EMC® Smarts® Health Monitor

Version 9.2

User Guide

P/N 300-999-861
REV 01

EMC²

Copyright © 2009 - 2013 EMC Corporation. All rights reserved. Published in the USA.

Published March, 2013

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC^2, EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to the technical documentation and advisories section on EMC Online Support.

CONTENTS

Preface

Chapter 1 Introduction

Overview ... 16
Software components ... 17
Software deployment ... 18
System dependencies ... 19
SHM agent support ... 19
Architecture ... 20
Automatic dynamic thresholding ... 23
- Requirements ... 23
- Operation ... 23
- Benefits ... 24
- Disabling the feature ... 24

Installation ... 25
Service command ... 27
- Startup ... 28
- Shutdown ... 28

Configuration ... 28
- Health Monitor configuration tasks ... 28
- Global Manager configuration tasks ... 28

Discovery ... 29
Monitoring ... 30
Analysis ... 30
Notifications ... 31

Chapter 2 Configuring the Health Monitor

Before you start ... 34
Configuration directory ... 34
User configuration parameters ... 34
Methods for modifying user configuration parameters ... 35
- Editing configuration files to modify configuration parameters ... 35
- Issuing the dmctl command to modify configuration parameters ... 36
- Using the Global Console to modify configuration parameters ... 38

Specifying a list of standby Managers for discovery ... 39
- Get port and PID of standby Managers ... 39
- Add standby Manager entries through stdby_servers.conf ... 40
- Add standby Manager entries through a script ... 41

Configuring filters that control which Managers are discovered ... 42
- Create an include filter for discovery ... 42
- Create an exclude filter for discovery ... 43
- Create both an include filter and an exclude filter for discovery ... 43

Disabling polling ... 44
Changing the default polling interval ... 45

Adding attribute entries to the history table 46
History file creation 46
History table additions 47
Procedure for adding an attribute entry to the history table 47
Configuring security 48

Chapter 3 Configuring the Global Manager

Overview 50
Opening the Global Manager Administration Console 50
Specifying the IC-SHM domain for configuration scenario 1 51
Specifying the SHM domain for configuration scenario 2 51
Loading the XML configuration file for the Health Monitor 53

Chapter 4 Customizing Polling and Thresholds

Before you start 56
Viewing the Polling and Thresholds tabs 57
Opening the Polling and Thresholds Console 59
User accounts and passwords 59
Procedure for opening the Polling and Thresholds Console 59
Understanding polling and thresholds 60
Understanding polling groups, threshold groups, and settings 61
Modifying polling and threshold groups 61
Default polling groups and settings 62
Queue_Groups polling group 62
Default threshold groups and settings 64
Host_Groups threshold group 64
SoftwareService_Groups threshold group 65
Working with groups and settings 68
Assign managed objects to groups 68
Modify the properties of a group 68
Add or remove settings 69
Modify the parameters of a setting 69
Modify the priority of groups 70
Edit matching criteria 70
Create new groups 71

Chapter 5 Topology Objects and Their Failures

Root-cause analysis 74
Data model 74
Summary of root-cause problems and events 75
Root-cause problems 76
Events 77
Topology object attributes, problems, and events 78
Host 79
InChargeClient 80
InChargeQueue 81
InChargeService 83
InChargeSession 90
InChargeSoftwareRequest 91
IP 91
SHM_UM_Agent 92

Appendix A **Terminology**

Overview 94
EMC Smarts terms and concepts 97
Computer-related terms and concepts 109

Appendix B **Wildcard Patterns**

Types of wildcard patterns 116

Appendix C **Topology Naming Conventions**

DisplayName and Name attributes 120
Naming convention for topology objects 121

Index

FIGURES

	Title	Page
1	EMC Smarts deployment	16
2	Health Monitor physical architecture	21
3	Health Monitor functional architecture	22
4	Health Monitor file tree	25
5	Preview of how the sm_edit utility works	35
6	Using the Domain Creation Wizard to create the SHM domain	52
7	Access to server tools for loading XML configuration files	53
8	Polling tab for the Health Monitor	57
9	Thresholds tab for the Health Monitor	58
10	Polling configuration and operation	60
11	Health Monitor data model	74
12	Basic components and their connections in an EMC Smarts deployment	94
13	EMC Smarts topology discovery and import	95
14	EMC Smarts monitoring and events and notifications	96
15	DisplayName and Name attributes for a topology object	120

TABLES

	Title	Page
1	EMC Smarts products that contain Health Monitor software	18
2	Health Monitor files	25
3	User configuration files for the Health Monitor	34
4	Default entries in the data-persistence.import file	46
5	Default Health Monitor polling group	62
6	Queues Polling Setting parameters and their values	63
7	Default Health Monitor threshold groups	64
8	System Health Metrics Thresholds parameters and their values	65
9	SoftwareService Health Metrics Thresholds parameters and their values	66
10	Root-cause problems diagnosed by the Health Monitor	76
11	Events computed by the Health Monitor	77
12	Attributes for Host	79
13	Root-cause problems for Host	80
14	Events for Host	80
15	Attributes for InChargeClient	81
16	Attributes for InChargeQueue	81
17	Events for InChargeQueue	82
18	Attributes for InChargeService	83
19	Root-cause problems for InChargeService	88
20	Events for InChargeService	89
21	Attributes for InChargeSession	90
22	Attributes for InChargeSoftwareRequest	91
23	Attributes for IP	91
24	Basic wildcard patterns	116
25	Compound wildcard patterns	117
26	Naming formats for discovered topology objects	121

PREFACE

As part of an effort to improve its product lines, EMC periodically releases revisions of its software and hardware. Therefore, some functions described in this document might not be supported by all versions of the software or hardware currently in use. The product release notes provide the most up-to-date information on product features.

Contact your EMC technical support professional if a product does not function properly or does not function as described in this document.

Note: This document was accurate at publication time. Go to EMC Online Support (https://support.emc.com) to ensure that you are using the latest version of this document.

Purpose

This document provides description, configuration, and operation information for the EMC® Smarts® Health Monitor, which is a tool that monitors and debugs the applications in an EMC Smarts deployment.

Audience

This document is part of the EMC Smarts Service Assurance Manager documentation set. It is intended for:

- Operators who are receiving and acting upon Health Monitor notifications.
- System administrators who are configuring and using the Health Monitor.
- IT managers who are seeking to better understand the value of the Health Monitor.

EMC Smarts Service Assurance Manager installation directory

In this document, the term BASEDIR represents the location where EMC Smarts software is installed:

- For UNIX, this location is /opt/InCharge/‹product›.
- For Windows, this location is C:\InCharge\‹product›.

On UNIX operating systems, EMC Smarts Service Assurance Manager is, by default, installed to /opt/InCharge/SAM/smarts. On Windows operating systems, this product is, by default, installed to C:\InCharge\SAM\smarts. This location is referred to as BASEDIR/smarts.

Optionally, you can specify the root of BASEDIR to be something other than /opt/InCharge (on UNIX) or C:\InCharge (on Windows), but you cannot change the ‹product› location under the root directory.

The *EMC Smarts System Administration Guide* provides detailed information about the directory structure for EMC Smarts software.

Related documentation

In addition to this document, EMC Corporation provides a Help system for command line programs as well as product documentation.

Help for command line programs

Descriptions of command line programs are available as HTML pages. The index.html file, which provides an index to the various commands, is located in the BASEDIR/smarts/doc/html/usage directory.

EMC Smarts documentation

Readers of this guide might find the following related documentation helpful. These documents are updated periodically. Electronic versions of the updated manuals are available on EMC Online Support:

- *EMC Smarts System Administration Guide*
- *EMC Smarts Common Information Model (ICIM) Reference for Service Assurance Manager*
- *EMC Smarts Common Information Model (ICIM) 1.11 Reference for Non-Service Assurance Manager Products*
- *EMC Smarts ASL Reference Guide*
- *EMC Smarts Perl Reference Guide*
- *EMC Smarts MODEL Reference Guide*
- *EMC Smarts Foundation EMC Data Access API (EDAA) Programmer Guide*
- *EMC Smarts Foundation Documentation Portfolio*

EMC Smarts Service Assurance Manager documentation

The following documents are relevant to users of the EMC Smarts Service Assurance Manager:

- *EMC Smarts Service Assurance Manager Release Notes*
- *EMC Smarts Installation Guide for SAM, IP, ESM, MPLS, VoIP, and NPM Managers*
- *EMC Smarts Open Source License and Copyright Information*
- *EMC Smarts Failover System User Guide*
- *EMC Smarts Service Assurance Manager Introduction*
- *EMC Smarts Service Assurance Manager Deployment Guide*
- *EMC Smarts Service Assurance Manager Configuration Guide*
- *EMC Smarts Service Assurance Manager Operator Guide*
- *EMC Smarts Service Assurance Manager Dashboard Configuration Guide*
- *EMC Smarts Business Impact Manager User Guide*
- *EMC Smarts Failover System User Guide*

- *EMC Smarts Service Assurance Manager Notification Adapters User Guide*
- *EMC Smarts Service Assurance Manager Adapter Platform User Guide*
- *EMC Smarts Adapter for Unified Infrastructure Manager/Operations Configuration Guide*
- *EMC Smarts XML Adapter User Guide*
- *EMC Smarts Notification Module User Guide*
- *EMC Smarts Service Assurance Manager Troubleshooting Guide*
- *EMC Smarts Companion UI Installation and Configuration Guide*
- *EMC Smarts Companion UI Online Help*
- *EMC Smarts Companion UI Open Source License and Copyright Information for GPLv3 as Included with a Distribution of SLES 11*
- *EMC Smarts Service Assurance Manager Documentation Portfolio*

Conventions used in this document

EMC uses the following conventions for special notices:

DANGER

DANGER indicates a hazardous situation which, if not avoided, will result in death or serious injury.

WARNING

WARNING indicates a hazardous situation which, if not avoided, could result in death or serious injury.

CAUTION

CAUTION, used with the safety alert symbol, indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to personal injury.

Note: A note presents information that is important, but not hazard-related.

IMPORTANT

An important notice contains information essential to software or hardware operation.

