



THE ENTERPRISE MIDDLEWARE SOLUTION

# BEA Manager

## Agent Integrator Reference Manual

Agent Integrator 4.2  
Document Edition 4.2  
October 1998

# Copyright

Copyright © 1997, 1998 BEA Systems, Inc. All Rights Reserved.

## Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

## Trademarks or Service Marks

BEA, ObjectBroker, TOP END, and TUXEDO are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Connect, BEA Manager, BEA MessageQ, Jolt and M3 are trademarks of BEA Systems, Inc.

All other company names may be trademarks of the respective companies with which they are associated.

## Agent Integrator Reference Manual

Document Edition	Part Number	Date	Software Version
4.2	815-001006-001	October 1998	Agent Integrator 4.2

---

# Contents

## Preface

Purpose of This Manual.....	xiii
Who Should Read This Manual .....	xiii
How This Manual Is Organized .....	xiii
Online Document Considerations.....	xiv
Opening the Manual in a Web Browser .....	xv
Printing the Manual from a Web Browser .....	xvii
How to Print the Complete Book .....	xvii
Documentation Conventions .....	xvii
Related Documentation .....	xix
Agent Integrator Documentation.....	xix
Other Related Publications .....	xx
Internet Standards Documents .....	xx
Contact Information.....	xxi
Documentation Support.....	xxi
Customer Support.....	xxi

## 1. Agent Integrator Overview

The Manager-Agent Model .....	1-2
Master Agents and Subagents .....	1-3
Polling.....	1-6
Object Identifiers .....	1-7
Relative and Absolute Object Identifiers .....	1-8
Specifying Object Identifiers .....	1-9
Supported MIB Objects.....	1-9
Directory Structure .....	1-11
Environment Variables .....	1-12

---

## 2. Getting Started

Administrative Tasks on UNIX Systems.....	2-1
Administrative Tasks on Windows NT Systems .....	2-4
Setting Up the Agent Integrator.....	2-6
Starting and Stopping the Agents .....	2-8
Starting the Agents on UNIX Systems .....	2-8
Starting the Agents on Windows NT Systems .....	2-9
Stopping the Agents .....	2-9

## 3. Using Multiple SNMP Agents

Agent Integrator Roles.....	3-2
Configuring Agent Integrator .....	3-3
Integrator Access to Managed Objects .....	3-3
SNMP Agents on Multiple Nodes .....	3-5

## 4. Using Agent Integrator for Polling

Procedure for Setting Up Local Polling .....	4-1
Introduction to Agent Integrator Rules.....	4-3
Conditions.....	4-3
States and Transitions.....	4-12
Actions.....	4-13
Starting and Stopping Polling .....	4-16
Creating New Polling Rules .....	4-16
Deleting or Modifying Polling Rules .....	4-17
Stopping Agent Integrator Polling Activity .....	4-17
Restarting Agent Integrator Polling Activity .....	4-18

## 5. Agent Integrator Commands

reinit_agents.....	5-1
snmp_integrator .....	5-2
stop_agents .....	5-4

## 6. Starting the Subagents

pm_snmpd .....	6-2
nt_snmpd .....	6-4
unix_snmpd .....	6-6

---

## 7. Configuration Files

BEA Manager Configuration File (beamgr.conf) .....	7-2
Default Location .....	7-2
Description .....	7-2
Keywords Used by All BEA Manager Products .....	7-3
Keywords Used by the Agent Integrator .....	7-4
Keywords Used by the Agent Connection .....	7-4
Keywords Used by unix_snmpd and nt_snmpd .....	7-7
The NON_SMUX_PEER Entry .....	7-8
The OID_CLASS Entry .....	7-13
The RULE_ACTION Entry .....	7-14
BEA Manager Passwords File (beamgr_snmpd.conf) .....	7-18
Default Location .....	7-18
Description .....	7-18

## 8. BEA SNMP Agent MIB

beaSystem Group .....	8-2
beaSysDescr .....	8-3
beaSysId .....	8-3
beaSysInstallTime .....	8-3
beaSysContact .....	8-4
beaSysName .....	8-4
beaSysLocation .....	8-4
beaSysServices .....	8-4
beaTrapHost .....	8-5
beaTrapCommunity .....	8-5
beaTrapPort .....	8-5
beaTrapDescr .....	8-5
beaSysHasDisk .....	8-6
beaSysSysname .....	8-6
beaSysNodename .....	8-6
beaSysRelease .....	8-6
beaSysVersion .....	8-6
beaSysMachine .....	8-7
beaSysAgentVersion .....	8-7

---

beaUnix Group .....	8-8
beaPsTable.....	8-9
beaPsUser.....	8-9
beaPsPid .....	8-10
beaPsCpu.....	8-10
beaPsMem .....	8-10
beaPsSize.....	8-10
beaPsRss.....	8-10
beaPsTty .....	8-11
beaPsStat .....	8-11
beaPsDate .....	8-11
beaPsCmd.....	8-11
beaPsCpuTime .....	8-12
beaPsThreadCount .....	8-12
beaPsHandleCount .....	8-12
beaDfTable .....	8-13
beaDfIndex .....	8-13
beaDfHostname.....	8-13
beaDfFilesystem.....	8-14
beaDfKbytes.....	8-14
beaDfUsed .....	8-14
beaDfAvail .....	8-14
beaDfCapacity.....	8-14
beaDfMountedon.....	8-15
System Statistics.....	8-15
beaPstatSwapAlloc.....	8-15
beaPstatSwapReser.....	8-15
beaPstatSwapUsed .....	8-15
beaPstatSwapAvail.....	8-16
beaMqTable .....	8-17
beaMqT .....	8-18
beaMqId .....	8-18
beaMqKey .....	8-18
beaMqMode .....	8-18
beaMqOwner.....	8-18

---

beaMqGroup .....	8-19
beaMqCreator.....	8-19
beaMqCgroup.....	8-19
beaMqCbytes .....	8-19
beaMqQnum.....	8-19
beaMqQbytes .....	8-20
beaMqLspid .....	8-20
beaMqLrpId.....	8-20
beaMqStime .....	8-20
beaMqRtime.....	8-20
beaMqCtime.....	8-21
beaShmTable .....	8-22
beaShmT .....	8-23
beaShmId .....	8-23
beaShmKey .....	8-23
beaShmMode .....	8-23
beaShmOwner .....	8-23
beaShmGroup.....	8-24
beaShmCreator.....	8-24
beaShmCgroup.....	8-24
beaShmNattch .....	8-24
beaShmSegsz .....	8-24
beaShmCpid .....	8-25
beaShmLpid .....	8-25
beaShmAtime.....	8-25
beaShmDtime.....	8-25
beaShmCtime .....	8-25
beaSemTable .....	8-26
beaSemT.....	8-26
beaSemId.....	8-27
beaSemKey .....	8-27
beaSemMode.....	8-27
beaSemOwner .....	8-27
beaSemGroup.....	8-27
beaSemCreator .....	8-28

---

beaSemCgroup .....	8-28
beaSemNsems .....	8-28
beaSemOtime .....	8-28
beaSemCtime .....	8-28
beaLclDfTable .....	8-29
beaLclDfFilesystem .....	8-29
beaLclDfKbytes .....	8-29
beaLclDfUsed .....	8-30
beaLclDfAvail .....	8-30
beaLclDfCapacity .....	8-30
beaLclDfMountedon .....	8-30
beaSmgr Group .....	8-31
beaSmgrTable .....	8-31
beaSmgrIndex .....	8-32
beaSmgrIpcKey .....	8-32
beaSmgrShmAllocated .....	8-32
beaSmgrSubsystem .....	8-32
beaSmgrCreationDate .....	8-33
beaSmgrLibVersion .....	8-33
beaSmgrMaxShmEntries .....	8-33
beaSmgrCurrentShmEntries .....	8-33
beaSmgrHWMShmUsed .....	8-33
beaSmgrPctShmUsed .....	8-34
beaSmgrHWMPPctShmUsed .....	8-34
beaSmgrTotalShmSize .....	8-34
beaSmgrUsegSize .....	8-34
beaSmgrAuxSize .....	8-34
beaSmgrSemId .....	8-35
beaSmgrShmId .....	8-35
beaSysPerf Group .....	8-36
beaSysPerfCpu .....	8-37
beaSysPerfCntxt .....	8-37
beaSysPerfSwap .....	8-37
beaSysPerfDisk .....	8-37
beaSysPerfIntr .....	8-37



---

beaSysPerfLoad .....	8-38
beaSysPerfPage .....	8-38
beaSysPerfIfNumber .....	8-38
beaSysPerfCntxtDelta .....	8-38
beaSysPerfSwapDelta .....	8-38
beaSysPerfDiskDelta .....	8-39
beaSysPerfIntrDelta .....	8-39
beaSysPerfLoadDelta .....	8-39
beaSysPerfPageDelta .....	8-39
beaSysPerfIfTable .....	8-40
beaSysPerfIfIndex .....	8-41
beaSysPerfIfDescr .....	8-41
beaSysPerfIfOperStatus .....	8-41
beaSysPerfIfInPackets .....	8-41
beaSysPerfIfInErrors .....	8-42
beaSysPerfIfOutPackets .....	8-42
beaSysPerfIfOutErrors .....	8-42
beaSysPerfIfCollisions .....	8-42
beaSysPerfIfInPacketsDelta .....	8-43
beaSysPerfIfInErrorsDelta .....	8-43
beaSysPerfIfOutPacketsDelta .....	8-43
beaSysPerfIfOutErrorsDelta .....	8-43
beaSysPerfIfCollisionsDelta .....	8-44
beaNTSysPerf Group .....	8-45
beaNTSysPerfCalls .....	8-46
beaNTSysPerfAlignFix .....	8-46
beaNTSysPerfExptDisp .....	8-46
beaNTSysPerfFloatEmul .....	8-46
beaNTSysPerfPgFaults .....	8-47
beaNTSysPerfPaging .....	8-47
beaNTSysPerfCacheBytes .....	8-47
beaNTSysPerfCodeTotal .....	8-48
beaNTSysPerfCodeRes .....	8-48
beaNTSysPerfDriverTotal .....	8-48
beaNTSysPerfDriverRes .....	8-48

---

beaNTSysPerfCacheRes.....	8-49
beaNTSysPerfPageUsed.....	8-49
beaNTSysPerfPagePeak .....	8-49
beaIntAgt Group .....	8-50
beaIntAgtTable .....	8-50
beaIntAgtRuleId .....	8-50
beaIntAgtScanIntvl.....	8-51
beaIntAgtRuleAction .....	8-51
beaIntAgtStatus .....	8-51

## **A. SNMP Information**

Reference Books .....	A-2
Obtaining MIBs .....	A-2
Enterprise ID Assignment .....	A-3
Obtaining RFCs .....	A-3
Obtaining Specifications.....	A-4
OSI NMF Documents .....	A-5
Mailing Lists and News Groups .....	A-5
Standards and Drafts.....	A-6
Accessing Internet-Drafts .....	A-7

## **Glossary**

## **Index**

---

# Preface

## Purpose of This Manual

The Agent Integrator Reference Manual provides reference information on the agents, utilities, configuration files, and MIBs shipped with the BEA Agent Integrator. Use this manual to understand how to configure and manage the Agent Integrator master agents.

## Who Should Read This Manual

This reference manual is particularly helpful for network or system administrators responsible for administering SNMP master agents and SMUX subagents.

## How This Manual Is Organized

The Agent Integrator Reference Manual is organized as follows:

- ◆ Chapter 1, “Agent Integrator Overview,” explains the agent/manager model, the structure of the SNMP agent software, its components, and its directory structure.
- ◆ Chapter 2, “Getting Started,” describes the administrative and user tasks that need to be performed after software installation.

- 
- ◆ Chapter 3, “Using Multiple SNMP Agents,” describes the use of the Agent Integrator to coordinate communication between an SNMP manager and multiple SNMP agents.
  - ◆ Chapter 4, “Using Agent Integrator for Polling,” describes the rules used to offload threshold-checking from the manager to the Agent Integrator master agent.
  - ◆ Chapter 5, “Agent Integrator Commands,” describes the syntax of the commands used to start or re-initialize the Agent Integrator.
  - ◆ Chapter 6, “Starting the Subagents,” explains how to use the SNMP Multiplex (SMUX) subagents shipped with the Agent Integrator.
  - ◆ Chapter 7, “Configuration Files,” describes the BEA Manager SNMP configuration files.
  - ◆ Chapter 8, “BEA SNMP Agent MIB,” describes the MIB objects supported by the BEA agent MIB, which is defined in the file `bea.asn1`.
  - ◆ Appendix A, “SNMP Information,” lists references and contacts for information concerning MIBs and the SNMP protocol.
  - ◆ “Glossary” contains a collection of acronyms and phrases used in discussing SNMP applications.

## Online Document Considerations

This document, Agent Integrator Reference Manual, is designed primarily as an online, hypertext guide. To get full use of the information in this document you should install and access it as an online document via a Web browser that supports HTML 3.0. Netscape Navigator 2.02 or Microsoft Internet Explorer 3.0 or later are recommended. (Information on how to install the online documentation is available in the *BEA Manager Installation Guide*.)

---

## Opening the Manual in a Web Browser

To access the interactive version of this document, open the following HTML file in a Web browser:

`$RELEASE_ROOT/DOCS/MGR/20/INTEG/INDEX.HTM`

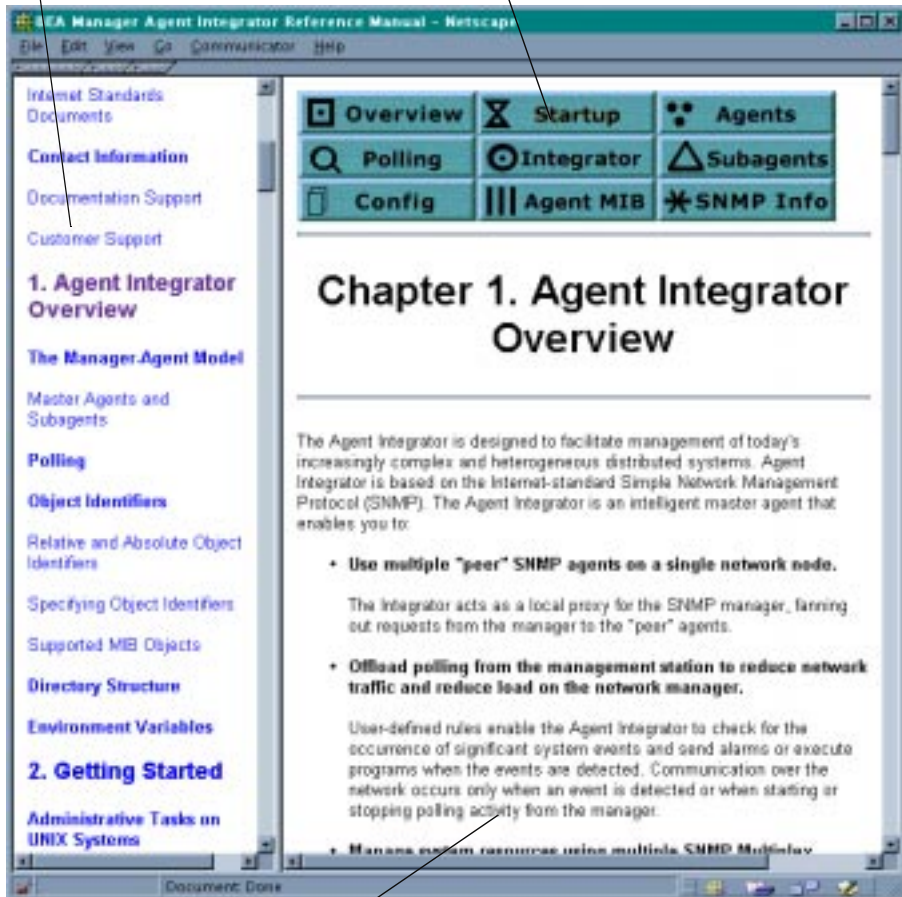
**Note:** The online documentation requires a Web browser that supports HTML 3.0. Netscape Navigator 2.02 or Microsoft Internet Explorer 3.0 or later are recommended.

Figure 1 shows the online manual with the clickable navigation bar and table of contents.

**Figure 1 Agent Integrator Reference Manual Displayed in Netscape Web Browser**

**Table of Contents**  
click on a topic to view it

**Navigation Bar**  
click on a button to view  
a main topic



**Document Display Area**

---

## Printing the Manual from a Web Browser

You can print a hardcopy version of this manual, one file at a time, from the Web browser. Before you print, make sure that the chapter or appendix you want is displayed and *selected* in your browser. (To select a file, click anywhere inside the frame you want to print. If your browser offers a Print Preview feature, you can use it to verify which file you are about to print.)

## How to Print the Complete Book

A PDF version of the *Agent Integrator Reference Manual* is available in the following location:

`YourDrive:\docs\mgr\20\pdf\integ.pdf`

To print the documentation, open a PDF file in an Adobe Acrobat Reader and choose the file print option.

If you do not have a reader, you can download one from the Adobe Web site at <http://www.adobe.com/>.

## Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
<b>boldface text</b>	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys sequentially.
<i>italics</i>	Indicates emphasis or book titles.

---

Convention	Item
<code>monospace text</code>	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard. <i>Examples:</i> <code>#include &lt;iostream.h&gt; void main ( ) the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float</code>
<b><code>monospace boldface text</code></b>	Identifies significant words in code. <i>Example:</i> <code>void <b>commit</b> ( )</code>
<i><code>monospace italic text</code></i>	Identifies variables in code. <i>Example:</i> <code>String <i>expr</i></code>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> <code>LPT1 SIGNON OR</code>
<code>{ }</code>	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
<code>[ ]</code>	Indicates optional items in a syntax line. The brackets themselves should never be typed. <i>Example:</i> <code>buildobjclient [-v] [-o name ] [-f <i>file-list</i>]... [-l <i>file-list</i>]...</code>
<code> </code>	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.



---

Convention	Item
...	Indicates one of the following in a command line: <ul style="list-style-type: none"><li>◆ That an argument can be repeated several times in a command line</li><li>◆ That the statement omits additional optional arguments</li><li>◆ That you can enter additional parameters, values, or other information</li></ul> The ellipsis itself should never be typed. <p><i>Example:</i></p> <pre>buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...</pre>
.	Indicates the omission of items from a code example or from a syntax line.
.	The vertical ellipsis itself should never be typed.
.	

---

## Related Documentation

The following sections list the documentation provided with the Agent Integrator software, and other publications related to SNMP technology.

## Agent Integrator Documentation

The Agent Integrator documentation consists of the following items:

- ◆ *Agent Integrator Reference Manual*
- ◆ *BEA Manager Installation Guide*
- ◆ *BEA Manager Release Notes*

---

## Other Related Publications

Other related BEA Manager SNMP publications:

- ◆ *Agent Connection for TUXEDO and M3 Reference Manual*
- ◆ *Agent Development Kit Programmer's Guide*

A comprehensive list of relevant publications can be found in the Appendix A, “SNMP Information.”

## Internet Standards Documents

You do not need to understand the SNMP or SMUX protocols in depth to use the Agent Integrator. We do recommend that you read the following Internet standards documents:

RFC 1155 — Structure of Management Information

RFC 1157 — Simple Network Management Protocol

RFC 1212 — Concise MIB Definitions

RFC 1213 — Management Information Base (MIB II)

RFC 1227 — SNMP Multiplex (SMUX)

Information on obtaining these RFC documents can be found in Appendix A, “SNMP Information.”

---

# Contact Information

The following sections provide information about how to obtain support for the documentation and software.

## Documentation Support

If you have questions or comments on the documentation, you can contact the BEA Information Engineering Group by e-mail at **docsupport@beasys.com**. (For information on how to contact Customer Support, refer to the section “Customer Support.”)

## Customer Support

If you have any questions about this version of BEA Agent Integrator, or if you have problems installing and running BEA Agent Integrator, contact BEA Customer Support through BEA WebSupport at [www.beasys.com](http://www.beasys.com). You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- ◆ Your name, e-mail address, phone number, and fax number
- ◆ Your company name and company address
- ◆ Your machine type and authorization codes
- ◆ The name and version of the product you are using
- ◆ A description of the problem and the content of pertinent error messages



# 1 Agent Integrator Overview

The Agent Integrator is designed to facilitate management of today's increasingly complex and heterogeneous distributed systems. Agent Integrator is based on the Internet-standard Simple Network Management Protocol (SNMP). The Agent Integrator is an intelligent master agent that enables you to:

- ◆ **Use multiple “peer” SNMP agents on a single network node.**

The Integrator acts as a local proxy for the SNMP manager, fanning out requests from the manager to the “peer” agents.

- ◆ **Offload polling from the management station to reduce network traffic and reduce load on the network manager.**

User-defined rules enable the Agent Integrator to check for the occurrence of significant system events and send alarms or execute programs when the events are detected. Communication over the network occurs only when an event is detected or when starting or stopping polling activity from the manager.

- ◆ **Manage system resources using multiple SNMP Multiplex (SMUX) subagents (such as those built using the BEA Manager Agent Development Kit).**

The Integrator “speaks” SMUX to the subagents and fans out requests from an SNMP-compliant system or network management station to the appropriate subagent.

- ◆ **Manage system resources using any other master agent/subagent protocol, such as Desktop Program Interface (DPI), where the master agent responds to SNMP management requests.**

The DPI master agent “speaks” Simple Network Management Protocol (SNMP) to management stations and is thus managed by Agent Integrator in the same manner as any other “peer” SNMP agent.

◆ **Coordinate communication between an SNMP manager and multiple SNMP agents on multiple network nodes.**

This feature is particularly useful when offloading polling to the Agent Integrator for management of a distributed system whose components are spread over a number of computers. To the Agent Integrator, the managed resources appear as if they were on a single computer.

This chapter provides a general introduction to SNMP concepts and the Agent Integrator software. It includes the following sections:

- ◆ “The Manager-Agent Model,” gives a brief overview of the SNMP manager-agent model and Agent Integrator capabilities.
- ◆ “Polling,” gives an overview of the Agent Integrator’s ability to offload threshold-checking from the management station.
- ◆ “Object Identifiers,” gives an overview of the tree structure that defines access to managed objects within the Management Information Base (MIB). The MIB provides a shared definition of managed resources that enables managers and agents to cooperate in management tasks.
- ◆ “Directory Structure,” describes the BEA Manager Agent Integrator directory structure.
- ◆ “Environment Variables,” summarizes the BEA Manager Agent Integrator environment variables.

## The Manager-Agent Model

The Agent Integrator software is based on the manager/agent model described in the network management standards defined by the International Organization for Standardization (ISO). In this model, a network manager exchanges monitoring and control information about network and system resources with distributed software processes called *agents*.

Any system or network resource that is manageable through this exchange of information is a *managed resource*. This could be a software resource, such as a message queue or TUXEDO application, or a hardware resource, such as a router or NFS file server.

Agents function as “collection devices” that typically gather and send data about the managed resource in response to a request from a manager. In addition, agents often have the ability to issue unsolicited reports to managers when they detect certain predefined thresholds or events on a managed resource. In SNMP terminology, these unsolicited event reports are called *trap notifications*.

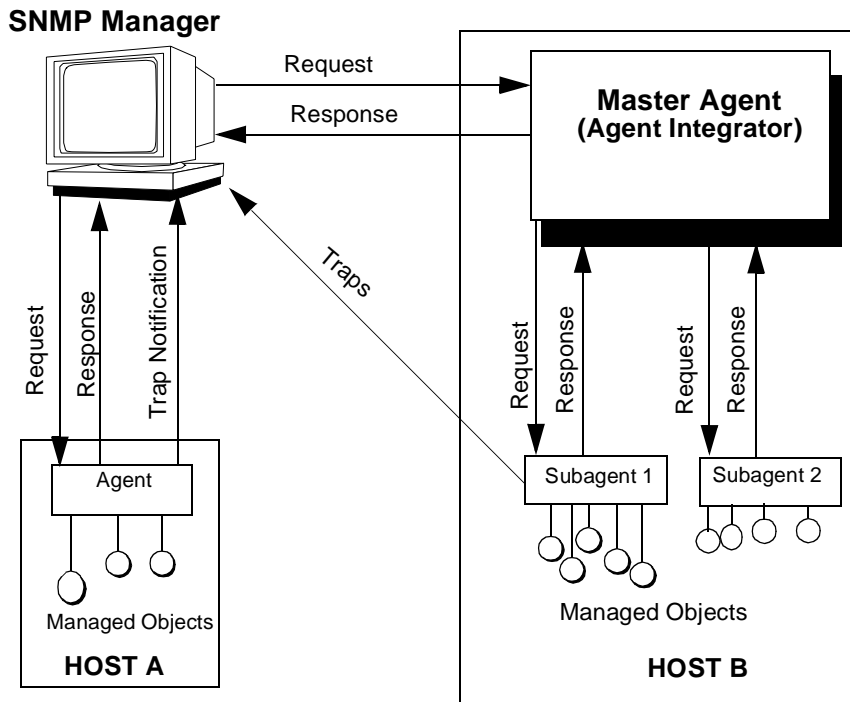
A manager relies upon a database of definitions and information about the properties of managed resources and the services the agents support — this comprises the Management Information Base (MIB). When new agents are added to extend the management domain of a manager, the manager must be provided with a new MIB component that defines the manageable features of the resources managed through that agent. The manageable features of resources, as defined in an SNMP-compliant MIB, are called *managed objects*. Defining the heterogeneous components of an enterprise’s distributed systems within a common MIB on the management station provides a unified perspective and single access point for managing system and network resources.

