the ntregperf.asn1 file to make sure there are no syntax errors. This procedure is specific to your management system and its corresponding MIB compiler.
3. Import the ntregperf.asn1 file to your management system software. This procedure is specific to your network management station and its corresponding MIB compiler.

   Note: If you do not reimport the MIB file, your management system software will not be able to access the new MIB objects.

Chapter 17: SystemEDGE AIMs

AIMs are plug-in modules that add additional monitoring functionality to the base SystemEDGE. This chapter covers configuring the AIMs that are provided with the agent. For more information about AIM architecture, see the chapter "Concepts."

This section contains the following topics:

Core AIMs (see page 409)
Top Processes AIM (see page 409)
Monitor Maintenance Windows AIM (see page 410)
Performance Cube AIM (see page 411)

Core AIMs

The monwin and perfcube AIMs are SystemEDGE Core AIMs. Core AIMs are installed with the SystemEDGE Core package. They do not require a sysedge_plugin statement in the sysedge.cf file. If the functionality is needed, SystemEDGE automatically loads SystemEDGE Core AIM libraries.

- If a monwin entry is found in the sysedge.cf file, the agent loads the Monitor Maintenance Windows AIM.
- If an emphistory entry with the collect flag (0x100) exists in the sysedge.cf file, the agent loads the Performance Cube AIM.

Versions prior to CA SystemEDGE 5.0 included an AIM called Top Processes AIM. In CA SystemEDGE 5.0, the functionality of the Top Processes AIM is dissolved into the Core agent. Therefore, this AIM is no longer a loadable plugin.

Top Processes AIM

The functionality of the previous Top Processes AIM is included in the SystemEDGE Core. You disable the Top Processes functionality when you activate the “no_topprocs_table” directive in the sysedge.cf file.

Monitor Maintenance Windows AIM

Use the Monitored Windows AIM for time-based control of the activity of SystemEDGE monitoring. This AIM applies to self-monitoring, process monitoring, process group monitoring, log file monitoring, history, and Windows event monitoring (for Windows only). Using the Monitored Windows AIM stops the production of traps (alarms) from such monitors during periods of the day when they would not be significant without stopping the monitor itself.

Enable the Monitor Maintenance Windows AIM

To start the Monitor Maintenance Windows AIM, you must specify an appropriate entry in the sysedge.cf file. SystemEDGE performs a warm-start and starts the AIM automatically.

The following entry specifies a monitoring (maintenance) time window for a monitor:

Syntax:

`monwin` *`monitor index timeOff timeOn status`*

monitor

Specifies the monitor table (selfmon, processmon, procgroupmon, nteventmon, logfilemon, emphistory).

index

Specifies the monitor's index.

timeOff

Specifies the time in 24-hour format (hhmm) at which the monitor is switched off (notInService). The maintenance window begins.

timeOn

Specfies the time in 24-hour format (hhmm) at which the monitor is switched on (active). The maintenance window ends.

status

Specifies if this entry is active (1) or notInService (2).

Example

```
monwin processmon 12 2200 2330 1
```

Limitations of the Monitor Maintenance Windows AIM

The Monitored Windows AIM functions only on a daily basis. There is no support for day of the week, date of the month, or other non-daily monitoring controls.

Any monitor specified in the AIM configuration file that does not actually exist in the monitor table is ignored until the time when it might exist. Any monitor upon which the AIM runs timing control is ignored when its index is removed from the monitor table.

The AIM attempts to run monitor activations and deactivations at the exact specified times. However, on systems with a significant load, the AIM may not experience a cycle at the exact moment. Therefore, the time off and the time on for any row should be greater than one minute apart.

Any single monitor may be the target of multiple rows in the AIM's configuration file. That is, a monitor may be switched off and on several times during the day. It is up to you to avoid the logical problems that might result from overlap of off and on times in this case.

Performance Cube AIM

The Performance Cube AIM reads the history table entries, performs metric averaging over the cube collection interval (default = 20 min), and writes the average metric value into the cube file (performance data file). The AIM uses the existing SystemEDGE history tables to define which metrics are collected and also facilitates averaging of the metric values.

Enable the Performance Cube AIM

SystemEDGE implicitly loads the Performance Cube AIM, if at least one emphistory entry with the collect flag (0x100) exists in the sysedge.cf file. The AIM does not require a sysedge_plugin entry in sysedge.cf.

Syntax

`emphistory` *`index interval oid number`* [*`'descr'`* [*`flags`*]]

index

Specifies the table index to be used.

interval

Specifies the collection interval in a multiple of 30 seconds.

oid

Specifies the OID of the object to collect (either a full OID in dot-notation or *attrName*.*instOID*).

number

Specifies the number of samples to collect.

descr

Specifies an arbitrary description.

flags

If specified as 0x100, it enables the Performance Cube AIM.

Example

```
emphistory 11 60 loadAverage1Min.0 60 '1-min load average history' 0x100
```

Chapter 18: Command Line Utilities

This chapter describes the usage of the SystemEDGE command line utilities.

This section contains the following topics:

SystemEDGE Command Line Utilities (see page 413)
SystemEDGE Additional Command Line Utilities (see page 414)
Configuring SystemEDGE Command Line Utilities (see page 414)

SystemEDGE Command Line Utilities

If you run SystemEDGE in unmanaged mode, you can use the SystemEDGE command line utilities to configure the parameters to operate CA SystemEDGE agent.

The SystemEDGE command line utilities communicate using SNMPv1, SNMPv2c, and SNMPv3 in both IPv4 and IPv6 environments. These utilities reside in the bin sub-directory of the SystemEDGE installation directory.

The information that the utilities pass should match the information in the configuration files of the agent: sysedge.cf and sysedgeV3.cf. For more information about the SystemEDGE configuration files, see the Agent Configuration chapter.

The agent supports the following SNMP command line utilities:

- diagsysedge
- edgemon
- edgewatch
- emphistory
- se_enc
- sendtrap
- snmpget
- snmpset
- sysvariable
- walktree
- edgetrapmon

SystemEDGE Additional Command Line Utilities

SystemEDGE provides additional command line utilities that are not SNMP-based. These utilities act as action items with SystemEDGE monitors. Actions run when a monitor table entry evaluates to true, or for stateful monitors, when the threshold breach results in a state change of the related managed object.

The agent supports the following additional command line utilities:

- bounce (Windows)
- checkfile
- email
- getver (Windows)
- restartproc.exe (Windows)
- restartproc.sh (UNIX/Linux)
- restartsvc (Windows)

Configuring SystemEDGE Command Line Utilities

Configure the parameters to operate the CA SystemEDGE agent by accessing the SystemEDGE command line utilities.

Follow these steps:

1. Log in to the managed node with administrator privileges.
2. Open a Command Prompt window and change to the Install_Dir/SystemEDGE/bin directory.

   You can now enter the required SystemEDGE commands in the command prompt.

Note: For more information about the SystemEDGE command line utilities, use the console help option (–h). For example, to learn about edgemon, enter the following command at the command prompt:

```
edgemon –h
```

Chapter 19: Examples Using State Management

The following examples use typical state management functionality of SystemEDGE.

This section contains the following topics:

Monitor the Windows Task Scheduler (see page 415)
Monitor the Existence of Quarantined Files (see page 417)
Monitor File Systems (see page 417)
Reconfigure Extension Scripts Through SNMP (see page 419)
Monitor Log Files and Send Notification Through Email (see page 423)
Monitor Processes (see page 424)
Create Monitored Objects (see page 425)

Monitor the Windows Task Scheduler

This example monitors the Task Scheduler for failed or excessive tasks on Windows. It consists of a set of entries for the sysedge.cf file and an associated schedlgu.cmd script. The schedlgu.cmd script counts *.job files, analyzes the *.txt files in the %SystemRoot%\Tasks directory, and writes the results into the %CASYSEDGE_DATA%\schedlgu.log file. SystemEDGE launches the script and monitors the schedlgu.log file to evaluate the current status.

Copy the schedlgu.cmd script into the %CASYSEDGE%\bin directory or add it to a configuration deployment package that delivers it to the %CASYSEDGE%\bin\managedscripts directory.

Entries for sysedge.cf

```
# Windows sample configuration for monitoring the Task Scheduler for failed or
excessive tasks
extension 55 integer readonly 'schedlgu.cmd %CASYSEDGE_DATA%\schedlgu.log'
watch logfile 55 0x0 '%CASYSEDGE_DATA%\schedlgu.log' 'The task completed with an exit
code of [(][0]*[^0].*[)]' 'Windows Task Scheduler Job Execution Errors' '' 10 warning
watch logfile 56 0x0 '%CASYSEDGE_DATA%\schedlgu.log' 'WARNING' 'Windows Task
Scheduler Warnings' '' 10 major
watch logfile 57 0x0 '%CASYSEDGE_DATA%\schedlgu.log' 'ERROR' 'Windows Task Scheduler
Errors' '' 10 critical
monitor oid extensionGroup.55.0 55 0x100 600 absolute > 15  'Number of Scheduled Tasks
> 15'  '' 'SysHealth' 'Scheduled Tasks' 'Count' 'minor'
```

Extension script schedlgu.cmd

```
@echo off

REM SystemEDGE extension script for monitoring Windows Task Scheduler

setlocal

set SLOG=.\schedlgu.log
if "%1" NEQ "" set SLOG=%1

set SDIR=%SystemRoot%\Tasks

set SREC=
set DONE=N

REM Logical records in the Task Scheduler log file span multiple lines.
REM Convert into single line format (as %SLOG%) for monitoring via SystemEDGE.
if exist %SLOG% del %SLOG%
for /f "tokens=*" %%i in ('type "%SDIR%\schedlgu.txt"') do set LINE=%%i&call
:parseline

REM Return number of configured tasks
set COUNT=0
for /f "tokens=*" %%i in ('dir /b "%SDIR%\*.job"') do call :countline
echo %COUNT%

endlocal

goto :EOF

:countline
set /a COUNT+=1
goto :EOF

:parseline
REM Replace double quotes with pound sign
set LINE=%LINE:"=#%
REM Check if we have already processed the most recent record in the log
if "%DONE%" EQU "Y" goto :EOF
REM Check for start of next logical record
if "%LINE:~0,1%" EQU "#" goto :startrecord
if "%LINE%" EQU "[ ***** Most recent entry is above this line ***** ]" set DONE=Y&goto
:startrecord
REM Concatenate lines for current record
set SREC=%SREC% %LINE%
goto :EOF
```

```
:startrecord
if "%SREC%"=="" goto :initrecord
REM Restore double quotes
set SREC=%SREC:#="%
REM Write current record
@echo %SREC%>>"%SLOG%"

:initrecord
set SREC=%LINE%
```

Monitor the Existence of Quarantined Files

This example monitors the existence of any files quarantined by eTrust Antivirus in the %ProgramFiles%\CA\eTrustITM\Move directory every ten minutes. The example consists of two entries for the sysedge.cf file and uses the logfile parameter to monitor the entries in a directory. The 0x1808 flag specifies to monitor directories recursively and not to log any traps.

```
# Windows sample configuration for monitoring the existence of any files quarantined
by eTrust Antivirus
watch logfile 45 0x1808 '%ProgramFiles%\CA\eTrustITM\Move' '' 'Monitor eTrust
Antivirus Quarantine directory' '' 10 warning
monitor logMonitorEntry '%ProgramFiles%\CA\eTrustITM\Move' logMonitorLogFileCount
45 0x100 600 absolute > 0 'eTrust Antivirus Quarantine directory not empty' '' '' ''
'' 'major'
```

Monitor File Systems

This example monitors file systems on Windows and consists of three groups of entries for the sysedge.cf file:

- Usage of object state model with different severities dependent from different thresholds.
- Usage of flag 0x800 in combination with the same class, instance, attribute, and severity. Flag 0x800 specifies the aggregate state of all monitors with same severity and applies the AND relation.
- Usage of aggregate_level control.