Typographical conventions

EMC uses the following type style conventions in this document:

Bold	Use for names of interface elements, such as names of windows, dialog boxes, buttons, fields, tab names, key names, and menu paths (what the user specifically selects or clicks)
Italic	Use for full titles of publications referenced in text
`Monospace`	Use for: • System output, such as an error message or script • System code • Pathnames, filenames, prompts, and syntax • Commands and options
`Monospace italic`	Use for variables.
`Monospace bold`	Use for user input.
[ ]	Square brackets enclose optional values
|	Vertical bar indicates alternate selections — the bar means “or”
{ }	Braces enclose content that the user must specify, such as x or y or z
...	Ellipses indicate nonessential information omitted from the example

Where to get help

EMC support, product, and licensing information can be obtained as follows:

Product information — For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

https://support.emc.com

Technical support — Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Send your opinions of this document to:

`techpubcomments@emc.com`

CHAPTER 1
Introduction

This chapter presents an overview of the EMC Smarts Health Monitor, highlights its configurations, and summarizes how it models, monitors, and analyzes the components in an EMC Smarts deployment. It contains the following sections:

- Overview 16
- Software components 17
- Software deployment 18
- System dependencies 19
- SHM agent support 19
- Architecture 20
- Automatic dynamic thresholding 23
- Installation 25
- Service command 27
- Configuration 28
- Discovery 29
- Monitoring 30
- Analysis 30
- Notifications 31

Overview

The EMC® Smarts® Health Monitor is a tool that monitors and analyzes the health of the Managers and their hosts in an EMC Smarts deployment. A Manager, as used here and throughout the rest of the document, represents an EMC Smarts Domain Manager, Global Manager, Adapter Platform, Adapter, or Broker, all of which are shown in Figure 1 on page 16.

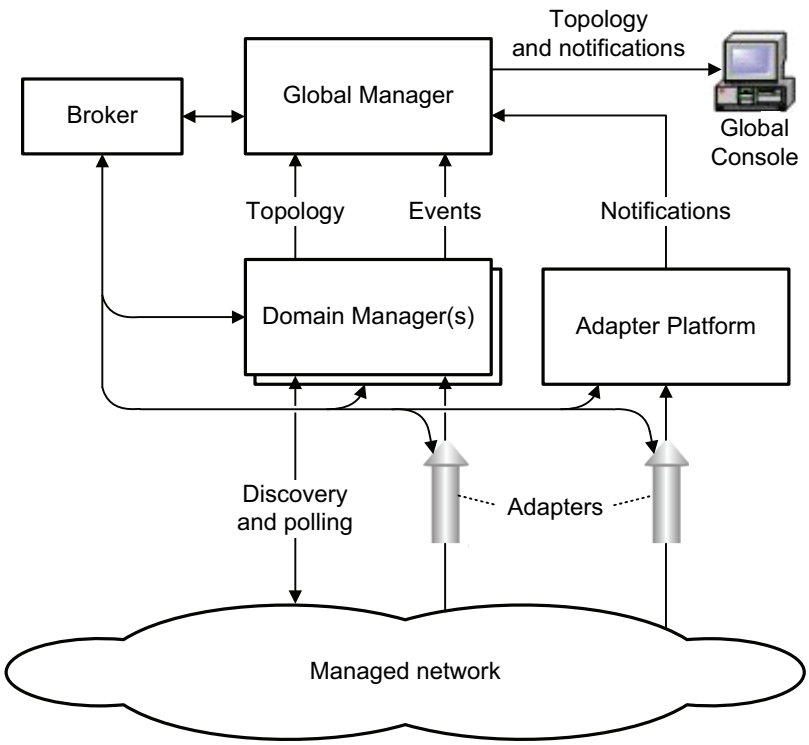


Figure 1 EMC Smarts deployment

NOTICE

If some of the terms in Figure 1 are unfamiliar to you, you might want to review their definitions in Appendix A, "Terminology."

At startup, the Health Monitor contacts the Broker for the EMC Smarts deployment to acquire a list of Managers that are registered with the Broker. It then discovers those Managers, the Broker, and their hosts, and discovers the connectivity and client/server relationships between the discovered Managers.

NOTICE

To be clear, the group of discovered Managers includes the Broker.

For the hosts on which the discovered Managers are running, the Health Monitor also discovers host CPU and memory performance metrics. For the discovered Managers, themselves, the Health Monitor also discovers:

- Queues
- Clients that are accessing the Managers
- Process queue, CPU, memory, heap, file-handle, and other performance metrics

When the discovery is complete, the Health Monitor begins monitoring the host- and Manager-specific performance metrics.

The Health Monitor uses the monitoring results to identify Managers that consume excessive CPU, memory, or other resources. In addition, it uses the client/server relationships between Managers to determine the impact that a failed or impaired Manager has on other Managers in the deployment.

The Health Monitor generates problem and impact events and exports them to the Global Manager. The Global Manager converts the events to notifications, for subsequent viewing in the Global Console.

Software components

The Health Monitor software consists of two major components:

- Health Monitor server

  Discovers and monitors the Managers in an EMC Smarts deployment and analyzes their health. The server consists of a set of library files that are included in any EMC Smarts Service Assurance Manager that is built on Foundation 9.0 or higher.

- Health Monitor component files

  A set of library files that are loaded into each Manager in an EMC Smarts deployment to facilitate the gathering of information by the Health Monitor server. The library files, which are different from the library files for the Health Monitor server, are included in any EMC Smarts product that is built on Foundation 9.0 or higher. They are loaded into a Manager when the Health Monitor server discovers and initially connects to the Manager.

Software deployment

Because the Health Monitor server is included in any EMC Smarts Service Assurance Manager that is built on Foundation 9.0 or higher, the server is installed when the Service Assurance Manager is installed. The Health Monitor component files are installed when any of the EMC Smarts products in Table 1 on page 18 are installed.

Table 1 EMC Smarts products that contain Health Monitor software (page 1 of 2)

Product	Product version	Foundation version [2, 3]
EMC Smarts Service Assurance Manager [1]	9.2	9.2
	9.1	9.1
	9.0	9.0.1
	8.1.4	9.0.4
	8.1.3	9.0.2
	8.1.2	9.0
EMC Smarts IP Manager	9.2	9.2
	9.1	9.1
	9.0	9.0.1
	8.1.4	9.0.4
	8.1.3	9.0.2
	8.1.1, 8.1.2	9.0
EMC Smarts MPLS Manager	9.2	9.2
	9.1	9.1
	9.0	9.0.1
	8.1.4	9.0.4
	8.1	9.0.2
	4.0, 4.1.1	9.0
EMC Smarts Network Protocol Manager	9.2	9.2
	8.1.4	9.0.4
	8.1	9.0.2
	3.1, 3.1.1	9.0

[1] The Health Monitor server is manually started from this product.
[2] Foundation 9.0.1, 9.1, and 9.2 support the SHM-agent version of the Health Monitor.
[3] Foundation 9.0, 9.0.2, and 9.0.4 support the SNMP-agent version of the Health Monitor.

Table 1 EMC Smarts products that contain Health Monitor software (page 2 of 2)

Product	Product version	Foundation version [2, 3]
EMC Smarts VoIP Manager	9.2	9.2
	8.1.4	9.0.4
	8.1.3	9.0.2
	4.0	9.0
EMC Smarts Multicast Manager	8.1.4	9.0.4
	8.1	9.0.2
	2.1, 2.1.1	9.0
EMC Smarts Server Manager	9.2	9.2
	9.1	9.1
	9.0	9.0.1
EMC Ionix Adapter for Alcatel-Lucent 5620 SAM EMS	3.2	9.0

[1] The Health Monitor server is manually started from this product.
[2] Foundation 9.0.1, 9.1, and 9.2 support the SHM-agent version of the Health Monitor.
[3] Foundation 9.0, 9.0.2, and 9.0.4 support the SNMP-agent version of the Health Monitor.

System dependencies

The EMC Smarts products that contain the Health Monitor software have been ported to Sun Solaris, Red Hat Linux, and Microsoft Windows. For a list of supported OS versions for an EMC Smarts product, see the installation guide for the particular product.

For a Windows-based Manager that is enabled by the Health Monitor server, the following functions are not supported:

- Discovery and monitoring of host virtual-memory performance metrics
- Discovery and monitoring of process heap performance metrics

Host virtual-memory performance metrics are not supported because the Windows operating system does not impose a host virtual-memory limitation value. Process heap performance metrics are not supported because the Health Monitor does not monitor those metrics on a Windows-based host.

SHM agent support

Previous to EMC Smarts 9.0, the Health Monitor server used SNMP and an SNMP agent on a Manager's host to discover and monitor host and process CPU and memory utilization metrics. In EMC Smarts 9.0, 9.1 and 9.2, the Health Monitor server uses EMC Smarts remote API and an "SHM agent" on a Manager's host to discover and monitor the CPU and memory utilization metrics. The SHM agent is an instance, named SHM-UM-Agent, of the SHM_UM_Agent class model file.

The 9.0/9.1/9.2 Health Monitor server loads the Health Monitor component files, including the SHM agent, into a 9.0/9.1/9.2 Manager when it discovers and initially connects to the Manager. The SHM agent gathers from the host operating system all host and process performance metrics for the discovered Manager *except* the process heap-headroom metric. That metric is computed once per minute by a history-only accessor that is running on the discovered Manager.

The Health Monitor server periodically calls the SHM agent of each discovered Manager to collect (1) the Manager's *host* CPU and memory performance metrics and (2) the Manager's *process* queue, CPU, memory, heap, file-handle, and other performance metrics. Through a remote attribute subscription, the Health Monitor server receives the process heap-headroom metric once per minute from each discovered Manager.

"SHM_UM_Agent" on page 92 describes the SHM agent object in more detail.

Architecture

Figure 2 on page 21 and Figure 3 on page 22 present physical and functional architectural views of the Health Monitor in an SHM-agent Health Monitor environment: The Health Monitor server and all of the discovered Managers are running product version 9.0, 9.1, or 9.2. Figure 2 includes just enough Managers and hosts to clarify the fundamentals of Health Monitor operation.