## Master Agents and Subagents

In the SNMP architecture, each managed object can be managed through only one SNMP agent, and each host may have only one agent communicating with an SNMP manager. The original SNMP management solution allowed for only a single, monolithic agent to carry out all management responsibilities on a given network node (IP address). This solution was soon discovered to be too inflexible to provide for effective management of increasingly complex systems. In addition to the agents typically provided by computer manufacturers for hardware and operating system information, agents are also being produced by vendors of other products, such as agents for SQL database systems. Complex and heterogeneous systems thus require the ability to accommodate multiple agents on a single network node.

The SNMP architecture was thus extended to allow a single master agent to communicate with subagents, allowing multiple agents to cooperate in managing diverse hardware and software components on a single host. This master agent functionality is provided by the BEA Manager Agent Integrator software. The extended SNMP Manager/Agent model is illustrated in Figure 1-1.

**Figure 1-1 SNMP Architecture**



A typical protocol used for communication between an SNMP master agent and subagents is the SNMP Multiplex (SMUX) protocol, defined in RFC 1227.

However, you may still wish to use one or more old-style monolithic agents that do not allow for a master agent/subagent architecture. Yet no standardized solution has emerged for the coexistence of non-SMUX SNMP agents on a single host. Master agents that “speak” SMUX protocol to subagents are typically able to communicate only with SMUX-compliant subagents, and cannot coexist with non-SMUX SNMP agents running on the same host.

However, the Agent Integrator can run on the same node with SNMP agents and SMUX subagents. The Agent Integrator can run on the same node with other master agent/subagent architectures, such as Distributed Program Interface (DPI) or EMANATE, so long as the master agent uses SNMP to respond to management requests, thus allowing multiple agents and subagents from any vendor to cooperate in the management of system components. The DPI master agent simply appears to the

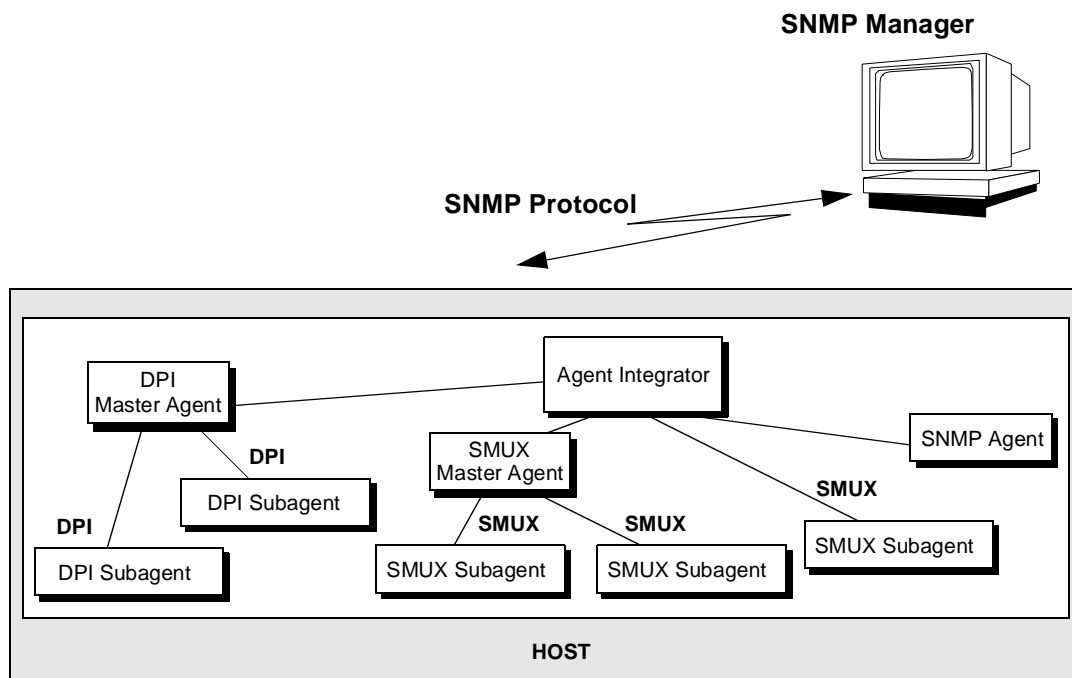


Agent Integrator as another SNMP agent. The multiple SNMP agents, SMUX subagents, and other subagents communicate with SNMP managers through the Agent Integrator and appear as a single SNMP agent to any SNMP manager.

In its communication with SMUX subagents (and DPI master agents and monolithic SNMP agents), the Agent Integrator acts as a proxy for the manager. The Agent Integrator distributes requests from the manager to specific SNMP agents or subagents, and receives the responses from the individual agents and forwards those responses back to the manager (as illustrated in Figure 1-1).

In Figure 1-2, the Agent Integrator master agent controls two master agents (one SMUX master agent and one DPI master agent) each controlling two subagents. It also directly controls one monolithic SNMP agent and a SMUX subagent.

**Figure 1-2 Agent Integrator Master/Subagent Architecture**



# Polling

Managers can request the current values of managed objects at periodic intervals; this is called *polling*. To track faults in critical system components or applications, management systems use polling to determine whether attributes of the managed resource have crossed some significant threshold. However, direct polling by a management station becomes increasingly inefficient as the number of components being polled increases — the load on network bandwidth increases as does the load on the management station itself.

The Agent Integrator allows you to offload polling from the management station to the Agent Integrator. In other words, the Agent Integrator can be configured to check for a specified condition on its own. This feature expands your system management capacity by reducing the load on the management station and network bandwidth caused by threshold-checking activity.

The user can specify the threshold to check, and polling can be activated during Agent Integrator startup or by an SNMP SET request from the management station. The Agent Integrator sends an enterprise-specific SNMP trap when the threshold is crossed. To indicate the cause of the alarm, the user can configure a specific-trap type number to be sent in the trap generated when a given threshold is crossed. Communication between the manager and Integrator occurs only when the manager sends a SET request to de-activate (or re-activate) the polling, or when the Agent Integrator sends an SNMP trap if it detects the specified event in the managed resource. The Integrator can also be configured to execute a script or program when a threshold is crossed.

The polling capability of the Agent Integrator is described in more detail in Chapter 4, “Using Agent Integrator for Polling.”

# Object Identifiers

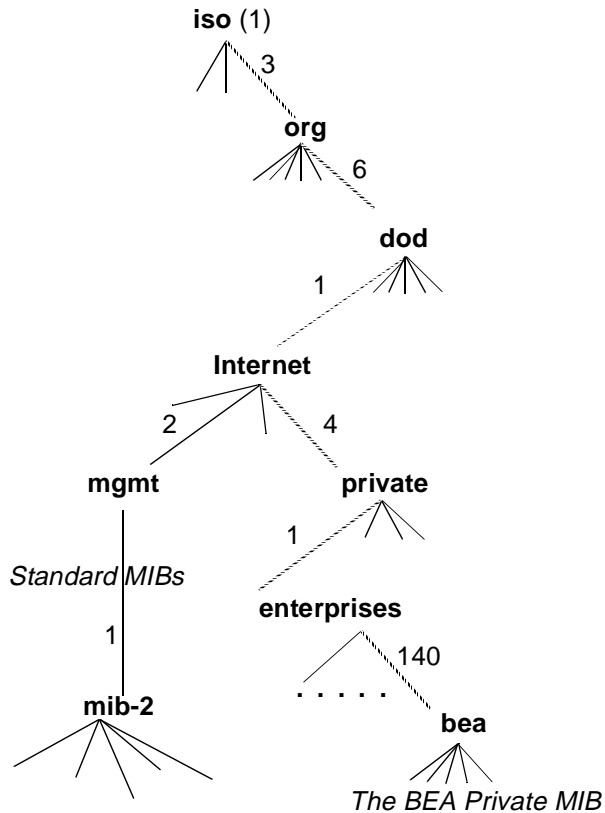
An SNMP management system sees the data it manages as a collection of managed objects. The managed objects make up the Management Information Base (MIB). Each object in the MIB has an *object identifier* (OID), which the manager uses to request its value from the agent.

To make use of the Agent Integrator polling feature, described in the previous section, OIDs are used to identify the managed objects whose values are retrieved by the Agent Integrator when checking for the occurrence of events in the managed resource.

The OID is a sequence of integers that uniquely identify a managed object by defining a path to that object through a tree-like structure, which is often called the *OID tree* or registration tree. When an SNMP agent wishes to access a specific object, it traverses the OID tree to find the object.

Figure 1-3 illustrates the OID tree for the BEA private MIB. Each BEA private MIB object that the SNMP agent software manages uses a prefix of .1.3.6.1.4.1.140 to identify it as an object in the BEA private MIB. This number sequence defines the path to this branch of the OID tree as illustrated in Figure 1-3.

**Figure 1-3 Object Identification Scheme**



## Relative and Absolute Object Identifiers

*Absolute* OIDs specify a path to an attribute from the root of the OID tree (as in Figure 1-3). *Relative* OIDs specify a path to the attribute relative to some node in the OID tree. For example, 2.1.1.7 specifies the `sysContact` object in the `system` group, relative to the `Internet` node in the OID tree.

Absolute OID names always begin with a dot and must specify every node of the OID tree from the top-most node to the specific managed object; for example:

```
.1.3.6.1.2.1.1.1
```

## Specifying Object Identifiers

Thus far we have used only a series of integers separated by dots — called “dot-dot” notation — to describe OIDs. However, an OID can also be expressed using textual symbols instead of numbers to represent nodes in the path to the object, or by a combination of both integers and textual symbols. A *symbolic* OID uses mnemonic keywords to specify the managed object; for example:

```
mgmt.mib-2.system.sysDescr
```

The following numeric OID uses integers to specify the same managed object:

```
2.1.1.1
```

Note that this example is a relative OID.

An OID may combine both symbolic and numeric representations of individual nodes of the OID tree; for example:

```
mgmt.mib-2.1.sysDescr
```

**Note:** When using OIDs to specify objects whose values are checked using Agent Integrator polling rules, only the numeric form of the OID may be used. For details, see Chapter 7, “Configuration Files.”

## Supported MIB Objects

In order to access MIB objects that are managed by agents or subagents, the scope of the OID tree that each agent or subagent is responsible for must be defined to the Agent Integrator. For monolithic SNMP agents, and SMUX or DPI master agents, this is done by specifying an OID in one or more NON\_SMUX\_PEER entries in the `beamgr.conf` configuration file (as described in Chapter 3, “Using Multiple SNMP Agents”). The Agent Integrator then knows to access the managed objects in that branch of the OID tree through the specified agent.

The Agent Integrator *directly* accesses MIB objects in the SMUX MIB, the MIB II system and snmp groups, and the beaIntAgtTable MIB object in the BEA Manager agent MIB. The beaIntAgtTable MIB objects define the polling capability of the Agent Integrator.

Using this MIB, the Agent Integrator can be configured to perform local polling and generate SNMP trap events or execute a system command when certain conditions are met. The same effect can be achieved by defining a RULE\_ACTION entry in the beamgr.conf file. This configuration file is described in Chapter 7, “Configuration Files.”

Three SMUX subagents are also shipped with Agent Integrator to support MIB groups in the BEA Manager agent MIBs as follows:

unix\_snmpd

Supports the beaSystem, beaUnix, beaSmgr, and beaSysPerf MIB groups. Supported on UNIX platforms only.

nt\_snmpd

Supports the beaSystem and beaNTSysPerf MIB groups. Also supports the following subgroups within the beaUnix group: the host process table (beaPsTable), the host file system table (beaDfTable), the system statistics group, and the local file system table (beaLc1DfTable). Does not support the message queue (beaMqTable) and semaphore (ipcs -sa) (beaSemTable) tables. Also, the beaShmTable (UNIX shared memory table) and beaSmgrTable are not supported. This is the Windows NT equivalent of unix\_snmpd.

pm\_snmpd

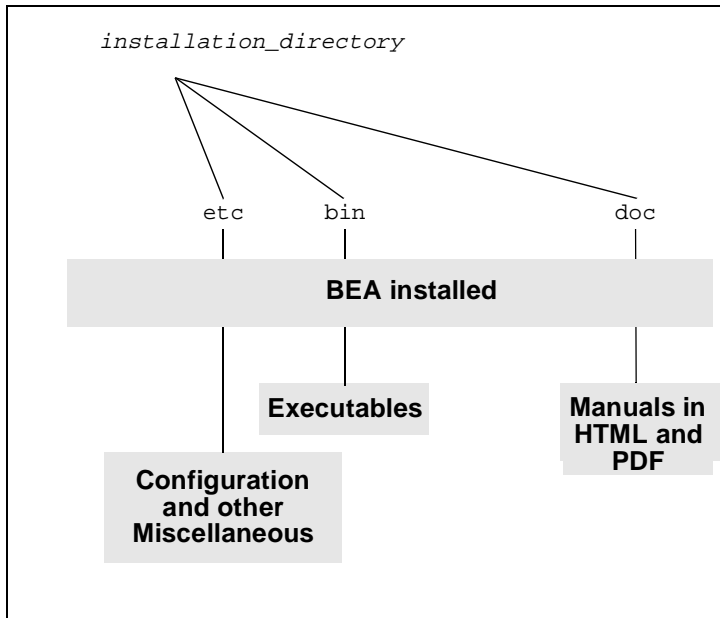
Supports the BEA Manager Log Central Process Monitor. Consult the *Log Central Administrator's Guide* for information on supported MIB objects.

Refer to Chapter 8, “BEA SNMP Agent MIB,” for more information about the BEA Manager MIB groups. The BEA Manager agent MIBs are defined in the BEA Manager MIB file bea.asn1. Refer to Chapter 6, “Starting the Subagents,” for more information about the subagents.

# Directory Structure

You can find the Agent Integrator files in the directories shown in Figure 1-4.

**Figure 1-4 Directory Structure**



**Note:** The items that appear in the gray boxes are descriptions only. They are not actual file names.

# Environment Variables

The BEA Manager Agent Integrator uses the following environment variables:

**BEA\_SMUX\_PASSWD**

Indicates the password that a SMUX subagent is to use when re-establishing communication with the Agent Integrator.

**BEA\_PEER\_MAX\_TRIES**

Indicates the number of times the Agent Integrator retries sending an SNMP request to peer SNMP agents, when no response is received within the established timeout interval.

**BEA\_PEER\_MAX\_WAIT**

Modifies the default time interval that the Agent Integrator waits for a reply to a request sent to a SMUX subagent or an SNMP peer agent. This value can also be set by adding a BEA\_PEER\_MAX\_WAIT entry to the BEA Manager configuration file. If this environment variable is not set, and there is no BEA\_PEER\_MAX\_WAIT entry in the configuration file, the default is three seconds. For peer SNMP agents, the default timeout value can be overridden for individual SNMP agents using the timeout parameter in NON\_SMUX\_PEER entries in the BEA Manager configuration file (beamgr.conf), as described in Chapter 7, “Configuration Files.”

**BEA\_SM\_BEAMGR\_CONF**

Specifies the absolute path to the BEA Manager configuration file (including the file name).



# 2 Getting Started

This chapter describes the tasks that you need to perform after installing the Agent Integrator software on the managed node where you want the master agent to run. These tasks differ somewhat depending upon whether you are using the software on a UNIX system or a Windows NT system. For information about installing the software, please consult the *BEA Manager Installation Guide*.

## Administrative Tasks on UNIX Systems

After the software is installed, there are additional tasks that must be carried out before the BEA Manager products can be used. This section describes post-installation tasks that are common to all BEA Manager products. To begin with, perform the following tasks:

### 1. Set your PATH to point to the BEA Manager executables

All users of the installed BEA Manager products need to update their PATH environment variable to include the location of the BEA Manager executable files. The following is an example in C shell:

```
% set path = ( $PATH installation_directory/bin )
```

### 2. Copy the configuration file:

Log in as root and copy the BEA Manager configuration file `beamgr.conf` from `installation_directory/etc` to the `/etc` directory.

```
%su
Password:
# cp installation_directory/etc/beamgr.conf /etc
```

### 3. Specify the destination for traps, if desired

The default destination for SNMP trap notifications is `localhost`. If you want traps to be sent to some other destination, use your favorite text editor to modify the BEA Manager configuration file (`beamgr.conf`) `TRAP_HOST` entry to specify the host name of the target destination machine for SNMP trap notifications, and the port number and community name to use in sending traps. For more information refer to Chapter 7, “Configuration Files.”

### 4. Specify non-default SNMP communities and SMUX password, if desired

By default, BEA Manager agents (such as the Agent Integrator or `tux_snmpd` when running as an SNMP agent) use `public` as the read-only community and `iview` as the read-write community when communicating with SNMP managers. If you want to specify different community names to be used by BEA Manager SNMP agents, this is specified in the BEA Manager passwords file. The passwords file can also be used to specify a password to be used by Agent Integrator for authenticating connection requests from SMUX subagents. To set up the passwords file, do the following:

- a. Copy the BEA Manager passwords file (`beamgr_snmpd.conf`) from the `installation_directory/etc` to the `/etc` directory and make the copy readable and writable only by root. For example:

```
# cp installation_directory/etc/beamgr_snmpd.conf /etc
# chmod 600 /etc/beamgr_snmpd.conf
```

- b. Now you can edit the copied file to update your SNMP communities. The keywords in this file are:

- ◆ `SMUX_PASSWD`
- ◆ `COMMUNITY_RO`
- ◆ `COMMUNITY_RW`
- ◆ `SET_DISABLE`

- c. If you want the agents to be read-only, there should be a `DISABLE_SET` entry in the passwords file as follows:

```
DISABLE_SET YES
```

If there is no `DISABLE_SET` entry in the passwords file, the agent has both SET and GET capability.

For more information refer to Chapter 7, “Configuration Files.”

## 5. Advertise services if you need to use non-standard ports

By default, BEA Manager agents assume the following port numbers as specified by SNMP and SMUX standards:

snmp	161/udp
snmp-trap	162/udp
smux	199/tcp

The default port assignments may be sufficient for your needs. If necessary, you can define these services on other ports, or use the appropriate command-line options when starting BEA Manager agents to assign them to non-default ports.

To modify or define the services, determine if the NIS is running. You can use the `ypwhich` command to determine if an NIS server or map master is available. For example:

```
% ypwhich
zort.kremvax.com
```

If an NIS server is available, you can use the `ypcat` command to determine if the services are available.

```
% ypcat services | grep snmp
snmp-trap    162/udp    snmptrap
snmp         161/udp
```

If an NIS server is not available and services are provided on the local host, you can examine the `/etc/services` file instead.

```
% cat /etc/services | grep snmp
snmp-trap    162/udp    snmptrap
snmp         161/udp
```

Refer to your UNIX system documentation, or consult your UNIX system administrator, for instructions specific to your UNIX platform to establish the SNMP services if necessary.

# Administrative Tasks on Windows NT Systems

After the software is installed, there are additional tasks that must be carried out before the BEA Manager products can be used. To begin with, perform the following tasks:

## 1. Install the BEA Manager configuration file

Copy the BEA Manager configuration file (`beamgr.conf`):

```
md c:\etc
copy installation-directory\etc\beamgr.conf c:\etc
```

## 2. Specify the destination for traps, if desired

The default destination for SNMP trap notifications is `localhost`. If you want to specify some other destination for SNMP traps, use your favorite text editor to modify the BEA Manager configuration file (`beamgr.conf`) `TRAP_HOST` entry to specify the host name of the target destination machine for SNMP trap notifications, and the port number and community name to use in sending traps. For more information refer to Chapter 7, “Configuration Files.”

## 3. Specify non-default SNMP communities and SMUX password, if desired

By default, BEA Manager agents (such as the Agent Integrator or `tux_snmpd` when running as an SNMP agent) use `public` as the read-only community and `iview` as the write-read community when communicating with SNMP managers. If you want to specify different community names to be used by BEA Manager SNMP agents, this is specified in the BEA Manager passwords file. The passwords file can also be used to specify a password to be used by Agent Integrator for authenticating connection requests from SMUX subagents. To set up the passwords file, do the following:

- a. Copy the BEA Manager passwords file (`beamgr_snmpd.conf`) to `c:\etc`. For example:

```
copy installation-directory\etc\beamgr_snmpd.conf c:\etc
```

- b. Now you can modify the SNMP communities in this file. The keywords used in this file are:

- ◆ `SMUX_PASSWD`
- ◆ `COMMUNITY_RO`

◆ COMMUNITY\_RW

◆ SET\_DISABLE

- c. If you want the agents to be read-only, there should be a DISABLE\_SET entry in the passwords file as follows:

```
DISABLE_SET YES
```

If there is no DISABLE\_SET entry in the passwords file, the agent has both SET and GET capability.

For more information refer to Chapter 7, “Configuration Files.”

#### **4. Advertise services if you need to use non-standard ports**

By default, BEA Manager agents assume the following port numbers as specified by SNMP and SMUX standards:

```
snmp          161/udp
snmp-trap     162/udp
smux          199/tcp
```

The default port assignments may be sufficient for your needs. If necessary, you can define these services on other ports, or use the appropriate command-line options when starting BEA Manager agents to assign them to non-default ports.

To modify or define the service locally, add the appropriate lines in the *NT-root-directory\system32\drivers\etc\services* file. You may wish to consult your system administrator.

# Setting Up the Agent Integrator

This section assumes that you have performed the post-installation administrative tasks described in the preceding sections. Setting up the Agent Integrator involves the following additional tasks:

- 1. Indicating the managed objects that are available from “peer” SNMP agents, if any**

The peer SNMP agents can be on the same managed node (IP address) as the Agent Integrator, or they can be on remote nodes. Access to the objects managed by the peer SNMP agents is defined through `NON_SMUX_PEER` entries in the `beamgr.conf` configuration file. Each entry defines or moves a branch of the OID tree that is accessible via that agent. This task is described in Chapter 3, “Using Multiple SNMP Agents.”

- 2. Indicating the managed objects that are available through Distributed Program Interface (DPI) master agents, if any**

Since a DPI master agent speaks SNMP, it appears to the Agent Integrator as just another peer SNMP agent. Setting up access to DPI subagents is thus done the same way as setting up access to peer SNMP agents, as described in step 1.

- 3. Modifying the management scope of SMUX subagents, if desired**

You can modify a SMUX subagent’s management scope — to avoid conflicts with other agents, for example — by specifying `OID_CLASS` entries in the `beamgr.conf` configuration file. By default, a SMUX subagent automatically indicates which section of the OID tree it is responsible for when it registers with the Agent Integrator master agent. The syntax for `OID_CLASS` entries is defined in Chapter 7, “Configuration Files.”

- 4. Defining local polling rules and the actions to be taken by the Agent Integrator if user-defined thresholds are crossed**

This task is necessary only if you want to use the Agent Integrator to offload polling from the management station. Polling rules are defined through `RULE_ACTION` entries in the `beamgr.conf` configuration file. Polling is automatically active when the Agent Integrator starts. Agent Integrator local polling can be de-activated or re-activated from a management station using SNMP SET commands. Agent Integrator polling rules, and how to start and stop polling, are described in Chapter 4, “Using Agent Integrator for Polling.”

## 5. Configuring your SNMP management system for Agent Integrator

This involves two steps:

### a. Loading the BEA MIB into the management system

The BEA MIB definitions are contained in the file `bea.asn1`, included with the Agent Integrator. By default, this file is installed in the `installation_directory/etc` directory. This MIB must be compiled into the management database of your management platform. Some management platforms refer to this process as importing or loading a MIB. The exact set of steps required varies depending upon the management system used. Consult your management system documentation for specific instructions.

### b. Configuring the management system for Agent Integrator traps

Some configuration will be required on your SNMP-compliant management system to make use of SNMP trap notifications that are generated by Agent Integrator. The exact set of steps you need to perform vary depending upon which management system you are using. Typically some configuration or mapping is required to get the management system to perform a desired action (such as turning an icon red) when a trap is received. Consult your management system documentation for specific instructions.

## 6. Modify other entries in the configuration file (if desired)

You may need to modify the following fields in the BEA Manager configuration file (`beamgr.conf`):

- ◆ `SYS_DESCR`
- ◆ `SYS_INSTALL`
- ◆ `SYS_CONTACT`
- ◆ `SYS_NAME`
- ◆ `SYS_LOCATION`
- ◆ `SYS_SERVICES`

These entries correspond to MIB objects in the `beaSystem` MIB group supported by the `nt_snmpd` subagent on Windows NT systems and supported by the `unix_snmpd` subagent on UNIX systems. For information on these objects, refer to the Chapter 3, “Using Multiple SNMP Agents.”

# Starting and Stopping the Agents

If you are using `tux_snmpd` or `m3_snmpd` as a subagent to manage TUXEDO 6.3 or 6.4 applications, you need to configure the Agent Integrator timeout to at least 30 seconds. This can be done by adding a `BEA_PEER_MAX_WAIT` entry to the BEA Manager configuration file (`beamgr.conf`) as follows:

```
BEA_PEER_MAX_WAIT 30
```

Alternatively, you could set this timeout value by setting the environment variable `BEA_PEER_MAX_WAIT` to 30. For C shell on UNIX platforms, use the following command:

```
# setenv BEA_PEER_MAX_WAIT 30
```

**Note:** All `NON_SMUX_PEERS` (if any) should be started before the Agent Integrator.

## Starting the Agents on UNIX Systems

To start the Agent Integrator and subagents, log in as root and start the following programs in the specified order:

```
# snmp_integrator
# unix_snmpd
```

**Note:** Users of legacy products from earlier versions of BEA Manager may also wish to start the `em_snmpd` subagent. In addition, other SMUX agents may now be started. For more information refer to Chapter 5, “Agent Integrator Commands,” and Chapter 6, “Starting the Subagents.”