Note: The prefix //./ which is added to the object instance per default (denoting localhost) is omitted in the example.

```
# Derives a state for file system C: (System).
# Creates an aggregate ‘FSys’ ‘System’ ‘PercentUsed’, which can be in state ok, minor,
major, or fatal.

monitor devTableEntry 'C:' devCapacity 51 0x0 120 absolute > 90 'Description' '' 'FSys'
'System' 'PercentUsed' minor
monitor devTableEntry 'C:' devCapacity 52 0x0 120 absolute > 95 'Description' '' 'FSys'
'System' 'PercentUsed' major
monitor devTableEntry 'C:' devCapacity 53 0x0 120 absolute > 99 'Description' '' 'FSys'
'System' 'PercentUsed' fatal

# Derives warning state if three conditions are met simultaneously, that is, only if
all three Backup file systems E:, F:, and G: are above 95%.
# Creates an aggregate ‘FSys’ ‘Backup’ ‘PercentUsed’, which can be in state ok or
warning.

monitor devTableEntry 'E:' devCapacity 54 0x800 120 absolute > 95 'Description' ''
'FSys' 'Backup' 'PercentUsed' warning
monitor devTableEntry 'F:' devCapacity 55 0x800 120 absolute > 95 'Description' ''
'FSys' 'Backup' 'PercentUsed' warning
monitor devTableEntry 'G:' devCapacity 56 0x800 120 absolute > 95 'Description' ''
'FSys' 'Backup' 'PercentUsed' warning

# Derives worst state of all file systems.
# Creates a super aggregate ‘FSys’ ‘*’ ‘PercentUsed’, which can be in state ok, warning,
minor, major, or fatal.

aggregate_level 0x0002
```

Reconfigure Extension Scripts Through SNMP

These scripts allow you to ping an arbitrary system using an extension script that you can reconfigure remotely using SNMP. Therefore, it is no necessary to edit sysedge.cf or any other custom script. You can set or change the name of the system you want to observe by using SNMP SET from any MIB Browser or event snmpset utility provided with SystemEDGE. Reading the extension variable provides a string with ping information.

To reconfigure extension scripts through SNMP

1. Add the following line to the configuration sysedge.cf file and restart SystemEDGE:

   ```
   extension 31 octetstring readwrite '/opt/CA/test/ping.sh' (UNIX, Linux)
   extension 31 octetstring readwrite 'c:\test\ping.bat' (Windows)
   ```

2. Copy the script (ping.sh, ping.bat) into the directory that you specified in the extension directive.

3. Enter a read request without setting up a ping target:

   ```
   C:\SystemEDGE\bin>.\snmpget -o 1.3.6.1.4.1.546.14.31.0
   1.3.6.1.4.1.546.14.31.0 Ping target must be set first
   ```

   An error message appears.

4. Set my-sles3 as the system which you want to ping:

   ```
   C:\SystemEDGE\bin>.\snmpset -c private -o 1.3.6.1.4.1.546.14.31.0 -s my-sles3
   1.3.6.1.4.1.546.14.31.0 my-sles3
   ```

5. Ping the system by reading the variable:

   ```
   C:\SystemEDGE\bin>.\snmpget -o 1.3.6.1.4.1.546.14.31.0
   1.3.6.1.4.1.546.14.31.0 my-sles3 is alive
   ```

The script checks that the extension has index 31. However, this check is technically not required and can be removed, so that the extension script can be assigned to any index.

The scripts ping.bat and ping.sh stores the configuration information in a temporary file.

Ping script for UNIX and Linux

```
#!/bin/sh
#
# ping.sh example extension script
#
# A simple example script for the SystemEDGE agent extension group.
# This script implements a "Ping" SNMP MIB variable whereby
# Sets to this variable indicate what system should be pinged
# and Gets to this variable cause the Ping to occur and the
# output to be returned as an OctetString.
#
# For reference, the following are passed as parameters by the SystemEDGE
# Agent. They are not used in this script.
#
# The Extension Group Leaf Number
LeafNumber=$1
# The SNMP Operation: GET, GETNEXT, or SET
Operation=$2
# The value if a SET Operation
SetValue=$3

# determine which ping program to use
OSNAME=`/bin/uname -r`
OSTYPE=`/bin/uname -s`

if [ "$LeafNumber" = "31" ]; then

   # setup who we should be pinging
   if [ "$Operation" = "SET" ]; then
      echo $SetValue > /tmp/sysedge.ping.target
      echo $SetValue
   fi

   if [ "$Operation" = "GETNEXT" ];  then
      Operation="GET"
   fi

   if [ "$Operation" = "GET" ]; then
      if [ ! -f /tmp/sysedge.ping.target ]; then
         echo "Ping target must be Set first"
         exit 0
      fi
```

```
        case $OSTYPE in
          Linux)
                  /bin/ping -c 1 -q -n `/bin/cat /tmp/sysedge.ping.target`
                  ;;

          HP-UX)
                  /etc/ping `/bin/cat /tmp/sysedge.ping.target` -n 1
                  ;;
          AIX)
                  /etc/ping -c 1 `/bin/cat /tmp/sysedge.ping.target`
                  ;;
          SunOS)
                  /usr/sbin/ping `/bin/cat /tmp/sysedge.ping.target` 2
                  ;;

          *)
                case $OSNAME in
                  6.*)
                    /usr/etc/ping -c 1 -q -n `/bin/cat /tmp/sysedge.ping.target`

                    # IRIX ping is very annoying since it doesnt appear you can get
                    # one-line output indicating if the machine is alive or not alive.
                    # If we were trickier shell programmers, we'd catch the output
                    # and only echo the last line.
                    ;;

                  5.*)
                    /usr/sbin/ping `/bin/cat /tmp/sysedge.ping.target` 2
                    ;;

                  4.*)
                    /usr/etc/ping `/bin/cat /tmp/sysedge.ping.target` 2
                    ;;

                  *)
                    echo "Cant determine ping command for system " $OSNAME
                    exit 0
                    ;;
                esac
          esac
      fi
fi
```

Ping script for Windows

```
@ECHO OFF
@rem ping.bat example extension script
@rem
@rem A simple example script for the SystemEDGE agent extension group.
@rem This script implements a "Ping" SNMP MIB variable whereby
@rem Sets to this variable indicate what system should be pinged
@rem and Gets to this variable cause the Ping to occur and the
@rem output to be returned as an OctetString.
@rem
@rem For reference, the following are passed as parameters by the SystemEDGE
@rem Agent.
@rem

@rem The Extension Group Leaf Number
set LeafNumber=%1%

@rem The SNMP Operation: GET, GETNEXT, or SET
set Operation=%2%

@rem The value if a SET Operation
set SetValue=%3%

@rem Which ping command should we use
set PING=%SystemRoot%\system32\ping.exe

@rem We store our target in the "environment" for future use
@rem using variable SYSEDGE_PING_TARGET

if not %LeafNumber% == 31 goto fi

   @rem if set, we want to ensure there is a value,
   @rem then store it away, then echo the value to STDOUT
   @rem so as to not confuse extension code
   if %Operation% == SET if !%SetValue%==! goto error
   if %Operation% == SET @echo set SYSEDGE_PING_TARGET=%SetValue% > %TMP%\pingtgt.bat
   if %Operation% == SET echo %SetValue%
   if %Operation% == SET goto fi
```

```
   @rem Run the script to recover the target name
   call %TMP%\pingtgt.bat
   if !%SYSEDGE_PING_TARGET% ==! goto error

   @rem echo pinging %SYSEDGE_PING_TARGET%
   %PING% -n 1 -w 250 %SYSEDGE_PING_TARGET% > nul

   if ERRORLEVEL 1 echo %SYSEDGE_PING_TARGET% not alive
   if not ERRORLEVEL 1 echo %SYSEDGE_PING_TARGET% is alive

:fi
goto exit

:error
echo Ping target must be set first

:exit
rem cleanup because command.com is really pathetic, sigh
set PING=
set Operation=
set LeafNumber=
set SetValue=
```

Monitor Log Files and Send Notification Through Email

This example monitors Windows Hotfix log files for fatal string occurrences. If the fatal string occurs, SystemEDGE sends a notification through email. The example consists of an entry for the sysedge.cf file and an associated Send_Notification_Mail.cmd script.

Note: The #no_actions directive must not be active. Prepend it with the comment (#) character.

```
watch logfile 44 0x8 ‘%windir%\KB*.log 'fatal' 'check KB for fatal occurrence in MS
hotfix log' "%CASYSEDGE%\My_Scripts\Send_Notification_Mail.cmd" 1 warning
```

Script Send_Notification_Mail.cmd

```
rem *** Get arguments in which we are interested ***
set regexpr=%3
set lasttime=%5
set description=%6

rem  *** Shift to get argument %10 as %9 ***
shift
set current_file=%9

rem *** Call email ***
“c:\Program Files\CA\SystemEDGE\bin\email.exe” -r smtp.company.com sysedge@%
COMPUTERNAME% you@company.com 'KB log fatal found' ’File monitor %description%
activated. %regexpr% pattern found in %current_file% at time %lasttime%’
```

Monitor Processes

Monitor a process including the arguments (note the flag value) and create a managed Object: Class=notepad, Instance=bad, Attribute=alive, state=*severity*

```
watch process procAlive 'notepad.*bad.*' 122 0x800 30 absolute nop 0 'Description:
procAlive(notepad)' '''notepad' ‘bad’ 'alive' 'minor'
```

Create a procgroup to monitor properties for a group of processes. Attributes (like RSS=Real memory Resident Set) of that proggroup can then be monitored like any other oid with a monitor directive.

```
watch procgroup 'svchost' 222 0x0 30 'Descr: procgroup(svchost)' ''
monitor processGroupMonEntry svchost pgmonRSS 20 0x0 60 absolute > 15000 'Descr:
procgroup svchost' '' ProcHealth svchost pgmonRSS critical
```

Create Monitored Objects

Check if a specific hotfix is installed and if so, create a monitored object.

```
monitor hrSWInstalledEntry 'Microsoft Hotfix Patch KB958644' hrSWInstalledIndex 41
0x100 30 absolute = 0 'Descr: Hot' '' 'SysHealth' 'SWInstalled' 'Microsoft Hotfix Patch
KB958644' 'ok'
```

Create a MIB attribute which holds the value of the performance counter 'Datagrams Sent/sec' from PerfObject 'UDPv4'.

Note: Check the available Performance Objects and Counters with the Microsoft command perfmon.

```
ntregperf 2 counter performance 'UDPv4' 'Datagrams Sent/sec' ''
```

Chapter 20: Troubleshooting and Usage Suggestions

This chapter presents helpful tips for using the SystemEDGE agent.