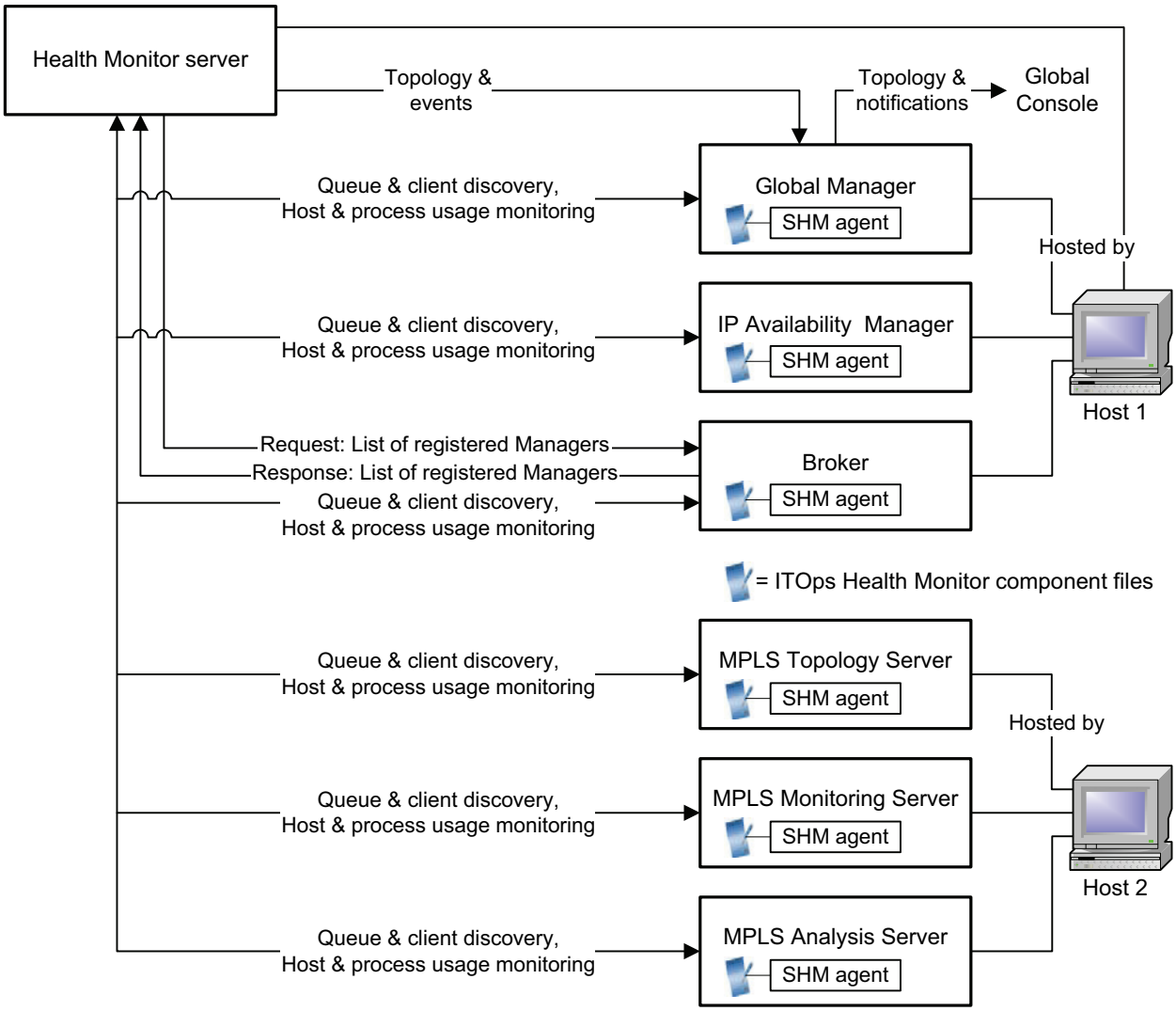


Figure 2 Health Monitor physical architecture

NOTICE

The descriptions in this document focus on the SHM-agent Health Monitor environment. The 9.0, 9.1, and 9.2 EMC Smarts products support the SHM-agent version, but not the SNMP-agent version, of the Health Monitor.

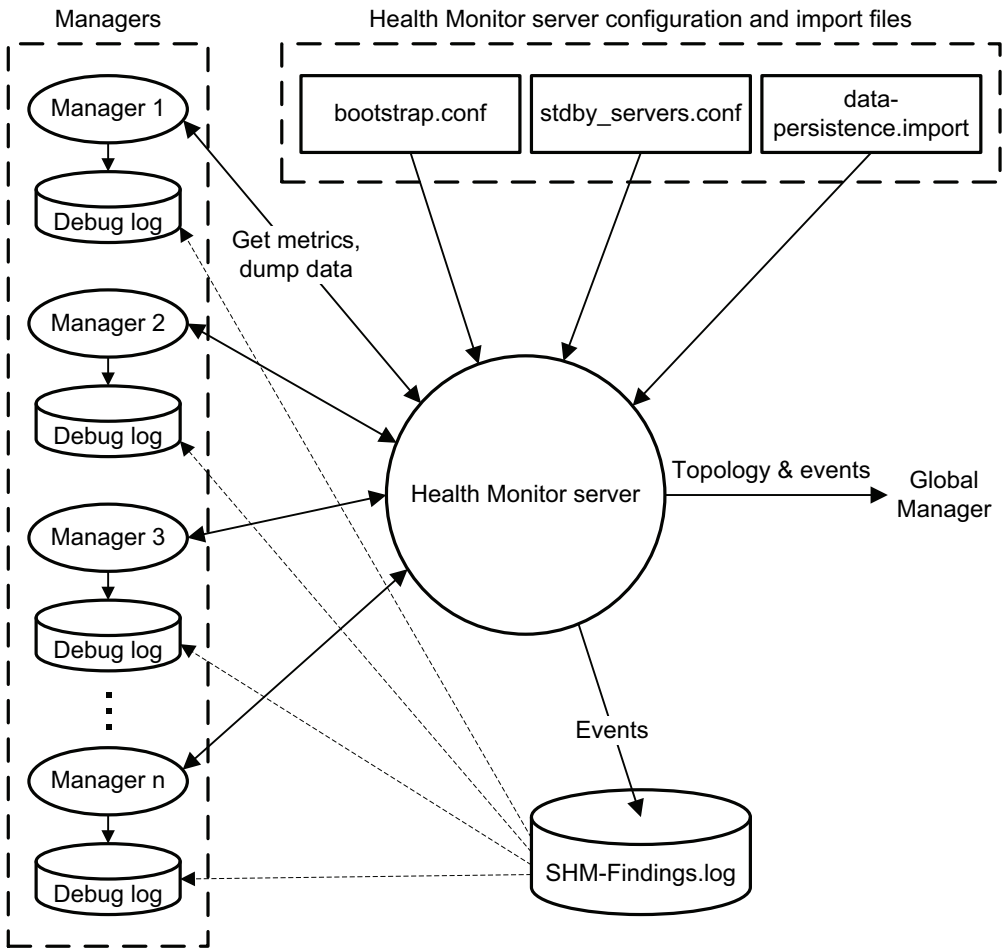


Figure 3 Health Monitor functional architecture

The Health Monitor server performs the following tasks in the deployment where it is started:

- Requests and receives from the deployment's Broker a list of Managers that are registered with the Broker. The list includes each Manager's name, process ID, and hostname, and the TCP port at which each Manager is listening for client connections, such as Global Console, dmctl, and other types of client connections.
- Discovers and monitors each Manager in the list and its host, and the Broker and its host. For each discovered Manager, the discovery includes queues and the clients that are accessing the Manager; host CPU and memory performance metrics; and process queue, CPU, memory, heap, file-handle, and other performance metrics.
- Analyzes the host and process performance metrics to identify Manager performance problems and their impacts. For each diagnosed event, writes the event to a local log named SHM-Findings.log, and dumps the associated debug data to a debug log or a set of debug logs on the host of the Manager for which the event was diagnosed. (Table 2 on page 25 provides more information about the SHM-Findings.log on the Health Monitor server's host and the debug log(s) on a Manager's host.)
- Exports topology and event information to the Global Manager.

Automatic dynamic thresholding

The Health Monitor server compares the polled host and process performance metrics to the thresholds that are listed in Table 6 on page 63, Table 8 on page 65, and Table 9 on page 66 to determine acceptable and unacceptable levels of performance. By default, the Health Monitor server performs automatic dynamic thresholding, which creates a customized set of threshold values for *each* discovered Manager in the monitored environment.

The threshold values that are calculated by the automatic dynamic thresholding feature override the default threshold values or the customized threshold values that a user sets through the Polling and Thresholds Console. Chapter 4, "Customizing Polling and Thresholds," identifies the default threshold values and provides instructions for manually customizing threshold values.

Requirements

The following requirements must be met to support automatic dynamic thresholding in an SHM-agent Health Monitor environment.

- The Health Monitor server must be started from Service Assurance Manager 9.0, 9.1, or 9.2.
- A discovered Manager must be running product version 9.0, 9.1, or 9.2.

Operation

When automatic dynamic thresholding is enabled (default), the Health Monitor server sets the value of an individual threshold in accordance to the following algorithm:

- For each discovered Manager, and for the metric to which the threshold applies, writes the initial polled value of the metric in a sample set for the metric.
- In any subsequent polling cycle, adds the polled value of the metric to the sample set *if and only if* the value is different than the value that was observed in the previous polling cycle. For example, if the initial polled value of the last full discovery time for a particular Manager is 500 seconds, the second polled value is 500 seconds, and the third polled value is 550 seconds, the sample set will contain two values: 500 and 550.

- Upon adding a new value to the sample set, and assuming that the sample set contains nine or less values, sets the metric's threshold to a value that is 10% greater than the maximum value in the sample set. For example, if the sample set contains two values, 500 and 550, the metric's threshold will be set to 550 + 10% times 550, which is 550 + 55 to equal 605.
- Upon adding a new value to the sample set, and assuming that the sample set contains ten values, sets the metric's threshold to a value that is 10% greater than the 90th percentile of the entire sample set. For example, if the sample set contains ten values, 500, 550, 570, 620, 530, 560, 580, 630, 700, and 590, the metric's threshold will be set to the 90th percentile of the ten ordered numbers (500, 530, 550, 560, 570, 580, 590, 620, 630, 700), which is 630, + 10% times 630, which is 63, to equal 693. The metric's threshold will be set to 693.
- Once the sample set contains ten values, deletes the oldest value upon receiving a new value, and recalculates a new threshold for the new sample set of ten. *The size of a sample set stays at ten*. This cycle continues until the Health Monitor server is shut down.

Two more points about automatic dynamic thresholding:

- For a metric that has a percentage-type threshold, the dynamically calculated threshold value is capped at 95 percent.
- If the polled values of a metric are between 0 and 1, the dynamically calculated threshold value for the metric might be the same as the previously calculated value, in which case the dynamically calculated threshold value is weighted appropriately to be between 0 and 10.

Sample sets and dynamically calculated threshold values are not persistent. The Health Monitor server builds new sample sets and calculates new thresholds after being shut down and restarted.

Benefits

Without the automatic dynamic thresholding feature, and because the default threshold values might not be applicable to the monitored environment, a user must set the threshold values manually through a Polling and Thresholds Console.

With the automatic dynamic thresholding feature, the Health Monitor server not only sets the threshold values automatically and dynamically for the monitored environment, but also creates a customized set of threshold values for each discovered Manager in the monitored environment.