## Starting the Agents on Windows NT Systems

Start the agents from the Windows NT service panel:

1. The Agent Integrator is installed as a Windows NT service (`snmp_integrator`).
2. A system subagent is installed as a Windows NT service (`nt_snmpd`).

The Integrator and agents must be started from the Services control panel. Expose the Windows TaskBar and select the **Start** Button. Select Settings from the menu, and open the Control Panel. Double-click on the Services applet. (In other words, select TaskBar→Start→Settings→Control Panel→Services.)

Locate each of the installed services. The Agent Integrator (`snmp_integrator`) should be started before the subagents. Click **Start** to start each service. There may be a short delay as each service is initiated.

Additionally, you can now start other SMUX subagents.

## Stopping the Agents

The following command is used to stop one or more BEA Manager agents:

```
stop_agents logical_agent_name | all [logical_agent_name]
```

For example,

```
stop_agents unix_snmpd
```

For all BEA Manager agents other than `tux_snmpd` and `m3_snmpd`, the logical agent name is always the name of the executable. If you specify `all`, all BEA Manager agents (including any agents built using the BEA Manager Agent Development Kit) will be stopped.



# 3 Using Multiple SNMP Agents

The Agent Integrator enables you to:

- ◆ **Run multiple SNMP agents on a single managed node (IP address).**

All communication between the agents and the SNMP manager is handled through the Agent Integrator master agent.

- ◆ **Coordinate communication between multiple SNMP agents and a network manager through the Agent Integrator when the SNMP agents are distributed on a number of different network nodes.**

This option is useful when you want to offload polling of these agents from the management station to distributed Integrator agents (as described in Chapter 4, “Using Agent Integrator for Polling”).

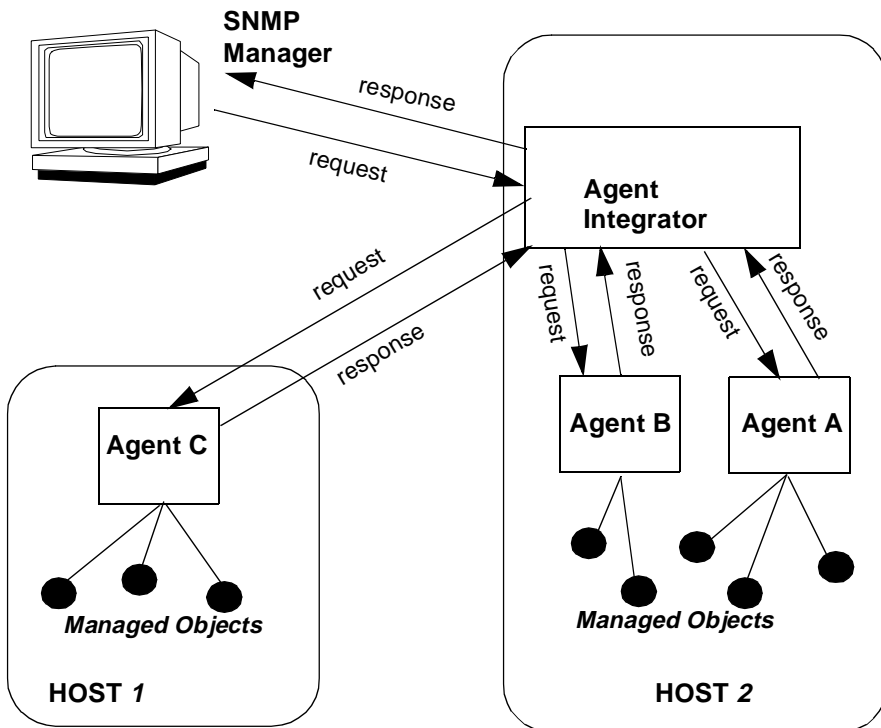
- ◆ **Manage resources through Desktop Program Interface (DPI) subagents using a DPI master agent (or manage resources through any other master agent/subagent architecture where the master agent speaks SNMP to a manager).**

The DPI master agent uses SNMP to respond to requests from network managers. Thus, the DPI master agent can be configured to communicate through the Agent Integrator in the same manner as other peer SNMP agents.

# Agent Integrator Roles

You can use the Agent Integrator to listen on UDP port 161, and handle all communications with the SNMP manager. SNMP requests received by the Integrator are fanned out to the appropriate “peer” SNMP agent (or SMUX subagent), and responses from the agents or subagents are passed on to the manager by the Agent Integrator. Thus, multiple SNMP agents (as well as SMUX subagents and other master agent/subagent architectures that use SNMP to communicate to a manager) can coexist on a single managed node. The various agents all appear to the manager as a single SNMP agent. Master agents that communicate with subagents using SMUX, DPI or other master agent/subagent architectures appear to the Agent Integrator as just another peer SNMP agent.

**Figure 3-1 Agent Integrator with Multiple SNMP Agents**



Handling manager/agent communications through the Agent Integrator has the advantage that you can offload polling of managed objects supported by the “peer” SNMP agents to the Agent Integrator, which reduces network polling traffic and management station load. The local polling capability of the Agent Integrator is described in Chapter 4, “Using Agent Integrator for Polling.”

## Configuring Agent Integrator

Each SNMP agent that is to run on the managed node with the Agent Integrator must have one or more `NON_SMUX_PEER` entry in the BEA Manager configuration file (`beamgr.conf`). The syntax of `NON_SMUX_PEER` entries is described in Chapter 7, “Configuration Files.”

Each `NON_SMUX_PEER` entry lists the port that the Agent Integrator is to use in communicating with the SNMP “peer” agent. When the agent is started, it must be configured to listen on the port assigned to it in the `NON_SMUX_PEER` entries for that agent.

**Note:** If an SNMP agent can only listen on default port 161, and has no ability to be reconfigured to listen on other ports, then any `NON_SMUX_PEER` entry for that agent must list 161 as its assigned port. In this case, that agent must be started before the Agent Integrator is started.

## Integrator Access to Managed Objects

Agent Integrator access to the managed objects that each agent is responsible for is defined through the `NON_SMUX_PEER` entries. Each `NON_SMUX_PEER` entry for an agent lists a branch of the OID tree that the SNMP agent is to be responsible for. If agent A is listed as responsible for a certain branch of the OID tree, then management requests for objects within that branch will be passed on to agent A by the Integrator.

For example:

```
NON_SMUX_PEER 2001 snmp .1.3.6.1.2.1.1,ro
NON_SMUX_PEER 2002 squid .1.3.6.1.4.1.141 .1.3.6.1.4.1.145
NON_SMUX_PEER 161 * .1.3.6.1.4.1.140 .1.3.6.1.4.1.145
```

The first entry tells the Agent Integrator to look for an SNMP agent at port 2001. All the requests from BEA Agent Integrator to this SNMP agent will use `snmp` as the community. The agent supports the subtree `.1.3.6.1.2.1.1`, and is available for read-only commands.

The second entry tells the Agent Integrator to look for an SNMP agent at port 2002. All the requests from BEA Agent Integrator to this SNMP agent will use `squid` as the community. The agent supports the subtrees `.1.3.6.1.4.1.141` and `.1.3.6.1.4.1.145`. Since no access option is specified, both subtrees default to read-write.

The third entry lists an agent at port 161. The asterisk means that the Agent Integrator will use the community string supplied by the management station. The agent supports two subtrees: `.1.3.6.1.4.1.140` and `.1.3.6.1.4.1.145`. The subtree entries list no access information, so access defaults to read-write.

**Note:** The SMUX protocol provides that SMUX subagents automatically register the sections of the OID tree that they support with the master agent. Hence, it is not necessary to add specific configuration file entries to specify which sections of the OID tree are accessible from the SMUX subagents. However, this default behavior can be overridden using a configuration file `OID_CLASS` entry. For more information about the `OID_CLASS` entry, refer to Chapter 7, “Configuration Files.”

## Assigning Priority for Conflicting Agents

If two agents, A and B, conflict in the portions of the OID tree that they are responsible for, then the `NON_SMUX_PEER` entries that define these responsibilities must assign a distinct priority to the two agents. The Agent Integrator thus refers requests for objects in the overlapping area to the agent with the lowest priority number. (The lower the priority number, the higher the priority.) For example:

```
NON_SMUX_PEER 2008 * .1.3.6.1.2.1.4,rw,8
NON_SMUX_PEER 2009 * .1.3.6.1.2.1.4,ro,5
```

In this example, the agents on port 2008 and port 2009 both supports the MIB-II `ip` group. Thus, both agents support the `ipAddrTable` (object `.1.3.6.1.2.1.4.20`). Since the agent at port 2009 has a higher priority (5 is a higher priority than 8), the Agent Integrator calls it for management requests for the `ipAddrTable`. Notice that this entry specifies read-only access. The other entry specifies read-write access, but since it has a lower priority, it is completely ignored for `ipAddrTable` requests.

The syntax of the `NON_SMUX_PEER` configuration file entry is described in detail, with examples, in Chapter 7, “Configuration Files.”

## SNMP Agents on Multiple Nodes

The Agent Integrator can be used as a proxy agent for the management station, conducting polling of SNMP agents on a number of machines, and sending an enterprise-specific trap when a user-defined condition is encountered. This is useful when resources of a single distributed system are spread over a number of machines.

From the perspective of the Agent Integrator, the resources managed by these multiple agents are viewed as if they were on a single machine. The polling capability of Agent Integrator is described in Chapter 4, “Using Agent Integrator for Polling.” Polling of agents on multiple machines assumes that `NON_SMUX_PEER` entries have been defined for these SNMP agents in the BEA Manager configuration file (`beamgr.conf`). The following is an example where the Integrator is used as proxy agent for communication with SNMP agents in a single subnet:

```
NON_SMUX_PEER 206.189.39.86.161 seahorse .1.3.6.1.2.1.4,rw,8
NON_SMUX_PEER 206.189.39.204.161 * \
.1.3.6.1.2.1.4.20,ro,5 .1.3.6.1.2.1.2
```

In this example, the SNMP agent on machine `206.189.39.86` communicates with the Integrator on port 161, uses a community string of `seahorse`, and is responsible for the MIB-II `ip` group. The SNMP agent on machine `206.189.39.204` also communicates with the Agent Integrator on port 161, using the community string passed from the SNMP manager.

The machine at IP address `206.189.39.204` is responsible for the SNMP interfaces group (`.1.3.6.1.2.1.2`) as well as the `ipAddrTable` (`.1.3.6.1.2.1.4.20`). The machine at IP address `206.189.39.86` is responsible for the MIB-II `ip` group which includes the `ipAddrTable`. Even though these agents are on physically distinct network nodes, there is still a conflict in the responsibilities of these two agents, as far as the Integrator is concerned, since the Integrator views the resources managed by the agents specified in its configuration file as if they were all on a single machine. Thus, the Integrator will only consult the machine at `206.189.39.204` for management requests for the `ipAddrTable` since the priority number for this entry is lower than the priority number for the machine at `206.189.39.86`. All other `ip` group requests, and requests for the interfaces group, will be sent to `206.189.39.86`, however.

This feature of the Agent Integrator is particularly useful when different functions of a distributed system are located on different machines, each with its own SNMP agent. The limitation to non-overlapping OID tree branches should not be a significant problem when different managed resources are located on the distinct nodes. If the same type of managed resource is located on multiple machines, then multiple Agent Integrators can be used to manage these resources.





# 4 Using Agent Integrator for Polling

To track faults in critical system components or applications, management systems use polling to determine whether attributes of the managed resource have crossed some significant threshold. Polling consists in checking the value of an attribute of the managed resource at some interval. The BEA Agent Integrator can be configured to act as a proxy for the manager, doing the polling locally on the managed node. By off-loading polling to distributed Integrator agents, the load on the management station is reduced and less network bandwidth is consumed. Communication between the manager and Agent Integrator occurs only when the manager sends a SET request to activate or de-activate the polling, or when the Agent Integrator sends an SNMP trap if it detects the specified event in the managed resource.

## Procedure for Setting Up Local Polling

The steps in using Agent Integrator for local polling can be summarized as follows:

- 1. Decide which resources you want to monitor.**

The attributes of the resource that you want to monitor must be defined as MIB objects. These MIB objects must be supported by an agent or subagent that has been installed on the managed node.

- 2. Make the managed resource accessible to the Agent Integrator.**

The Agent Integrator must know how to access the managed object. This means the object identifier for that object must lie within branches of the OID tree that

are known to the Agent Integrator. If the managed object you want to monitor is supported by a SMUX subagent that has been installed on the managed node, the subagent automatically registers its section of the OID tree with the Agent Integrator when the subagent is started. This can be modified using `OID_CLASS` entries in the BEA Manager configuration file, as described in Chapter 7, “Configuration Files.” For peer SNMP agents (or DPI or SMUX master agents), you must define the segments of the OID tree supported by those agents in `NON_SMUX_PEER` entries in the BEA Manager configuration file. This is described in the section “Integrator Access to Managed Objects” in Chapter 3, “Using Multiple SNMP Agents.” The Agent Integrator directly supports the following MIB groups: MIB II `system` and `snmp` groups, the SMUX MIB, and the BEA Manager `beaintAgtTable` in the BEA Manager agent MIB. Additional MIB groups are supported by the `unix_snmpd` and `nt_snmpd` SMUX subagents, shipped with Agent Integrator. These are listed in Chapter 6, “Starting the Subagents.”

### 3. Define polling instructions for the Agent Integrator

We can divide this task into two main subtasks:

- ◆ **Define the desired threshold.**
- ◆ **Specify the action to take if the threshold is crossed.**

Each polling instruction for the SNMP Integrator is called a *rule*. Rules are defined under the `RULE_ACTION` entry in the BEA Manager configuration file, `beamgr.conf`. You can use your favorite text editor to modify this file. (For information on how to create rules, see the section “Creating New Polling Rules.”) Rules are explained below under “Introduction to Agent Integrator Rules.” Chapter 7, “Configuration Files,” provides the complete syntax for the `RULE_ACTION` entries.

### 4. Configure your SNMP management system for Agent Integrator traps

When a polling threshold is crossed, Agent Integrator sends an enterprise-specific SNMP trap notification to the destinations specified by the `TRAP_HOST` entries in the BEA Manager configuration file. Some configuration will be required on your SNMP-compliant management system to make use of the traps that are thus generated. The exact set of steps you need to perform vary depending upon which management system you are using. Typically some configuration or mapping is required to get the management system to perform a desired action (such as turning an icon red) when a trap is received. Consult your management system documentation for specific instructions.

**5. Start Agent Integrator polling.**

The Agent Integrator begins executing all valid polling rules when it is started. Refer to the section “Starting and Stopping Polling” for more details.

**6. De-activate or re-activate Agent Integrator polling, when desired.**

Polling rules are available as MIB objects; thus an operator can de-activate or re-activate polling from the management station by means of an SNMP SET request. This is described in the section “Starting and Stopping Polling.”

## Introduction to Agent Integrator Rules

An Agent Integrator rule consists of the following parts:

- ◆ A unique name for the rule (no more than eight characters in length)
- ◆ A *condition* (or threshold) that the Integrator is to check for. (This is described further in the section, “Conditions.”)
- ◆ An *action* to take if the specified threshold is crossed. (This is described in the section “States and Transitions.”)
- ◆ A *polling frequency* (specified in seconds), that is, the time delay between each access of the specified object value

## Conditions

When the Agent Integrator polls, it checks to determine if a specified condition holds. A *condition* is defined as a relationship between an object (specified by its object identifier) and a value. (Object identifiers are described in Chapter 1, “Agent Integrator Overview.” Polling is described in “Polling.”)

## Relations for Defining Conditions

The condition obtains (the threshold is crossed) if and only if the specified relation holds between the object and the value. For example, the relation greater than defines the following condition:

disk capacity in use greater than 90 percent

In this case, the condition holds (evaluates to true) if the object (percentage of disk capacity in use) has a value that is greater than 90. (In this example, the condition is described in English, not the actual code used to define Agent Integrator polling rules.)

Any of the relations listed in the following table can be used to define conditions.

**Table 4-1 Relations for Defining Conditions**

Symbol	Meaning
==	is identical to
!=	is not identical to
<	is less than (for numeric values) is a substring of (for strings)
>	is greater than (for numeric values) contains (for strings)
<=	is less than or equal to
>=	is greater than or equal to

## Polling with a SMUX Subagent

For example, suppose that we want the Agent Integrator to check if CPU usage has exceeded 80 percent. This feature of the CPU is represented by the `beaSysPerfCpu` object in the `beaSysPerf` group. This MIB group is supported by the `unix_snmpd` subagent, which is shipped with the Agent Integrator. This subagent uses SNMP Multiplex (SMUX) protocol to talk to the Agent Integrator. Thus, the Agent Integrator can obtain the value of this object from the `unix_snmpd` subagent if it is running on the same machine. We could use the following condition to define a polling rule for the Agent Integrator:

```
(VAL(.1.3.6.1.4.1.140.11.1.0) > 80)
```

The expression `VAL()` is used to obtain the value of the `beaSysPerfCpu` object. The specified condition obtains if the percentage of CPU capacity in use exceeds 80 percent. In this example, the initial dot indicates that this is an absolute OID, that is, the path to the `beaSysPerfCpu` object is defined from the root of the OID tree. The actual OID for the `beaSysPerfCpu` object is `.1.3.6.1.4.1.140.11.1`. However, when retrieving the value, it is necessary to specify the *instance* of the object to be retrieved. The last numeral, 0, is the instance index. Because `beaSysPerfCPU` is a scalar object — an object that can have only one instance — an index of zero is specified in this case. (How to specify non-scalar objects is discussed in the section “Instance Indexes.”)

The following is an example of an Agent Integrator rule that uses the condition previously specified:

```
RULE_ACTION checkcpu 600 \  
if (VAL(.1.3.6.1.4.1.140.11.1.0) > 80) {TRAPID_ERR = 200}
```

In this example, `checkcpu` is the name of the rule. The Agent Integrator checks the CPU usage every ten minutes. If the value of `beaSysPerfCpu` is greater than 80 percent, `TRAPID_ERR = 200` instructs the Agent Integrator to generate an enterprise-specific trap with a specific-trap type number of 200. This type number can be used by a system administrator to identify the cause of the trap.

**Note:** The MIB objects whose values the Agent Integrator can obtain depends on the MIB objects supported by the agents or subagents that the Agent Integrator is managing. In the previous example, the Agent Integrator can poll for the `beaSysPerfCpu` object value only if the `unix_snmpd` subagent is running on the managed node. The MIB objects that Agent Integrator can access through a “peer” SNMP agent depends on the `NON_SMUX_PEER` entries in the BEA Manager configuration file, as explained in Chapter 3, “Using Multiple SNMP Agents.”

## Polling with SNMP Peer Agents

The Agent Integrator can also obtain MIB object values from SNMP “peer” agents on either the same machine or other machines in the network. For example, suppose that we have a peer SNMP agent that supports the MIB II interfaces group. If so, we might want the Integrator to check if a physical interface is not operational. This feature of the interface is represented by the `ifOperStatus` object in the `ifTable` in the MIB II interfaces group. In this case, we want to know whether the value of `ifOperStatus` is not equal to 1. (An interface is operational if its `ifOperStatus` value is 1.) If we want to check the `ifOperStatus` value for the first interface on the machine, we could use the following condition:

```
(VAL(.1.3.6.1.2.1.2.2.1.8.1) != 1)
```

This condition holds if and only if the first interface in the `ifTable` is operational. The last numeral, 1, specifies the instance index — the first interface entry in the table.

If the condition is satisfied, we want the Agent Integrator to take some action. For example, if the `ifOperStatus` value for an interface is not 1 (i.e., the interface is not up), we might want the Agent Integrator to notify the management station. To do this, we can specify that the Agent Integrator send an enterprise-specific SNMP trap to the management station with a special specific-trap value, which identifies the cause of the trap to the systems administrator.

Instead of requesting this notification if a specific interface (such as the first one in the `ifTable`) is down, we might want to be notified if *any* of the interfaces is down.

Here is an example of a rule entry that would do this:

```
RULE_ACTION checkIf 120 \  
if (VAL(.1.3.6.1.2.1.2.2.1.8.*) != 1) {TRAPID_ERR=300}
```

In this example, `checkIf` is a name we have given to this particular rule. We have indicated that Agent Integrator should check the interface every two minutes (120). By using the asterisk wildcard for the instance index, the condition will be satisfied if any interface in the `ifTable` has an `ifOperStatus` not equal to 1, that is, all instances will be checked. If the value of the OID is not equal to 1 (the interface is not up) for any instance, an enterprise-specific trap is sent with a specific trap ID of 300.

**Note:** This rule only causes a trap to be generated when Agent Integrator first detects that an interface is down. If the interface continues to be down, it does not generate additional traps.

## Use of Logical Operators in Conditions

Conditions are of two types, simple and complex. A *simple* condition consists of a relation between a managed object and a value. All of the examples in the previous sections have been simple conditions.

You can use the logical operators AND, OR, and NOT to define *complex* conditions. For example, if A and B are two simple conditions, you can specify a complex condition that consists of *both* A and B occurring. The symbols listed in the following table can be used to define complex conditions.

**Table 4-2 Logical Operators for Specifying Complex Conditions**

Symbol	Meaning
<code>!(condition_A)</code>	Logical negation. The threshold is crossed if and only if <i>condition_A</i> does not hold.
<code>(condition_A    condition_B)</code>	Logical disjunction. The threshold is crossed if and only if either <i>condition_A</i> or <i>condition_B</i> obtain.
<code>(condition_A &amp;&amp; condition_B)</code>	Logical conjunction. The threshold is crossed if and only if both <i>condition_A</i> and <i>condition_B</i> obtain.

### SCENARIO FOR USING A COMPLEX CONDITION

For example, we might not want the Agent Integrator to send an alarm when `ifOperStatus` is not up for an interface if a system administrator has taken that interface down for repair. In that case, we could define a rule that asks the Agent Integrator to determine if two conditions hold: `ifOperStatus` is not up AND `ifAdminStatus` is up. In other words, we want to be notified if the interface should be up but is not.

**Note:** The MIB objects whose values the Agent Integrator can obtain depends on the MIB objects supported by the agents or subagents that the Agent Integrator is managing.

### SAMPLE CODE FOR THIS SCENARIO

To do this, we might modify our `checkIf` rule as follows:

```
RULE_ACTION checkIf 60\  
if ((VAL(.1.3.6.1.2.1.2.2.1.8.*) != 1) && \  
(VAL(.1.3.6.1.2.1.2.2.1.7.*) == 1)) \  
{TRAPID_ERR=301}
```

### HOW THIS RULE WORKS

In this example, the Agent Integrator checks the interfaces every minute (60) and generates an enterprise-specific trap, with a specific trap value of 301, if any of the interfaces is not up (`ifOperStatus` not equal to 1) but has an `ifAdminStatus` value of up (i.e., the interface should be up but it is not).

**Note:** This rule causes this trap to be generated only when the condition first evaluates to true. As long as the interface continues in the same state, a new trap is not generated.

## Data Types for Defining Conditions

The syntax for a simple condition is as follows:

```
(VAL(oid) relation value)
```

where

*relation*

Is one of the relations described in Table 4-2.

*oid*

Is specified in one of the formats described in the section “Specifying Object Identifiers in Conditions.”

*value*

Can be one of the following data types:

- ◆ integer
- ◆ string
- ◆ IP address (in the form *number1.number2.number3.number4*)
- ◆ Object identifier, surrounded by single quotes (‘). The OID should be specified exactly as returned by the agent managing that object.



## Specifying Object Identifiers in Conditions

In defining polling conditions, the object identifier (OID) must be specified numerically, not using textual symbols (other than `mib-2` or `enterprises` as indicated in the following list). One of the following formats can be used to specify the object identifier:

- ◆ An absolute object identifier, that is, the full path to the object is specified from the root of the OID tree. An initial dot is used to indicate that the path starts at root, (for example, `.1.3.6.1.2.1.1.1.0`). Note that the trailing zero in this example is the instance index.
- ◆ A relative OID under the MIB II branch can be specified in the form:

`mib-2.number.number ...`

When the reserved word `mib-2` appears as the leading sub-oid, `.1.3.6.1.2.1` is assumed to be prefixed to the rest of OID. For example:

`mib-2.1.1.0`

represents the absolute OID:

`.1.3.6.1.2.1.1.1.0`

- ◆ A relative OID under the enterprises branch can be specified in the form:

`enterprises.number.number ...`

When the reserved word `enterprises` appears as the leading sub-oid, `.1.3.6.1.4.1` is assumed to be prefixed to the rest of OID. For example:

`enterprises.140.1.0`

represents the absolute OID:

`.1.3.6.1.4.1.140.1.0`

- ◆ A relative OID under the enterprises branch can also be specified in purely numeric form:

`number.number.number ... ,`

If there is no leading “.” and the OID starts with a number, `.1.3.6.1.4.1` is assumed to be prefixed to the rest of OID. For example:

`140.1.1.0`

represents the absolute OID:

`.1.3.6.1.4.1.140.1.1.0`

### INSTANCE INDEXES

*Columnar* objects are used to represent a column of a tabular MIB group. Columnar objects accordingly can have multiple instances. To specify an instance, the index is appended to the rest of the OID. If the index is a single attribute, the last number in an OID is used to specify the particular instance. If the more than one attribute is required to uniquely identify an instance, an instance number for each attribute is appended to the OID, separated by a dot, in the order specified by the INDEX definition in the ASN.1 file.