On Windows, the Windows Control Panel is the central location to

- View Agent basics: System Information and SNMP communities
- View and modify the Agent Configuration File
- View the Agent Diagnostic File
- Start and stop the Agent

This section contains the following topics:

Using diagsysedge.exe (see page 427)
Common Problems and Questions (see page 429)

Using diagsysedge.exe

You can use the diagsysedge.exe program to verify that the agent is running and to obtain information about the agent that you can use for troubleshooting

The examples in this section assume that the agent is configured to use SNMPv1 or SNMPv2c, has a read community of public, and is on port 161 or 1691.

Note: For more information about all of the options supported by diagsysedge.exe, see the chapter "Command Line Utilities."

Determine Whether the Agent Is Running with diagsysedge

You can run diagsysedge.exe with the -B option to determine whether the agent is running.

To determine whether the agent is running

1. Open a command prompt and change to the bin directory of SystemEDGE.

2. Enter the following:

   UNIX:

   ```
   ./diagsysedge -B
   ```

   Note: Instead of diagsysedge you can also use the symbolic link diagsysedge.exe on UNIX to run the utility.

   Windows:

   ```
   diagsysedge.exe -B
   ```

   The command provides output similar to the following:

   ```
   #./diagsysedge.exe -B
   diagsysedge: (V1.03 - LINUX)
   egyptian
   2.6.9-11.ELsmp, Red Hat Enterprise Linux AS release 4 (Nahant Update 1)#1 SMP Fri
   May 20 18:26:27 EDT 2005
   ```

   If the agent is not running, you receive an error message that the SNMP operation failed. Check the port number, and if the agent is started on a port other than 161, specify -p *port* on the command line and try again, or you can try to start the agent again, using the instructions in the chapter “Starting the Agent."

Obtain a Report for Troubleshooting

If you are encountering problems with SystemEDGE, Technical Support may ask you to run diagsysedge.exe to generate a report that they can use for troubleshooting.

To generate a report that you can send to Technical Support

1. Open a command prompt and change to the sysedge/bin directory.
2. Enter the following:

   UNIX:

   ```
   ./diagsysedge
   ```

   Note: Instead of diagsysedge you can also use the symbolic link diagsysedge.exe on UNIX to run the utility.

   Windows:

   Use Windows Control Panel to create diagsysedge output.

   If the agent is not running, you receive an error message that the SNMP operation failed. Attempt to start the agent again, using the instructions in this chapter for your operating system.

Validate AIM Plugins

You can run diagsysedge.exe with the -Q option to determine whether the AIMs are running.

To determine whether the AIMs are running

1. Open a command prompt and change to the bin directory of SystemEDGE.
2. Enter the following:

   UNIX/Linux:

   ```
   ./diagsysedge -Q
   ```

   Note: Instead of diagsysedge you can also use the symbolic link diagsysedge.exe on UNIX to run the utility.

   Windows:

   ```
   diagsysedge.exe -Q
   ```

   The command provides output similar to the following:

   ```
   C:\Program Files\CA\SystemEDGE\bin>diagsysedge -Q
   --- diagsysedge plugin query result ---
   monwin : RESULT: ok BRANCH: 1.3.6.1.4.1.546.16.20 DESCR: CA SystemEDGE AIM for
   Monitor Timing Windows, VERSION:5.0.0 BUILD:10092 REQUIRES:5.0.0
   perfcube : RESULT: ok BRANCH: 1.3.6.1.4.1.546.16.21 DESCR: CA SystemEDGE AIM for
   Performance Cubes, VERSION:5.0.0 BUILD:10092 REQUIRES:5.0.0
   cavmvcaim : RESULT: ok BRANCH: 1.3.6.1.4.1.546.16.52 DESCR: CA VMware VC Server
   AIM 2.0 PL0
   rmonwbem : RESULT: ok BRANCH: 1.3.6.1.4.1.546.16.22 DESCR: SysEDGE RMONWBEM AIM,
   VERSION:1.0.0 BUILD:10102 REQUIRES:5.0.0

   C:\Program Files\CA\SystemEDGE\bin>
   ```

   If the agent is not running, you receive an error message that the SNMP operation failed. Check the port number, and if the agent is started on a port other than 161, specify -p *port* on the command line and try again, or you can try to start the agent again, using the instructions in the chapter “Starting the Agent."

Common Problems and Questions

This section describes problems that might occur when you are installing and using SystemEDGE. It also provides instructions for resolving these problems. This section is organized by observable symptoms of a problem and verification steps that you can perform to isolate the cause.

Diagnostic Logging Information for Troubleshooting

If you encounter problems with SystemEDGE, you need to adjust diagnostic logging settings by editing the sysedge.cf file. The procedure to edit the sysedge.cf file is described in chapter “Configure Diagnostic Logging Mechanism”.

Typically, the amount of information logged increases and the log file maximum size and log file number must be adjusted.

Example:

```
sysedge_logfile sysedge.log 20000 2
sysedge_loglevel debug
```

SystemEDGE is instructed to use a set of two additional log files with a maximum size of 20000 KB each.

Agent Not Responding to SNMP Requests

When the agent is not responding to SNMP requests. To resolve this problem, you must do the following to isolate the cause:

- Verify that the agent is running (see page 431)
- Verify that the agent is starting at system initialization (see page 431)
- Verify that the agent is responding to queries (see page 432)
- Verify that the agent binaries and configuration files are installed
- Verify that the agent is co-existing correctly with other agents
- Verify the agent configuration (see page 433)
- Verify the management software configuration (see page 433)

Verify that the Agent is Running

You can verify that the agent is running in one of the following ways, depending on your platform.

To verify that the agent is running on UNIX

1. Enter netstat -a and look for UDP/161, UDP/1691, or SNMP.
2. Enter ps -aux or ps -aef and look for sysedge.
3. Run walktree or snmpget.
4. Examine sysedge.log file for agent error messages.

To verify that the agent is running on Windows

1. Select Start, Control Panel, CA SystemEDGE and view the status.
2. Open the Windows Task Manager and select the Processes tab. Look for the process named sysedge.exe. If this process appears in the list, CA SystemEDGE is running.
3. Select Start, Control Panel, Administrative Tools, Services, and look for CA SystemEDGE in the Services dialog.
4. Run walktree or snmpget.
5. Examine sysedge.log file for agent error messages.

Verify Agent Startup at System Initialization

To verify that the agent is started at system initialization, do one of the following depending on your platform:

- Check the UNIX startup script.
- Verify that the Windows SystemEDGE service is configured for automatic startup.

Verify Agent Response to Queries

Use the sysvariable utility to query a system and prove that the agent is responding to queries. You must specify the port number (unless you are using the default port of 161) for the agent and the community string for your system if the agent is configured to accept SNMPv1 or SNMPv2c communication. Otherwise, you must specify all necessary SNMPv3 communication (depending on the security level - Authpasswd, Authprotocol, Privpasswd, Privprotocol).

To display the operating system release number for a UNIX system where the agent is on port 1691 and the community string is public, enter the following for SNMPv1:

```
#./sysvariable -h 127.0.0.1 -p 1691 -c public -v 1 -o systype
```

To display the operating system release number for a UNIX system where the agent is on port 1691 and the community string is public, enter the following for SNMPv2c:

```
#./sysvariable -h 127.0.0.1 -p 1691 -c public  -v 2c -o systype
```

To display the operating system release number for a UNIX system where the agent is on port 1691 and the SecLevel is configured to AuthPriv(3) with secuser as the USM user name, authpasswd as the Authentication password, MD5 as the Authentication protocol, privpasswd as the Privacy password and DES as the Privacy protocol, enter the following for SNMPv3:

```
#./sysvariable -h 127.0.0.1 -p 1691 -v 3 -u secuser -s 3 -a authpasswd -A MD5 -x
privpasswd -X DES -o systype
```

For a Windows system where the agent is on port 1691 and the community string is public, enter the following for SNMPv1:

```
C:\sysedge\bin>sysvariable.exe -h 127.0.0.1 -p 1691 -c public -v 1 -o systype
```

For a Windows system where the agent is on port 1691 and the community string is public, enter the following for SNMPv2c:

```
C:\sysedge\bin>sysvariable.exe  -h 127.0.0.1 -p 1691 -c public  -v 2c -o systype
```

For a Windows system where the agent is on port 1691 and the SecLevel is configured to AuthPriv(3) with secuser as the USM user name, authpasswd as the Authentication password, MD5 as the Authentication protocol, privpasswd as the Privacy password and DES as the Privacy protocol, enter the following for SNMPv3:

```
C:\sysedge\bin>sysvariable.exe -h 127.0.0.1 -p 1691 -v 3 -u secuser -s 3 -a authpasswd
-A MD5 -x privpasswd -X DES -o systype
```

Verify Agent Configuration

To verify that the agent is configured correctly, check the access control lists. For SMPv2 and SNMPv2c, check the communities in sysedge.cf. For SNMPv3, check the USM security configuration in sysedgeV3.cf.

Performs these verifications as follows for each platform:

- On UNIX systems, SYSEDGE_DATADIR/sysedge.cf for community strings and SYSEDGE_DATADIR/sysedgeV3.cf for USM security settings.
- On Windows systems, verify SNMP v1 and SNMPv2c communities through the CA SystemEDGE Control Panel. Verify USM security information in SYSEDGE_DATADIR\sysedgeV3.cf.

Note: sysedgeV3.cf may be encrypted.

For more information, see se_enc Utility--Encrypt the SNMPv3 Configuration File in the chapter "Command Line Utilities."

Verify Management System Configuration

To ensure that the management system software is configured correctly, verify that it queries the correct system and port number if SystemEDGE is running on an alternate UDP port (for example, on UDP/1691).

Management System Not Receiving SNMP Trap Messages

If the management system is not receiving SNMP traps from SystemEDGE, you must verify that the agent is sending Trap PDUs and that it is sending them to the correct addresses. If you have already verified those conditions, the problem is most likely with the configuration of the management system. Perform the following verifications to isolate the root of the problem:

- Verify that the agent is running (see page 431)
- Verify that the agent is configured to send trap messages (see page 434)
- Verify that the agent is sending traps (see page 434)
- Verify management system trap configuration (see page 434)

Verify Agent Trap Configuration

To verify that the agent is configured to send trap messages to the appropriate addresses, do one of the following:

- For SNMPv1, verify the trap communities in the sysedge.cf file.
- For SNMPv3, verify the USM security configuration in the sysedgeV3.cf file.
- On UNIX systems, verify the trap communities in the sysedge.cf file. For example, make sure that only one IP address is specified for each trap community statement. For more information about trap community configuration, see the chapter “Agent Configuration.”
- On Windows systems, verify the trap configuration (SNMPv1 only) through the Control Panel application. For more information, see Configure Access Communities in the chapter “Agent Configuration.”

Verify that the Agent is Sending Traps

To verify that the agent is sending traps, do the following:

- On UNIX systems, examine CASYSEDGE_DATADIR/sysedge.log.
- On Windows systems, examineCASYSEDGE_DATADIR\sysedge.log

Verify Management System Trap Configuration

To verify the proper configuration of the management system, make sure that the empire.asn1 file has been imported into the management station (so that the management system knows the format of SNMP Trap PDUs generated by the SystemEDGE agent). For more information about troubleshooting the management system software, contact your management system vendor.