Disabling the feature

EMC recommends that you do not disable automatic dynamic thresholding. However, if you want to disable the feature, you can do so through the Polling and Thresholds Console by setting the DynamicThresholding parameter to DISABLED. The Dynamic Thresholding parameter is described in Table 6 on page 63, Table 8 on page 65, and Table 9 on page 66.

Disabling automatic dynamic thresholding activates the default threshold values or the customized threshold values that are set through the Polling and Thresholds Console.

Installation

The Health Monitor software is included in all of the EMC Smarts products that are listed in Table 1 on page 18. When installing any of those products, you will not see any mention of the Health Monitor.

The installed Health Monitor file tree is shown in Figure 4 on page 25 and described in Table 2 on page 25.

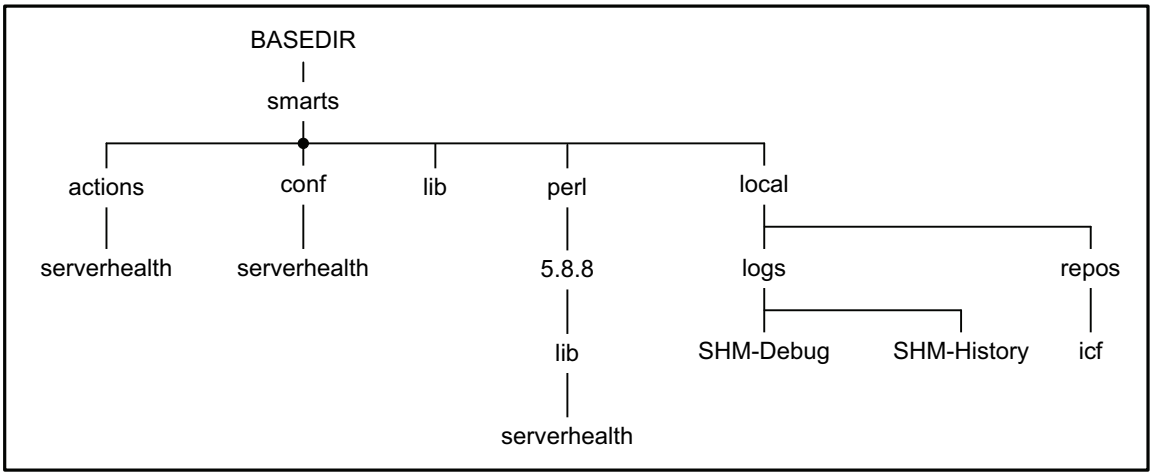


Figure 4 Health Monitor file tree

Table 2 Health Monitor files (page 1 of 3)

Filename	Description
Files in actions/serverhealth directory	
host-data-dump.pl	Scripts that a Health Monitor component invokes, when instructed to do so by the Health Monitor server, to complete such tasks as dumping debug data.
linux_prstat.pl	
Files in conf/serverhealth directory	
bootstrap.conf	Bootstrap file for the Health Monitor server. Also, the file in which a user configures filters that control which Managers are discovered; disables polling; and changes the default polling interval.
stdby_servers.conf	Text file in which a user specifies a list of standby Managers that the user wants the Health Monitor server to discover. Standby Managers are not registered with the Broker.
data-persistence.import	Text file in which a user adds attribute entries to the history table.
Files in lib directory	
libsm_serverhealth_datacollection.so	Libraries that contain dump and metric-fetching functionalities. The libsm_serverhealth_metrics.so library contains the Health Monitor component files, including the SHM_UM_Agent class model file, an instance of which gathers host and process performance metrics for the hosting Manager.
libsm_serverhealth_metrics.so	

Table 2 Health Monitor files (page 2 of 3)

Filename	Description
Files in perl/5.8.8/lib/serverhealth directory	
CmdExec.pm	Commands that a Health Monitor component invokes, when instructed to do so by the Health Monitor server, to perform such tasks as dumping debug data for a specific OS. DataDumpCmds.pm contains a list of Solaris, Linux, and Windows dump commands from which the host-data-dump.pl script in the BASEDIR/smarts/actions/serverhealth directory executes the appropriate command to dump data for a Solaris-, Linux-, or Windows-based Manager.
DataDumpCmds.pm	
logging.pm	
Files in local/logs directory	
SHM-Findings.log	An event log in which the Health Monitor server writes diagnosed Manager events. The event log is created on the host where the Health Monitor server is running. For each event, the event log includes the path to the associated debug log or set of debug logs, which are created on the host of the Manager that is associated with the event.
Server log file Example: IC-SHM.log	A server log in which startup, progression, and error messages are written for the Health Monitor server. The server log is created on the host where the Health Monitor server is running.
Files in local/logs/SHM-Debug directory	
Debug log files For *older* Health Monitor component software, a debug log file resides in the following 2-level subdirectory structure: ‹topology object type›-‹topology object name›-‹event name›/‹timestamp when event was notified›/‹debug log file› Example: Host-zach.emc.com-HighCpuUtilization/July_10,_2011_2_59_42_PM_EDT/top For *newer* Health Monitor component software, a set of debug log files resides in the following 1-level subdirectory structure: ‹timestamp when event was notified›.‹index›/‹debug log files› Example: 2011.07.10.02.59.42.0/Host-HighCpuUtilization-dmctl.output.2	Debug logs in which a Health Monitor component dumps data for diagnosed Manager events. For each event, a debug log or a set of debug logs are created on the host of the Manager that is associated with the event.

Table 2 Health Monitor files (page 3 of 3)

Filename	Description
Files in local/logs/SHM-History directory	
Comma-separated values (.csv) history files, where a history file is named in accordance to the following naming format: ‹topology object type›_‹topology object name›_‹attribute name›.csv Example: Host_zach.emc.com_CpuUtilization.csv	History files in which the Health Monitor server writes monitored attribute values for certain topology objects. Each time that an attribute value changes, the new attribute value is written to the appropriate history file. The history files are created on the host where the Health Monitor server is running.
Files in local/repos/icf directory	
Repository file Example: IC-SHM.repos	A repository file in which the repository of the Health Monitor server is stored. The repository file is created on the host where the Health Monitor server is running.

Service command

From the BASEDIR/smarts/bin directory in the Service Assurance Manager installation area, a user manually installs the Health Monitor server as a service after the installation. Here is the Health Monitor service command.

UNIX:

```
/opt/InCharge/SAM/smarts/bin/sm_service install
   --force
   --unmanaged
   --name=ic-shm-server
   --startmode=runonce
   --description="EMC Smarts Health Monitor Server"
   /opt/InCharge/SAM/smarts/bin/sm_server
   --name=IC-SHM
   --config=serverhealth
   --port=0
   --ignore-restore-errors
   --output
```

Windows:

```
C:\InCharge\SAM\smarts\bin\sm_service.exe install
   --force
   --name=ic-shm-server
   --startmode=runonce
   --description="EMC Smarts Health Monitor Server"
   C:\InCharge\SAM\smarts\bin\sm_server.exe
   --name=IC-SHM
   --config=serverhealth
   --port=0
   --ignore-restore-errors
   --output
```

To install the service, invoke the service command on one line.

Startup

To manually start the service and the Health Monitor server, invoke the following command:

sm_service start ic-shm-server

Shutdown

To manually stop the service and the Health Monitor server, invoke the following command:

sm_service stop ic-shm-server

Configuration

Introducing the Health Monitor into an EMC Smarts deployment involves the configuration tasks that are presented in this section. Throughout this section and the rest of the document, assume that "Health Monitor" means "Health Monitor server."

Health Monitor configuration tasks

Configuring the Health Monitor consists of the following tasks:

- Specify a list of standby Managers for discovery (optional).

  NOTICE

  Standby Managers are not registered with the Broker.

- Configure filters that control which Managers are discovered (optional).
- Disable polling (optional).
- Change the default polling interval (optional).
- Add attribute entries to the history table (optional).
- Disable automatic dynamic thresholding (optional).
- Customize polling for the Health Monitor (optional).
- Customize thresholds for the Health Monitor (optional).

Chapter 2, "Configuring the Health Monitor," and Chapter 4, "Customizing Polling and Thresholds," provide the procedures for completing these tasks.

Global Manager configuration tasks

Set up a communication link between the Global Manager and the Health Monitor. Chapter 3, "Configuring the Global Manager," provides the procedure for completing this task.

Discovery

The Health Monitor initiates a discovery of the Managers in its deployment whenever it is started or restarted, or whenever **Topology › Discover All** is selected in a Domain Manager Administration Console that is attached to the Health Monitor. Instructions for opening the Domain Manager Administration Console are included in "Opening the Polling and Thresholds Console" on page 59.

During a discovery, the Health Monitor performs host-based discovery and Manager-based discovery for each Manager that is registered with the Broker, and for the Broker, itself.

- Host-based discovery:
  - Host CPU performance metrics
  - Host physical-memory performance metrics
  - Host virtual-memory performance metrics
- Manager-based discovery:
  - Queues and clients
  - Manager metrics
    - Process queue performance
    - Process CPU performance
    - Process physical-memory performance
    - Process virtual-memory performance
    - Process heap performance
    - Process file-handle performance
    - Process correlation time, discovery time, recompute time, and reconfigure time
  - Manager interdependencies, such as which Manager is a client and which is a server

NOTICE

If so configured, the Health Monitor server also discovers standby Managers.

The Health Monitor uses the Manager queue and client information to determine which queues are being used by which clients that are accessing the Manager. It uses the Manager interdependencies to determine whether the client or the server process is preventing the timely flow of data.

During the discovery, the Health Monitor uses instances of certain EMC Smarts object classes to create within its repository a data model representation of the discovered Managers and hosts, their relationships, their internal components, and their logical connections. The data model is described in Chapter 5, "Topology Objects and Their Failures."

Monitoring

After the initial discovery and modeling, the Health Monitor uses EMC Smarts remote API polling to monitor the performance metrics of the discovered Managers and hosts. Chapter 4, “Customizing Polling and Thresholds,” identifies which objects are being monitored and what performance threshold levels are being observed.

Analysis

The performance metric updates that are acquired from periodic monitoring serve as input to the data model. The Health Monitor applies the metric updates to the appropriate attributes of the corresponding objects in the data model.

The data model involves a codebook that is created by EMC engineers as a result of their knowledge of host computers and EMC Smarts Managers. The codebook identifies host and Manager problems and their symptoms, and presents a causality mapping between each problem and its set of symptoms. Most symptoms are abnormal performance conditions that are detected when performance thresholds are exceeded.