For example, suppose that you want to check for the condition when the state of a particular server is anything but active. To uniquely specify a server instance, we require both the group number and the server ID. The INDEX entry for `tuxTsrvrTbl` in the ASN.1 file specifies the following as an INDEX to particular instances.

```
INDEX (tuxTsrvrGrpNo, tuxTsrvrId)
```

The relative OID for `tuxTsrvrState` is the following:

```
140.300.20.1.1.5
```

Thus, to specify the particular server instance for group 55 and server ID 3, you use the following OID:

```
140.300.20.1.1.5.55.3
```

Note that the order of the two attribute instances added to the `tuxTsrvrState` OID is indicated by the INDEX definition above: `tuxTsrvrGrpNo` followed by `tuxTsrvrId`.

You can thus define the condition that you want to check as follows:

```
VAL(140.300.20.1.1.5.55.3) != 1
```

This condition will evaluate to true whenever this particular server instance is not active.

A specific number can be used to specify a particular instance or the asterisk wildcard can be used to specify all instances. Zero is used as the instance index in the case of *scalar* objects (objects that can have only one instance). The asterisk wildcard is only used to represent all instances of a columnar object. For example:

```
.1.3.6.1.4.1.140.1.1.0
```

specifies the single instance of a scalar object while:

```
.1.3.6.1.4.1.140.2.22.1.2.*
```

specifies all of the instances of a columnar object. When a wildcard is used to define a condition, the condition will be satisfied if *any* instance satisfies the condition.

## NOTE

- ◆ When specifying multiple OIDs in a complex rule, the OIDs should either all use wildcards or none should. That is, you should not combine an OID that specifies a particular instance with an OID that uses a wildcard in the same rule.
- ◆ When multiple OIDs with wildcards are used in a single rule, all of the OIDs should specify objects only within the same table.
- ◆ In complex conditions, when using asterisk (\*) as the indices for the OIDs, the condition is checked between columns of the same row, for all rows available in that table.
- ◆ When a wildcard is used to define a condition, the condition is satisfied if any instance satisfies the condition. For example,

```
VAL(.1.3.6.1.2.1.2.1.2.2.1.8.*) == 1
```

is satisfied if 1 is the value of any instance. Once the condition is satisfied, the rule is in the ERR state. The rule remains in the ERR state as long as any instance satisfies the condition. The rule transitions to the OK state only when no instance satisfies the rule.

## States and Transitions

Associated with each active polling rule is a *state*. There are two possible states for an active rule:

OK — A rule is in the OK state when the specified condition does not hold (threshold is not crossed).

ERR — A rule is in the ERR state when the specified condition does hold (threshold is crossed).

A *transition* takes place when the value obtained from a poll of an object results in the state of a rule changing, either from OK to ERR or from ERR to OK. Transitions determine when an action is to be taken in response to a poll, as described in the following section. That is, Agent Integrator polling rules execute an action (such as generating a trap notification) only when a polling rule undergoes a transition from one state to a different state.

When the Agent Integrator begins executing a polling rule, the rule is initially in the OK state. As long as the threshold is not crossed, the rule remains in the OK state. If the threshold is crossed, the rule undergoes a transition from the OK state to the ERR state. As long as the condition continues to evaluate as true, the rule remains in the ERR state. If the condition subsequently evaluates to false, the rule then transitions back to the OK state. Thus, there are two types of transition:

- ◆ A transition from the OK state to the ERR state
- ◆ A transition from the ERR state to the OK state

**Note:** When conditions are defined for all instances of a columnar object using a wildcard (“\*”), the rule will transition from the OK state to the ERR state if the column in any row of the table evaluates to true for the defined condition. The rule will transition to OK if the condition evaluates to false for the column in all rows of the table.

## Actions

An Agent Integrator rule can specify an action to be taken if the polling rule undergoes one of these transitions. Two different types of action can be specified:

- ◆ Agent Integrator can generate an enterprise-specific trap with a user-specified specific trap number.
- ◆ Agent Integrator can execute a specified program or script (or batch file).

Both types of action can be specified in the same rule.

**Note:** Agent Integrator carries out an action *only* when a transition occurs. Continued polling does not result in duplicate actions as long as the rule remains in the same state. This prevents duplicate traps from being generated in response to detection of a single event.

Four keywords are used to define actions:

`TRAPID_ERR = specific-trap-number`

Indicates that a trap should be sent if the state of the rule transitions from OK to ERR.

`TRAPID_OK = specific-trap-number`

Indicates that a trap should be sent if the state of the rule transitions from ERR to OK.

`COMMAND_ERR = "command"`

The program specified by *command* is executed if the state of the rule transitions from OK to ERR.

`COMMAND_OK = "command"`

The program specified by *command* is executed if the state of the rule transitions from ERR to OK.

**Note:** The string specifying the command to be executed must be in quotes. For example:

```
COMMAND_ERR = "usr/mybin/test.ksh"
```

If you do not specify the absolute path to the executable or script, the path should be specified in the Agent Integrator's environment settings.

These statements specifying actions must be placed within curly braces. When multiple commands are specified in a rule, the commands must be separated by spaces. *command* must be enclosed in quotes.

A string containing the name of the rule and the direction of the state transition (OK to ERR or ERR to OK) is passed as an argument to the script or program called by the `COMMAND_ERR` or `COMMAND_OK` actions.

### Trap Information

The following information is passed in the enterprise-specific traps generated by Integrator polling rules:

- ◆ User-defined specific trap number

- ◆ Rule name and state change

A `TRAPID_ERR` action passes a string in the variable bindings of the trap that takes the following form:

Rule id *rule-name* has triggered from OK to ERR state

A `TRAPID_OK` action passes a string in the variable bindings of the trap that takes the following form:

Rule id *rule-name* has triggered from ERR to OK state

- ◆ Enterprise ID

When a threshold is crossed, the Agent Integrator generates an SNMP trap packet (PDU) that has the following enterprise OID in its enterprise field:

.1.3.6.1.4.1.140.1.1

**Note:** Agent Integrator polling alarms have a different enterprise identifier in the enterprise field of the trap than the Agent Connection traps that forward TUXEDO or M3 system events. Agent Integrator polling alarms use `bea` as an enterprise identifier whereas Agent Connection TUXEDO system traps use `tuxedo` as the enterprise identifier.

## Examples

In the following example, the Agent Integrator polls every ten minutes (600) to determine if disk capacity in use is greater than 90 percent. If any file system has more than 90 percent capacity in use, an enterprise-specific trap with number 102 is generated. If subsequently all the file systems have less than or equal to 90 percent of capacity in use, an enterprise-specific trap with trap number 202 is generated.

```
RULE_ACTION diskchk 600 \  
if (VAL(140.2.22.1.5.*) > 90) {TRAPID_ERR = 102 TRAPID_OK = 202}
```

In the next example, a TUXEDO application is checked to determine if the transaction triptime exceeds 36 mSec. If the threshold is crossed, an enterprise-specific trap is generated and a user script, logtime, is invoked to log the time of the event. If the triptime is subsequently less than 36 mSec after having crossed that threshold on the previous poll, an enterprise-specific trap with a number of 302 is generated.

```
RULE_ACTION triptime 20 \  
if (VAL(140.150.1.3.*) > 35) \  
{TRAPID_ERR = 301 TRAPID_OK = 302 \  
COMMAND_ERR = "/usr/sbin/logtime"}
```

**Note:** The object identifier in this example is not defined in the BEA MIB. This is an example of an object that might be defined in a user-supplied custom MIB.

In the next example, Agent Integrator polls every five seconds to check whether the number of requests completed by the TUXEDO server `Server1` is greater than six. If it is, an enterprise-specific trap is generated with a specific trap number of 210 and the command `c:/etc/srv_reqs.cmd` is executed.

```
RULE_ACTION Server1 5 \  
if ((VAL(140.300.20.2.1.12.*) > 6)) \  
{ TRAPID_ERR=210  COMMAND_ERR="c:/etc/srv_reqs.cmd" }
```

In the next example, Agent Integrator is checking a particular server instance in any state other than active. The server that is being checked is uniquely identified by its group number and server ID: group number 55 and server ID 3.

```
RULE_ACTION srvrUp 60 if (VAL(140.300.20.1.1.5.55.3) != 1 \  
                        {TRAPID_ERR = 306 TRAPID_OK = 307}
```

Whenever the server satisfies the condition, the rule transitions to the ERR state and generates an enterprise-specific trap with the specific trap number of 306. Whenever the server becomes active again, it transitions back to the OK state and issues a trap with the specific trap number of 307.

# Starting and Stopping Polling

Polling rules are defined as `RULE_ACTION` entries in the BEA Manager configuration file, `beamgr.conf`. The default location of this file is `/etc` on UNIX machines or `C:\etc` on Windows NT machines. Individual rules are MIB objects, stored as an entry (row) in the `beaIntAgtTable`.

The status of each rule entry determines whether the Agent Integrator will execute that rule, that is, actively check the condition specified in the rule. The status of each rule entry is stored in the `beaIntAgtStatus` object. Polling is active for a rule if the status of that rule is valid (integer value of 1). Polling is inactive for a rule if its status has been set to inactive (integer value of 3). The specific rule can be SET from a management station (such as OpenView or SunNet Manager) by using the unique name of the rule as the key field used to specify the entry instance (row).

**Note:** The Agent Integrator must be running in order to successfully SET objects in the `beaIntAgtTable`.

The Agent Integrator begins executing all polling rules defined in `RULE_ACTION` entries in the BEA Manager configuration file (`beamgr.conf`) when it first starts up. The status of each rule object in the `beaIntAgtTable` is valid at startup.

## Creating New Polling Rules

Rules can be added to the configuration file in two ways:

- ◆ Use a text editor, such as `vi`, to add a `RULE_ACTION` entry to file, taking care to conform to the syntax of the rule, as described in Chapter 7, “Configuration Files.” However, if the Integrator is already running, the new rule will not take effect until you execute the following command:

```
reinit_agents snmp_integrator
```

This causes the Agent Integrator to re-read its configuration file.

- ◆ Since individual rules are MIB objects, stored as an entry (row) in the `beaIntAgtTable`, you can use an SNMP manager (or the `snmptest` utility) to



create a new entry (row) in the `beaIntMgtTable`. (The SNMP manager must have the ability to issue SNMP SET requests that contain multiple objects in a single SET request.) To create the new row, issue a SET request after specifying a new index value that does not already exist in the table. This causes a new `RULE_ACTION` entry to be created in the configuration file.

## Deleting or Modifying Polling Rules

Agent Integrator polling rules can be modified in the same two ways they can be created:

- ◆ Using a text editor to delete (or comment out) or modify a `RULE_ACTION` entry in the `beamgr.conf` file. This change does not take effect unless you issue the following command to force the Agent Integrator to re-read its configuration file:

```
reinit_agents snmp_integrator
```

- ◆ SNMP SET commands can also be used to delete or modify rules.

## Stopping Agent Integrator Polling Activity

Polling can be de-activated in one of two ways:

- ◆ **Remove the `RULE_ACTION` entry in the configuration file.**

You can turn off a polling rule by commenting out or deleting that `RULE_ACTION` entry in the BEA Manager configuration file (`beamgr.conf`). However, for this to take effect, you will need to execute `reinit_integrator`, which causes the Agent Integrator to re-read its configuration file.

- ◆ **Use `snmpset` or an SNMP-compliant manager to SET the value of the rule status to inactive.**

Polling for that rule can be de-activated from the management station (or by using the `snmpset` utility packaged with the Agent Integrator) by setting the value of that object to inactive (an integer value of 3). Setting the value to 2 (invalid) causes the `RULE_ACTION` entry to be deleted from the configuration file. Figure 4-1 shows our rule `diskchk`, discussed earlier, being set to inactive. Note that the read/write community string (in this example, “iview”) is required for SET permission.

**Figure 4-1** Setting a Polling Rule to Inactive

The image shows a window titled "SunNet Manager - Set : diamond". At the top are three buttons: "Get", "Set", and "Unset". Below these are four dropdown menus: "Agent" set to "BEA-MIB", "Group" set to "beaIntAgtTable", "Key" set to "7.100.105.115.107.99.104.107", and "Options" set to "iview". Below the dropdowns is a table with three columns: "Attribute Name", "Current Value", and "New Value".

Attribute Name	Current Value	New Value
beaIntAgtRuleId	diskchk	
beaIntAgtScanIntvl	600	
<input checked="" type="checkbox"/> tRuleAction	<input checked="" type="checkbox"/> = 102 TRAPID_OK <input type="checkbox"/>	
beaIntAgtStatus	valid	<input type="checkbox"/> inactive

## Restarting Agent Integrator Polling Activity

When a polling rule has been de-activated using a SET request from a management station, the rule can be re-activated using a SET request to set the value of the corresponding `beaIntAgtStatus` object to valid (integer value of 1).

# 5 Agent Integrator Commands

This chapter explains the commands for using the Agent Integrator.

## reinit\_agents

### Syntax

```
reinit_agents all | logical_agent_name [logical_agent_name]
```

### Description

This utility causes the specified agents to re-read their configuration file. This utility must be run with root permissions. Using `all` causes all BEA Manager agents, including any custom agents built using the Agent Development Kit, to re-initialize. For all BEA Manager agents other than `tux_snmpd` or `m3_snmpd`, *logical\_agent\_name* is the name of the executable. For example, the command:

```
reinit_agents snmp_integrator
```

causes the Agent Integrator to re-read its configuration file.

# snmp\_integrator

## Syntax

```
snmp_integrator [-d] [-n] [-p port | -r smux_port] [-b ipaddr_list  
| hostname_list ]
```

## Arguments

- d**  
Causes the program to display a message for each SNMP/SMUX packet sent or received.
- n**  
The program is not run as a daemon (UNIX only).
- p *port***  
Specifies the port on which the Agent Integrator listens for SNMP requests (default: 161/udp).
- r *smux\_port***  
Specifies the port used to communicate with SMUX subagents (default: 199/tcp).
- b *ipaddr\_list* | *hostname\_list***  
If the machine where the Agent Integrator is running has multiple IP addresses, by default the Agent Integrator listens on all IP addresses. The **-b** option can be used to specify a subset of IP addresses to monitor for incoming SNMP requests.
- ipaddr\_list***  
Can consist of a single IP address or a blank-separated list of IP addresses.
- hostname\_list***  
Can consist of one hostname or a blank-separated list of hostnames.

For example, if the machine on which the Agent Integrator is running has the following IP addresses:

```
130.86.34.3
130.86.33.13
130.86.23.1
```

Agent Integrator can be configured to only service requests addressed to 130.86.23.1 by starting it with the following command:

```
snmp_integrator -b 130.86.23.1
```

## Description

`snmp_integrator` is the BEA Manager Agent Integrator executable. It allows multiple SNMP agents and SMUX subagents from any vendor to coexist on the same node and to appear as a single SNMP agent to any SNMP manager.

The Agent Integrator can simultaneously support any number of:

- ◆ SNMP agents
- ◆ SMUX subagents
- ◆ Master agents such as other SMUX or DPI master agents or any other proprietary master agents. (The master agent must respond to requests from a manager using SNMP.)

Also, the Agent Integrator can coexist on the standard SNMP port (161/udp) with any other SNMP agent. It directly supports the SMUX MIB (RFC 1227) in addition to the `system(1)` and `snmp(3)` groups of MIB II.

When the program is running as an SNMP agent, it generates a `coldStart` trap to the host specified by the `TRAP_HOST` entry in the `beamgr.conf` file at startup. If there is no `TRAP_HOST` entry, the trap is sent to port 162 on the host where the utility is running, with a community defined as `public`.

Read-write and read-only communities supported by the Integrator can be specified in the `beamgr_snmpd.conf` file. By default, read-only community is `public` and read-write community is `beamgr`.

You may configure the Agent Integrator to expect a password from SMUX subagents registering with it. For more information, refer to “BEA Manager Passwords File (`beamgr_snmpd.conf`).”

### UNIX

The `-d` argument is usually used when the program is executed on the command line for debugging purposes. Messages displayed are sent to the standard output of the program. If the program is started by `init(1M)`, the destination of these messages is determined by the UNIX platform and version. These messages are most frequently sent to the console.

The `-n` argument is usually used when the program is started by `init(1M)` with the `respawn` option.

### Windows NT

Messages displayed with the `-d` argument are sent to the NT Event Log.

The `-n` argument has no effect.

# stop\_agents

## Syntax

```
stop_agents logical_agent_name | all [logical_agent_name]
```

## Arguments

`all`

Stops all BEA Manager agents (including any agents built using the BEA Manager Agent Development Kit).

`logical_agent_name`

For all BEA Manager agents other than `tux_snmpd` and `m3_snmpd`, the logical agent name is always the name of the executable.

# 6 Starting the Subagents

The Agent Integrator software includes three subagents that can be used as either standalone SNMP agents or as SNMP Multiplex (SMUX) subagents in concert with a master agent such as the Agent Integrator. This chapter describes the use of these subagents:

`pm_snmpd`

A SMUX subagent that supports the BEA Log Central Process Monitor. Consult the *Log Central Administrator's Guide* for further information.

`unix_snmpd`

A SMUX subagent that supports `beaSystem`, `beaUnix`, `beaSmgr`, and `beaSysPerf` MIB groups.

`nt_snmpd`

Supports the `beaNTSysPerf` and `beaSystem` MIB groups. Also supports the following subgroups within the `beaUnix` group: the host process table (`beaPsTable`), the host file system table (`beaDfTable`), the system statistics group, and the local file system table (`beaLclDfTable`). Does not support the message queue (`beaMqTable`) and semaphore (`ipcs -sa`) (`beaSemTable`) tables. Also, the `beaShmTable` (UNIX shared memory table) and `beaSmgrTable` are not supported.

Refer to Chapter 8, “BEA SNMP Agent MIB,” for more information about the supported MIB objects.

# pm\_snmpd

A SMUX subagent which supports the BEA Log Central Process Monitor.

## Syntax

```
pm_snmpd [-d] [-n] [-s] [-p port | -r smux_port]
```

## Arguments

- d  
Causes the program to display a message for each SNMP/SMUX packet sent or received.
- n  
The program is not run as a daemon (UNIX only).
- s  
Causes the program to run as an SNMP agent. If this option is not specified, the program runs as a SMUX subagent.
- p *port*  
Specifies the port used to communicate with SNMP managers (default: 161). This option affects the program only when it is executed as an SNMP agent.
- r *smux\_port*  
Specifies the port used to communicate with the SMUX master agent (default: 199). This option affects the program only when it is executed as a SMUX subagent.

## Description

The `pm_snmpd` program is the BEA Manager SMUX subagent that supports the BEA Manager Log Central Process Monitor MIB groups (`beaPm` and `beaPmProcTable`). This subagent is capable of running as an SNMP agent as well as a SMUX subagent.



When the program is running as an SNMP agent, it generates a coldStart trap to the host specified by the TRAP\_HOST entry in the `beamgr.conf` file. If there is no TRAP\_HOST entry, the trap is sent to port 162 on the host where the utility is running, with a community defined as `public`.

If the program is running as an SNMP agent:

- ◆ The supported read-write and read-only communities can be specified in the `beamgr_snmpd.conf` file. By default, the read-only community is `public` and the read-write community is `beamgr`.
- ◆ It also supports the MIB-II SNMP group.

If the program is running as a SMUX subagent:

- ◆ It specifies a password to the SMUX master when it establishes communication if the environment variable `BEA_SMUX_PASSWD` is defined.
- ◆ It advertises all MIB groups it knows about: `beaPm`, `beaPmProcTable`, and `beaSmgr`. You may limit the advertised MIB groups by specifying an `OID_CLASS` entry in the `beamgr.conf` file. For more information, refer to “BEA Manager Configuration File (`beamgr.conf`).”

## UNIX

The `-d` argument is usually used when the program is executed on the command line for debugging purposes. Displayed messages are sent to the standard output of the program. If the program is started by `init`, the destination of these messages is determined by the UNIX platform and version. These messages are most frequently sent to the console.

The `-n` argument is usually used when the program is started by `init` with the `respawn` option.

## Windows NT

Messages displayed with the `-d` argument are sent to the Windows NT Event Log.

The `-n` argument has no effect.

# nt\_snmpd

## Syntax

```
nt_snmpd [-d] [-s] [-p port] [-r smux_port]
```

## Arguments

- d  
This option dumps the SNMP/SMUX packets received and sent by the agent to the Event Log.
- s  
Specifies that `nt_snmpd` runs as an SNMP agent. If this option is not specified, `nt_snmpd` runs as a SMUX subagent.
- p *port*  
This option allows running the `nt_snmpd` on a port other than the standard SNMP port 161. This option is meaningful if and only if `nt_snmpd` is running as a standalone SNMP agent.
- r *smux\_port*  
This option specifies the port to use when connecting to a SMUX master agent. (The default is port 199). This option is meaningful if and only if `nt_snmpd` is running as a SMUX subagent.

## Description

This subagent is capable of running as an SNMP agent as well as a SMUX subagent. In either case, it runs as a Windows NT service.

When the program is running as an SNMP agent, it generates a coldStart trap to the host specified by the TRAP\_HOST entry in the `beamgr.conf` file at startup. If there is no TRAP\_HOST entry, the trap is sent to port 162 on the host where the utility is running, with a community defined as `public`.

Read-write and read-only communities supported by the `nt_snmpd` (community) can be specified in the `beamgr_snmpd.conf` file. (Community is meaningful only when running as an SNMP agent.) By default, the read-only community is `public` and the read-write community is `beamgr`.

When the program is running as a SMUX subagent, it can (if desired) specify a password to the SMUX master when it establishes communication. The password used is the value of the environment variable `BEA_SMUX_PASSWD`.

This utility supports the MIB-II `snmp` group when running as an SNMP agent.

By default, when running as a SMUX subagent, `nt_snmpd` advertises all the MIBs it knows about. The `nt_snmpd` subagent supports the following MIB groups: `beaSystem`, `beaNTSysPerf`, `beaSysPerf`, and some objects under the `beaUnix` group.

You may limit the MIBs to be advertised by `nt_snmpd` by specifying an `OID_CLASS` entry in the `beamgr.conf` file. For more information, refer to “BEA Manager Configuration File (`beamgr.conf`).”

Messages displayed with the `-d` argument are sent to the NT Event Log.

# unix\_snmpd

## Syntax

```
unix_snmpd [-d] [-n] [-s] [-p port] [-r smux_port]
```

## Arguments

- d**  
This option dumps the SNMP/SMUX packets received and sent by the agent to stdout.
- n**  
The program is not run as a daemon.
- s**  
Specifies that `unix_snmpd` runs as an SNMP agent. If this option is not specified, `unix_snmpd` runs as a SMUX subagent.
- p *port***  
This option allows running the `unix_snmpd` on a port other than the standard SNMP port 161. This option is meaningful if and only if `unix_snmpd` is running as a standalone SNMP agent.
- r *smux\_port***  
This option specifies the port to use when `unix_snmpd` registers with the SMUX master agent. (The default is port 199.) This option is meaningful if and only if `unix_snmpd` is running as a SMUX subagent.

## Description

This subagent is capable of running as an SNMP agent as well as a SMUX subagent.

When the program is running as an SNMP agent, it generates a coldStart trap to the host specified by the TRAP\_HOST entry in the `beamgr.conf` file at startup. If there is no TRAP\_HOST entry, the trap is sent to port 162 on the host where the utility is running, with a community defined as `public`.

Read-write and read-only communities supported by the `unix_snmpd` (community) can be specified in the `beamgr_snmpd.conf` file. (Community is meaningful only when running as an SNMP agent.) By default, the read-only community is `public` and the read-write community is `beamgr`.

When the program is running as a SMUX subagent, it can (if desired) specify a password to the SMUX master when it establishes communication. The password used is the value of the environment variable `BEA_SMUX_PASSWD`.

This utility supports the MIB-II `snmp` group when running as an SNMP agent.

By default, when running as a SMUX subagent, `unix_snmpd` advertises all the MIBs it knows about. The `unix_snmpd` subagent supports the following MIB groups on UNIX platforms: `beaSystem`, `beaUnix`, `beaSmgr` and `beaSysPerf` groups.

You may limit the MIBs to be advertised by `unix_snmpd` by specifying an `OID_CLASS` entry in the `beamgr.conf` file. For more information, refer to “BEA Manager Configuration File (`beamgr.conf`).”

The `-d` argument is usually used when the program is executed on the command line for debugging purposes. Messages displayed are sent to the standard output of the program. If the program is started by `init`, the destination of these messages is determined by the UNIX platform and version.

The `-n` argument is usually used when the program is started by `init(1M)` with the `respawn` option.



# 7 Configuration Files

This chapter describes the configuration files used with the Agent Integrator and other BEA Manager SNMP products. It includes the following sections:

- ◆ “BEA Manager Configuration File (`beamgr.conf`)” describes `beamgr.conf`, which is the configuration file for the Agent Integrator and other BEA Manager applications.
- ◆ “BEA Manager Passwords File (`beamgr_snmpd.conf`)” describes `beamgr_snmpd.conf`, which contains the community strings used as passwords in communication between managers and agents, and passwords used in communication with SMUX subagents. This file is separate from the `beamgr.conf` file because it should be secured with read and write access permitted for root only. This file is used by the subagents shipped with the Agent Integrator (or subagents generated using the BEA Manager Agent Development Kit) when they are run as standalone agents. This file is also used by the Agent Integrator master agent and the agents shipped with the Agent Connection.

# BEA Manager Configuration File (beamgr.conf)

`beamgr.conf` is the configuration file for the Agent Integrator and other BEA Manager applications.