Bind Failed: Address Already In Use

If a bind fails because the address is in use, the agent displays a message similar to the following when it first starts:

```
sysedge: bind call failed: Address already in use
sysedge: another agent is probably running on port X
```

This message most often occurs when another SNMP agent is already running and bound to port UDP/161, which is the default SNMP agent port.

For more information about using the SystemEDGE agent with other SNMP agents, see the appendix “Co-existence with Other SNMP Agents.” For more information about using the agent on an alternative UDP port, see the chapter “Starting the Agent.”

CA SystemEDGE: xtrapmon.exe Utility Replaced by edgetrapmon.exe

The edgetrapmon utility replaces xtrapmon on Windows, Linux, and UNIX. The xtrapmon utility is no longer available. See also *SystemEDGE Release Notes*.

Appendix A: FIPS 140-2 Encryption

This appendix describes how to install and enable FIPS mode for SystemEDGE.

This section contains the following topics:

FIPS 140-2 Mode (see page 437)
FIPS Library Installation (see page 437)
Platform Support (see page 438)
Supported Encryption Protocols (see page 438)
Supported Authentication Protocols (see page 439)
Configure FIPS 140-2 Mode (see page 439)
FIPS Mode Considerations (see page 440)
Key Protection (see page 440)

FIPS 140-2 Mode

US Federal regulations require that all new software product sales to the US Federal government use Federal Information Processing Standard (FIPS 140-2) validated encryption algorithms if that product contains encryption.

You can operate SystemEDGE in FIPS mode using a version of the cryptographic library that has been certified according to the rules of the FIPS 140-2 standard.

FIPS Library Installation

To use FIPS mode of operation, you must install the SystemEDGE Advanced Encryption package in addition to the SystemEDGE Core package. The SystemEDGE Advanced Encryption package installs FIPS libraries along with libraries providing more advanced encryption than what the core agent package provides.

The following FIPS-certified files are installed by the SystemEDGE Advanced Encryption package:

Windows

- cryptocme2.dll [FIPS library]
- cryptocme2.sig [FIPS library signature file]

UNIX

- libcryptocme2.so [FIPS library for UNIX except for HP-UX PA-RISC]
- libcryptocme2.sl [FIPS library for HP-UX PA-RISC]
- libcryptocme2.sig [FIPS library signature file]

For more information about how to install SystemEDGE Advanced Encryption, see the appendix "SystemEDGE Advanced Encryption."

Platform Support

RSA B-safe Crypto FIPS-compliant libraries are currently only available for the following supported agent platforms:

- Windows 32-bit
- Windows 64-bit non-Itanium
- Redhat 32 and 64-bit non-Itanium
- SuSE Linux 32 and 64-bit non-Itanium
- Solaris SPARC
- Solaris 32 and 64-bit non-Intel

  Note: Solaris Intel architecture platforms do not support FIPS 140-2.
- HP-UX IA-64
- HP-UX 11.31 PA-Risc 64-bit

  Note: FIPS 140-2 is not supported on earlier versions of HP-UX PA-Risc.
- AIX 32 and 64-bit

Supported Encryption Protocols

The following encryption protocols are supported by SystemEDGE in the FIPS mode of operation. Anything not listed below is not supported.

- Triple Data Encryption Standard (3DES)
- Advanced Encryption Standard (AES) using cryptographic keys of 128 or 256 bits

Supported Authentication Protocols

Only the following authentication protocol is supported by SystemEDGE in the FIPS-only mode of operation:

- Secure Hash Algorithm-1 (SHA)

Configure FIPS 140-2 Mode

You can configure how the agent should treat its encryption using the sysedge_fips_mode parameter in the sysedge.cf file.

The options for the sysedge_fips_mode parameter are as follows:

0 (zero)

Indicates non-FIPS mode. The agent enables the CA eTrust Public Key Infrastructure libraries, and if this method fails, it uses basic built-in encryption and authentication protocols that are based on the non-certified FIPS code.

1

Indicates FIPS co-existence mode. The agent tries to use FIPS-certified encryption and authentication protocols using FIPS-certified libraries. However, if FIPS-certified libraries could not be located or if the agent fails to load these libraries, the agent's functionality falls back to using the non-FIPS compliant CA eTrust Public Key Infrastructure libraries. This is the default mode if sysedge_fips_mode is not configured.

2

Indicates FIPS only mode. Only FIPS-certified encryption and authentication code and protocols are supported, and all of the non-certified FIPS code and protocols are disallowed. No encryption is performed if the FIPS libraries fail.

Example

Add the following line to the sysedge.cf file to run the agent in FIPS only mode:

```
sysedge_fips_mode 2
```

FIPS Mode Considerations

Note the following considerations when using FIPS mode:

- When the agent is enabled in FIPS mode (sysedge_fips_mode 1 or sysedge_fips_mode 2) on a platform that does not support FIPS mode, its behavior is the same as a platform that supports FIPS mode.

  For example, if you enable FIPS only mode (sysedge_fips_mode 2), and if the authorization protocol MD5 is used on a platform that does not have support for FIPS mode, access using MD5 fails (which is the same behavior as if FIPS only mode was supported on the platform).

Key Protection

SystemEDGE provides SNMPv3 user configuration using a configuration file, sysedgeV3.cf. This file can be encrypted to protect keys and passwords. For more information, see the chapter "Agent Configuration."

Appendix B: SystemEDGE Advanced Encryption

This chapter describes how to install the SystemEDGE Advanced Encryption package for providing additional encryption capability and FIPS 140-2 mode of operation for SystemEDGE.

This section contains the following topics:

SystemEDGE Advanced Encryption (see page 441)
Supported Platforms (see page 441)
Supported Encryption Protocols (see page 441)
Supported Authentication Protocols (see page 442)
FIPS Compatibility (see page 442)
Installation Prerequisites (see page 442)
Install SystemEDGE Advanced Encryption (see page 442)
Installed Files (see page 445)

SystemEDGE Advanced Encryption

SystemEDGE Advanced Encryption provides additional encryption to what the SystemEDGE Core package provides. This package contains CA eTrust Public Key Infrastructure (ETPKI) libraries that provide additional encryption capability and include RSA BSAFE Crypto libraries for FIPS mode of operation.

Supported Platforms

SystemEDGE Advanced Encryption is supported and installable on any platform that SystemEDGE supports except for Solaris 8 and 9 32-bit operating systems..

Supported Encryption Protocols

The following encryption protocols are supported by SystemEDGE with the SystemEDGE Advanced Encryption package installed:

- Data Encryption Standard (DES)
- Triple Data Encryption Standard (3DES)
- Advanced Encryption Standard (AES) using cryptographic keys of 128 and 256 bits

Supported Authentication Protocols

The following encryption protocols are supported by SystemEDGE with the SystemEDGE Advanced Encryption package installed:

- Message-Digest algorithm 5 (MD5)
- Secure Hash Algorithm-1 (SHA)

FIPS Compatibility

The SystemEDGE Advanced Encryption package installs RSA B-safe Crypto FIPS-compliant libraries automatically. For detailed information about platforms supported by FIPS mode and any specific considerations, see the appendix “FIPS 140-2 Encryption."

Installation Prerequisites

Having SystemEDGE installed is a prerequisite for installing SystemEDGE Advanced Encryption.

Install SystemEDGE Advanced Encryption

The following section describes how to install and uninstall SystemEDGE Advanced Encryption on Windows and UNIX.

Install SystemEDGE Advanced Encryption on Windows

SystemEDGE Advanced Encryption packages are available for all supported Windows hardware architectures. The installation automatically installs the appropriate version for your system architecture.

To install SystemEDGE Advanced Encryption on Windows

1. Log in to the Windows system as an Administrator, Open the DVD1\Installers\Windows\Agent\SysMan\CA_SystemEDGE_AdvancedEncryption folder, and double-click ca-setup.exe.

   The SystemEDGE Advanced Encryption installation wizard appears.

2. Click Next.

   The License Agreement page appears.

3. Select I accept the terms of the License Agreement and click Next.

   The Review Settings page appears.

4. Click Install.

   The Installation Completed page appears after the installation finishes.

To verify the installation, check for the presence of the encryption libraries in the SystemEDGE installation directory. For more information, see Installed Files.

Uninstall SystemEDGE Advanced Encryption on Windows

Verify that you do not have packages installed which depend on SystemEDGE Advanced Encryption. Perform the following steps:

To uninstall SystemEDGE Advanced Encryption on Windows

1. Open the Add or Remove Programs dialog, select SystemEDGE Advanced Encryption, and click Remove.

   A confirmation dialog appears.

2. Click Yes.

   The package is uninstalled.

Install SystemEDGE Advanced Encryption on UNIX

SystemEDGE Advanced Encryption installation packages are available for all UNIX platforms and architectures that SystemEDGE supports. Install the appropriate package suitable to your system and architecture.

To install SystemEDGE Advanced Encryption on UNIX

1. Log in to the system as the root user, and make a local copy of the DVD2\Installers\<*platform*>\Agent\SysMan\CA_SystemEDGE_AdvancedEncryption folder, selecting the *platform* folder that corresponds to your operating system.
2. Run the installation from this directory as follows:

   ```
   ./ca-setup.sh CA_SETUP_JRE_DIR=""
   ```

   CA_SETUP_JRE_DIR

   Specifies the Java directory on the system, so that the installer can display the installation wizard dialogs. If Java is not available, the dialogs are displayed in the console in text mode.

   The Installation Completed page appears after the installation finishes.

To verify the installation, check for the presence of the encryption libraries in the SystemEDGE installation directory. For more information, see Installed Files.

Uninstall SystemEDGE Advanced Encryption on UNIX

Verify that you do not have packages installed which depend on SystemEDGE Advanced Encryption. Perform the following steps:

To uninstall Advanced Encryption on UNIX

1. Open a terminal console.
2. Navigate to the CA_SystemEDGE_AdvancedEncryption folder on the system and run the following command:

   ```
   ./ca-setup.sh -x
   ```

3. The agent uninstalls.

Installed Files

When you install SystemEDGE Advanced Encryption, the following files are installed on Windows systems:

- The following installed files provide authentication and encryption (non-FIPS mode):
  - libcapki.dll
  - libcapki_thread.dll
  - libcaopenssl_crypto.dll
  - libcaopenssl_ssl.dll
  - libcapki_ipthread.dll
- The following files are installed for FIPS mode of operation:
  - cryptocme2.dll
  - cryptocme2.sig
  - ccme_base.dll
  - ccmd_ecc.dll
  - ccme_eccaccel.dll

  For more information, see the appendix "FIPS 140-2 Encryption."

When you install SytemEDGE Advanced Encryption, the following files are installed on UNIX systems:

- The following installed files provide authentication and encryption (non-FIPS mode):
  - libcapki
  - libcapki_thread_posix
  - libcaopenssl_crypto
  - libcaopenssl_ssl
  - libetpki_openssl_ssl

  Note: The file extensions are .so, .sl, or .dylib, depending on the operating system.