The Health Monitor uses its codebook to diagnose root-cause problems by cross correlating:

- Host status
- Manager status
- Manager-to-Manager interconnectivity status

From the correlation, the Health Monitor finds the problem that can best explain the observed statuses.

Notifications

The Health Monitor exports the results of its root-cause and impact analysis to the Global Manager, where the results are displayed in the form of EMC Smarts notifications in the Notification Log Console view of the Global Console. The notifications identify the Health Monitor by its domain name (IC-SHM, for example) in the Source attribute. Users can double-click a notification to view detailed information about the notification.

The Health Monitor creates two types of notifications:

- Root-cause problem notifications

  Each notification indicates a point of failure or an unacceptable performance level that is diagnosed by the Health Monitor.

- Event notifications

  Each notification indicates an abnormal availability or performance condition that is computed by the Health Monitor. It computes these events primarily from performance metric updates and performance thresholds, and uses them as symptoms to determine root-cause problems.

Chapter 5, “Topology Objects and Their Failures,” provides more information about the notifications that are created by the Health Monitor server.

CHAPTER 2
Configuring the Health Monitor

This chapter identifies the Health Monitor configuration parameters that are relevant to discovery or monitoring, describes their settings, and provides instructions for changing their settings. It consists of the following sections:

- Before you start 34
- Configuration directory 34
- User configuration parameters 34
- Methods for modifying user configuration parameters 35
- Specifying a list of standby Managers for discovery 39
- Configuring filters that control which Managers are discovered 42
- Disabling polling 44
- Changing the default polling interval 45
- Adding attribute entries to the history table 46
- Configuring security 48

Before you start

All configuration procedures in this chapter are performed from the Service Assurance Manager installation area from which the Health Monitor server is started. Also, unless otherwise stated, assume that “Health Monitor” means “Health Monitor server.”

Configuration directory

The Health Monitor configuration directory is BASEDIR/smarts/conf/serverhealth, shown in Figure 4 on page 25. BASEDIR, which is not an environment variable, is used in documentation to represent the top-level directory structure of an EMC Smarts product installation. For the Service Assurance Manager, BASEDIR represents ‹installation_root_directory›/SAM.

The configuration directory contains the bootstrap and configuration files for the Health Monitor. The --config=serverhealth option in a Health Monitor startup command points to these files in the serverhealth directory. The default name of the Health Monitor is IC-SHM.

The bootstrap and configuration files for the Health Monitor set up its environment.

User configuration parameters

Part of configuring the Health Monitor involves the setting of user configuration parameters in the files that are identified in Table 3 on page 34.

Table 3 User configuration files for the Health Monitor

File under BASEDIR/smarts	Description
conf/serverhealth/bootstrap.conf	File in which you: • Configure filters that control which Managers are discovered. • Disable polling. • Change the default polling interval.
conf/serverhealth/stdby_servers.conf	File in which you specify a list of standby Managers for discovery.
conf/serverhealth/data-persistence.import	File in which you add attribute entries to the history table.

In general, most EMC Smarts configuration files are used to set attributes for certain objects within an EMC Smarts server’s environment when the server starts up. The parameters and their values in a configuration file become the attributes and their values for the created objects.

For the conf/serverhealth/bootstrap.conf file, as an example, one of the associated objects is an instance of the *SHM_InChargeServerInterface* class and is named *SHM-InChargeServerInterface*. This object has attributes for configuring various aspects of the Health Monitor.

Methods for modifying user configuration parameters

There are three methods to modify user configuration parameters:

1. Edit the user configuration files and modify configuration parameters.
2. Issue the dmctl command to modify configuration parameters when the Health Monitor is running.
3. Use the Global Console to modify configuration parameters when the Health Monitor is running.

Parameters that are modified by using methods 2 and 3 will be reconfigured to the settings in the user configuration files when the Health Monitor is shut down and restarted. Every time that the Health Monitor starts up, it sets its configurations in accordance to the settings in the user (and other) configuration files.

Editing configuration files to modify configuration parameters

EMC Smarts provides the sm_edit utility to ensure that modified files are always saved to the appropriate local area, and that base (original) copies of the files remain unchanged. Figure 5 on page 35 previews the operation of the sm_edit utility.

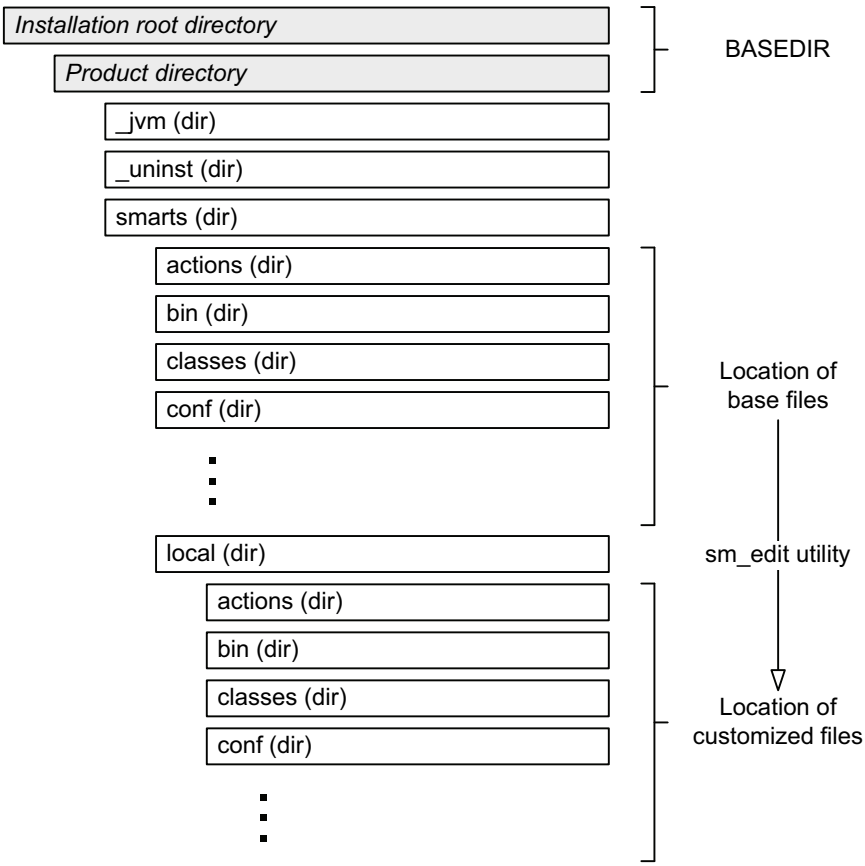


Figure 5 Preview of how the sm_edit utility works

All EMC Smarts products use the same basic installation directory structure that is shown in Figure 5. For the Service Assurance Manager, from which the Health Monitor is launched, the product directory is named SAM.

To invoke the sm_edit utility, go to the BASEDIR/smarts/bin directory and specify the path and the name of the file, relative to the BASEDIR/smarts/local or BASEDIR/smarts directory, that you want to edit. For example,

```
sm_edit conf/serverhealth/bootstrap.conf
```

opens in a text editor *either* a local copy of the bootstrap.conf file in BASEDIR/smarts/local/conf/serverhealth *or* a base (original) copy of the bootstrap.conf file in BASEDIR/smarts/conf/serverhealth if no local copy exists.

After you modify and save the bootstrap.conf file, the sm_edit utility saves the modified version of the file to the BASEDIR/smarts/local/conf/serverhealth directory.

NOTICE

Base (original) versions of files may be changed or updated as part of an EMC Smarts software upgrade. However, files in the BASEDIR/smarts/local directory are retained during an upgrade.

The *EMC Smarts System Administration Guide* provides additional information about the sm_edit utility.

Issuing the dmctl command to modify configuration parameters

You can use the following dmctl command to *view* the value of any basic-type attribute (string, boolean, integer, float, and so on) of any EMC Smarts object:

```
dmctl -s <EMC Smarts application instance name> get
   <class::instance::attribute>
```

You can use the following dmctl command to *change* the value of any basic-type attribute (string, boolean, integer, float, and so on) of any EMC Smarts object:

```
dmctl -s <EMC Smarts application instance name> put
   <class::instance::attribute> <attribute_value>
```

For the Health Monitor, as an example, you could use the dmctl put command to change (override) parameter values for the SHM_InChargeServerInterface object in the conf/serverhealth/bootstrap.conf file.

Upon issuing a dmctl command, you might be prompted for a username and password. Respond with your user account. For example, to specify the default administrative account, enter username *admin* and password *changeme*.

The dmctl utility is described in the HTML pages that are located in the BASEDIR/smarts/doc/html/usage directory of any EMC Smarts product installation area.

Example 1

As an example of using the dmctl command to *view* an object's attribute value, enter the following dmctl command from the BASEDIR/smarts/bin directory of the target Health Monitor to view the *polling_period* attribute value of the SHM_InChargeServerInterface object:

```
dmctl -s <Health Monitor instance name> get
   SHM_InChargeServerInterface::SHM-InChargeServerInterface::
      polling_period
```

NOTICE

The default value for the *polling_period* attribute is 240 seconds (four minutes).

To view all attribute values of the SHM_InChargeServerInterface object in one invocation, enter the following dmctl command:

```
dmctl -s <Health Monitor instance name> get
   SHM_InChargeServerInterface::SHM-InChargeServerInterface | more
```

Example 2

As an example of using the dmctl command to *change* an object's attribute value, enter the following dmctl command from the BASEDIR/smarts/bin directory of the target Health Monitor to change the *polling_period* attribute value of the SHM_InChargeServerInterface object:

```
dmctl -s <Health Monitor instance name> put
   SHM_InChargeServerInterface::SHM-InChargeServerInterface::
      polling_period 180
```

Your change will take effect in the next polling cycle, without any need to reconfigure ( ) the Health Monitor.

Because the SHM_InChargeServerInterface object is not saved to disk, that is, is not saved to a Health Monitor repository file in the BASEDIR/smarts/local/repos/icf directory, your change is forgotten when the Health Monitor is restarted. To make your change permanent, use the sm_edit utility to change the parameter value in the conf/serverhealth/bootstrap.conf file.

Using the Global Console to modify configuration parameters

You can use the Global Console to view or change the attributes of any EMC Smarts object. For the Health Monitor, as an example, you could use the Global Console to change (override) parameter values for the SHM_InChargeServerInterface object in the conf/serverhealth/bootstrap.conf file.

To use the Global Console to change (override) parameter values for the SHM_InChargeServerInterface object:

1. Attach the Global Console to the target Health Monitor.

   The *EMC Smarts Service Assurance Manager Operator Guide* provides information about starting the Global Console, logging in, and attaching to an EMC Smarts application. Also, see "Opening the Polling and Thresholds Console" on page 59.