## Default Location

On UNIX systems: `/etc/beamgr.conf`

On Windows NT systems: `C:\etc\beamgr.conf`

## Description

The `beamgr.conf` file contains information used by the BEA Manager products (including Agent Integrator, Agent Connection, and Log Central) and the SNMP agents and SMUX subagents created using the BEA Manager Agent Development Kit. The location of this configuration file may be specified by the environment variable `BEA_SM_BEAMGR_CONF`. A configuration entry is composed of two or more blank or tab-separated fields:

*KEYWORD parameters*

If an entry happens to be too long, the backslash (`\`) character can be used as a continuation character at the end of the line. There should be a newline character immediately following the `\` character.

The following keywords have corresponding MIB objects: `TMEVENT_FILTER`, `SYS_INSTALL`, `SYS_CONTACT`, `SYS_NAME`, `SYS_LOCATION`, `SYS_SERVICES`, and `RULE_ACTION`.



## Keywords Used by All BEA Manager Products

### TRAP\_HOST

Host name, port number, and community name necessary to send SNMP traps. The parameters are:

```
host_name trap_port trap_community
```

*host\_name* is the name of the target destination machine for the trap notification. Default is the local host.

You may have multiple TRAP\_HOST entries in the configuration file if you wish to send traps to multiple destinations.

The TRAP\_HOST entry is used by the following software components to determine trap destinations when generating SNMP trap notifications:

- ◆ Agent Integrator
- ◆ Agent Connection (BEA TUXEDO and M3 SNMP agents)
- ◆ SNMP agents and SMUX subagents created using the Agent Development Kit
- ◆ Log Central Message Sender
- ◆ Log Central Message Processor

### SNMP\_ENABLE\_AUTH\_TRAP

This field contains a value to indicate if the SNMP agent is permitted to generate authentication-failure traps. If the value is 1, the SNMP agent generates authentication-failure traps if an invalid request (as per the community profile) is received. If the value is not 1, no authentication-failure trap is generated.

### OID\_CLASS

This entry is used by the SMUX subagents created by the BEA Agent Development Kit. This entry is discussed in the section “The OID\_CLASS Entry.”

## Keywords Used by the Agent Integrator

### INTEGRATOR\_TIMEOUT

Sets the Agent Integrator timeout in waiting for responses to requests. The default timeout is 3 seconds. If you are using `tux_snmpd` or `m3_snmpd` as a subagent to manage TUXEDO 6.3 or 6.4 applications, you need to configure the Agent Integrator timeout to at least 30 seconds. This can be done by adding a `INTEGRATOR_TIMEOUT` entry as follows:

```
BEA_PEER MAX_WAIT 30
```

### INTEGRATOR\_MAX\_TIMEOUTS

Sets the maximum number of times the Agent Integrator allows requests to an SNMP peer or SMUX subagent to timeout before disregarding any further requests for that agent or subagent. The default is 3.

### NON\_SMUX\_PEER

This keyword specifies the OIDs supported by an SNMP agent when it is running as a peer of the Agent Integrator. This entry is discussed in the next section, “The `NON_SMUX_PEER` Entry.”

### RULE\_ACTION

This keyword is used to specify an Agent Integrator polling rule. This allows threshold-checking to be offloaded from the network management system to the distributed Integrator agents on managed nodes. This entry is discussed in the section “The `RULE_ACTION` Entry.”

## Keywords Used by the Agent Connection

### TMAGENT

This entry is used to define the TUXEDO or M3 domain that an agent is to monitor. There must be one `TMAGENT` entry for each TUXEDO or M3 agent on a managed node. The format of the entry is as follows:

```
TMAGENT logical_agent_name tuxdir tuxconfig_path
```

Monitoring of multiple domains is done by running a separate TUXEDO or M3 agent for each domain being monitored. These agents must be run as subagents under the Agent Integrator.

When there are multiple TUXEDO or M3 SNMP agents running on a managed node, the logical agent name must be used to modify the community in SET or GET requests. The community must be of the following form:

*community@logical\_agent\_name*

For example:

*public@payrollagent*

The community does not need to be qualified with the logical agent name when there is only one TUXEDO or M3 SNMP agent running on the managed node.

#### TMEVENT\_FILTER

This entry is used to define a subset of TUXEDO event notifications that are to be forwarded as SNMP trap notifications. By default, if no filter is provided, the Agent Connection forwards all TUXEDO events as SNMP traps. The format of the entry is:

*TMEVENT\_FILTER filter\_id logical\_agent\_name tux\_evt\_expr  
tux\_evt\_filter status*

**Note:** The strings that are used for each of the parameters must not have blanks or white space.

The parameters have the following interpretation:

*filter\_id*

Is a string that must be unique among all TMEVENT\_FILTER entries in the BEA Manager configuration file. Maximum length of the string is 16 characters.

*logical\_agent\_name*

Maps the event filter to a particular agent on that node. A logical agent name is a string up to 32 characters in length. The logical agent name is as specified in the `-l` option used when starting the agent (on UNIX systems) or the name of the Windows NT service (on Windows NT systems).

*tux\_evt\_expr*

Is a TUXEDO event name expression. This is a string up to 255 characters in length. Refer to the `recomp` entry in section 3 of the *BEA TUXEDO Reference Manual* for information about event name expressions. This name must match a TUXEDO event name for that

event to be forwarded as an SNMP trap. See the `EVENTS` entry in the *BEA TUXEDO Reference Manual* for a list of event names. The default is all events. If `NONE` is used, no events are forwarded and the other parameters in the `TMEVENT_FILTER` entry can be omitted. An entry of `NONE` overrides all other event filter entries for the same logical agent name. Examples:

An entry of

```
\.Sys.*
```

matches all events. An entry of

```
\.SysServer.*
```

matches all system events related to servers.

*tux\_evt\_filter*

Is an event filter expression. This is a string up to 255 characters in length. Each TUXEDO event is accompanied by an FML buffer that contains pertinent information about the event. The buffer is evaluated with respect to this filter if the filter is present. If the buffer content is evaluated to `TRUE`, the event is forwarded, otherwise not. The Agent Connection uses this filter as an argument for a call to `tpssubscribe()`. Refer to the `tpssubscribe()` entry in section 3 of the *BEA TUXEDO Reference Manual* for more information.

*status*

Is either `active` or `inactive`. If the status is `active`, the filter is being used, otherwise not.

There is a MIB table that corresponds to the `TMEVENT_FILTER` entries in the BEA Manager configuration file. These entries can be updated dynamically using an SNMP SET request. For more information, refer to the `beaEvtFilterTable` section in the *Agent Connection for TUXEDO and M3 Reference Manual*.

## Keywords Used by `unix_snmpd` and `nt_snmpd`

The following keywords are used by the subagents (`unix_snmpd` and `nt_snmpd`) shipped with Agent Integrator:

### `SYS_INSTALL`

The field associated with this keyword indicates when the SNMP agent was installed and configured on the host.

### `SYSOBJID`

The field associated with this keyword refers to the `sysObjectId` MIB-II object. If this keyword is not specified in the configuration file, `.1.3.6.1.4.1.140.1.1` is used as the value of the `sysObjectId` object. This keyword is also used by the Agent Development Kit trap functions to determine default enterprise identifier (OID). Refer to the “Tools and Functions” chapter in the *Agent Development Kit Programmer’s Guide* for more information.

### `SYS_DESCR`

The field associated with this keyword refers to the `sysDescr` MIB-II object.

### `SYS_CONTACT`

A description of the contact person in case of problems.

### `SYS_NAME`

Any string to describe the system running on that host.

### `SYS_LOCATION`

Any information about the physical location of the host.

### `SYS_SERVICES`

List of services offered by the host. If running the BEA Manager platform with applications, the list of application services running on the host may be listed here.

### `SYSSERVICES`

List of layers supported. This field refers to the `sysServices` object in the MIB-II `system` group. See the MIB-II definition in RFC 1213 for more information about this field. This is a distinct entry from `SYS_SERVICES`.

## The NON\_SMUX\_PEER Entry

To configure the BEA Agent Integrator to access MIB objects through a peer SNMP agent, (e.g., a non-SMUX master agent), add the NON\_SMUX\_PEER entry to the `beamgr.conf` file in the following format:

```
NON_SMUX_PEER port community[* OID_Node[,ro|rw][,priority][,timeout] ...
```

The explanation of the terms used in the previous entry follows.

NON\_SMUX\_PEER

Is the keyword for the entry.

*port*

Specifies the UDP port number on which the SNMP agent is listening. This value can be specified in the forms:

*ip\_address.port*  
*hostname.port*

when the SNMP agent is remote from the Agent Integrator. If *ip\_address* or *hostname* is not specified, the Agent Integrator assumes that the peer SNMP agent is on the same managed node as the Agent Integrator.

*community*

Specifies the community to be used by the Agent Integrator when the SNMP agent is polled. The special value `*` is used to specify that the Agent Integrator should use the community supplied by the SNMP manager.

*OID\_Node*

Specifies the OID of the root of the MIB tree that is supported by this SNMP agent.

*ro|rw*

Specifies whether this OID tree is being exported as read-only or as read-write. The default is `rw`.

*priority*

Is a positive number that specifies the priority at which the OID tree is being exported. The lower the number, the higher the priority. If there are multiple agents/subagents supporting the same MIB tree, the subagent with the highest priority is consulted. Multiple SNMP agents and SMUX subagents may register the same subtree. However, they must do so at different priorities.

If an SNMP agent tries to register a subtree at a priority that is already taken, the Agent Integrator repeatedly increments the integer value (lowering the priority) until an unused priority is found. A special priority -1, causes the selection of the highest available priority. When a request is made to register with priority -1, registration is made at the highest available number below 20. If the priority field is missing, the MIB tree is exported at a -1 priority.

`timeout`

Specifies the time interval in seconds for which the Agent Integrator waits for the replies from this SNMP agent for the particular MIB group. The default value is three seconds. This default may be changed by setting the `BEA_PEER_MAX_WAIT` environment variable to a different value.

The `access` (`ro` or `rw`), `priority`, and `timeout` fields are optional. But you must specify `access` and `priority` if you wish to specify `timeout`, and you must specify `access` if you wish to specify `priority` for a MIB tree.

Multiple OID nodes can be listed.

**Note:** A subtree registration will hide the registrations by other SNMP agents/subagents of objects within the subtree. So, if an agent A registers subtree `.1.3.6.1.4.1.140` and another agent/subagent, B, registers subtree `.1.3.6.1.4.1.140.1`, agent/subagent A will be consulted for all the objects under the `.1.3.6.1.4.1.140.1` subtree.

Also, when the Agent Integrator reads this entry, an SNMP agent should be running on the specified port. Otherwise, the Agent Integrator disregards this entry. Also, if three consecutive requests to this SNMP agent time out, it is assumed that the SNMP agent specified by this entry is no longer alive and this entry is disregarded.

At any point, the utility `reinit_integrator` can be used to force the Agent Integrator to re-read its configuration file.

The Agent Integrator will disallow/disregard any attempt to register above, at, or below the SNMP (`mib2.snmp`) and SMUX subtrees of the MIB.

## NON\_SMUX\_PEER Examples

- ◆ Consider the following sample entries in the `beamgr.conf` file:

```
NON_SMUX_PEER 2001 snmp .1.3.6.1.2.1.1,ro
NON_SMUX_PEER 2002 squid .1.3.6.1.4.1.141 .1.3.6.1.4.1.145
NON_SMUX_PEER 2008 * .1.3.6.1.4.1,ro
NON_SMUX_PEER 2005 * .1.3.6.1.4.1.140 .1.3.6.1.4.1.145,rw
```

The first entry tells the Agent Integrator to look for an SNMP agent at port 2001. All the requests from Agent Integrator to this SNMP agent will use `snmp` as the community. The agent supports the subtree `.1.3.6.1.2.1.1`, and is available for read-only commands.

The second entry tells the Agent Integrator to look for an SNMP agent at port 2002. All the requests from Agent Integrator to this SNMP agent will use `squid` as the community. The agent supports the subtrees `.1.3.6.1.4.1.141` and `.1.3.6.1.4.1.145`. Since no access option is specified, both subtrees default to `rw`.

The third entry specifies a non-SMUX agent at port 2008 with a community of `*`. The `*` tells the Agent Integrator to pass along the same community information it receives from the SNMP manager. For example, if the SNMP manager sends the community `nevus`, the Agent Integrator sends `nevus` along to the subagent. (Of course, `nevus` must be a valid community for the Agent Integrator in the first place.)

The fourth entry lists an agent at port 2005 with a community of `*`. The agent supports two subtrees: `.1.3.6.1.4.1.140` and `.1.3.6.1.4.1.145`. The first subtree lists no access information, so access defaults to `rw`. The second subtree specifically lists `rw`. This means exactly the same thing. The `rw` could have been left off with no effect.

If the `rw` is redundant, then why have it at all? Because each OID node can have three arguments: access (`ro` or `rw`), priority, and time-out. If you specify any of these arguments, you must also specify all the arguments that come before it. For example, if you specify priority, you must also specify access. If you specify time-out, you must specify both access and priority.

- ◆ The following entry tells the Agent Integrator to look for an agent at port 2008 with a community of `*`. The agent supports two subtrees. The first has an access of `rw`, a priority of `-1`, and a timeout of two seconds; the second has an access of



`rw`, a priority of `-1`, and a timeout of ten seconds. Although `rw` and `-1` are the defaults for access and priority, these values must be stated explicitly in order to include a timeout value.

```
NON_SMUX_PEER 2008 * .1.3.6.1.2.1.1,rw,-1,2
.1.3.6.1.2.1.2,rw,-1,10
```

The time-out values are the maximum amount of time the Agent Integrator waits for a response from the SNMP agent for a given object. In this case, two seconds for a response if the object falls under MIB tree `.1.3.6.1.2.1.1`, and ten seconds for a response if the object falls under the `.1.3.6.1.2.1.2` MIB group. The default value is three seconds. This default value can be changed by setting the environment variable `BEA_PEER_MAX_WAIT`.

- ◆ The priority value decides which agent is consulted by the Agent Integrator, in the event of a conflict. If more than one agent handles the same object, the one with the lowest number as a priority value is called. In the following entries, the agents on ports 2008 and 2009 both support object `.1.3.6.1.2.1.1`. The agent at port 2009 has a higher priority (5 is a higher priority than 8), so that is the one the Agent Integrator calls. Notice that this entry specifies `rw` access. The other entry specifies `ro` access, but since it has a lower priority, it is completely ignored. As far as the Agent Integrator is concerned, the only agent supporting `.1.3.6.1.2.1.1` is at port 2009.

```
NON_SMUX_PEER 2008 * .1.3.6.1.2.1.1,ro,8
NON_SMUX_PEER 2009 * .1.3.6.1.2.1.1,rw,5
```

If you do not specify a priority, the default is `-1`. The Agent Integrator reads through the file sequentially. When it comes to an object with a `-1` priority, it tries to assign it a priority of 20. If 20 has already been assigned to that MIB group (in another entry), it tries to assign 19. It keeps trying each successive lower number until it finds one that is not taken, or until it reaches 0. In that case it gives an error.

- ◆ In the following entries, the Agent Integrator assigns a priority of 20 to the first entry and a priority of 19 to the second entry. Since 19 is a higher priority, the agent at port 2009 handles object `.1.3.6.1.2.1.1` and the first entry is ignored. As before, the access is `rw`, since the entry specifying `ro` access is ignored.

```
NON_SMUX_PEER 2008 * .1.3.6.1.2.1.1,ro
NON_SMUX_PEER 2009 * .1.3.6.1.2.1.1
```

- ◆ The MIB tree `.1.3.6.1.2.1.11` is a special case, called `mib2.snmp`. This MIB group is always handled by the Agent Integrator itself, and should not be exported by any agent or subagent. Any registration at, above, or below this MIB tree is not allowed. For example, none of the following entries is allowed:

```
NON_SMUX_PEER 2005 * .1.3.6.1.2.1
NON_SMUX_PEER 2006 * .1.3.6.1.2.1.11
NON_SMUX_PEER 2007 * .1.3.6.1.2.1.11.7
```

The first includes `mib2.snmp`, the second specifies it exactly, and the third specifies a part of it.

If an SNMP agent wants to support `mib2` (except for the `snmp` group, as it is not allowed), you need to enter each of the `mib2` subtrees explicitly:

```
NON_SMUX_PEER 2002 * .1.3.6.1.2.1.1 .1.3.6.1.2.1.2 .1.3.6.1.2.1.3 \
                    .1.3.6.1.2.1.4 .1.3.6.1.2.1.5 .1.3.6.1.2.1.6 \
                    .1.3.6.1.2.1.7 .1.3.6.1.2.1.8 .1.3.6.1.2.1.9 \
                    .1.3.6.1.2.1.10
```

**Note:** To continue an entry on another line, use a backslash. Make sure that there are no characters (other than the carriage return) immediately following the backslash.

- ◆ The Agent Integrator supports row-level registration. Consider a table with five columns and two rows. Each item in the table is a separate object, and can be identified by the column and row identifiers, in that order. The following entry:

```
NON_SMUX_PEER 2000 * .1.3.6.1.4.1.140.100.5.1.2.1
```

specifies row 1 of the second column of table object

`.1.3.6.1.4.1.140.100.5.1`. Row 1 is not necessarily the first row. 1 is simply a unique identifier for the row.

Notice that `.1.3.6.1.4.1.140.100.5.1.2` refers to the whole second column.

If you want to associate the two rows with different subagents, you need to specify each object in the row:

```
NON_SMUX_PEER 2000 * .1.3.6.1.4.1.140.100.5.1.1.1 .1.3.6.1.4.1.140.100.5.1.2.1 \
                    .1.3.6.1.4.1.140.100.5.1.3.1 .1.3.6.1.4.1.140.100.5.1.4.1 \
                    .1.3.6.1.4.1.140.100.5.1.5.1
NON_SMUX_PEER 2002 * .1.3.6.1.4.1.140.100.5.1.1.2 .1.3.6.1.4.1.140.100.5.1.2.2 \
                    .1.3.6.1.4.1.140.100.5.1.3.2 .1.3.6.1.4.1.140.100.5.1.4.2 \
                    .1.3.6.1.4.1.140.100.5.1.5.2
```

The first entry lists the object in row 1 in each of the five columns. The second entry lists the object in row 2 in each of the five columns.

## The **OID\_CLASS** Entry

This entry is used by the SMUX subagents created by the Agent Development Kit. The `unix_snmpd`, `nt_snmpd`, and `pm_snmpd` subagents are SMUX subagents created using the Agent Development Kit which are shipped along with `snmp_integrator`. Normally, SMUX subagents register all the MIB groups they know about with the SMUX master agent. But you may limit the MIB groups exported by these SMUX subagents. In order to do so, you need to add an `OID_CLASS` entry to `beamgr.conf` in the following format:

```
OID_CLASS agent_name OID_Node[,ro|rw][,priority] ..
```

An explanation of the terms used in the previous entry follows.

`OID_CLASS`

Is the keyword for the entry.

`agent_name`

Specifies the name of the SMUX subagent for which this entry is applicable.

`OID_Node`

Specifies the OID of the tree that is supported by this subagent.

`ro|rw`

Specifies whether this OID tree is being exported as read-only or as read-write. The default is `rw`.

`priority`

Is a positive number that specifies the priority at which the OID tree is being exported. The lower the number, the higher the priority. If there are multiple subagents supporting the same MIB tree, the subagent with the highest priority is consulted. If the priority field is missing, the MIB tree is exported at the highest available priority. This entry is mainly used to limit the OID subtrees being exported by the Agent Integrator or SMUX subagents generated using the Agent Development Kit. If this entry is not present, the SMUX subagent exports all the MIB groups it knows about.

Multiple OID nodes can be listed.

## Example

Assume you have a SMUX subagent called `my_snmpd`. This subagent is created using the Agent Development Kit and supports the MIB groups `.1.3.6.1.4.1.140.1`, `.1.3.6.1.4.1.140.2`, and `.1.3.6.1.2.1.1`. When this subagent starts, it registers all three MIB groups as read-write with the SMUX master agent. If, for example, you want `my_snmpd` to only export `.1.3.6.1.4.1.140.1` as read-only and `.1.3.6.1.2.1` as read-write, you would need to insert an entry of the following form:

```
OID_CLASS my_snmpd .1.3.6.1.4.1.140.1,ro .1.3.6.1.2.1.1
```

## The RULE\_ACTION Entry

The BEA Agent Integrator can be configured to do management locally on the managed node and inform the SNMP manager selectively to reduce polling traffic generated by SNMP manager. The user can define rules in terms of MIB objects available locally using a C-like “IF” syntax and accordingly send SNMP traps or execute commands locally (or both). These MIB objects may be the one supported by the Agent Integrator itself or the one supported by one of its SMUX subagents or SNMP agents. For a discussion of RULE\_ACTION entries, with examples, refer to Chapter 4, “Using Agent Integrator for Polling.”

The configuration is done in the `beamgr.conf` file. Syntax of the entry looks like following:

```
RULE_ACTION rule-name frequency_in_secs \  
if (VAL(oid) rel_operator value) logical_op ( cond_2 ) ...\  
{ \  
TRAPID_ERR=enterprise-specific-trapid\  
TRAPID_OK=enterprise-specific-trapid\  
COMMAND_ERR=executable\  
COMMAND_OK=executable\  
}
```

**Note:** The whole entry should appear on the same line, else the backslash (\) should be used as a continuation character. If \ is being used as a continuation character, a newline character should immediately follow it.

RULE\_ACTION

Is a keyword to identify this entry.

*rule-name*

Is a unique identifier for each RULE\_ACTION entry and is passed as a command-line argument to any commands specified as actions in the rule. This identifier cannot be more than 8 characters long.

*frequency*

Is the polling frequency (in seconds) at which the `snmp_integrator` should check the condition.

*VAL*

Each left-hand side of a condition should have this keyword which should be followed by the object identifier (within parenthesis) of the MIB object. Each rule can contain a maximum of ten *VAL* keywords.

*oid*

Is an object identifier (OID). The OID must be specified only in numeric form and can be in one of the following formats:

- ◆ An absolute OID, that is, the full path to the object is specified from the root of the OID tree. An initial dot is used to indicate that the path starts at root. For example: `.1.3.6.1.2.1.1.1.0`. Note that the trailing zero in this example is the instance index.

- ◆ A relative OID under the MIB II branch can be specified in the form  
`mib-2.number.number ...`

When the reserved word `mib-2` appears as the leading sub-oid, `.1.3.6.1.2.1.` is assumed to be prefixed to the rest of the OID. For example:

`mib-2.1.1.0`

represents the absolute OID:

`.1.3.6.1.2.1.1.1.0`

- ◆ A relative OID under the enterprises branch can be specified in the form:

`enterprises.number.number ...`

When the reserved word `enterprises` appears as the leading sub-oid, `.1.3.6.1.4.1.` is assumed to be prefixed to the rest of the OID. For example:

`enterprises.140.1.0`

represents the absolute OID:

.1.3.6.1.4.1.140.1.0

- ◆ A relative OID under the enterprises branch can also be specified in purely numeric form:

number.number.number . . . ,

If there is no leading “.” and the OID starts with a number,

.1.3.6.1.4.1. is assumed to be prefixed to the rest of OID. For example:

140.1.1.0

represents the absolute OID:

.1.3.6.1.4.1.140.1.1.0

*Columnar* objects are used to represent a column of a tabular MIB group. Columnar objects accordingly can have multiple instances. The last number in an OID is used to specify the particular instance. A specific number can be used to specify a particular instance or the asterisk wildcard can be used to specify all instances. Zero is used as the instance index in the case of *scalar* objects (objects that can have only one instance). The asterisk wildcard is only used to represent all instances of a columnar object. For example:

.1.3.6.1.4.1.140.1.1.0

specifies the single instance of a scalar object while

1.3.6.1.4.1.140.2.22.1.2.\*

specifies all of the instances of a columnar object.

**Note:** When specifying multiple OIDs in a complex rule, the OIDs should either all use wildcards or none should. That is, you should not combine an OID that specifies a particular instance with an OID that uses a wildcard in the same rule. Also, when multiple OIDs with wildcards are used in a single rule, all of the OIDs should specify objects only within the same table.

*rel\_operator*

Valid relational operators are listed in the table under “Relations for Defining Conditions.”

*value*

The RHS in a condition can be one of the following: number, string, IP address: number1.number2.number3.number4, or OID (as previously explained).

If RHS is an OID it must be enclosed in single quotes. Also the type of value on RHS should correspond to the type of VALUE of the OID in LHS of the condition.

*logical\_op*

Valid logical operators are listed in Table 4-2.

## Specifying Actions

Two actions can be taken whenever there is a transition in the state of a rule from true (ERR) to false (OK) and false (OK) to true (ERR) — namely execute a command and/or generate an SNMP\_TRAP. When generic OIDs (an asterisk used to specify all instances of a columnar object) are used to define a rule, the rule state transitions from the OK state to the ERR state if the threshold evaluates to true for any row in the table and the rule state transitions from ERR to OK if the threshold evaluates to false for all rows in the MIB table. Initially, the rule states of all rules are set to OK when the Integrator starts up (or is re-initialized using the `reinit_agent` command). Actions are specified using the following keywords:

TRAPID\_ERR = *number*

This indicates that an enterprise specific SNMP trap with trapid of *number* should be generated whenever there is a transition from false (OK) to true (ERR).

TRAPID\_OK = *number*

This indicates that an enterprise specific SNMP trap with trapid of *number* should be generated whenever there is a transition from true (ERR) to false (OK).

COMMAND\_ERR = *command*

This indicates that *command* should be executed whenever there is a transition from false (OK) to true (ERR).

COMMAND\_OK = *command*

This indicates that *command* should be executed whenever there is a transition from true (ERR) to false (OK).

# BEA Manager Passwords File (beamgr\_snmpd.conf)

`beamgr_snmpd.conf` is a configuration file used by the BEA Manager Agent Integrator and subagents generated using the Agent Development Kit.