- The following files are installed for FIPS mode of operation:
  - libcryptocme2
  - libcryptocme2.sig
  - libccme_base
  - libccme_ecc
  - libccme_eccaccel

  For more information, see the appendix "FIPS 140-2 Encryption."

  Note: The file extensions are .so or .sl (except for the .sig file), depending on the operating system.

Appendix C: Co-existence with Other SNMP Agents

This chapter describes how to use the SystemEDGE agent with other SNMP agents.

This section contains the following topics:

Multiple SNMP Agents Support (see page 447)
Multiple SystemEDGE Instance Support (see page 448)
Coexistence with the Microsoft Windows SNMP Agent (see page 448)
Coexistence with the HP SNMP Agent (see page 449)
Coexistence with the AIX SNMP Agent (see page 449)

Multiple SNMP Agents Support

By default, SNMP agents attempt to use UDP port 161, but only a single process can bind to a given port at any one time. Therefore, only one SNMP agent can use UDP port 161 at a time. If you are running multiple SNMP agents simultaneously, you must develop strategies for their coexistence.

The following are potential solutions for enabling several SNMP agents to share UDP port 161:

- Multiplex UDP port 161 among several SNMP agents, usually in a master/slave relationship.
- Run agents simultaneously if each binds to a separate UDP port and management software is configured to communicate with them individually.

  Note: UDP port 1691 is reserved for use by the SystemEDGE agent and is the default port for agent deployment from CA Virtual Assurance. You can configure the agent to use that port instead of UDP/161. For more information about starting the agent on an alternate port, see the chapter “Starting the Agent."
- Code agents to conform to every existing proprietary master/subagent API. This solution has an exceedingly high cost in terms of coding, testing, and licensing the numerous proprietary APIs.

The most common notion of agent multiplexing involves a master agent bound to UDP port 161 that communicates with separate subagents or slave agents that implement different MIBs. The master and subagents communicate using a defined protocol for registering the subagent and exchanging messages between them. Several protocols exist for governing such a communication protocol between master and subagent.

The SystemEDGE installation also provides you the opportunity to replace the native operating system agent with the SystemEDGE agent. If you do replace the agent, you have the option of automatically inheriting the native agent's settings. For more information about replacing the native SNMP agent during installation, see the chapter "Installation and Deployment."

Multiple SystemEDGE Instance Support

While it is not an officially supported feature, you can run multiple instances of the SystemEDGE agent from a command shell on a single server such as test configuration changes in a test instance. The command line options for starting the agent let you specify a listening port and configuration file locations. You can start multiple instances of the agent at one time by entering different ports on the command line when you start each instance. You can also edit and configure each agent to use different configuration files if necessary.

Important! Multiple SystemEDGE instances are not supported by the agent installation or by CA Virtual Assurance deployment and management. Only the instance running on the default port that you specified during installation is supported. If you need to uninstall the agent, only the default instance is removed. You must manually configure, implement, and maintain any additional instances on the same server.

For more information about running multiple agent instances on one server, see the chapter "Starting the Agent."

Coexistence with the Microsoft Windows SNMP Agent

The SystemEDGE agent is no longer a subagent of the Microsoft SNMP agent. It can coexist with the Microsoft Windows extensible agent.

Take either of the following coexistence approaches:

- Run the Microsoft and SystemEDGE agents together on the system, running the SystemEDGE agent on port 1691 and the Microsoft agent on port 161.
- Turn off or disable the Microsoft Master agent, and run the SystemEDGE agent on port 161. Replacing the native agent during installation implements this approach.

Coexistence with the HP SNMP Agent

HP-UX systems ship with an extensible agent and subagents that provide support for MIB-II and the HP private-enterprise MIB. The master/subagent interface is based on a proprietary API that the SystemEDGE agent does not support.

Take either of the following coexistence approaches:

- Run the HP and SystemEDGE agents together on the system, running the SystemEDGE agent on port UDP/1691.
- Turn off or disable the HP master or subagent and run the SystemEDGE agent on port UDP/161. Replacing the native agent during installation implements this approach.

Coexistence with the AIX SNMP Agent

AIX Release 4.1 and later systems ship with a MIB-II SNMP agent that also supports SMUX. You can run the SystemEDGE agent in addition to or in place of the AIX agent.

Take either of the following coexistence approaches:

- Run the SystemEDGE agent and AIX agents together. To do so, make sure that the SystemEDGE agent is invoked with the -p 1691 option in the AIX TCP/IP startup script, /etc/rc.tcpip.
- Run the SystemEDGE agent instead of the AIX agent. To do so, comment out the AIX agent's invocation in /etc/rc.tcpip by adding a pound sign (#) at the beginning of the following lines:

  ```
  # Start up the Simple Network Management Protocol (SNMP) daemon
  # start /usr/sbin/snmpd "$src_running"
  ```

  Replacing the native agent during installation also implements this approach.

Appendix D: Host Resources MIB

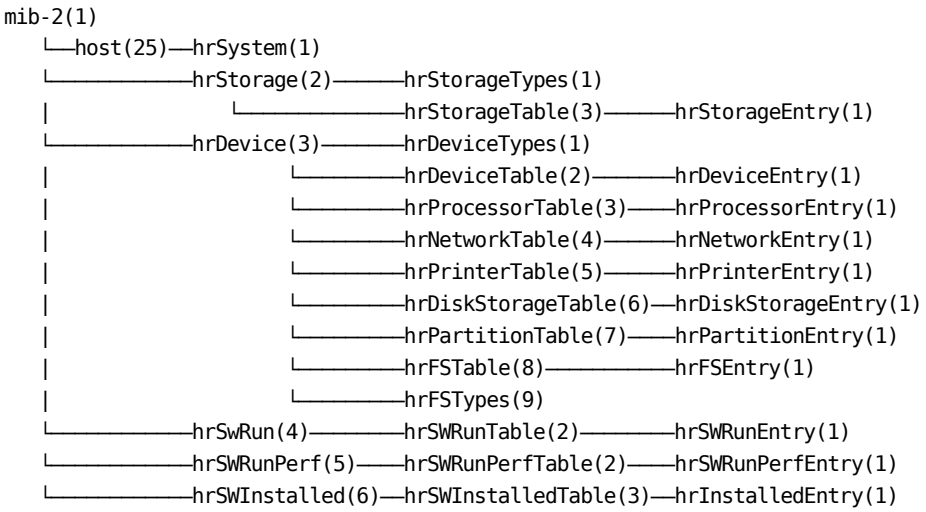
This chapter describes the management information available through the IETF Host Resources MIB (RFC 1514).

This section contains the following topics:

Host Resources MIB Overview (see page 451)
Host Resources System Group (see page 452)
Host Resources Storage Group (see page 452)
Host Resources Device Group (see page 453)
Host Resources Running Software Group (see page 456)
Host Resources Installed Software Group (see page 456)
Unsupported Host Resources MIB Objects (see page 457)

Host Resources MIB Overview

The Host Resources MIB defines a set of objects that can manage host computers, which are independent of the operating system, network services, or any software application. The objects defined in the Host Resources MIB are common across many computer system architectures. The following illustration shows the overall organization of the Host Resources MIB:



Note: The examples in this chapter do not describe the entire Host Resources MIB. For a description of the entire MIB, see the hostmib.asn1 file, which is included in the mib subdirectory of the agent installation.

Host Resources System Group

The Host Resources System (hrSystem) group provides information that pertains to the host system as a whole. The following illustration shows the organization of the hrSystem group:

```
mib-2(1)
    └─host(25)─┬─hrSystem(1)
               ├─────────hrSsystemUptime(1)
               ├─────────hrSystemDate(2)
               ├─────────hrSystemInstalLoadDevice(3)
               ├─────────hrSystemInstalLoadParameters(4)
               ├─────────hrSystemNumUsers(5)
               ├─────────hrSystemProcesses(6)
               └─────────hrSystemMaxProcesses(7)
```

The following information is available through the Host Resources System group:

- System uptime and date
- Operating system loading device, path name, and parameters
- Number of user sessions for which the host is storing state information
- Number of processes currently loaded or running
- Maximum number of processes that can be running

Host Resources Storage Group

The Host Resources Storage group (hrStorageTable) lists the logical areas of storage allocated on the host system. These storage areas include file systems and disk partitions that might be seen by an application, instead of physical storage such as tapes and floppy drives.

This table is intended to be a useful diagnostic for out of memory and out of buffers types of failures. It can also be a useful performance monitoring tool for tracking memory, disk, or buffer usage. The following illustration shows sample output of the Host Resources Storage group:

```
Table: hrStorageTable   Path: 1.3.6.1.2.1.25.2.3.1 Rows:5 Columns:7

Type                  Descr                    Units  Size       Used     Failures

hrStorageRam           Physical RAM              1      24801280   4567030  0
hrStorageVirtualMemory  Virtual Memory           1      103849984  0        0
hrStorageFixedDisk      Disk partition /dev/sd0a  1024   15487      11080    0
hrStorageFixedDisk      Disk partition /dev/sd0g  1024   160423     130421   0
hrStorageFixedDisk      Disk partition /dev/sd0h  1024   123911     29154    0
```

Host Resources Device Group

The Host Resources Device (hrDevice) group contains several tables that provide information about the devices on the host system. The main table in this group is the hrDeviceTable. It lists all of the devices that the host contains. The hrDevice group also includes a number of device-specific tables that provide more detailed information for particular device types.

For example, the group includes device-specific tables for the following:

- Processors
- Networks
- Printers
- Disk storage
- Partitions
- File systems

If the hrDeviceTable shows that the host contains a Printer device, for example, you can retrieve more detailed information about the printer from the hrPrinterTable. The following sections describe the hrDeviceTable and the device-specific tables.

Device Table

The hrDeviceTable lists the devices on the host system. The table lists the following for each device:

- Device type
- Description of the device
- Status (for example, running or down)
- Number of errors detected for the device

Inventory Tracking and Asset Management

The hrDeviceTable is especially useful for inventory tracking and asset management. Instead of manually opening and inspecting each workstation on your network to determine the number and types of ethernet cards, serial devices, audio devices, disk storage devices, and so on, you can use your central management system to retrieve the hrDeviceTable from each of the systems on your network.

Processor Table

The hrProcessorTable lists device-specific information about the system's processors. If the system contains only a single processor, the hrProcessorTable includes only one row. The columns of the hrProcessorTable provide the product ID of the firmware associated with the processor (if such an ID has been assigned and made available by the processor vendor) and the processor load. The load is the average over the last minute of the percentage of time that the processor was not idle.

Processor Load Tracking

By monitoring (graphing) the Processor Load average, you can visually track the load put on the processor. For example, if the graph shows constant high levels (indicating that the processor was heavily loaded), you may decide that the host system is being overworked with too many users and processes, requiring you to limit the number of users or have some of the processes run on another system.

A temporary spike in a Processor Load graph could be an indication that a processor-intensive process is running. To avoid problems that might result from such a heavy processor load, you can reschedule that process to run at a time when processor load is less heavy.