2. Press the **Ctrl-Alt-M** keys simultaneously to make all Health Monitor classes available to the Global Console, including the meta classes.

3. In the topology tree panel of the Global Console, right-click an open space and select **Refresh tree** from the pop-up menu to refresh the content of the topology tree.

4. In the topology tree panel, click the **+** signs to expand the **SHM_InChargeServerInterface** class, then click **SHM-InChargeServerInterface** to display the **Attributes** tab in the right panel.

   The **Attributes** tab lists the attribute names, data types, and values for an instance of the **SHM_InChargeServerInterface** class named **SHM-InChargeServerInterface**.

5. In the **Attributes** tab, use the scroll bar to scroll to the attribute that you want to modify (**polling_period**, for example), double-click its value in the **Value** column, modify the value, and press **Enter**.

6. Click **Apply** to save your changes.

7. In the topology tree panel, right-click an open space and select **Refresh tree** from the pop-up menu to refresh the content of the topology tree.

8. Press the **Ctrl-Alt-M** keys simultaneously to make only the default Health Monitor classes available.

9. In the topology tree panel, click the **+** sign next to the Health Monitor's name to expand the topology tree. Only the default classes appear in the topology tree.

Your change will take effect in the next polling cycle, without any need to reconfigure ( ) the Health Monitor.

Because the SHM_InChargeServerInterface object is not saved to disk, that is, is not saved to a Health Monitor repository file in the BASEDIR/smarts/local/repos/icf directory, your change is forgotten when the Health Monitor is restarted. To make your change permanent, use the sm_edit utility to change the parameter value in the conf/serverhealth/bootstrap.conf file.

Specifying a list of standby Managers for discovery

By default, the Health Monitor does not discover standby Managers because standby Managers are *not* registered with the Broker.

NOTICE

In a hot-hot failover configuration, a standby Manager runs in parallel with its primary Manager to provide continuous availability and high reliability. If the primary Manager fails, the standby Manager is automatically switched into operation.

For the Health Monitor to discover standby Managers, you must make them known to the Health Monitor. You do so in one of two ways:

- "Add standby Manager entries through stdby_servers.conf" on page 40
- "Add standby Manager entries through a script" on page 41

The syntax for a standby Manager entry is:

```
<hostname>:<port of standby Manager>/<name of standby Manager> <PID
   of standby Manager> [name of primary Manager]
```

For example:

```
kirk.emc.com:6357/INCHARGE-AM2 12838 [INCHARGE-AM1]
```

The name of the primary Manager is optional. If not specified, the name of the primary Manager is assumed to be same as the name of the standby Manager.

Get port and PID of standby Managers

To get the port and PID that are assigned to a standby Manager:

1. Go to the BASEDIR/smarts/bin directory in the Manager's installation area and enter the following command to start the Manager:

   ```
   sm_server -n <standby Manager's name> -c <standby Manager's
      configuration directory> -b <Broker host or IP>:<Broker port>
         --noregister --output
   ```

   Note that any Manager that is started with the --noregister option will not register with the Broker.

2. Open the Manager's log file in the BASEDIR/smarts/local/logs directory and find the port and PID that are assigned to the Manager.

Add standby Manager entries through stdby_servers.conf

By default, the Health Monitor reads the stdby_servers.conf file at startup because of the following line in the conf/serverhealth/bootstrap.conf file:

```
fileName = "$SM_SITEMOD/conf/serverhealth/stdby_servers.conf"
```

If you specify a standby Manager in stdby_servers.conf, do not also specify that Manager in the script that is described in "Add standby Manager entries through a script" on page 41.

To add standby Manager entries through stdby_servers.conf:

1. Go to the BASEDIR/smarts/bin directory in the Service Assurance Manager installation area and enter the following command to open the stdby_servers.conf file:

   sm_edit conf/serverhealth/stdby_servers.conf

2. Add your standby Manager entries, one entry per line.
3. Save and close the file.

   The modified version of the stdby_servers.conf file is saved to the BASEDIR/smarts/local/conf/serverhealth directory.

4. If the Health Monitor was running before you edited the stdby_servers.conf file, restart the Health Monitor.

At startup, the Health Monitor will read the stdby_servers.conf file and discover the standby Managers.

If the Health Monitor is not able to discover the standby Manager for a particular standby Manager entry, or if the Health Monitor discovered the standby Manager but that Manager is now down, the Health Monitor will generate a StdbyMissingOrDown event on the object that represents the primary Manager.

If the Health Monitor is not able to discover the primary Manager for a particular standby Manager entry, or if the Health Monitor discovered the primary Manager but that Manager is now down, the Health Monitor will generate a PrimaryMissingOrDown event on the object that represents the standby Manager.

Add standby Manager entries through a script

Instead of using stdby_servers.conf, you can create an executable program/script and add the standby Manager entries to the script. To complete the configuration, you specify the absolute path and name of the script in the *stdby_fe_script* attribute of the following object definition in the conf/serverhealth/bootstrap.conf file:

```
SHM_InChargeServerInterface::SHM-InChargeServerInterface
{
    .
    .
    .
    #stdby_fe_script = . . .
}
```

To add standby Manager entries through a script:

1. Go to the BASEDIR/smarts/bin directory in the Service Assurance Manager installation area and enter the following command to open the bootstrap.conf file:

   sm_edit conf/serverhealth/bootstrap.conf

2. Find and uncomment the following attribute definition:

   ```
   #stdby_fe_script = "/emc/mirkhr/sitemod/scripts/stdby_servers.sh"
   ```

3. Specify the absolute path and name of your script. For example:

   ```
   stdby_fe_script = "/emc/sylves/sitemod/scripts/stdby-srvs.sh”
   ```

4. Save and close the file.

   The modified version of the bootstrap.conf file is saved to the BASEDIR/smarts/local/conf/serverhealth directory.

5. If the Health Monitor was running before you edited the bootstrap.conf file, restart the Health Monitor.

At startup, the Health Monitor will read the script and discover the standby Managers.

Configuring filters that control which Managers are discovered

By default, the Health Monitor discovers all Managers that are registered with the Broker.

You can edit the conf/serverhealth/bootstrap.conf file and create a filter that includes a specific set of Managers for discovery, or excludes a specific set of Managers for discovery. The line to be edited is the *dm_incl_namepattern* attribute of the following object definition in the bootstrap.conf file:

```
SHM_InChargeServerInterface::SHM-InChargeServerInterface
{
    .
    .
    .
    #dm_incl_namepattern = "AM"
    .
    .
    .
}
```

Create an include filter for discovery

To create an include filter for discovery:

1. Go to the BASEDIR/smarts/bin directory in the Service Assurance Manager installation area and enter the following command to open the bootstrap.conf file:

   sm_edit conf/serverhealth/bootstrap.conf

2. Find and uncomment the following attribute definition:

   ```
   #dm_incl_namepattern = "AM"
   ```

3. Replace **AM** with the Manager names that should be discovered. Use the wildcard characters that are described in Appendix B, "Wildcard Patterns," to specify name patterns; for example:

   dm_incl_namepattern = "***Cust1*|*cust2***"

   For ***Cust1*|*cust2***, any Manager that has either the string **Cust1** or **cust2** anywhere in its name will become a member of the group. The Managers that become members of the group will be *included* in discovery, and the Managers that do not become members of the group will be *excluded* from discovery.

   NOTICE

   To learn how to use the glob function to specify all types of name patterns, consult the *EMC Smarts ASL Reference Guide*.

4. Save and close the file.

   The modified version of the bootstrap.conf file is saved to the BASEDIR/smarts/local/conf/serverhealth directory.

5. If the Health Monitor was running before you edited the bootstrap.conf file, restart the Health Monitor.

Create an exclude filter for discovery

To create an exclude filter for discovery:

1. Go to the BASEDIR/smarts/bin directory in the Service Assurance Manager installation area and enter the following command to open the bootstrap.conf file:

   sm_edit conf/serverhealth/bootstrap.conf

2. Find and uncomment the following attribute definition:

   `#dm_incl_namepattern = "AM"`

3. Replace **AM** with the Manager names that should *not* be discovered. For example

   `dm_incl_namepattern = "~usr1*"`

   For **~usr1***, any Manager that does *not* have a name that begins with the string **usr1** will become a member of the group. The Managers that become members of the group will be *included* in discovery, and the Managers that do not become members of the group (all those with names that begin with **usr1**) will be *excluded* from discovery.

   Another way to create this example exclude filter is to add the following attribute definition right after `#dm_incl_namepattern = "AM":`

   dm_excl_namepattern = "usr1*"

4. Save and close the file.

   The modified version of the bootstrap.conf file is saved to the BASEDIR/smarts/local/conf/serverhealth directory.

5. If the Health Monitor was running before you edited the bootstrap.conf file, restart the Health Monitor.

Create both an include filter and an exclude filter for discovery

As demonstrated by the following examples and clarified in the previous two procedures, you can create both an include filter and an exclude filter for discovery:

```
dm_incl_namepattern = "*Cust1*|*cust2*"
dm_excl_namepattern = "usr1*"
```

Disabling polling

By default, the Health Monitor polls the discovered Managers for host and process performance metrics.

You can edit the conf/serverhealth/bootstrap.conf file and disable the polling of host and process performance metrics. The object definition to be edited in the bootstrap.conf file is:

```
SHM_InChargeServerInterface::SHM-InChargeServerInterface
{
    .
    .
    .
}
```

To disable the polling of all host and process performance metrics:

1. Go to the BASEDIR/smarts/bin directory in the Service Assurance Manager installation area and enter the following command to open the bootstrap.conf file:

   ```
   sm_edit conf/serverhealth/bootstrap.conf
   ```

2. Find the following object definition:

   ```
   SHM_InChargeServerInterface::SHM-InChargeServerInterface
   {
       .
       .
       .
   }
   ```

3. Add the following attribute definition to the object definition:

   ```
   SHM_InChargeServerInterface::SHM-InChargeServerInterface
   {
       .
       .
       .
       disable_polling = TRUE
   }
   ```

4. Save and close the file.

   The modified version of the bootstrap.conf file is saved to the BASEDIR/smarts/local/conf/serverhealth directory.

5. If the Health Monitor was running before you edited the bootstrap.conf file, restart the Health Monitor.

Changing the default polling interval

By default, the Health Monitor polls the discovered Managers *every four minutes* for host and process performance metrics.