## Default Location

`/etc/beamgr_snmpd.conf` on UNIX platforms.  
`C:\etc\beamgr_snmpd.conf` on Windows NT systems.

## Description

The file `beamgr_snmpd.conf` contains information used by the BEA Manager Agent Integrator, or by programs created using the Agent Development Kit when they are used as SNMP agents, as opposed to SMUX subagents. This file is separate from the `beamgr.conf` file because it contains the SNMP community strings which are used as passwords in communication between agents and managers.

This file will be installed with access privileges for root only. For password security, the read and write permissions for the `beamgr_snmpd.conf` file should be set to allow access only by root.

A configuration entry is composed of two or more blank or tab-separated fields:

*KEYWORD parameters*

Recognized values for *KEYWORD* are:

### COMMUNITY\_RW

The string following this keyword specifies the read-write community for the agent. If this keyword is not present in the configuration file, the SNMP agent takes `beamgr` as the read-write community. Entries with this keyword may be repeated more than one time, in order to specify more than one read-write community.



#### COMMUNITY\_RO

The string following this keyword specifies the read-only community for the agent. If this keyword is not present in the configuration file, the SNMP agent takes `public` as the read-only community. Entries with this keyword may be repeated more than one time, in order to specify more than one read-only community.

#### SMUX\_PASSWD

The string following this keyword specifies the SMUX password. Any SMUX subagent wishing to register with the Agent Integrator should specify this password. If this keyword is not present, no authentication is done by the Agent Integrator.

#### DISABLE\_SET

The possible values for this keyword are `YES` or `NO`. `NO` is the default. If set to `YES`, SET access for all BEA Manager agents (including agents built using the Agent Development Kit) is disabled.



# 8 BEA SNMP Agent MIB

This chapter provides an overview of the BEA MIB which is supported by the SNMP agents shipped with the Agent Integrator product. The following groups are included.

Group Name	Attributes
beaSystem	General-purpose BEA Manager objects such as UNIX operating system name and version and fields in the <code>beamgr.conf</code> file
beaUnix	Managed objects specific to UNIX systems such as the process table, file system table, the <code>ipcs</code> tables, performance measurements ( <code>rstat</code> command), and <code>pstat -a</code> command
beaSmgr	Shared memory manager group
beaSysPerf	BEA system performance group
beaNTSysPerf	System performance specific to Windows NT machines
beaIntAgt	BEA Intelligent Agent MIB, which includes tables for addition or modification of polling and event-generation rules

# beaSystem Group

This group contains general purpose objects such as operating system name and version and the fields in the `beamgr.conf` configuration file. Fields in the configuration file can be changed through an SNMP SET on the corresponding MIB object. This MIB is supported by the `unix_snmpd` and `nt_snmpd` subagents, provided with Agent Integrator.

Variable Name	Object ID
beaSysDescr	.1.3.6.1.4.1.140.1.1
beaSysId	.1.3.6.1.4.1.140.1.2
beaSysInstallTime	.1.3.6.1.4.1.140.1.3
beaSysContact	.1.3.6.1.4.1.140.1.4
beaSysName	.1.3.6.1.4.1.140.1.5
beaSysLocation	.1.3.6.1.4.1.140.1.6
beaSysServices	.1.3.6.1.4.1.140.1.7
beaTrapHost	.1.3.6.1.4.1.140.1.8
beaTrapCommunity	.1.3.6.1.4.1.140.1.9
beaTrapPort	.1.3.6.1.4.1.140.1.10
beaTrapDescr	.1.3.6.1.4.1.140.1.11
beaSysHasDisk	.1.3.6.1.4.1.140.1.12
beaSysLmServers	.1.3.6.1.4.1.140.1.13
beaSysLmLogFile	.1.3.6.1.4.1.140.1.14
beaSysEmServers	.1.3.6.1.4.1.140.1.15
beaSysSysname	.1.3.6.1.4.1.140.1.16
beaSysNodename	.1.3.6.1.4.1.140.1.17

Variable Name	Object ID
beaSysRelease	.1.3.6.1.4.1.140.1.18
beaSysVersion	.1.3.6.1.4.1.140.1.19
beaSysMachine	.1.3.6.1.4.1.140.1.20
beaSysAgentVersion	.1.3.6.1.4.1.140.1.21

## beaSysDescr

Syntax	DisplayString
Access	read-only
Description	Contains the agent version and copyright notice as well as the architecture that the agent runs on.

## beaSysId

Syntax	Object Identifier
Access	read-only
Description	The value of this object is an OID that consists of the BEA OID followed by the IP address of the managed node, (i.e., .1.3.6.1.4.1.140.IP_Address).

## beaSysInstallTime

Syntax	DisplayString
Access	read-only
Description	A string that specifies when the agent was installed on the host. This is specified in the SYS_INSTALL entry in the BEA Manager configuration file.

### **beaSysContact**

Syntax	DisplayString
Access	read-only
Description	A user-specified string that specifies the contact person for problems on the host. This field is specified the SYS_CONTACT entry in the <code>beamgr.conf</code> file.

### **beaSysName**

Syntax	DisplayString
Access	read-only
Description	A user-defined string that specifies the system name for the host. This field is specified in the <code>beamgr.conf</code> file.

### **beaSysLocation**

Syntax	DisplayString
Access	read-only
Description	A user-defined string that specifies the location of the host. This field is specified in the <code>beamgr.conf</code> file by the keyword SYS_LOCATION.

### **beaSysServices**

Syntax	DisplayString
Access	read-only
Description	A user-defined string that specifies the system services provided by the host. This field is specified in the <code>beamgr.conf</code> file by the keyword SYS_SERVICES.

## beaTrapHost

Syntax	DisplayString
Access	read-only
Description	A user-defined string that specifies the name of the host that is the destination for traps sent from this host. This field is specified by the TRAP_HOST entry in the <code>beamgr.conf</code> file.

## beaTrapCommunity

Syntax	DisplayString
Access	read-only
Description	Community string to use when sending a trap to the host identified by <code>beaTrapHost</code> . This field is specified in the TRAP_HOST entry in the BEA Manager configuration file ( <code>beamgr.conf</code> ).

## beaTrapPort

Syntax	Integer
Access	read-only
Description	The port number to use when sending a trap to the host designated by <code>beaTrapHost</code> . This field is specified in the TRAP_HOST entry in the <code>beamgr.conf</code> file.

## beaTrapDescr

Syntax	DisplayString
Access	read-only
Description	This string contains a message explaining the reason for a trap. This object is passed as part of the variable bindings in the trap. This field is used to describe transition (OK to ERR, or ERR to OK) for traps sent as actions in response to local Agent Integrator polling.

### **beaSysHasDisk**

Syntax	Integer
Access	read-only
Description	Indicates whether the host has a hard disk. Acceptable values, and their interpretation, are: 0 : Unknown 1 : No 2 : Yes

### **beaSysSysname**

Syntax	DisplayString
Access	read-only
Description	Operating system name as given by the <code>uname -s</code> command. For example, SunOS.

### **beaSysNodename**

Syntax	DisplayString
Access	read-only
Description	Node (machine) name as given by the <code>uname -n</code> command. For example, blueberry.

### **beaSysRelease**

Syntax	DisplayString
Access	read-only
Description	Operating system release as given by the <code>uname -r</code> command. For example, 4.1.1.

### **beaSysVersion**

Syntax	DisplayString
Access	read-only
Description	Operating system version as given by the <code>uname -m</code> command. For example, 1.



## beaSysMachine

Syntax    DisplayString

Access    read-only

Description    Machine type as given by the `uname -m` command. For example, `sun4c`.

## beaSysAgentVersion

Syntax    DisplayString

Access    read-only

Description    Software revision level of the BEA Manager SNMP agent. For example, `3.6`.

# beaUnix Group

This group includes such objects as the process table, file system table, the ipcs tables, the performance measurements (`rstat`), and the `pstat -s` command. The message queue and semaphore (`ipcs -sa`) tables are not supported on Windows NT systems. This MIB group is supported by the `unix_snmpd` and `nt_snmpd` subagents provided with Agent Integrator. However, the `nt_snmpd` subagent does not support the `beaMqTable`, `beaShmTable`, and `beaSemTable` subgroups.

Variable Name	Object ID
beaPsTable	.1.3.6.1.4.1.140.2.1
beaDfTable	.1.3.6.1.4.1.140.2.3
beaPstatSwapAlloc	.1.3.6.1.4.1.140.2.15
beaPstatSwapReser	.1.3.6.1.4.1.140.2.16
beaPstatSwapUsed	.1.3.6.1.4.1.140.2.17
beaPstatSwapAvail	.1.3.6.1.4.1.140.2.18
beaMqTable	.1.3.6.1.4.1.140.2.19
beaShmTable	.1.3.6.1.4.1.140.2.20
beaSemTable	.1.3.6.1.4.1.140.2.21
beaLclDfTable	.1.3.6.1.4.1.140.2.22

# beaPsTable

Each entry (row) in the beaPsTable is a sequence of the following columnar objects. These objects represent the attributes of a software process executing on the host. A process is an entity created when a program is executed.

Variable Name	Object ID
beaPsUser	.1.3.6.1.4.1.140.2.1.1.2
beaPsPid	.1.3.6.1.4.1.140.2.1.1.3
beaPsCpu	.1.3.6.1.4.1.140.2.1.1.4
beaPsMem	.1.3.6.1.4.1.140.2.1.1.5
beaPsSize	.1.3.6.1.4.1.140.2.1.1.6
beaPsRss	.1.3.6.1.4.1.140.2.1.1.7
beaPsTty	.1.3.6.1.4.1.140.2.1.1.8
beaPsStat	.1.3.6.1.4.1.140.2.1.1.9
beaPsDate	.1.3.6.1.4.1.140.2.1.1.10
beaPsCmd	.1.3.6.1.4.1.140.2.1.1.11
beaPsCpuTime	.1.3.6.1.4.1.140.2.1.1.12
beaPsThreadCount	.1.3.6.1.4.1.140.2.1.1.13
beaPsHandleCount	.1.3.6.1.4.1.140.2.1.1.14

## beaPsUser

Syntax	DisplayString
Access	read-only
Description	Name of the owner of the process.

### **beaPsPid**

Syntax	Integer
Access	read-only
Description	Process ID number.

### **beaPsCpu**

Syntax	DisplayString
Access	read-only
Description	Percentage of CPU capacity in use.

### **beaPsMem**

Syntax	DisplayString
Access	read-only
Description	Percentage of real memory being used by this process.

### **beaPsSize**

Syntax	Integer
Access	read-only
Description	The combined size of the data and stack segments (in kilobytes).

### **beaPsRss**

Syntax	Integer
Access	read-only
Description	The real memory (resident set) size of this process (in kilobytes). On systems where this attribute is not available, the value for all processes is set to zero.

## beaPsTty

Syntax	DisplayString
Access	read-only
Description	TTY used by this process.

## beaPsStat

Syntax	DisplayString
Access	read-only
Description	Process status. This field may contain multiple characters such as R, T, P, D, I, Z.

## beaPsDate

Syntax	DisplayString
Access	read-only
Description	Date and time when the process started.

## beaPsCmd

Syntax	DisplayString
Access	read-only
Description	The command that was used to start the process.

**beaPsCpuTime**

Syntax	Integer
Access	read-only
Description	CPU time is the cumulative time, in timeticks, that all of the threads of this process have made use of the processor to execute instructions. Instructions are the basic unit of execution in a computer, and threads are the object that executes instructions. A process is an object created when a program is executed. Code executed to handle certain hardware interrupts or hardware traps may be counted towards the total value for this object. This object is supported only for Windows NT systems.

**beaPsThreadCount**

Syntax	Integer
Access	read-only
Description	The number of threads currently active in this process. A thread is an object that executes instructions. Instructions are the basic unit of execution in a processor. Every running process has at least one thread. This object is supported for Windows NT systems only.

**beaPsHandleCount**

Syntax	Integer
Access	read-only
Description	The total number of handles open for this process. This number is the sum of the number of handles open for each thread in this process.

# beaDfTable

Each entry (row) in the beaDfTable is a sequence of the following columnar objects. These objects represent the attributes of file systems mounted on the host. The objects in this table are equivalent to the output of the UNIX `df` command.

Variable Name	Object ID
beaDfIndex	.1.3.6.1.4.1.140.2.3.1.1
beaDfHostname	.1.3.6.1.4.1.140.2.3.1.2
beaDfFilesystem	.1.3.6.1.4.1.140.2.3.1.3
beaDfKbytes	.1.3.6.1.4.1.140.2.3.1.4
beaDfUsed	.1.3.6.1.4.1.140.2.3.1.5
beaDfAvail	.1.3.6.1.4.1.140.2.3.1.6
beaDfCapacity	.1.3.6.1.4.1.140.2.3.1.7
beaDfMountedon	.1.3.6.1.4.1.140.2.3.1.8

## beaDfIndex

Syntax	Integer
Access	read-only
Description	Index of the entry (row) in the file system table.

## beaDfHostname

Syntax	DisplayString
Access	read-only
Description	The name of the host on which the file system is mounted. If the file system is local, this field is a hyphen (-).

### **beaDfFilesystem**

Syntax	DisplayString
Access	read-only
Description	Name of the file system.

### **beaDfKbytes**

Syntax	Integer
Access	read-only
Description	Size of the file system in kilobytes.

### **beaDfUsed**

Syntax	Integer
Access	read-only
Description	The number of kilobytes used in the file system.

### **beaDfAvail**

Syntax	Integer
Access	read-only
Description	The number of kilobytes available in the file system.

### **beaDfCapacity**

Syntax	Integer
Access	read-only
Description	Percentage of the total capacity of the file system that is in use.



## beaDfMountedon

Syntax	DisplayString
Access	read-only
Description	Directory where the file system is mounted.

## System Statistics

The scalar objects in this section are equivalent to the information provided by the UNIX `pstat -s` command.

### beaPstatSwapAlloc

Syntax	Integer
Access	read-only
Description	Amount of swap space (in kilobytes) allocated to private pages.

### beaPstatSwapReser

Syntax	Integer
Access	read-only
Description	Number of kilobytes of swap space that are not currently allocated but which are claimed by memory mappings that have not yet created private pages.

### beaPstatSwapUsed

Syntax	Integer
Access	read-only
Description	Total amount of swap space, in kilobytes, that is either allocated or reserved.

**beaPstatSwapAvail**

Syntax	Integer
Access	read-only
Description	Total swap space, in kilobytes, that is currently available for future reservation or allocation.

# beaMqTable

Each entry (row) in the `beaMqTable` is a sequence of the following columnar objects. These objects represent the attributes of message queue entries. This table is not supported by the `nt_snmpd` subagent.

Variable Name	Object ID
<code>beaMqT</code>	<code>.1.3.6.1.4.1.140.2.19.1.2</code>
<code>beaMqId</code>	<code>.1.3.6.1.4.1.140.2.19.1.3</code>
<code>beaMqKey</code>	<code>.1.3.6.1.4.1.140.2.19.1.4</code>
<code>beaMqMode</code>	<code>.1.3.6.1.4.1.140.2.19.1.5</code>
<code>beaMqOwner</code>	<code>.1.3.6.1.4.1.140.2.19.1.6</code>
<code>beaMqGroup</code>	<code>.1.3.6.1.4.1.140.2.19.1.7</code>
<code>beaMqCreator</code>	<code>.1.3.6.1.4.1.140.2.19.1.8</code>
<code>beaMqCgroup</code>	<code>.1.3.6.1.4.1.140.2.19.1.9</code>
<code>beaMqCbytes</code>	<code>.1.3.6.1.4.1.140.2.19.1.10</code>
<code>beaMqQnum</code>	<code>.1.3.6.1.4.1.140.2.19.1.11</code>
<code>beaMqQbytes</code>	<code>.1.3.6.1.4.1.140.2.19.1.12</code>
<code>beaMqLspid</code>	<code>.1.3.6.1.4.1.140.2.19.1.13</code>
<code>beaMqLrpid</code>	<code>.1.3.6.1.4.1.140.2.19.1.14</code>
<code>beaMqStime</code>	<code>.1.3.6.1.4.1.140.2.19.1.15</code>
<code>beaMqRtime</code>	<code>.1.3.6.1.4.1.140.2.19.1.16</code>
<code>beaMqCtime</code>	<code>.1.3.6.1.4.1.140.2.19.1.17</code>

### **beaMqT**

Syntax	DisplayString
Access	read-only
Description	Type of facility. This value is always set to q.

### **beaMqId**

Syntax	Integer
Access	read-only
Description	The identifier for the facility entry.

### **beaMqKey**

Syntax	DisplayString
Access	read-only
Description	The interprocess communication (IPC) key for the message queue.

### **beaMqMode**

Syntax	DisplayString
Access	read-only
Description	Access modes and flags for the facility.

### **beaMqOwner**

Syntax	DisplayString
Access	read-only
Description	Login name of the owner of the facility entry.

## beaMqGroup

Syntax	DisplayString
Access	read-only
Description	Name of the group of the owner of the facility entry.

## beaMqCreator

Syntax	DisplayString
Access	read-only
Description	Login name of the creator of the facility entry.

## beaMqCgroup

Syntax	DisplayString
Access	read-only
Description	Name of the group of the creator of the facility entry.

## beaMqCbytes

Syntax	Integer
Access	read-only
Description	The number of bytes in messages currently outstanding in the associated message queue.

## beaMqQnum

Syntax	Integer
Access	read-only
Description	Number of messages currently outstanding in the associated message queue.

### **beaMqQbytes**

Syntax	Integer
Access	read-only
Description	Maximum number of bytes allowed in messages outstanding in the associated message queue.

### **beaMqLspid**

Syntax	Integer
Access	read-only
Description	Process ID of the last process to send a message to the associated queue.

### **beaMqLrpid**

Syntax	Integer
Access	read-only
Description	Process ID of the last process to receive a message from the associated queue.

### **beaMqStime**

Syntax	DisplayString
Access	read-only
Description	Time when the last message was sent to the associated queue.

### **beaMqRtime**

Syntax	DisplayString
Access	read-only
Description	Time when the last message was received from the associated queue.

## beaMqCtime

Syntax	DisplayString
Access	read-only
Description	Time when the associated entry was created or changed.

# beaShmTable

Each entry (row) in the beaShmTable is a sequence of the following columnar objects. These objects represent the attributes of entries in the UNIX shared memory table. This is equivalent to the UNIX `ipcs -ma` command. This table is not supported by the `nt_snmpd` subagent.

Variable Name	Object ID
beaShmT	.1.3.6.1.4.1.140.2.20.1.2
beaShmId	.1.3.6.1.4.1.140.2.20.1.3
beaShmKey	.1.3.6.1.4.1.140.2.20.1.4
beaShmMode	.1.3.6.1.4.1.140.2.20.1.5
beaShmOwner	.1.3.6.1.4.1.140.2.20.1.6
beaShmGroup	.1.3.6.1.4.1.140.2.20.1.7
beaShmCreator	.1.3.6.1.4.1.140.2.20.1.8
beaShmCgroup	.1.3.6.1.4.1.140.2.20.1.9
beaShmNattch	.1.3.6.1.4.1.140.2.20.1.10
beaShmSegsz	.1.3.6.1.4.1.140.2.20.1.11
beaShmCpid	.1.3.6.1.4.1.140.2.20.1.12
beaShmLpid	.1.3.6.1.4.1.140.2.20.1.13
beaShmAtime	.1.3.6.1.4.1.140.2.20.1.14
beaShmDtime	.1.3.6.1.4.1.140.2.20.1.15
beaShmCtime	.1.3.6.1.4.1.140.2.20.1.16



## beaShmT

Syntax	DisplayString
Access	read-only
Description	Type of facility. This value is always set to m.

## beaShmId

Syntax	Integer
Access	read-only
Description	The identifier for the facility entry.

## beaShmKey

Syntax	DisplayString
Access	read-only
Description	The interprocess communication (IPC) key for the shared memory.

## beaShmMode

Syntax	DisplayString
Access	read-only
Description	The access modes and flags for the facility.

## beaShmOwner

Syntax	DisplayString
Access	read-only
Description	Login name of the owner of the facility entry.

### **beaShmGroup**

Syntax	DisplayString
Access	read-only
Description	Name of the group of the owner of the facility entry.

### **beaShmCreator**

Syntax	DisplayString
Access	read-only
Description	Login name of the creator of the facility entry.

### **beaShmCgroup**

Syntax	DisplayString
Access	read-only
Description	Name of the group of the creator of the facility entry.

### **beaShmNattch**

Syntax	Integer
Access	read-only
Description	Number of processes attached to the associated shared memory segment.

### **beaShmSegsz**

Syntax	Integer
Access	read-only
Description	The size of the associated shared memory segment.

## beaShmCpid

Syntax	Integer
Access	read-only
Description	The process ID of the creator of the shared memory entry.

## beaShmLpid

Syntax	Integer
Access	read-only
Description	Process ID of the last process to attach to or detach from the shared memory segment.

## beaShmAtime

Syntax	DisplayString
Access	read-only
Description	The time of the last attachment of a process to the associated shared memory segment.

## beaShmDtime

Syntax	DisplayString
Access	read-only
Description	The time of the last detachment of a process from the associated shared memory segment.

## beaShmCtime

Syntax	DisplayString
Access	read-only
Description	The time when the associated entry was created or changed.

# beaSemTable

Each entry (row) in the beaSemTable is a sequence of the following columnar objects. These objects represent the attributes of entries in the UNIX semaphore table. This is equivalent to the UNIX `ipcs -sa` command. This table is not supported by the `nt_snmpd` subagent.

Variable Name	Object ID
beaSemT	.1.3.6.1.4.1.140.2.21.1.2
beaSemId	.1.3.6.1.4.1.140.2.21.1.3
beaSemKey	.1.3.6.1.4.1.140.2.21.1.4
beaSemMode	.1.3.6.1.4.1.140.2.21.1.5
beaSemOwner	.1.3.6.1.4.1.140.2.21.1.6
beaSemGroup	.1.3.6.1.4.1.140.2.21.1.7
beaSemCreator	.1.3.6.1.4.1.140.2.21.1.8
beaSemCgroup	.1.3.6.1.4.1.140.2.21.1.9
beaSemNsems	.1.3.6.1.4.1.140.2.21.1.10
beaSemOtime	.1.3.6.1.4.1.140.2.21.1.11
beaSemCtime	.1.3.6.1.4.1.140.2.21.1.12

## beaSemT

Syntax	DisplayString
Access	read-only
Description	Type of facility. This value is always set to s.

## **beaSemId**

Syntax	Integer
Access	read-only
Description	The identifier for the facility entry.

## **beaSemKey**

Syntax	DisplayString
Access	read-only
Description	The interprocess communication (IPC) key for the semaphore.

## **beaSemMode**

Syntax	DisplayString
Access	read-only
Description	Access modes and flags for the facility.

## **beaSemOwner**

Syntax	DisplayString
Access	read-only
Description	Login name of the owner of the facility entry.

## **beaSemGroup**

Syntax	DisplayString
Access	read-only
Description	Name of the group of the owner of the facility entry.

### **beaSemCreator**

Syntax	DisplayString
Access	read-only
Description	Login name of the creator of the facility entry.

### **beaSemCgroup**

Syntax	DisplayString
Access	read-only
Description	Name of the group of the creator of the facility entry.

### **beaSemNsems**

Syntax	Integer
Access	read-only
Description	The number of semaphores in the set associated with the semaphore entry.

### **beaSemOtime**

Syntax	DisplayString
Access	read-only
Description	The time of completion of the last operation on the set associated with the semaphore entry.

### **beaSemCtime**

Syntax	DisplayString
Access	read-only
Description	The time when the associated entry was created or changed.

## beaLclDfTable

Each entry (row) in the `beaLclDfTable` is a sequence of the following columnar objects. These objects represent the attributes of local file systems. This is equivalent to the output of the UNIX `df` command.

Variable Name	Object ID
<code>beaLclDfFilesystem</code>	<code>.1.3.6.1.4.1.140.2.22.1.1</code>
<code>beaLclDfKbytes</code>	<code>.1.3.6.1.4.1.140.2.22.1.2</code>
<code>beaLclDfUsed</code>	<code>.1.3.6.1.4.1.140.2.22.1.3</code>
<code>beaLclDfAvail</code>	<code>.1.3.6.1.4.1.140.2.22.1.4</code>
<code>beaLclDfCapacity</code>	<code>.1.3.6.1.4.1.140.2.22.1.5</code>
<code>beaLclDfMountedon</code>	<code>.1.3.6.1.4.1.140.2.22.1.6</code>

### beaLclDfFilesystem

Syntax	<code>DisplayString</code>
Access	<code>read-only</code>
Description	Name of the file system.

### beaLclDfKbytes

Syntax	<code>Integer</code>
Access	<code>read-only</code>
Description	File system size in kilobytes.

### **beaLclDfUsed**

Syntax	Integer
Access	read-only
Description	The number of kilobytes used in the file system.

### **beaLclDfAvail**

Syntax	Integer
Access	read-only
Description	The number of kilobytes available in the file system.

### **beaLclDfCapacity**

Syntax	Integer
Access	read-only
Description	Percentage of the file system capacity that is in use.

### **beaLclDfMountedon**

Syntax	DisplayString
Access	read-only
Description	Directory where the file system is mounted.



# beaSmgr Group

The shared memory manager table is the only object under the beaSmgr group.

## beaSmgrTable

The shared memory manager table is supported by the `unix_snmpd` subagent shipped with the Agent Integrator. Each entry (row) in the `beaSmgrTable` is a sequence of the following columnar objects. Each row is a sequence of attributes for a block of shared memory.