Disk Storage Table

The hrDiskStorageTable provides information about the host's disk storage devices. The table lists the following for each disk storage device:

- Instance
- Whether the disk storage device permits read-write or read-only access
- Storage media type (for example, hard disk or floppy disk)
- Whether the disk is removable
- Storage capacity (in KB)

Partition Table

The hrPartitionTable shows the partitions that have been configured for the disk storage devices. The table lists the following for each partition:

- Instance
- Partition Index
- Partition Label
- Partition ID
- Partition Size (in KB)
- Index of the file system mounted on that partition

File System Table

The hrFSTable provides information about the host's file systems, both local and remote. The table lists the following for each file system:

- Local mount point
- Remote mount point (if the file system is being mounted from a remote server)
- Type of file system (for example, BerkeleyFFS)
- Whether the file system permits read-write or read-only access

The following illustration shows sample output of the hrFSTable:

```
Table: hrFSTable   Path: 1.3.6.1.2.1.25.3.8.1 Rows:5 Columns:5

Mount Point         Remote Mount Point    Type       Access        Index

/                                         hrFSBtrfs  readWrite(1)  3
/usr                                      hrFSBtrfs  readWrite(1)  4
/net                                      hrFSOther  readOnly(2)   0
/tmp_mnt/net/ht1    cantor:/ht1           hrFSNFS    readWrite(1)  0
/tmp_mnt/net/data   hamilton:/export/data hrFSNFS    readWrite(1)  0
```

Host Resources Running Software Group

The hrSWRun table lists the software currently running on the host system. The table lists the following for each running software process:

- Unique identification number
- Name
- Product ID
- Directory path where the software resides
- Run-time parameters with which the software was started
- Type of software (for example, operating system or application)
- Status of the software (for example, running, runnable, not runnable, or invalid)

Host Resources Installed Software Group

The hrSWInstalled table lists the software currently installed on the host system. The table lists the following for each software package:

- Unique identifier
- Software package name
- Description
- Installation date

This table shows the operating system patches and versions of software installed and is ideal for checking software version consistency between systems. The following illustration shows part of a sample hrSWInstalledTable:

```
Table: hrSWInstalledTable   Path: 1.3.6.1.2.1.25.6.3.1 Rows:6  Columns:3

Installed Name                                          ID      Type

/kernel/unix: SunOS: 5.10 Generic_Patch                 null    operatingSystem(2)
SUNWkvm:Core Architecture, (Kvm):11.5.0,REV=2.0.18      null    operatingSystem(2)
SUNWcsu:Core Sparc, (Usr):11.5.0,REV=2.0.18             null    operatingSystem(2)
SUNWcsr:Core Sparc, (Root):11.5.0,REV=2.0.18            null    operatingSystem(2)
SUNWcsd:Core Sparc, Devices:11.5.0,REV=2.0.18           null    operatingSystem(2)
SUNWcar:Core Architecture, (Root):11.5.0,REV=2.0.18     null    operatingSystem(2)
```

Note: Not all systems can support this table. To determine whether your operating system supports this table, see the *SystemEDGE Release Notes*.

Unsupported Host Resources MIB Objects

The SystemEDGE agent supports the Host Resources MIB (RFC 1514) on all supported operating systems, but some of the MIB objects within this MIB module are not supported by the agent or the underlying operating systems. Those objects, therefore, cannot be implemented by the version of the agent installed on the affected operating system.

For a list of unsupported Host Resources MIB objects on all operating systems, see the *SystemEDGE Release Notes*.

Appendix E: Private Enterprise Traps

In addition to supporting the standard MIB-II traps, the SystemEDGE agent supports a number of private-enterprise trap types defined for use with the agent's self-monitoring capabilities. This appendix describes the format of the Trap PDUs that the SystemEDGE agent can send.

This section contains the following topics:

Format of Trap PDUs (see page 459)
SNMPv1 Trap Format (see page 468)

Format of Trap PDUs

The Systems Management Empire MIB file, empire.asn1, defines enterprise-specific traps for use with the SystemEDGE agent. This section describes the PDU format, including the information in the variable-bindings fields, for each type of trap.

Variable bindings are optional fields in a Trap PDU that provide additional important information associated with the trap.

Note: The text in this chapter is taken from the Systems Management Empire MIB definition, which exists in the mib subdirectory of the agent installation.

All traps sent by the SystemEDGE agent contain the following enterprise system object identifier (sysObjectID): empire(546).1.1.

monitor Trap

The following shows the format of a monitor trap, which the SystemEDGE agent sends to indicate that a self monitor event has occurred. The agent sends this trap when a monitored attribute's threshold has been reached and the self monitor has a severity of none(1); otherwise it sends an aggregateState trap.

```
monitorEvent OBJECT IDENTIFIER ::= { traps 1 }

monitorTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { monDescr, monOID, monCurrVal, monValue, monRowStatus, monOperator,
monIndex, monFlags, monObjClass, monObjInstamce, monObjAttribute, monCurrState }

DESCRIPTION
"The threshold condition of a self monitor (see monitorTable) is breached."

::= 1
```

monitorEntryNotReady Trap

The following shows the format of a monitorEntryNotReady trap, which the SystemEDGE agent sends to indicate that the monRowStatus field of a Self Monitor table entry is set to notReady(3). The agent only sends this trap if a monitor has a severity of none(1).

```
monitorEntryNotReadyEvent OBJECT IDENTIFIER ::= { traps 3 }

monitorEntryNotReadyTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { monDescr, monOID, monCurrVal, monValue, monRowStatus, monOperator,
monIndex, monFlags, monObjClass, monObjInstance, monObjAttribute, monCurrState }

DESCRIPTION
"The row status of a self monitor (see monitorTable) has become notReady(3), typically
because the monitored OID does not exist."

::= 3
```

logMonMatch Trap

The following shows the format of a logMonMatch trap, which the SystemEDGE agent sends to indicate that the log file monitoring subsystem has detected a match in a log file that the agent is currently monitoring. The monitor severity is included in the variable binding.

```
logMonMatchEvent OBJECT IDENTIFIER ::= { traps 4 }

logMonMatchTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { logMonitorLogFile, logMonitorRegularExpression,
logMonitorLastTrap, logMonitorLastMatch,
logMonitorDescr, logMonitorIndex, logMonitorFlags,
logMonitorLogFileName, logMonitorSeverity }

DESCRIPTION
"The regular expression pattern of a log file monitor (see logMonitorTable) matched."

::= 4
```

logMonNotReady Trap

The following shows the format of a logMonNotReady trap, which the SystemEDGE agent sends to indicate that the status of an entry in the Log Monitor table has changed to notReady.

```
logMonNotReadyEvent OBJECT IDENTIFIER ::= { traps 5 }

logMonNotReadyTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { logMonitorLogFile, logMonitorRegularExpression,
logMonitorLastTrap, logMonitorLastMatch,
logMonitorDescr, logMonitorIndex, logMonitorFlags,
logMonitorLogFileName, logMonitorSeverity }

DESCRIPTION
"The row status of a log file monitor (see logMonitorTable) has become notReady(3),
typically because the log file does not exist."

::= 5
```

ntEventMonMatch Trap

The following shows the format of an ntEventMonMatch trap, which the SystemEDGE agent sends to indicate that the event log monitoring subsystem has detected a match in a Windows event log file that the agent is currently monitoring. The monitor severity is included in the variable binding.

```
ntEventMonMatchEvent OBJECT IDENTIFIER ::= { traps 7 }

ntEventMonMatchTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { ntEventMonLog, ntEventMonTypeLastMatch,
ntEventMonTime, ntEventMonSrcLastMatch,
ntEventMonDescLastMatch, ntEventMonDescr,
ntEventMonIndex, ntEventMonFlags, ntEventMonSeverity }

DESCRIPTION
"All filters (log, type, src, desc) of an NT event log monitor (see ntEventMonTable)
matched."

::= 7
```

ntEventMonNotReady Trap

The following shows the format of an ntEventMonNotReady trap, which the SystemEDGE agent sends to indicate that the status of a Windows event log monitoring entry has become notReady(3).

```
ntEventMonNotReadyEvent OBJECT IDENTIFIER ::= { traps 8 }

ntEventMonNotReadyTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { ntEventMonLog, ntEventMonTypeFilter,
ntEventMonSrcFilter, ntEventMonDescFilter,
ntEventMonDescr, ntEventMonIndex, ntEventMonFlags, ntEventMonSeverity }

DESCRIPTION
"The row status of an Windows event log monitor (see ntEventMonTable) has become
notReady(3), typically because an event log does not exist."

::= 8
```

monitorClear Trap

The following shows the format of a monitorClear trap, which the SystemEDGE agent sends to indicate that a threshold breach condition that previously existed no longer exists. The SystemEDGE agent sends this trap when a Self Monitor table entry that has a Send Clear Trap flag transitions from True to False. The agent only sends this trap if a monitor has a severity of none(1); otherwise it sends an aggregateState trap.

```
monitorClearEvent OBJECT IDENTIFIER ::= { traps 9 }

monitorClearTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { monDescr, monOID, monCurrVal, monValue, monRowStatus, monOperator,
monIndex, monFlags, monObjClass, monObjInstance, monObjAttribute, monCurrState }

DESCRIPTION
"The threshold condition of a self monitor (see monitorTable) is no more breached."

::= 9
```

processStop Trap

The following shows the format of a processStop trap, which the SystemEDGE agent sends to indicate that a process it was monitoring has either stopped running or is in a state where it cannot run. The agent only sends this trap if a process monitor has a severity of none(1); otherwise it sends an aggregateState trap.

```
processStopEvent OBJECT IDENTIFIER ::= { traps 10 }

processStopTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { pmonIndex, pmonDescr, pmonAttribute, pmonCurrVal, pmonOperator,
pmonValue, pmonFlags, pmonRegExpr, pmonCurrentPID, pmonObjClass, pmonObjInstance,
pmonObjAttribute, pmonCurrState, pmonProcName}

DESCRIPTION
"The process (or Windows service) monitored by a process monitor (see processMonTable)
using procAlive is not running."

::= 10
```

processStart Trap

The following shows the format of a processStart trap, which the SystemEDGE agent sends to indicate that a stopped process has restarted (and has been reacquired by the SystemEDGE agent). The agent only sends this trap if a process monitor has a severity of none(1); otherwise it sends an aggregateState trap.

```
processStartEvent OBJECT IDENTIFIER ::= { traps 11 }

processStartTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { pmonIndex, pmonDescr, pmonAttribute, pmonCurrVal, pmonOperator,
pmonValue, pmonFlags, pmonRegExpr, pmonCurrentPID, pmonObjClass, pmonObjInstance,
pmonObjAttribute, pmonCurrState, pmonProcName}

DESCRIPTION
"The process (or Windows service) monitored by a process monitor (see processMonTable)
using procAlive has been re-started."

::= 11
```

processThreshold Trap

The following shows the format of a processThreshold trap, which the SystemEDGE agent sends to indicate that a monitored attribute of a process has breached a specified threshold condition. The agent only sends this trap if a process monitor has a severity of none(1); otherwise it sends an aggregateState trap.

```
processThresholdEvent OBJECT IDENTIFIER ::= { traps 12 }

processThresholdTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { pmonIndex, pmonDescr, pmonAttribute, pmonCurrVal, pmonOperator,
pmonValue, pmonFlags, pmonRegExpr, pmonCurrentPID, pmonObjClass, pmonObjInstance,
pmonObjAttribute, pmonCurrState, pmonProcName}

DESCRIPTION
"The threshold condition of a process monitor (see processMonTable) is breached."

::= 12
```