You can edit the conf/serverhealth/bootstrap.conf file and change the polling interval, which is the time (in seconds) between successive polls. The line to be edited is the *polling_interval* attribute of the following object definition in the bootstrap.conf file:

```
SHM_InChargeServerInterface::SHM-InChargeServerInterface
{
    .
    .
    .
   polling_interval = 240
    .
    .
    .
}
```

To change the polling interval:

1. Go to the BASEDIR/smarts/bin directory in the Service Assurance Manager installation area and enter the following command to open the bootstrap.conf file:

   `sm_edit conf/serverhealth/bootstrap.conf`

2. Find the following attribute definition:

   `polling_interval = 240`

3. Replace **240** (seconds) with a different polling interval. Allowed values are 30 to 3600 inclusive.

4. Save and close the file.

   The modified version of the bootstrap.conf file is saved to the BASEDIR/smarts/local/conf/serverhealth directory.

5. If the Health Monitor was running before you edited the bootstrap.conf file, restart the Health Monitor.

Adding attribute entries to the history table

The history table maintains a list of monitored attribute entries for certain topology objects. The entries are defined in the data-persistence.import file and shown in Table 4 on page 46.

Table 4 Default entries in the data-persistence.import file

Host attributes	InChargeService attributes
CpuUtilization	AvailableHeapHeadRoom
TotalPhysicalMemoryUtilization	CorrelationTime
TotalVirtualMemoryUtilization	CpuUtilization
	lastFullDiscoveryDuration_long
	lastPendingDiscoveryDuration_long
	PhysicalMemoryUtilization
	ProcessVirtualMemorySize
	RecomputeTime
	ReconfigureTime
	VirtualMemoryUtilization

Table 12 on page 79 describes Host attributes, and Table 18 on page 83 describes InChargeService attributes.

History file creation

A software module that runs within the Health Monitor subscribes to the monitored attributes and writes their values to individual history files in the BASEDIR/smarts/local/logs/SHM-History directory. The values are stored in the comma-separated values (.csv) file format.

Here is the naming format for a history file:

‹topology object type›_‹topology object name›_‹attribute name›.csv

For example:

```
Host_zach.emc.com_CpuUtilization.csv
```

Here is an example entry in a history file:

```
Host_zach.emc.com_CpuUtilization 2011-Jan-12;17:44:28 9.50787
```

An entry is added to the history file each time that the attribute value changes.

History table additions

Because system engineers read the history files to spot patterns and trends, EMC recommends that you do not delete any default attribute entries from the data-persistence.import file. EMC also encourages you to add any attribute entry for which you want to keep a record of the attribute's changing values.

Use the following attribute entry template to add attribute entries to the data-persistence.import file:

```
SubscribesTo += GA_PropertyChoiceSubscription::DATA-PERSISTENCE-<attr ref name>-PS
     {
     classPattern = "<class name, for example,  InChargeService>"
     instancePattern = "<instance name filter; to accept all instances, type .*>"
     propertyPattern = "<attribute name>"
     expandSubClasses = <TRUE or FALSE; typically, you set this value to FALSE>
     }
```

Example of an attribute entry in the data-persistence.import file:

```
SubscribesTo += GA_PropertyChoiceSubscription::DATA-PERSISTENCE-RECOMP-PS
     {
     classPattern = "InChargeService"
     instancePattern = ".*"
     propertyPattern = "RecomputeTime"
     expandSubClasses = FALSE
     }
```

Procedure for adding an attribute entry to the history table

To add an attribute entry to the data-persistence.import file:

1. Go to the BASEDIR/smarts/bin directory in the Service Assurance Manager installation area and enter the following command to open the data-persistence.import file:

   sm_edit conf/serverhealth/data-persistence.import

2. Find the line, **GA_SubscriberFE::DATA-PERSISTENCE-FE {**, and then find the attribute entry that follows that line:

   ```
   SubscribesTo +=
   GA_PropertyChoiceSubscription::DATA-PERSISTENCE-CPU-PS
        {
        classPattern = "InChargeService"
        instancePattern = ".*"
        propertyPattern = "CpuUtilization"
        expandSubClasses = FALSE
        }
   ```

3. Copy and paste the attribute entry to right below the line that is identified in step 2.

4. Edit the copied attribute entry to reflect your attribute entry. For example:

   ```
   SubscribesTo +=
   GA_PropertyChoiceSubscription::DATA-PERSISTENCE-PROCCYCLES-PS
        {
        classPattern = "InChargeService"
        instancePattern = ".*"
        propertyPattern = "ProcessorCycles"
        expandSubClasses = FALSE
        }
   ```

NOTICE

If you want, consult Appendix B, "Wildcard Patterns," and specify a restrictive matching pattern for the instancePattern parameter in your attribute entry.

5. Save and close the file.

   The modified version of the data-persistence.import file is saved to the BASEDIR/smarts/local/conf/serverhealth directory.

6. If the Health Monitor was running before you edited the data-persistence.import file, restart the Health Monitor.

Configuring security

The security configuration files, clientConnect.conf and serverConnect.conf, located in the BASEDIR/smarts/conf directory of each EMC Smarts software installation, enable you to set up secure connections between the component applications (including the Health Monitor) in an EMC Smarts deployment. By default, the configuration option settings in the clientConnect.conf file and the serverConnect.conf files enable minimally secure connections between the components.

The *EMC Smarts System Administration Guide* presents detailed information about EMC Smarts secure communications.

CHAPTER 3
Configuring the Global Manager

This chapter provides instructions for configuring the Global Manager to communicate with the Health Monitor. It consists of the following sections:

- Overview 50
- Opening the Global Manager Administration Console 50
- Specifying the IC-SHM domain for configuration scenario 1 51
- Specifying the SHM domain for configuration scenario 2 51
- Loading the XML configuration file for the Health Monitor 53

Overview

Through a Global Manager Administration Console that is attached to the Global Manager, an administrator specifies the Health Monitor domain. Doing so associates the Health Monitor with the proper Global Manager DXA file, which identifies the topology and events that the Global Manager should import from the Health Monitor.

The expectation is that an administrator will either accept the default Health Monitor domain named IC-SHM, or make a copy of the default domain to create a new domain that has a different name. For example, an administrator might make a copy of IC-SHM and name it SHM.

The procedures that follow configure two configuration scenarios:

- Configuration scenario 1: A Health Monitor named IC-SHM
- Configuration scenario 2: A Health Monitor named SHM

For either scenario, only one instance of the Health Monitor will be started in the managed environment. For scenario 1, that instance will be named IC-SHM; for scenario 2, that instance will be named SHM.

Opening the Global Manager Administration Console

Accessing the Global Manager Administration Console view of the Global Console requires attaching the Global Console to the Global Manager (for example, INCHARGE-SA) in your deployment. The *EMC Smarts Service Assurance Manager Operator Guide* provides information about starting the Global Console, logging in, and attaching to an EMC Smarts application.

NOTICE

Also, see "Opening the Polling and Thresholds Console" on page 59.

Upon attaching the Global Console to the Global Manager in your deployment, a Notification Log Console opens. In that console, select **Configure › Global Manager Administration Console** to open the Global Manager Administration Console. In the left panel of the Global Manager Administration Console, expand the configuration tree to display the Domains objects.

NOTICE

If the IC-SHM domain does not appear, perform the procedure in "Loading the XML configuration file for the Health Monitor" on page 53.

Specifying the IC-SHM domain for configuration scenario 1

By default, the IC-SHM domain is disabled. The names of disabled domains appear as gray text in the Global Manager Administration Console, and the names of enabled domains appear as black text.

To specify the IC-SHM domain for configuration scenario 1:

1. In the left panel of the Global Manager Administration Console, select **IC-SHM** to display its configuration window.
2. In the configuration window, click the **Enabled** checkbox.
3. In the configuration window, click **Apply** to save your configuration changes.
4. In the **Server Reconfiguration** information dialog box that appears, click **OK**.
5. In the toolbar area of the Global Manager Administration Console, click the **Reconfigure** toolbar button ( ).
6. In the **Server Reconfiguration** confirmation dialog box that appears, click **Yes**.
7. In the **Server Reconfiguration** information dialog box that appears, click **Close**.

In response, the Global Manager:

- Creates an InChargeDomain object named IC-SHM and starts the appropriate topology and event drivers.
- Consults the Broker for the deployment and establishes a connection to the IC-SHM domain.
- Subscribes to and imports topology and events from the IC-SHM domain.

Specifying the SHM domain for configuration scenario 2

To specify the SHM domain for configuration scenario 2:

1. In the left panel of the Global Manager Administration Console, right-click **Domains** and then click the pop-up menu to launch the **Domain Creation Wizard**.
2. Consult Figure 6 on page 52 and follow the directions to create a Health Monitor domain named **SHM**.
3. In the toolbar area of the Global Manager Administration Console, click the **Reconfigure** toolbar button ( ).
4. In the **Server Reconfiguration** confirmation dialog box that appears, click **Yes**.
5. In the **Server Reconfiguration** information dialog box that appears, click **Close**.

In response, the Global Manager:

- Creates an InChargeDomain object named SHM and starts the appropriate topology and event drivers.
- Consults the Broker for the deployment and establishes a connection to the SHM domain.
- Subscribes to and imports topology and events from the SHM domain.