Variable Name	Object ID
beaSmgrIndex	.1.3.6.1.4.1.140.5.1.1
beaSmgrIpKey	.1.3.6.1.4.1.140.5.1.2
beaSmgrShmAllocated	.1.3.6.1.4.1.140.5.1.3
beaSmgrSubsystem	.1.3.6.1.4.1.140.5.1.4
beaSmgrCreationDate	.1.3.6.1.4.1.140.5.1.5
beaSmgrLibVersion	.1.3.6.1.4.1.140.5.1.6
beaSmgrMaxShmEntries	.1.3.6.1.4.1.140.5.1.7
beaSmgrCurrentShmEntries	.1.3.6.1.4.1.140.5.1.8
beaSmgrHWMShmUsed	.1.3.6.1.4.1.140.5.1.9
beaSmgrPctShmUsed	.1.3.6.1.4.1.140.5.1.10
beaSmgrHWMPctShmUsed	.1.3.6.1.4.1.140.5.1.11
beaSmgrTotalShmSize	.1.3.6.1.4.1.140.5.1.12
beaSmgrUsegSize	.1.3.6.1.4.1.140.5.1.13
beaSmgrAuxSize	.1.3.6.1.4.1.140.5.1.14

Variable Name	Object ID
beaSmgrSemId	.1.3.6.1.4.1.140.5.1.15
beaSmgrShmId	.1.3.6.1.4.1.140.5.1.16

**beaSmgrIndex**

- Syntax Integer
- Access read-only
- Description Index into the table of shared memory blocks.

**beaSmgrIpcKey**

- Syntax DisplayString
- Access read-only
- Description Interprocess communication (IPC) key used by the block of shared memory.

**beaSmgrShmAllocated**

- Syntax Integer {  
no (1),  
yes (2) }
- Access read-only
- Description Status of the block of shared memory. This object can be monitored to detect when shared memory is corrupted or removed accidentally.

**beaSmgrSubsystem**

- Syntax DisplayString
- Access read-only
- Description Name of the subsystem using the block of shared memory.

## beaSmgrCreationDate

Syntax	DisplayString
Access	read-only
Description	Timestamp when the block of shared memory was allocated.

## beaSmgrLibVersion

Syntax	DisplayString
Access	read-only
Description	Software version of the shared memory manager library which was linked in to allocate the block of shared memory.

## beaSmgrMaxShmEntries

Syntax	Integer
Access	read-only
Description	Maximum number of entries configured for shared memory.

## beaSmgrCurrentShmEntries

Syntax	Integer
Access	read-only
Description	Current number of entries in shared memory.

## beaSmgrHWMShmUsed

Syntax	Integer
Access	read-only
Description	High-water mark of the number of entries used in shared memory.

## **beaSmgrPctShmUsed**

Syntax	Integer
Access	read-only
Description	Percentage of the current number of entries in shared memory that are in use.

## **beaSmgrHWMPctShmUsed**

Syntax	Integer
Access	read-only
Description	High-water mark of the percentage of entries in use in shared memory. This object can be used to determine if more entries need to be allocated without having to poll continuously for the maximum current number of entries.

## **beaSmgrTotalShmSize**

Syntax	Integer
Access	read-only
Description	Total size of the shared memory segment.

## **beaSmgrUsegSize**

Syntax	Integer
Access	read-only
Description	Size of the user segment in the shared memory segment.

## **beaSmgrAuxSize**

Syntax	Integer
Access	read-only
Description	Size of the auxiliary segment in the shared memory segment.

## **beaSmgrSemId**

Syntax	Integer
Access	read-only
Description	Semaphore ID used to arbitrate allocation of user segments within the shared memory segment.

## **beaSmgrShmId**

Syntax	Integer
Access	read-only
Description	Shared memory ID.

# beaSysPerf Group

Workstation performance attributes. This MIB group is supported by the `unix_snmpd` subagent shipped with the Agent Integrator. The objects in this group are similar to those supported by the Sun `perfmeter` on Sun workstations. The `beaSysPerfIfTable` object is described following the descriptions of all the scalar objects in this group.

Variable Name	Object ID
<code>beaSysPerfCpu</code>	<code>.1.3.6.1.4.1.140.11.1</code>
<code>beaSysPerfCntxt</code>	<code>.1.3.6.1.4.1.140.11.2</code>
<code>beaSysPerfSwap</code>	<code>.1.3.6.1.4.1.140.11.3</code>
<code>beaSysPerfDisk</code>	<code>.1.3.6.1.4.1.140.11.4</code>
<code>beaSysPerfIntr</code>	<code>.1.3.6.1.4.1.140.11.5</code>
<code>beaSysPerfLoad</code>	<code>.1.3.6.1.4.1.140.11.6</code>
<code>beaSysPerfPage</code>	<code>.1.3.6.1.4.1.140.11.7</code>
<code>beaSysPerfIfNumber</code>	<code>.1.3.6.1.4.1.140.11.8</code>
<code>beaSysPerfIfTable</code>	<code>.1.3.6.1.4.1.140.11.9</code>
<code>beaSysPerfCntxtDelta</code>	<code>.1.3.6.1.4.1.140.11.10</code>
<code>beaSysPerfSwapDelta</code>	<code>.1.3.6.1.4.1.140.11.11</code>
<code>beaSysPerfDiskDelta</code>	<code>.1.3.6.1.4.1.140.11.12</code>
<code>beaSysPerfIntrDelta</code>	<code>.1.3.6.1.4.1.140.11.13</code>
<code>beaSysPerfLoadDelta</code>	<code>.1.3.6.1.4.1.140.11.14</code>
<code>beaSysPerfPageDelta</code>	<code>.1.3.6.1.4.1.140.11.15</code>

## beaSysPerfCpu

Syntax	Integer
Access	read-only
Description	Percentage of CPU capacity being utilized between polls.

## beaSysPerfCntxt

Syntax	Integer
Access	read-only
Description	Number of context switches. This is a cumulative counter.

## beaSysPerfSwap

Syntax	Integer
Access	read-only
Description	Jobs swapped. This is a cumulative counter.

## beaSysPerfDisk

Syntax	Integer
Access	read-only
Description	Disk traffic in numbers of transfers in blocks. This is a cumulative counter.

## beaSysPerfIntr

Syntax	Integer
Access	read-only
Description	Number of device interrupts. This includes receive, transmit, and modem type interrupts from console, ports, and hiports boards. This is a cumulative counter.

### **beaSysPerfLoad**

Syntax Integer

Access read-only

Description The size of the run queue every second. This is a cumulative counter.

### **beaSysPerfPage**

Syntax Integer

Access read-only

Description The paging activity in the number of pages. This is a cumulative counter.

### **beaSysPerfIfNumber**

Syntax Integer

Access read-only

Description The number of network interfaces (regardless of their current state) present on this system.

### **beaSysPerfCntxtDelta**

Syntax Integer

Access read-only

Description The number of context switches. This represents the change since the last poll.

### **beaSysPerfSwapDelta**

Syntax Integer

Access read-only

Description The number of jobs swapped. This represents the change since the last poll.



## beaSysPerfDiskDelta

Syntax	Integer
Access	read-only
Description	Disk traffic in numbers of transfers in blocks. This represents the change since the last poll.

## beaSysPerfIntrDelta

Syntax	Integer
Access	read-only
Description	The number of device interrupts. This includes receive, transmit, and modem types of interrupts from the console, ports, and hiports boards. This represents the change since the last poll.

## beaSysPerfLoadDelta

Syntax	Integer
Access	read-only
Description	The size of the run queue every second. This represents the change since the last poll.

## beaSysPerfPageDelta

Syntax	Integer
Access	read-only
Description	The paging activity in the number of pages. This represents the change since the last poll.

# beaSysPerfIfTable

This tabular object within the `beaSysPerf` MIB group represents the physical interfaces on a system. Each entry (row) in the `beaSysPerfIfTable` is a sequence of the following columnar objects. Each row represents attributes of a particular interface. This table is supported by the `unix_snmpd` subagent shipped with the Agent Integrator. The last five columnar objects provide differences between last and current poll. This is intended to compensate for the fact that some network managers cannot effectively use absolute values to calculate differences needed for defining thresholds.

Variable Name	Object ID
<code>beaSysPerfIfIndex</code>	<code>.1.3.6.1.4.1.140.11.9.1.1</code>
<code>beaSysPerfIfDescr</code>	<code>.1.3.6.1.4.1.140.11.9.1.2</code>
<code>beaSysPerfIfOperStatus</code>	<code>.1.3.6.1.4.1.140.11.9.1.3</code>
<code>beaSysPerfIfInPackets</code>	<code>.1.3.6.1.4.1.140.11.9.1.4</code>
<code>beaSysPerfIfInErrors</code>	<code>.1.3.6.1.4.1.140.11.9.1.5</code>
<code>beaSysPerfIfOutPackets</code>	<code>.1.3.6.1.4.1.140.11.9.1.6</code>
<code>beaSysPerfIfOutErrors</code>	<code>.1.3.6.1.4.1.140.11.9.1.7</code>
<code>beaSysPerfIfCollisions</code>	<code>.1.3.6.1.4.1.140.11.9.1.8</code>
<code>beaSysPerfIfInPacketsDelta</code>	<code>.1.3.6.1.4.1.140.11.9.1.9</code>
<code>beaSysPerfIfInErrorsDelta</code>	<code>.1.3.6.1.4.1.140.11.9.1.10</code>
<code>beaSysPerfIfOutPacketsDelta</code>	<code>.1.3.6.1.4.1.140.11.9.1.11</code>
<code>beaSysPerfIfOutErrorsDelta</code>	<code>.1.3.6.1.4.1.140.11.9.1.12</code>
<code>beaSysPerfIfCollisionsDelta</code>	<code>.1.3.6.1.4.1.140.11.9.1.13</code>

## beaSysPerfIfIndex

Syntax	Integer
Access	read-only
Description	A unique number for each interface. The value ranges between 1 and <code>beaSysPerfIfNumber</code> . The value for each interface must remain constant between re-initializations of the entity's network management system.

## beaSysPerfIfDescr

Syntax	DisplayString (SIZE 0..255)
Access	read-only
Description	A string containing information about the interface. This should include the name of the manufacturer, the product name, and the version of the hardware interface.

## beaSysPerfIfOperStatus

Syntax	Integer { up (1), down (2), testing (3) }
Access	read-only
Description	The current operational state of the interface. An interface is up if it is ready to pass packets. An interface is not ready to pass packets if it is in the <code>testing</code> state.

## beaSysPerfIfInPackets

Syntax	Counter
Access	read-only
Description	The total number of packets received on the interface. This object differs from the MIB II <code>interfaces</code> group <code>ifInPackets</code> object in that the latter represents number of octets rather than packets.

### **beaSysPerflInErrors**

Syntax	Counter
Access	read-only
Description	The number of inbound packets that contained errors that prevented them from being delivered to a higher-layer protocol.

### **beaSysPerflOutPackets**

Syntax	Counter
Access	read-only
Description	The total number of packets transmitted out of the interface.

### **beaSysPerflOutErrors**

Syntax	Counter
Access	read-only
Description	The number of outbound packets that could not be transmitted because of errors. This differs from the MIB II <code>interfaces</code> group <code>ifOutErrors</code> object in that the latter represents octets rather than packets.

### **beaSysPerflCollisions**

Syntax	Counter
Access	read-only
Description	The number of collisions on CSMA interfaces.

## beaSysPerfIfInPacketsDelta

Syntax	Counter
Access	read-only
Description	The total number of packets received on the interface. This represents the change since the last poll. This differs from the MIB II <code>interfaces</code> group <code>ifInPackets</code> object in that the latter represents octets rather than packets.

## beaSysPerfIfInErrorsDelta

Syntax	Counter
Access	read-only
Description	The number of inbound packets that contained errors preventing them from being delivered to a higher-level protocol. This represents the difference since the last poll.

## beaSysPerfIfOutPacketsDelta

Syntax	Counter
Access	read-only
Description	The total number of packets transmitted out of the interface. This represents the change since the last poll.

## beaSysPerfIfOutErrorsDelta

Syntax	Counter
Access	read-only
Description	The number of outbound packets that could not be transmitted because of errors. This represents the change since the last poll. This object differs from the MIB II <code>interfaces</code> group <code>ifOutErrors</code> object in that the latter counts octets rather than packets.

**beaSysPerflfCollisionsDelta**

Syntax	Counter
Access	read-only
Description	The number of collisions on CSMA interfaces. This represents the change since the last poll.

# beaNTSysPerf Group

This MIB group represents system performance attributes specific to the Windows NT system. This MIB group is supported by the `nt_snmpd` subagent shipped with the Agent Integrator. This MIB group contains the following scalar objects.

Variable Name	Object ID
beaNTSysPerfCalls	.1.3.6.1.4.1.140.12.1.1
beaNTSysPerfAlignFix	.1.3.6.1.4.1.140.12.1.2
beaNTSysPerfExptDisp	.1.3.6.1.4.1.140.12.1.3
beaNTSysPerfFloatEmul	.1.3.6.1.4.1.140.12.1.4
beaNTSysPerfPgFaults	.1.3.6.1.4.1.140.12.1.5
beaNTSysPerfPaging	.1.3.6.1.4.1.140.12.1.6
beaNTSysPerfCacheBytes	.1.3.6.1.4.1.140.12.1.7
beaNTSysPerfCodeTotal	.1.3.6.1.4.1.140.12.1.8
beaNTSysPerfCodeRes	.1.3.6.1.4.1.140.12.1.9
beaNTSysPerfDriverTotal	.1.3.6.1.4.1.140.12.1.10
beaNTSysPerfDriverRes	.1.3.6.1.4.1.140.12.1.11
beaNTSysPerfCacheRes	.1.3.6.1.4.1.140.12.1.12
beaNTSysPerfPageUsed	.1.3.6.1.4.1.140.12.1.13
beaNTSysPerfPagePeak	.1.3.6.1.4.1.140.12.1.14

## **beaNTSysPerfCalls**

Syntax	Integer
Access	read-only
Description	This is a cumulative counter of calls to Windows NT system service routines. These routines perform all of the basic scheduling and synchronization tasks on the computer and provide access to non-graphical devices, memory management, and name-space management.

## **beaNTSysPerfAlignFix**

Syntax	Integer
Access	read-only
Description	This is a cumulative counter of all alignment faults fixed by the system.

## **beaNTSysPerfExptDisp**

Syntax	Integer
Access	read-only
Description	This is a cumulative counter of all exceptions dispatched by the system.

## **beaNTSysPerfFloatEmul**

Syntax	Integer
Access	read-only
Description	This is a cumulative counter of all floating emulations performed by the system.



## beNTSysPerfPgFaults

Syntax	Integer
Access	read-only
Description	This is a count of the page faults in the processor. A page fault occurs when a process refers to a virtual memory page that is not in its working set in main memory. A page fault will not cause the page to be fetched from disk if that page is on the standby list, and hence already in main memory, or if it is in use by another process when the page is shared.

## beNTSysPerfPaging

Syntax	Integer
Access	read-only
Description	The number of pages read from disk or written to the disk to resolve memory references to pages that were not in memory at the time of the reference. This is the sum of pages input and output. This counter includes paging traffic on behalf of the system cache to access file data for applications. This is the primary counter to observe if you are concerned about excessive pressure on memory (that is, memory thrashing), and the excessive paging that may result.

## beNTSysPerfCacheBytes

Syntax	Integer
Access	read-only
Description	Measures the number of bytes currently in use by the system cache. The system cache is used to buffer data retrieved from disk or local area network (LAN) resources. The system cache uses memory not in use by the active processes in the computer.

## beaNTSysPerfCodeTotal

Syntax	Integer
Access	read-only
Description	The number of kilobytes of pages that can be paged in <code>ntoskrnl.exe</code> , <code>hal.dll</code> , and the boot drivers and file systems loaded by <code>ntldr/osloader</code> .

## beaNTSysPerfCodeRes

Syntax	Integer
Access	read-only
Description	The number of kilobytes of system code currently resident in core memory. This is the working code set of the pagable executive. In addition to this, there is roughly an additional 300 kilobytes non-paged kernel code.

## beaNTSysPerfDriverTotal

Syntax	Integer
Access	read-only
Description	The number of kilobytes of pagable pages in all loaded device drivers other than those counted in <code>beaNTSysPerfCodeTotal</code> .

## beaNTSysPerfDriverRes

Syntax	Integer
Access	read-only
Description	The number of kilobytes of <code>beaNTSysPerfDriverTotal</code> currently resident in core memory. This number is the working code set of the pagable drivers. In addition to this, there is roughly another 300 kilobytes of non-paged driver code.

## beaNTSysPerfCacheRes

Syntax Integer

Access read-only

Description The number of bytes currently resident in the global disk cache.

## beaNTSysPerfPageUsed

Syntax DisplayString

Access read-only

Description The percent of the page file instance that is in use.

## beaNTSysPerfPagePeak

Syntax DisplayString

Access read-only

Description The peak usage of the page file instance in percent.

# beaIntAgt Group

The only object in the `beaIntAgt` group is the BEA intelligent agent table.

## beaIntAgtTable

This MIB group is supported directly by the Agent Integrator master agent. The `beaIntAgtTable` permits the addition or deletion or modification of `RULE_ACTION` entries that are used to control local polling by the Agent Integrator. Each entry (row) in the `beaIntAgtTable` supports a particular polling rule. The `beaIntAgtStatus` object is a read-write object that can be used to de-activate or activate polling execution by particular rules. For more information about Agent Integrator polling, refer to Chapter 4, “Using Agent Integrator for Polling.”

Variable Name	Object ID
<code>beaIntAgtRuleId</code>	<code>.1.3.6.1.4.1.140.200.1.1</code>
<code>beaIntAgtScanIntvl</code>	<code>.1.3.6.1.4.1.140.200.1.2</code>
<code>beaIntAgtRuleAction</code>	<code>.1.3.6.1.4.1.140.200.1.3</code>
<code>beaIntAgtStatus</code>	<code>.1.3.6.1.4.1.140.200.1.4</code>

### beaIntAgtRuleId

Syntax	DisplayString
Access	read-write
Description	A name of the rule that must be unique within the table. This provides an index into the table. The string must not be longer than eight characters.

## bealntAgtScanIntvl

Syntax	Integer
Access	read-write
Description	Polling interval in seconds. This is the length of delay from the previous evaluation of the rule entry.

## bealntAgtRuleAction

Syntax	DisplayString
Access	read-write
Description	This string contains the actual rule that defines the threshold to be checked and the action to be taken if there is a change of state in the rule. There are two possible states for the rule, OK or ERR. If the rule condition is satisfied, the state is ERR. If the rule condition is not satisfied, the state is OK. Actions are executed only if a transition from one state to the other occurs. RULE_ACTION entries are explained in Chapter 4, “Using Agent Integrator for Polling.” The syntax of the RULE_ACTION entry can be found in Chapter 7, “Configuration Files.”

## bealntAgtStatus

Syntax	Integer { active(1), invalid (2), inactive (3) }
Access	read-write
Description	Status of the row (RULE_ACTION entry). Polling is active if and only if the status is <i>active</i> . If the status of the rule is set to <i>invalid</i> , the row (RULE_ACTION entry) is deleted. Polling is disabled without deleting the RULE_ACTION entry if the row status is set to <i>inactive</i> .



# A SNMP Information

This appendix deals with some frequently expressed questions and concerns about the SNMP protocol and Management Information Bases (MIBs). It includes the following sections:

- ◆ “Reference Books” lists books with additional information about MIBs, agents, or the SNMP protocol.
- ◆ “Obtaining MIBs” lists vendor-specific MIBs and MIBs under development.
- ◆ “Enterprise ID Assignment” lists information on where to obtain an enterprise ID assignment if you want to develop your own private MIB.
- ◆ “Obtaining RFCs” contains information on how and where to obtain Requests for Comments (RFCs).
- ◆ “Obtaining Specifications” lists information on how and where to obtain specifications for SNMP.
- ◆ “Mailing Lists and News Groups” lists news and mail groups that provide general information about SNMP.
- ◆ “Standards and Drafts” lists the available standards and drafts for SNMP.
- ◆ “Accessing Internet-Drafts” lists all internet draft directories that are available world-wide.

## Reference Books

If you need additional information about MIBs, agents, or the SNMP protocol, refer to these books:

- ◆ Comer, Douglas; *Internetworking with TCP/IP, Vol. 2*; Prentice-Hall, Englewood Cliffs, New Jersey, 1991
- ◆ Leinwand, Allan and Fang, Karen; *Network Management: A Practical Perspective*; Addison-Wesley, Reading, Massachusetts, 1993
- ◆ Rose, Marshall T.; *The Simple Book: An Introduction to Management of TCP/IP-based Internets*; Prentice-Hall, Englewood Cliffs, New Jersey, 1991
- ◆ Rose, Marshall T.; *The Open Book: A Practical Perspective on Open Systems Interconnection*; Prentice-Hall, Englewood Cliffs, New Jersey, 1989
- ◆ Miller, Mark; *Managing Internetworks with SNMP*, M & T Books
- ◆ Stallings, William; *SNMP, SNMPv2 and CMIP: The Practical Guide to Network Management Standards*, Addison-Wesley, Reading, Massachusetts, 1993

## Obtaining MIBs

The following is a list of vendor-specific MIBs and MIBs under development. These MIBs are available via anonymous FTP at the indicated IP address:

- ◆ venera.isi.edu [128.9.0.32] in directory mib (for vendor MIBs)
- ◆ nic.ddn.mil [192.67.67.20] in directory internet-drafts
- ◆ nnsc.nsf.net [192.31.103.6] in directory internet-drafts
- ◆ munnari.oz.au [128.250.1.21] in directory internet-drafts (Pacific Rim)
- ◆ nic.nordu.net [192.36.148.17] in directory internet-drafts (Europe)



---

# Enterprise ID Assignment

If you want to develop your own private MIB, you must obtain an enterprise ID assignment from the Internet Assigned Numbers Authority.

Contact	Joyce K. Reynolds
Mailing Address	Internet Assigned Numbers Authority USC/Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292-6695
Phone	+1.213.822.1511
e-mail	iana@isi.edu

## Obtaining RFCs

You can obtain Requests for Comments in the following ways:

- ◆ Download them from almost anywhere on the Internet
- ◆ Obtain them from SRI International

Mailing Address	SRI International, EJ291 DDN Network Information Center 333 Ravenswood Ave. Menlo Park CA 94025
Phone	+1.800.235.3155
e-mail	MAIL-SERVER@nisc.sri.com Leave the subject field blank. In the body, enter: SEND RFCnnnn.TXT-1
FTP	ftp://ftp.nisc.sri.com/rfc/rfcNNNN.txt

# Obtaining Specifications

If you need specifications for SNMP, refer to the following sources:

◆ IEEE and ISO/IEC[IEEE] Standards

---

Mailing Address	Service Center 445 Hoes Lane PO Box 1331 Piscataway NJ 08855-1331
Phone	+1.800.678.4333

---

◆ IEEE drafts

---

Mailing Address	IEEE Computer Society Documents c/o AlphaGraphics ATTN: P. Thrush 10215 N. 35th Ave., Suite A & B Phoenix AZ 85051
-----------------	--

---

◆ ISO and ISO/IEC documents

---

Mailing Address	American National Standards Institute 1430 Broadway New York NY 10018 USA
-----------------	---

---

## OSI NMF Documents

◆ ITU-T (formerly CCITT) Blue Book documents:

---

FTP	<a href="ftp://bruno.cs.colorado.edu">ftp://bruno.cs.colorado.edu</a> <a href="ftp://gatekeeper.dec.com/pub/bruno.cs.colorado.edu/pub/standards">ftp://gatekeeper.dec.com/pub/bruno.cs.colorado.edu/pub/standards</a> <a href="ftp://ftp.uu.net/doc/standards">ftp://ftp.uu.net/doc/standards</a>
-----	---

---

FTP	Europe: <a href="ftp://src.doc.ic.ac.uk/doc/ccitt-standards">ftp://src.doc.ic.ac.uk/doc/ccitt-standards</a> <a href="ftp://nic.ja.net/doc/ccitt-standards">ftp://nic.ja.net/doc/ccitt-standards</a>
-----	---

---

## Mailing Lists and News Groups

The following news and mail groups provide general information about SNMP. Also, contact BEA Manager Customer Support. (See the customer support card included in your BEA Manager product box).

SNMP in general

[snmp-request@uu.psi.com](mailto:snmp-request@uu.psi.com)

RMON MIB

[rmonmib-request@lexcel.com](mailto:rmonmib-request@lexcel.com)

Security issues

[snmp-sec-dev-request@tis.com](mailto:snmp-sec-dev-request@tis.com)

Device discovery

[finder-request@emerald.acc.com](mailto:finder-request@emerald.acc.com)

# Standards and Drafts

The following standards and drafts are available for SNMP.

RFC Number	Description
052	IAB Recommendations
1089	SNMP over Ethernet
1109	Ad-hoc Review
1155	Structure of Management Information
1156	Management Information Base (MIB-I)
1157	SNMP
1161	SNMP over OSI
1187	Bulk table retrieval
1212	Concise MIB definitions
1213	Management Information Base (MIB-II)
1214	OSI MIB
1215	Traps
1227	SNMP Multiplex (SMUX)
1228	SNMP-DPI
1229	Generic-interface MIB extensions
1230 IEEE 802.4	Token Bus MIB
1231 IEEE 802.5	Token Ring MIB
1239	Reassignment of MIBs
1243	AppleTalk MIB
1248	OSPF MIB
ISO 8824	ASN.1
ISO 8825	BER for ASN.1

# Accessing Internet-Drafts

Internet-Drafts are available by anonymous FTP. Log in with the username `anonymous` and password `guest`. After logging in, type `cd internet-drafts`.