processClear Trap

The following shows the format of a processClear trap, which the SystemEDGE agent sends to indicate that a process attribute for which it previously sent a processThreshold trap is no longer at the threshold level. The agent only sends this trap if a process monitor has a severity of none(1); otherwise it sends an aggregateState trap.

```
processClearEvent OBJECT IDENTIFIER ::= { traps 13 }

processClearTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { pmonIndex, pmonDescr, pmonAttribute, pmonCurrVal, pmonOperator,
pmonValue, pmonFlags, pmonRegExpr, pmonCurrentPID, pmonObjClass, pmonObjInstance,
pmonObjAttribute, pmonCurrState, pmonProcName}

DESCRIPTION
"The threshold condition of a process monitor (see processMonTable) is no more
breached."

::= 13
```

processNotReady Trap

The following shows the format of a processNotReady trap, which the SystemEDGE agent sends to indicate that the row status of a Process Monitor table entry is set to notReady(3). If the process is being monitored using procAlive, a processStop trap is sent instead. The agent only sends this trap if a process monitor has a severity of none(1).

```
processNotReadyEvent OBJECT IDENTIFIER ::= { traps 14 }

processNotReady TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { pmonIndex, pmonDescr, pmonAttribute, pmonCurrVal, pmonOperator,
pmonValue, pmonFlags, pmonRegExpr, pmonCurrentPID, pmonObjClass, pmonObjInstance,
pmonObjAttribute, pmonCurrState, pmonProcName}

DESCRIPTION
"The row status of a process monitor (see processMonTable) not using procAlive has
become notReady(3), typically because the process does not exist."

::= 14
```

addrChange Trap

The following shows the format of an addrChange trap, which the SystemEDGE agent sends to indicate that the underlying IP address has changed.

```
addrChangeEvent OBJECT IDENTIFIER ::= { traps 18 }

addrChangeTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { nodename, sysedgeAddressList }

DESCRIPTION
"The IP address of the agent's machine has changed, typically due to DCHP or other
administrative means.”

::= 18
```

procGroupChange Trap

The following shows the format of a procGroupChange trap, which the SystemEDGE agent sends to indicate that the process group has changed. The monitor severity is included in the variable binding.

```
procGroupChangeEvent OBJECT IDENTIFIER ::= { traps 19 }

procGroupChangeTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { pgmonIndex, pgmonDescr, pgmonFlags, pgmonNumProcs, pgmonProcRegExpr,
pgmonRowStatus, pgmonPIDList, pgmonStatusList, pgmonSeverity, pgmonProcNameList}

DESCRIPTION
"The membership of a process group monitor (see processGroupMonTable) has changed,
typically because processes have joined or left."

::= 19
```

aggregateState Trap

The following shows the format of an aggregateState trap, which the SystemEDGE agent sends to indicate that the state of an aggregate has changed. The agent only sends this trap when a threshold breach of a monitor changes the aggregate state. For example, if a monitor with a severity of critical is breached, but the aggregate current state is fatal, the agent does not send a trap because the aggregate state is not affected.

```
aggregateStateEvent OBJECT IDENTIFIER ::= {traps 20}

aggregateStateTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { aggIndex, aggObjClass, aggObjInstance, aggObjAttribute, aggCurrState,
aggPrevState, aggMonTable, aggMonIndex, aggMonCurrVal, aggMonOperator, aggMonValue
}

DESCRIPTION
"The state of an aggregate (see aggregateTable) has changed."

::= 20
```

aggregateAdd Trap

The following shows the format of an aggregateAdd trap, which the SystemEDGE agent sends to indicate that an aggregate has been added to the Aggregate table.

```
aggregateAddEvent OBJECT IDENTIFIER ::= {traps 21}

aggregateAddTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { aggIndex, aggObjClass, aggObjInstance, aggObjAttribute }

DESCRIPTION
"A new aggregate (see aggregateTable) has been added."


::= 21
```

aggregateDelete Trap

The following shows the format of an aggregateDelete trap, which the SystemEDGE agent sends to indicate that an aggregate has been deleted from the Aggregate table.

```
aggregateDeleteEvent OBJECT IDENTIFIER ::= {traps 22}

aggregateDeleteTrap TRAP-TYPE

ENTERPRISE sysmgmt

VARIABLES { aggIndex, aggObjClass, aggObjInstance, aggObjAttribute }

DESCRIPTION
"An aggregate (see aggregateTable) has been deleted."


::= 22
```

SNMPv1 Trap Format

The SystemEDGE agent sends Trap PDUs to the SNMPv1 Trap port (UDP/162). Following are the components of the SNMPv1 Trap PDU:

Enterprise

Specifies the System Object ID of the sender: empire(546).1.1.

Agent address

Specifies the IP address of the sending host.

Generic Trap Type

Specifies the generic trap type, which can be one of the following:

- coldStart(0)
- warmStart(1)
- linkDown(2)
- linkup(3)
- authenFailure(4)
- egpNeighborloss(5)
- enterpriseSpecific(6)

Specific Trap Type

Specifies the enterprise-specific trap type.

Time Stamp

Specifies the value of sysUptime when the trap was sent. The value is always 0 for sendtrap.

Variable Bindings

Specifies the array of MIB variables and their values.

Index

A

Access Lists for SNMPv1 Communities • 127
Additional Parameters • 392
addrChange Trap • 466
Address Filtering for SNMPv3 Users • 158
Agent Addresses of Traps from SystemEDGE • 130
Agent and AIM Upgrades • 113
Agent Configuration • 119
Agent Configuration During Installation • 78
Agent Deployment • 106
Agent Not Responding to SNMP Requests • 430
Agent Upgrade from SystemEDGE 4.3.4 • 114
Agent Warm Start • 169
Aggregate State Table • 191, 196, 241
aggregateAdd Trap • 467
aggregateDelete Trap • 468
aggregateState Trap • 467
AIM Architecture • 73
AIM Integration with CA Virtual Assurance • 74
AIX Logical Partition Group • 180
Application Insight Modules (AIMs) • 31
Architecture Overview • 39
Audience • 17
Automatic Agent Startup at System Boot • 168
Autowatcher Tables • 191
Autowatchers • 285

B

Bind Failed
    Address Already In Use • 434
Boot Configuration Group • 174
Buffers Group • 179

C

CA eHealth • 36
CA NSM • 37
CA Server Automation • 34
CA Spectrum • 35
CA SystemEDGE • 17
    xtrapmon.exe Utility Replaced by edgetrapmon.exe • 435
CA Systems Management Empire MIB • 26
CA Technologies Product References • 3
CA Virtual Assurance • 33
CA Virtual Assurance Security Options • 72
Change the nice Value of a Process (UNIX only) • 176
Co-existence with Other SNMP Agents • 447
Coexistence with the AIX SNMP Agent • 449
Coexistence with the HP SNMP Agent • 449
Coexistence with the Microsoft Windows SNMP Agent • 448
Command Line Startup Options • 167
Command Line Utilities • 413
Common Problems and Questions • 429
Concepts • 39
Configuration Basics • 119
Configuration Files • 43
Configuration Using CA Virtual Assurance • 121
Configure Access Communities • 126
Configure and Use a Response File • 104
Configure Authentication Failure Traps • 130
Configure Device Status Checking Restrictions • 138
Configure Diagnostic Logging Mechanism • 134
Configure Disk Performance Statistics Collection for AIX Systems • 188
Configure Federal Information Processing Standard Mode • 133
Configure FIPS 140-2 Mode • 439
Configure GetBulk Response Message Size • 132
Configure IP Family for SNMP User Datagram Protocol Communications • 123
Configure Linux Free Memory Calculation • 141
Configure Maintenance Mode • 136
Configure MIB Table Restrictions • 137
Configure Monitor Aggregation in a Multi-tier Hierarchy • 141
Configure SNMP Information • 123
Configure SNMP Set Notifications • 131
Configure SNMP Set Restrictions • 131
Configure SNMPv1 Trap Destinations • 128
Configure SNMPv2c and SNMPv3 Trap Destinations • 161
Configure SNMPv3 User Information • 156
Configure Support for Actions • 136
Configure System Information • 122
Configure Text Pattern Exclusion • 152
Configure the SNMP Bind Address • 124
Configure the SNMPv3 Engine ID • 156

Configure User and Group Permissions for Subprograms (UNIX Only) • 137
Configure Warm Start Discovery • 140
Configuring SystemEDGE Command Line Utilities • 414
Contact CA Technologies • 4
Conventions • 18
Core AIMs • 409
Corrective Actions • 30
CPU Statistics Table • 189
Create Monitor by Instance • 43
Create Monitored Objects • 425
Custom MIB Objects • 389

D

Delete IPC Mechanisms • 179
Deploy the Agent with CA Virtual Assurance • 107
Determine Whether the Agent Is Running with diagsysedge • 427
Device Table • 173, 453
Diagnostic Logging Information for Troubleshooting • 430
Directory Monitoring • 328
Directory Name Lookup Cache Group • 180
Disable SNMPv1 and SNMPv2c • 164
Disk Statistics Table • 188
Disk Storage Table • 454
Distributed Systems Group • 185
Dynamic Configuration During Operation • 383

E

edgemon Commands for Self Monitoring • 230
edgemon Examples • 233
edgewatch Autowatcher Examples • 296
edgewatch Commands for Autowatchers • 292
edgewatch Commands for Log File Monitoring • 345
edgewatch Commands for Process Group Monitoring • 323
edgewatch Commands for Process Monitoring • 275
edgewatch Commands for Windows Event Monitoring • 372
edgewatch Examples • 278, 324, 347, 373
edgewatch Utility--Monitor Log Files • 342
edgewatch Utility--Monitor Process Groups • 319
edgewatch Utility--Monitor Windows Events • 368
edgewatch Utility--Using Autowatchers • 288
emphistory Commands for Managing Entries in the History Control Table • 385
emphistory Directive--Add Entries to History Control Table • 384
emphistory Examples • 387
Enable the Monitor Maintenance Windows AIM • 410
Enable the Performance Cube AIM • 411
Encrypt the SNMPv3 Configuration File • 163
Event Engine • 72
Example Script Test • 395
Examples Using State Management • 415
Extension Examples • 392
Extension Group • 190
Extension Group Features • 390
extension Keyword--Add Entries to the Extension Group • 391
Extension Scripts • 394
Extension Variable Configuration • 390

F

Features and Integrations Overview • 21
Features Overview • 21
File System Space Monitoring • 174
File System Table • 455
File-Based Configuration • 121
FIPS 140-2 Encryption • 437
FIPS 140-2 Mode • 437
FIPS Compatibility • 442
FIPS Library Installation • 437
FIPS Mode Considerations • 440
Format of Trap PDUs • 459

G

Generic Autowatcher Examples • 300
Generic Autowatchers • 287
Group Table • 175
Guidelines for Using the SystemEDGE Agent • 74

H

History Collection • 60, 377
History Collection Example • 385
History Collection Overview • 377
History Control Table and History Data Table • 378
History Control Table Columns • 378
History Control Table Configuration • 383
History Sampling • 377
History Sampling Examples • 381
History Table • 186
History Table Columns • 380