Internet-Drafts directories are located at:

US East Coast	<code>ftp://ds.intermic.net/internet-drafts</code>
US West Coast	<code>ftp://ftp.isi.edu/internet-drafts</code>
Pacific Rim	<code>ftp://munnari.oz.au/internet-drafts</code>
Europe	<code>ftp://nic.nordu.net/internet-drafts</code>

Internet-Drafts are also available by mail. Send e-mail to `mailserv@ds.intermic.net`. In the body, type:

```
FILE/internet-drafts-ietf-rdbmsmib-mib-02.txt
```

For questions, please send e-mail to:

```
Internet-Drafts@cnri.seston.va.us
```



---

# Glossary

## **absolute OID**

An *object identifier* (OID) that specifies a unique path to a managed object from the root of the OID tree.

## **Abstract Syntax Notation One (ASN.1)**

A formal notation used to define data types and encode data values. A language that describes the data structures that make up an abstract syntax. ITU-T (formerly CCITT) specification X.409 is equivalent to ASN.1. ASN.1 is used to define a management information base (MIB). ASN.1 is a common requirement of the SNMP and CMIP network management protocols.

## **agent**

1) A component of a network management system that exchanges data about managed objects with a manager at a network management workstation. Agents provide a software interface to managed resources, and gather data about them at a manager's request. 2) In a two-phase commit syncpointing sequence (LU6.2 or MRO), a task that receives syncpoint requests from an initiator (a task that initiates syncpointing activity). 3) See *SNMP agent*.

## **agent-manager model**

A model where a manager communicates with many distributed agents via a system management protocol.

## **alarm**

A means of reporting that a managed object is in an abnormal state (i.e., a managed object has passed a predefined threshold).

## **API**

See *application programming interface (API)*.

---

**application programming interface (API)**

1) The verbs and environment that exist at the application level to support a particular system software product. 2) A set of code that enables a developer to initiate and complete client/server requests within an application. 3) A set of calling conventions that define how to invoke a service.

**architecture**

The structure and interrelationship of components in a system or in an environment.

**ASN.1**

See *Abstract Syntax Notation One (ASN.1)*.

**atomic set**

A behavior of SNMP agents where, if an SNMP agent receives an SNMP set request containing more than one variable, it either sets all requested objects or none are set. This behavior is a requirement of the SNMP standard.

**bandwidth**

The transmission capacity of a computer or communications channel.

**BEA TUXEDO**

A robust middleware engine for developing and deploying business-critical client/server applications. It handles distributed transaction processing, application messaging, and the full complement of services necessary to build and run enterprise-wide applications.

**client/server**

1) A model used in distributed systems where one host acts as a system server, and the other host acts as a client. 2) A distribution model in which there are two types of applications: client applications that request that tasks be performed, and server applications that perform those tasks. 3) A programming model in which application programs are structured as clients or servers. A client program is an application program that requests services to be performed. A server program is an entity that dispatches service routines to satisfy requests from client programs. A service routine is an application program module that performs one or more specific functions on behalf of client programs.

**CMIP**

See *Common Management Interface Protocol*.



---

**columnar object**

A MIB “leaf” object — that is, a MIB object that does not have any objects below it in the OID tree — which can have zero or more instances. A columnar object represents one column in a table.

**Common Management Interface Protocol (CMIP)**

An protocol for network management defined by ISO standards.

**database**

A collection of interrelated or independent data items stored together without redundancy to serve one or more applications.

**database management system (DBMS)**

A program or set of programs that let users structure and manipulate the data in the tables of a database. A DBMS ensures privacy, recovery, and integrity of data in a multi-user environment.

**DBMS**

See *database management system (DBMS)*.

**Distributed Program Interface (DPI)**

The Distributed Program Interface (DPI) protocol extension to SNMP agents that permits end-users to dynamically add, delete or replace variables in the local MIB without requiring recompilation of the SNMP agent by creating a subagent that communicates with the agent via the SNMP-DPI protocol.

**DPI subagent**

See *Distributed Program Interface*.

**Factory**

An interface used by a client to obtain an object reference to a CORBA object. Object references to factories are obtained by the client using an object reference to a FactoryFinder interface. The FactoryFinder interface is advertised by the system and is made available to the client as part of client bootstrap processing.

**Factory-based routing**

A feature of M3 used to distribute processing to specific server groups. Routing is done when a factory creates an object reference in its call to a TP framework. The framework executes the routing algorithm based on the routing criteria specified by the administrator.

---

**field**

1) In a record, a specified area used for a particular category of data. 2) An area within a segment that is the smallest referable unit of data. 3) Any designated portion of a segment. 4) A way of addressing a single item of data in a database table. A field is also an area of a window where data displays.

**graphical user interface (GUI)**

A high-level interface that uses windows and menus with graphic symbols instead of system commands typed at a prompt to provide an interactive environment for a user.

**group**

See *MIB group*.

**GUI**

See *graphical user interface (GUI)*.

**host**

A computer that is attached to a network and provides services other than acting as a communication switch.

**host computer**

The primary or controlling computer in a data communication system.

**ident string**

See *identification string*.

**identification string**

Portions of a file that get expanded by RCS and BEA Manager utilities to contain file and identification information. If compiled, these strings are placed into object file functions, where the information is made available.

**instrumentation**

Facilities that provide access to the attributes of managed resources, to retrieve or modify values of these attributes. Access to managed resources used by agents to respond to management requests.

---

## **International Organization for Standardization (ISO)**

An international organization whose membership includes standards and research groups from various nations. ISO establishes standards for computer network communications and many other technologies.

## **ISO**

See *International Organization for Standardization (ISO)*.

## **managed object**

A software entity, defined within the *management information base*, that represents a feature of a *managed resource* (such as a process, a piece of hardware, or system performance attribute) and is controlled by a management device.

## **managed resource**

The physical resource whose attributes are represented by *managed objects* in a *management information base*. A managed resource can be a software entity such as an application or queue, or a hardware device, such as an interface card or hub.

## **management framework**

A system that provides a unified view of hardware and software resources on distributed systems and enterprise-wide networks to aid network administrators to manage and control these resources.

## **management information base (MIB)**

1) A virtual storage database that uses ASN.1 notation. The MIB contains each object that the System Manager software monitors and controls. These objects are written in ASN.1 notation. Each has a unique object name and a unique object identifier. 2) A TUXEDO System component that provides a complete definition of the object classes and their attributes that together comprise the TUXEDO System. The total TUXEDO System management information base is organized into a generic MIB and component-specific MIBs for each major component. Configuration and administration of the TUXEDO System can be done programmatically.

## **mask**

An SNMP means for hiding selected SNMP traps, so that alarms are generated only for specified instances.

---

**master agent**

The single point of contact for the SNMP manager on a managed node. The master agent receives requests from the SNMP manager and contacts the appropriate subagents to fulfill the requests.

**message (log message)**

A means for sending data and values across applications. Messages represent statistical or status information about application processes, and consist of a header, containing message ID data, and a body containing user-defined information.

**message definition block**

The total body of data that comprises a message definition, such as the command name, the subsystem name, and the internal and external recommendations.

**MIB**

See *management information base (MIB)*.

**MIB group**

Ancestor object of MIB objects within the OID (or registration) tree. A MIB group may contain other MIB groups, or it may contain scalar or tabular objects.

**monitor**

Allows an SNMP capable management station to watch the status of the TUXEDO system.

**MOPS**

Management operations per second

**object identifier (OID)**

A unique number assigned to each object in the MIB. These OIDs fall into specific categories and form a tree. When the SNMP agent accesses a specific object, it traverses the OID tree in the MIB file to find the object. An OID identifies an object by specifying a unique path to the object from the root of the OID tree.

**OID**

See *object identifier (OID)*.

**OLTP**

See *online transaction processing (OLTP)*.

---

**online transaction processing (OLTP)**

The execution of units of work in a performance-critical environment that appears to the user as immediate; real-time; usually having internal recoverability, history-keeping and consistency-assurance features.

**Open Systems Interconnection (OSI)**

A consortium that facilitates communication between different types of computer systems.

**OSI**

See *Open Systems Interconnection (OSI)*.

**PID**

See *process ID (PID)*.

**polling**

An activity where a manager interrogates an agent at a periodic interval. The agents report the values of the specified managed objects. Checking at periodic intervals to determine if a managed object value has crossed a specified threshold.

**private MIB**

A MIB that is defined under the private MIB directory

**process ID (PID)**

A unique number that identifies a process.

**relative OID**

An *object identifier* (OID) that specifies a path to a managed object only relative to some node in the OID tree below root.

**requester**

A process that receives messages from clients, converts these messages to a common internal form, determines the appropriate server or servers for the transaction request, and forwards the request to a server.

**scalar object**

A MIB “leaf” object — that is, a MIB object that does not contain other MIB objects below it in the OID tree — which can have only one instance.

---

**server**

1) Software that performs a task requested of it by a client. 2) In client/server terminology, a server application typically stores and manipulates the data, as opposed to the client, which contains the user interface. 3) A software module that accepts requests from clients and other servers and returns responses.

**Simple Network Management Protocol (SNMP)**

A de facto standard network management protocol developed by the Internet community.

**SMUX**

Stands for SNMP multiplexing. A protocol for master agent/subagent communication defined by RFC 1227.

**SMUX subagent**

The SNMP Multiplexing (SMUX) protocol allows the creation of subagents that communicate with the agent and resolve management operations for specific objects in the MIB module.

**SNMP**

See *Simple Network Management Protocol (SNMP)*.

**SNMP agent**

An agent using the SNMP protocol to exchange data with a system manager.

**standard MIB**

A MIB developed as a standard by the Internet community. Examples are MIB I and MIB II.

**subagent**

A component of the master agent protocol that fulfills requests and replies to the master agent.

**system manager**

The part of a network management system that requests data from an agent and takes actions based on that data.

**TCP/IP**

See *Transmission Control Protocol/Internet Protocol (TCP/IP)*.

---

**token**

An individual element in the message definition block, such as the command or the subsystem name.

**Transmission Control Protocol/Internet Protocol (TCP/IP)**

- 1) A provider of network services that is supported by the transport layer interface.
- 2) Communications protocol standard.

**trap**

An SNMP data packet containing information about an error that occurred with a managed object. Traps are unsolicited event notifications, that is, notifications generated by an agent on its own initiative.

**trap notification**

See *trap*.

**UDP**

See *user datagram protocol (UDP)*.

**user datagram protocol (UDP)**

The TCP/IP datagram transport layer protocol.





---

# Index

## A

- absolute OIDs 1-8
- access to managed objects
  - through SMUX subagents 3-4
- access, to MIB objects
  - specifying in NON\_SMUX\_PEER entries 7-8
- actions
  - executing a command 7-17
  - in response to polling 4-13
- actions, in polling
  - syntax of 7-14
- Agent Integrator
  - capabilities of 1-1, 3-1
  - commands 5-1
  - configuration file 7-2
  - configuring for access to agents 3-3
  - functionality of 1-3
  - how to use local polling feature of 4-1
  - re-initializing 5-1
  - setting up 2-6
  - stopping 5-4
  - supported MIB objects 1-10
  - types of agents supported 1-4
- alarms from local threshold-checking 7-17
- AND, use in defining polling thresholds 4-8
- architecture of SNMP 1-3
- asterisks
  - use in complex polling rules 4-11

## B

- bea.asn1
  - default location of 2-7
- BEA\_PEER\_MAX\_TRIES 1-12
- BEA\_PEER\_MAX\_WAIT 1-12, 7-9
- BEA\_SM\_BEAMGR\_CONF 1-12, 7-2
- BEA\_SMUX\_PASSWD 1-12
- beaDfAvail 8-14
- beaDfCapacity 8-14
- beaDfFilesystem 8-14
- beaDfHostname 8-13
- beaDfIndex 8-13
- beaDfKbytes 8-14
- beaDfMounted 8-15
- beaDfTable 8-13
- beaDfUsed 8-14
- beaIntAgtRuleAction 8-51
- beaIntAgtRuleId 8-50
- beaIntAgtScanIntvl 8-51
- beaIntAgtStatus MIB object 8-51
- beaIntAgtTable 8-50
  - role of 1-10
- beaLclDfAvail 8-30
- beaLclDfCapacity 8-30
- beaLclDfFilesystem 8-29
- beaLclDfKbytes 8-29
- beaLclDfMountedon 8-30
- beaLclDfTable 8-29
- beaLclDfUsed 8-30
- beamgr.conf 7-2, 8-4, 8-5
- beamgr\_snmpd.conf 7-18

---

beaMqCbytes 8-19  
beaMqCgroup 8-19  
beaMqCreator 8-19  
beaMqCtime 8-21  
beaMqGroup 8-19  
beaMqId 8-18  
beaMqKey 8-18  
beaMqLrpId 8-20  
beaMqLspid 8-20  
beaMqMode 8-18  
beaMqOwner 8-18  
beaMqQbytes 8-20  
beaMqQnum 8-19  
beaMqRtime 8-20  
beaMqStime 8-20  
beaMqT 8-18  
beaNTSysPerf MIB group 8-45  
beaNTSysPerfAlignFix 8-46  
beaNTSysPerfCacheBytes 8-47  
beaNTSysPerfCacheRes 8-49  
beaNTSysPerfCalls 8-46  
beaNTSysPerfCodeRes 8-48  
beaNTSysPerfCodeTotal 8-48  
beaNTSysPerfDriverRes 8-48  
beaNTSysPerfDriverTotal 8-48  
beaNTSysPerfExptDisp 8-46  
beaNTSysPerfFloatEmul 8-46  
beaNTSysPerfPagePeak 8-49  
beaNTSysPerfPageUsed 8-49  
beaNTSysPerfPaging 8-47  
beaNTSysPerfPgFaults 8-47  
beaPsCmd 8-11  
beaPsCpu 8-10  
beaPsCpuTime 8-12  
beaPsDate 8-11  
beaPsHandleCount 8-12  
beaPsMem 8-10  
beaPsPid 8-10  
beaPsRss 8-10  
beaPsSize 8-10  
beaPsStat 8-11

beaPsTable 8-9  
beaPstatSwapAlloc 8-15  
beaPstatSwapAvail 8-16  
beaPstatSwapReser 8-15  
beaPstatSwapUsed 8-15  
beaPsThreadCount 8-12  
beaPsTty 8-11  
beaPsUser 8-9  
beaSemCgroup 8-28  
beaSemCtime 8-28  
beaSemGroup 8-27  
beaSemId 8-27  
beaSemKey 8-27  
beaSemMode 8-27  
beaSemNsems 8-28  
beaSemOtime 8-28  
beaSemOwner 8-27  
beaSemT 8-26  
beaSemTable 8-26  
beaShmAtime 8-25  
beaShmCgroup 8-24  
beaShmCpid 8-25  
beaShmCreator 8-24  
beaShmCtime 8-25  
beaShmDtime 8-25  
beaShmGroup 8-24  
beaShmId 8-23  
beaShmKey 8-23  
beaShmLpid 8-25  
beaShmMode 8-23  
beaShmNattch 8-24  
beaShmOwner 8-23  
beaShmSegsz 8-24  
beaShmT 8-23  
beaSmgrAuxSize 8-34  
beaSmgrCreationDate 8-33  
beaSmgrCurrentShmEntries 8-33  
beaSmgrHWMpctShmUsed 8-34  
beaSmgrHWMShmUsed 8-33  
beaSmgrIndex 8-32  
beaSmgrIpcKey 8-32

---

beaSmgrLibVersion 8-33  
beaSmgrMaxShmEntries 8-33  
beaSmgrPctShmUsed 8-34  
beaSmgrSemId 8-35  
beaSmgrShmAllocated 8-32  
beaSmgrShmId 8-35  
beaSmgrSubsystem 8-32  
beaSmgrTable 8-31  
beaSmgrTotalShmSize 8-34  
beaSmgrUsegSize 8-34  
beaSysAgentVersion 8-7  
beaSysContact 8-4  
beaSysDescr 8-3  
beaSysHasDisk 8-6  
beaSysId 8-3  
beaSysInstallTime 8-3  
beaSysLocation 8-4  
beaSysMachine 8-7  
beaSysName 8-4  
beaSysNodeName 8-6  
beaSysPerf MIB group 8-36  
beaSysPerfCntxt 8-37  
beaSysPerfCntxtDelta 8-38  
beaSysPerfCpu 8-37  
beaSysPerfDisk 8-37  
beaSysPerfDiskDelta 8-39  
beaSysPerfIfCollisions 8-42  
beaSysPerfIfCollisionsDelta 8-44  
beaSysPerfIfDescr 8-41  
beaSysPerfIfIndex 8-41  
beaSysPerfIfInErrors 8-42  
beaSysPerfIfInErrorsDelta 8-43  
beaSysPerfIfInPackets 8-41  
beaSysPerfIfNumber 8-38  
beaSysPerfIfOperStatus 8-41  
beaSysPerfIfOutErrors 8-42  
beaSysPerfIfOutErrorsDelta 8-43  
beaSysPerfIfOutPackets 8-42  
beaSysPerfIfOutPacketsDelta 8-43  
beaSysPerfIfPacketsDelta 8-43  
beaSysPerfIfTable 8-40  
beaSysPerfIntr 8-37

beaSysPerfIntrDelta 8-39  
beaSysPerfLoad 8-38  
beaSysPerfLoadDelta 8-39  
beaSysPerfPage 8-38  
beaSysPerfPageDelta 8-39  
beaSysPerfSwap 8-37  
beaSysPerfSwapDelta 8-38  
beaSysRelease 8-6  
beaSysServices 8-4  
beaSysSysName 8-6  
beaSystem MIB group 8-2  
    objects in 8-2  
beaSysVersion 8-6  
beaTrapDescr 8-5  
beaTrapHost 8-5  
beaTrapPort 8-5  
beaUnix MIB group 8-8

## C

columnar object  
    what it is 4-10, 7-16  
COMMAND\_ERR 4-13  
COMMAND\_OK 4-13  
commands  
    as an action in polling 4-14  
    specifying more than one 4-14  
community strings  
    configuring Agent Integrator to use with  
        agents 3-3  
    specifying in passwords file 7-18  
compilation  
    of MIB on management platform 2-7  
condition  
    syntax for 4-8  
conditions, complex 4-7  
conditions, simple 4-7  
configuration file  
    environment variable for 1-12  
    for Agent Integrator 7-2  
    for BEA Manager 7-2  
configuring for access to agents 3-3

---

## D

- data types
  - for polling conditions 4-8
  - in polling thresholds 4-8
- disk accesses
  - monitoring on Windows NT systems 8-47
- dot-dot notation 1-9
- DPI 1-1
  - subagents 1-4

## E

- em\_snmpd
  - MIB groups supported by 6-1
- em\_snmpd, MIB groups supported by 1-10
- enterprise OID
  - in traps generated by local polling 4-14
- environment variables
  - summary of 1-12
- ERR state of a polling rule 4-12

## F

- file system attributes
  - as MIB table 8-13
- file systems, local
  - as MIB objects 8-29

## H

- host contact person
  - MIB object for 8-4
- hostname
  - as MIB object 8-4

## I

- instance
  - of a managed object 4-5
- instance index
  - in an OID 4-5

INTEGRATOR\_MAX\_TIMEOUTS entry 7-4

INTEGRATOR\_TIMEOUT 7-4

interfaces

- features of that can be polled with beaSysPerf MIB group 8-40

IP address 1-3

- specifying for remote agents 7-8
- use by Integrator in managing remote agents 3-5
- use for access to remote agents 3-5
- use in polling thresholds 4-8

IP addresses

- used by Agent Integrator 5-2

ISO 1-2

## L

location, of system

- as MIB object 8-4

logical agent name

- defining 7-4

## M

managed objects

- configuring Agent Integrator access to 3-3
- making accessible to Agent Integrator 4-1
- what they are 1-3

managed resource

- what it is 1-3

Management Information Base

- See MIB 1-3

master agents 1-4

message queue table

- as MIB table 8-17

MIB

- for BEA Manager agents 8-1
- role in system management 1-3
- what it does 1-2

---

MIB file, for TUXEDO  
    default location of 2-7

MIB groups  
    in BEA Agent MIB 8-1  
    supported by subagents 1-10, 6-1

MIB objects  
    directly supported by Agent Integrator 4-2

## N

network management platforms  
    role of 1-3

NON\_SMUX PEER  
    configuring Agent Integrator for 3-3

NON\_SMUX\_PEER entries  
    in configuration file 3-3  
    role of 4-2

NON\_SMUX\_PEER entry  
    syntax of 7-8

NT, Windows  
    MIB objects specific to 8-12, 8-45  
    polling for memory thrashing on 8-47  
    system performance attributes of as MIB objects 8-45

## O

object identifier  
    See OID 1-7

OID  
    expressing with textual names 1-9  
    format of in specifying objects to be polled 7-15  
    relative and absolute 1-8  
    use with Agent Integrator polling rules 1-9  
    what it is 1-7

OID tree  
    what it is 1-7  
    when agents overlap in 3-4

OID\_CLASS entry  
    syntax of 7-13

OK state of a polling rule 4-12

## P

page faults  
    as MIB objects 8-47

password, of SMUX agents  
    environment variable for 1-12

passwords file 7-18

peer SNMP agents  
    use in polling 4-6

polling  
    a specific instance 4-15  
    activating or de-activating 8-51  
    availability of MIB objects for 4-5, 4-7  
    data types in 4-8  
    how to de-activate 4-17  
    re-activating 4-18  
    relations for defining thresholds in 4-4  
    starting 4-16  
    use of deltas in 8-40  
    use of peer SNMP agents in 4-6

polling interval  
    as MIB object 8-51

polling rules  
    as MIB objects 8-50  
    creating through a manager 4-16  
    deleting or modifying 4-17  
    for Agent Integrator 4-2

polling, by Agent Integrator  
    how to use 4-1  
    of agents on remote nodes 3-5

port  
    specifying for peer agents 7-8

port 161  
    use of for peer SNMP agents 3-3

priority  
    of agent access to MIB objects 7-8  
    of SMUX subagents 7-13

---

- setting 7-9
- use in resolving overlap in agent OID registration 3-4

processes, attributes of

- as objects in MIB process table 8-9

## R

read-write access to MIB objects 7-8

reinit\_agents 5-1

relations

- use of defining thresholds 4-4

relative OIDs 1-8

remote nodes

- managing agents on 3-5

retries

- environment variable for 1-12

RFC 1227 1-4

RULE\_ACTION entry

- syntax of 7-14

rules, for polling 4-3, 7-14

## S

scalar object

- what it is 4-5, 4-10, 7-16

semaphores

- as MIB objects 8-26

SET

- used to turn off polling 4-17

set up

- of Agent Integrator 2-6

shared memory, attributes of

- as MIB objects 8-22
- as MIB table 8-31

Simple Network Management Protocol

- See SNMP 1-2

SMUX 1-1

- automatic registration of OID tree under 3-4
- what it is 1-4

SMUX subagent passwords

- specifying in passwords file 7-19

SMUX subagents

- modifying management scope of 7-13
- registration of managed objects with master 3-4

SNMP agents

- more than one on a single host 1-5
- on multiple nodes 3-5
- use in Integrator polling 4-7

SNMP agents, managing

- on remote nodes 3-5

SNMP architecture 1-2

SNMP trap notification

- TRAP\_HOST determines destination of 7-3

SNMP traps 7-3

SQL database, agents for 1-3

state, change of

- in polling 4-12

states

- of polling rules 4-12

status, of polling rules

- as MIB object 8-51

stop\_agents command 5-4

stopping agents 2-9

subagents 1-4

- use in polling 4-5

support

- customer xix
- documentation xix

support, customer xix

support, documentation xix

swap space

- attributes of as MIB objects 8-15

swap space statistics

- as MIB objects 8-15

syntax

- of actions when polling 7-14
- of NON\_SMUX\_PEER entry 7-8
- of OID\_CLASS entry 7-13
- of RULE\_ACTION entry 7-14

---

- SYS\_CONTACT 7-7
- SYS\_DESCR 7-7
- SYS\_INSTALL 7-7
- SYS\_LOCATION 7-7
- SYS\_NAME 7-7
- SYS\_SERVICES 7-7
- SYSOBJID 7-7
- SYSSERVICES 7-7
- system attributes
  - as MIB objects 8-2
- system statistics
  - as MIB objects 8-15

## T

- timeout
  - environment variable for 1-12
  - for replies to Agent Integrator from SNMP agents 7-9
- TAGENT entry 7-4
- TMEVENT\_FILTER entry 7-5
- transition
  - of polling rules 4-12
  - what it is 4-12
- trap destination
  - as MIB object 8-5
- trap notification
  - what it is 1-3
- TRAP\_HOST entry 7-3
  - components making use of 7-3
  - more than one allowed 7-3
- TRAPID\_ERR 4-13
- TRAPID\_OK 4-13

- traps
  - sending to multiple destinations 7-3
- traps, enterprise-specific
  - use in polling 7-17
- traps, polling
  - information passed in 4-14
- TUXEDO event filters 7-5
- TUXEDO system events
  - subscribing for 7-5

## U

- unix\_snmpd
  - configuration file keywords used by 7-7
  - MIB groups supported by 1-10, 6-1
- unsolicited messages from agents 1-3

## V

- VAL, use of in defining thresholds 4-4
- variable bindings
  - of trap generated by Agent Integrator 8-5
- variable bindings, of Agent Integrator traps
  - information in 4-14

## W

- wildcards
  - use in OIDs 4-10, 7-16
- Windows NT
  - see also NT, Windows 8-45