Host Resources Device Group • 453
Host Resources Installed Software Group • 456
Host Resources MIB • 29, 451
Host Resources MIB Overview • 451
Host Resources Running Software Group • 456
Host Resources Storage Group • 452
Host Resources System Group • 452
How Autowatchers Work • 285
How History Collection Works • 60
How State Management and Aggregation Works • 66
How Stateful Monitoring Works • 47
How Stateless Monitoring Works • 48
How to Add a Separate MIB Specification for ntRegPerf Variables • 407
How to Edit a Separate MIB Specification for Extension Variables • 397
How to Edit empire.asn1 for Extension Variables • 397
How to Edit empire.asn1 for ntRegPerf Variables • 406

I

I/O Buffer Cache Group • 180
Import SystemEDGE Monitors into a Policy • 116
Initial Configuration During Startup • 383
Install SystemEDGE Advanced Encryption • 442
Install SystemEDGE Advanced Encryption on UNIX • 444
Install SystemEDGE Advanced Encryption on Windows • 443
Install the Agent in Legacy Mode • 104
Install the Agent on UNIX and Linux Systems • 90
Install the Agent on UNIX from the Command Line • 96
Install the Agent on Windows • 79
Install the Agent on Windows from the Command Line • 84
Installation and Deployment • 77
Installation Notes • 77
Installation on UNIX and Linux Systems • 90
Installation on Windows Systems • 78
Installation Prerequisites • 442
Installation with CA Virtual Assurance • 78
Installed Files • 445
Integration with CA Virtual Assurance • 70
Interactions Between sysedge.cf and sysedge.mon • 121
Interprocess Communication Group • 178
Introduction • 17
Inventory Tracking and Asset Management • 453
IP-MIB Tables • 29

K

Kernel Configuration Group • 174
Key Protection • 440

L

Legacy Mode Operation • 69
Legacy Monitors • 69
Legacy Support of the $CASYSEDGE Variable • 102
Limitations of the Monitor Maintenance Windows AIM • 411
Load SystemEDGE AIMs • 145
Log File Monitor Traps • 58
Log File Monitoring • 58, 327
Log File Monitoring Examples • 340
Log File Monitoring Overview • 327
Log Monitor Table • 187, 328
Log Monitor Table Action Parameters • 336
Log Monitor Table Columns • 328
Log Monitor Table Flags • 333
Log Monitoring Configuration • 337
logMonMatch Trap • 461
logMonNotReady Trap • 461
LPAR Group • 191

M

Maintenance Mode • 169
Managed Object Creation • 194, 240
Management System Not Receiving SNMP Trap Messages • 433
Message Buffers Group • 179
MIB Extensions • 30
MIB-II • 28
Mirror Tables • 191
monitor Directive--Add Entries to the Self Monitor Table • 212
Monitor Examples • 69
Monitor File Systems • 417
Monitor Log Files and Send Notification Through Email • 423
Monitor Maintenance Window Module Configuration • 146
Monitor Maintenance Windows AIM • 410
Monitor Processes • 424

Monitor the Existence of Quarantined Files • 417
Monitor the Windows Task Scheduler • 415
monitor Trap • 460
monitorClear Trap • 463
monitorEntryNotReady Trap • 460
Monitoring Architecture • 45
Monitoring Configuration • 144
Monitoring Features • 22
Multiline Regular Expressions • 151
Multiple SNMP Agents Support • 447
Multiple SystemEDGE Instance Support • 448

N

NFS Statistics Group • 186
NT Cache Performance Group • 183
NT Event Monitor Group • 184
NT Event Monitor Table • 355
NT Event Monitor Table Action Parameters • 362
NT Event Monitor Table Columns • 355
NT Event Monitor Table Flags • 360
NT Memory Performance Group • 183
NT Page File Performance Group • 184
NT Performance Groups • 183
NT Registry and Performance Extension Group • 184
NT Registry Group • 182
NT Services Table • 182
NT System Group • 181
NT System Performance Group • 183
NT Thread Table • 182
ntEventMonMatch Trap • 462
ntEventMonNotReady Trap • 462
ntregperf Keyword—Add Windows Registry and Performance MIB Objects • 404

O

Object Aggregation • 63
Object Status • 62
Obtain a Report for Troubleshooting • 428
On-Demand Modules • 73

P

Partition Table • 455
Perfcube Module Configuration • 146
Performance Cube AIM • 411
Performance Data • 401
Performance Group • 178
Perl Compatible Regular Expression (PCRE) Support • 150
Platform Management Modules • 70
Platform Support • 438
Plug-ins and Integrations • 31
Private Enterprise Traps • 459
Process and Service Autowatchers • 287
Process and Service Monitoring • 55, 237
Process and Service Monitoring Overview • 237
Process Attributes • 249
Process Autowatcher Examples • 301
Process Group Monitor Table • 190, 304
Process Group Monitor Table Action Parameters • 312
Process Group Monitor Table Columns • 304
Process Group Monitor Table Flags • 310
Process Group Monitor Traps • 57
Process Group Monitoring • 57, 303
Process Group Monitoring Configuration • 313
Process Group Monitoring Examples • 318
Process Group Monitoring Overview • 303
Process Monitor Table • 190, 241
Process Monitor Table Action Parameters • 259
Process Monitor Table Columns • 242
Process Monitor Table Flags • 253
Process Monitor Traps • 56
Process Monitoring Configuration • 261
Process Monitoring Examples • 268
Process Table • 175
processClear Trap • 465
processNotReady Trap • 465
Processor Load Tracking • 454
Processor Table • 454
processStart Trap • 464
processStop Trap • 463
processThreshold Trap • 464
procGroupChange Trap • 466

R

Recommendations for Configuring Security • 147
Recommendations for Log File Monitoring • 349
Recommendations for Process and Service Monitoring • 283
Recommendations for Using Extensions • 398
Reconfigure Extension Scripts Through SNMP • 419
Registry Data • 400
Regular Expression Examples • 152
Regular Expression Examples for Monitors and Autowatchers • 147
Reinstallation • 112

Related Publications • 17
Remote Configuration and Deployment • 71
Remote Shell Group • 177
Remove Log Monitoring Entries • 348
Remove Process Group Monitoring Entries • 324
Remove Process Monitoring Entries • 281
Remove Self Monitoring Entries • 235
Remove Windows Event Monitoring Entries • 375
Reverse OID Lookup • 215
Rotating Log Files • 351
Row Creation • 204
Row Creation Objects • 251, 309, 332, 358, 381
RPC Statistics Group • 186
Run a Remote Command • 177

S

Sample Process Monitor Table Entry • 248
Sample Self Monitor Table Entry • 203
Self Monitor Table • 185, 196
Self Monitor Table Action Parameters • 208
Self Monitor Table Columns • 197
Self Monitor Table Flags • 205
Self Monitor Traps • 54
Self Monitoring • 50, 193
Self Monitoring Configuration • 211
Self Monitoring Examples • 216
Self Monitoring Overview • 193
Send a Signal to a Process • 176
Service Autowatcher Example • 301
Service Startup Script for UNIX Systems • 166
Setting the Bind Address • 124
SNMP Tables • 42
SNMP Traps • 41
SNMP Versions and Access • 40
SNMPv1 and SNMPv2c Communities • 40
SNMPv1 Trap Format • 468
SNMPv2 Row Status • 46
SNMPv3 Configuration • 155
SNMPv3 Users and Security • 41
Solaris Zone Process Group Monitoring • 325
Solaris Zone Process Monitoring • 282
Specify a SNMPv1 Trap Source • 128
Specify an Autowatcher in sysedge.cf • 297
Start or Stop the Agent on UNIX Systems • 166
Start or Stop the Agent on Windows Systems • 165
Start the Agent Automatically on UNIX Systems • 168
Start the Agent Automatically on Windows Systems • 168
Starting the Agent • 165
State Management • 25
State Management Configuration Options • 68
State Management Model • 61
State Management of Process Monitors • 239
State Management of Self Monitors • 194
State Management Traps • 67
Streams Buffers Allocation Table • 180
Streams Group • 175
Super Aggregation • 64
Supported Authentication Protocols • 439, 442
Supported Encryption Protocols • 438, 441
Supported MIBs • 25, 51
Supported Platforms • 441
System Group • 173
SystemEDGE Additional Command Line Utilities • 414
SystemEDGE Advanced Encryption • 441
SystemEDGE AIMs • 409
SystemEDGE Command Line Utilities • 413
SystemEDGE Control Panel for Windows • 164
SystemEDGE Installation on 64-bit Linux Releases Fails • 95
Systems Management Empire MIB • 171
Systems Management Empire MIB Extension Group • 389
Systems Management Empire MIB Groups • 52
Systems Management Empire MIB ntRegPerf Group • 400
Systems Management Empire MIB Overview • 172

T

Table of Work Load Manager (WLM) Classes • 181
Terminology • 20
The NT Event Log Monitoring Table • 184
The Streams Buffers Group • 179
Top Processes AIM • 409
Trap Community Table • 181
Troubleshooting and Usage Suggestions • 427

U

Uninstall SystemEDGE Advanced Encryption on UNIX • 444
Uninstall SystemEDGE Advanced Encryption on Windows • 443

Uninstall SystemEDGE and the AIMs on UNIX Systems • 111
Uninstall SystemEDGE and the AIMs on Windows • 109
Uninstalling SystemEDGE • 108
Unmount a Mounted Device • 174
Unsupported Host Resources MIB Objects • 457
Unsupported Performance Data Types • 403
Unsupported Systems Management Empire MIB Objects • 192
Upgrade Managed Nodes and AIM Servers • 113
User Table • 175
Using diagsysedge.exe • 427
Using Extension Variables with Your Management Software • 396
Using Windows Registry and Performance Variables with Your Management Software • 406

V

Validate AIM Plugins • 429
Verify Agent Configuration • 433
Verify Agent Response to Queries • 432
Verify Agent Startup at System Initialization • 431
Verify Agent Trap Configuration • 434
Verify Management System Configuration • 433
Verify Management System Trap Configuration • 434
Verify that the Agent is Running • 431
Verify that the Agent is Sending Traps • 434
View the History Control Table • 383
View the Log Monitor Table • 337
View the NT Event Monitor Table • 363
View the Process Group Monitor Table • 313
View the Process Monitor Table • 261
View the Self Monitor Table • 210

W

Warm Start Exceptions • 169
watch logfile Directive--Add Entries to the Log Monitor Table • 338
watch ntevent Directive--Add Entries to NT Event Monitor Table • 365
watch ntservice Directive--Add Service Monitoring Entries to the Process Monitor Table • 266
watch process Directive--Add Entries to the Process Monitor Table • 263
watch procgroup Directive--Add Entries to the Process Group Monitor Table • 314
watch procgroupex Directive--Add Entries to the Process Group Monitor Table (UNIX only) • 316
Who Table • 176
Windows Event Monitor Traps • 59
Windows Event Monitoring • 59, 353
Windows Event Monitoring Configuration • 364
Windows Event Monitoring Examples • 367
Windows Event Monitoring Overview • 353
Windows Event Search Criteria • 354
Windows Registry and Performance Examples • 405
Windows Registry and Performance Functionality • 399
Windows Registry and Performance MIB Objects • 399
Windows Registry and Performance Variable Configuration • 403
Windows Service Monitoring • 238