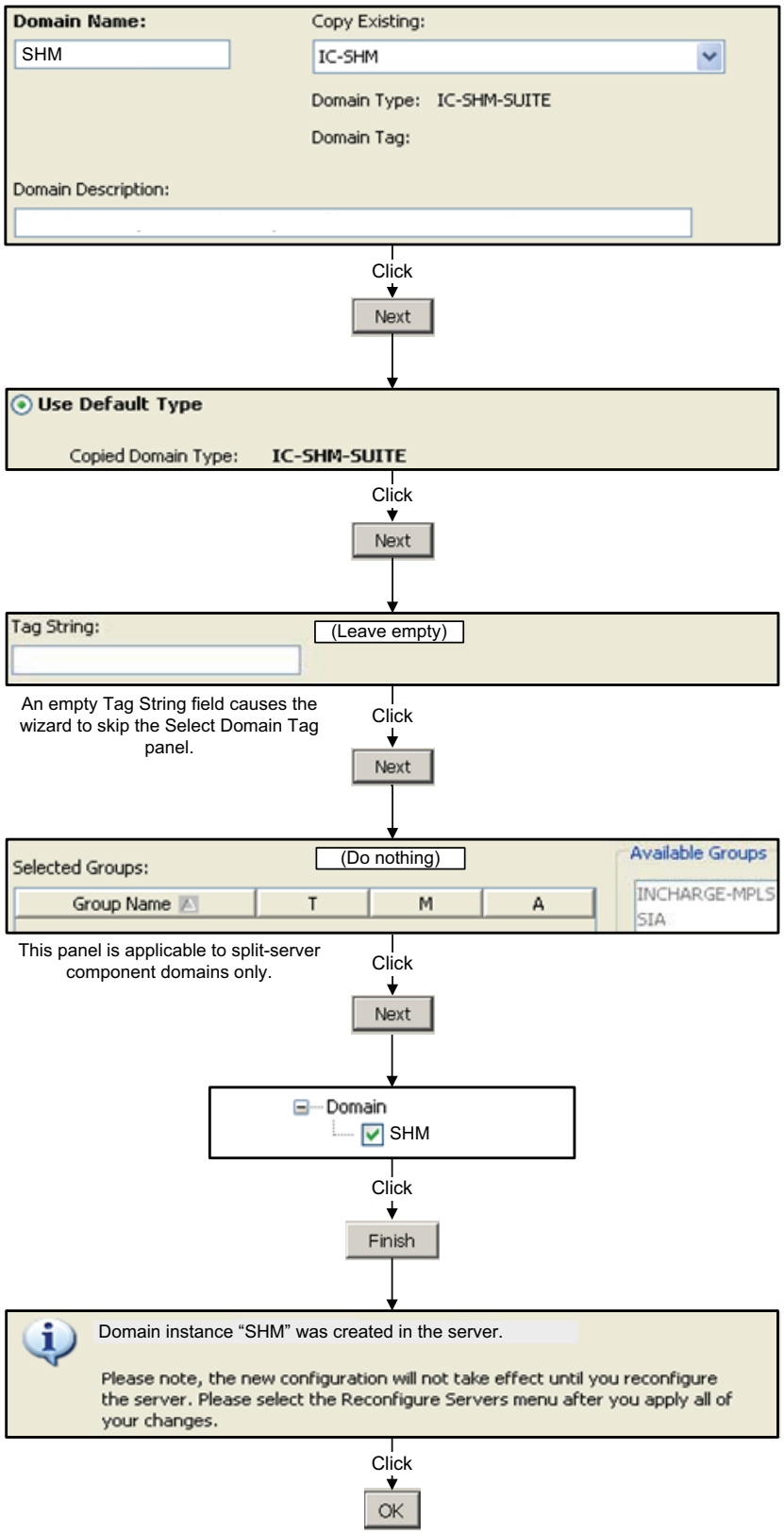


Figure 6 Using the Domain Creation Wizard to create the SHM domain

Loading the XML configuration file for the Health Monitor

If the IC-SHM domain does not appear in the Domains configuration window of the Global Manager Administration Console, use the Import XML Domain Configuration server tool that is shown in Figure 7 on page 53 to load the XML domain configuration file for the Health Monitor.

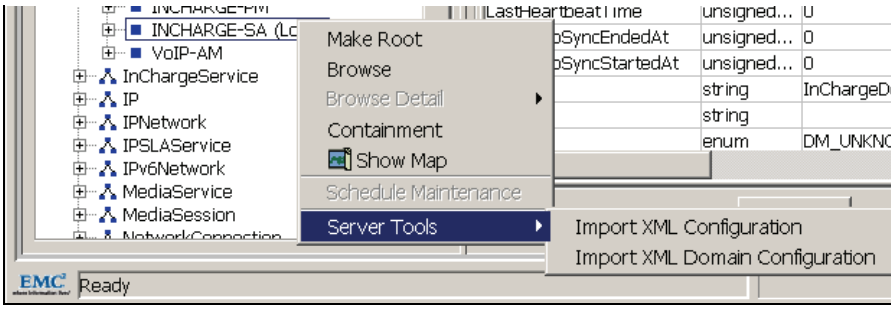


Figure 7 Access to server tools for loading XML configuration files

The Import XML Domain Configuration server tool is described in the *EMC Smarts Service Assurance Manager Configuration Guide*.

To load the XML domain configuration file for the Health Monitor:

1. In the Global Manager Administration Console, select **File › New › Topology Browser Console** to launch a Topology Browser Console.
2. In the Topology Browser Console, expand the topology tree to display the **InChargeDomain** objects.
3. Right-click the local **InChargeDomain** object that represents the Global Manager (for example, **INCHARGE-SA (Local)**) and select **Server Tools › Import XML Domain Configuration** from the pop-up menu.

   The **Import XML Domain Configuration** dialog box appears.
4. To load the XML domain configuration file for the Health Monitor, type **shm** in the **Domain Type Descriptor** field and click **OK**.
5. Switch back to the Global Manager Administration Console.
6. In the left panel of the Global Manager Administration Console, expand the configuration tree to display the **Domains**.

   The Health Monitor domain appears in the **Domains** list as **IC-SHM**. Because the name appears as gray text, the domain is disabled.

CHAPTER 4
Customizing Polling and Thresholds

This chapter describes how to configure polling and threshold groups for the Health Monitor. It consists of the following sections:

- Before you start 56
- Viewing the Polling and Thresholds tabs 57
- Opening the Polling and Thresholds Console 59
- Understanding polling and thresholds 60
- Understanding polling groups, threshold groups, and settings 61
- Modifying polling and threshold groups 61
- Default polling groups and settings 62
- Default threshold groups and settings 64
- Working with groups and settings 68

Before you start

By default, the default threshold values or the customized threshold values that a user sets through the Polling and Thresholds Console are overridden by the values that are calculated by the automatic dynamic thresholding feature. Thresholds and their default values are identified in Table 6 on page 63, Table 8 on page 65, and Table 9 on page 66, and automatic dynamic thresholding is described in "Automatic dynamic thresholding" on page 23.

The default or customized threshold values are *global* in the sense that a single set of threshold values applies to *all* discovered Managers.

The dynamically calculated threshold values are *local* in the sense that a set of threshold values is created for *each* discovered Manager.

You *selectively* disable automatic dynamic thresholding through the Polling and Thresholds Console by setting the "DynamicThresholding parameter" to DISABLED. The DynamicThresholding parameter is described in Table 6 on page 63, Table 8 on page 65, and Table 9 on page 66. Disabling automatic dynamic thresholding activates the default threshold values or the customized threshold values that are set through the Polling and Thresholds Console.

For accurate baseline values for the various thresholds, EMC recommends that you do not disable automatic dynamic thresholding.

Viewing the Polling and Thresholds tabs

Figure 8 on page 57 shows the Polling tab of a Polling and Thresholds Console that is connected to a Health Monitor named IC-SHM. The right panel of the console identifies "Queues Polling Setting" as the setting for the default polling group "Queue_Groups."

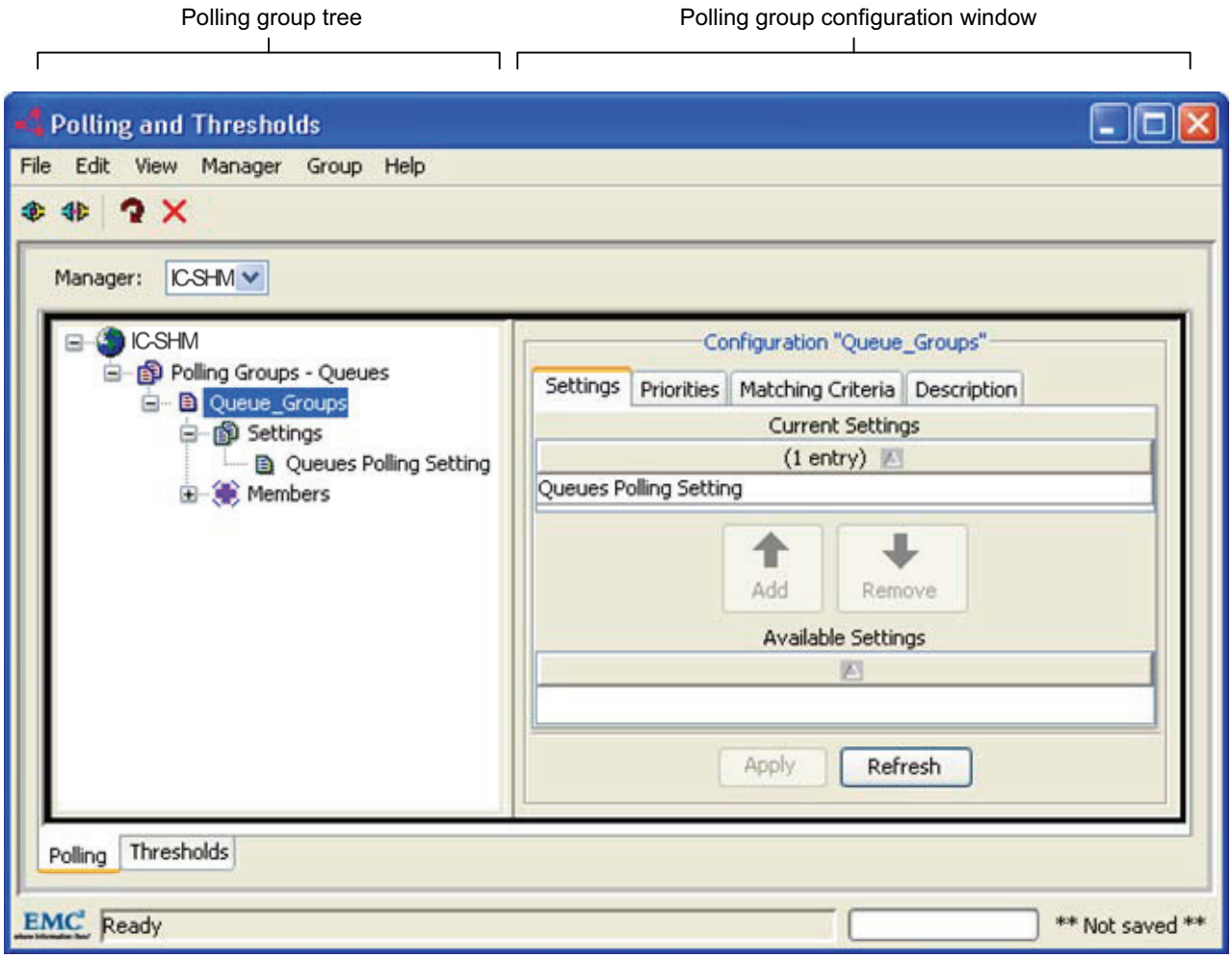


Figure 8 Polling tab for the Health Monitor

Figure 9 on page 58 shows the Thresholds tab of the Polling and Thresholds Console that is connected to a Health Monitor named IC-SHM. The right panel of the console identifies "System Health Metrics Thresholds" as the setting for the default threshold group "Host_Groups."

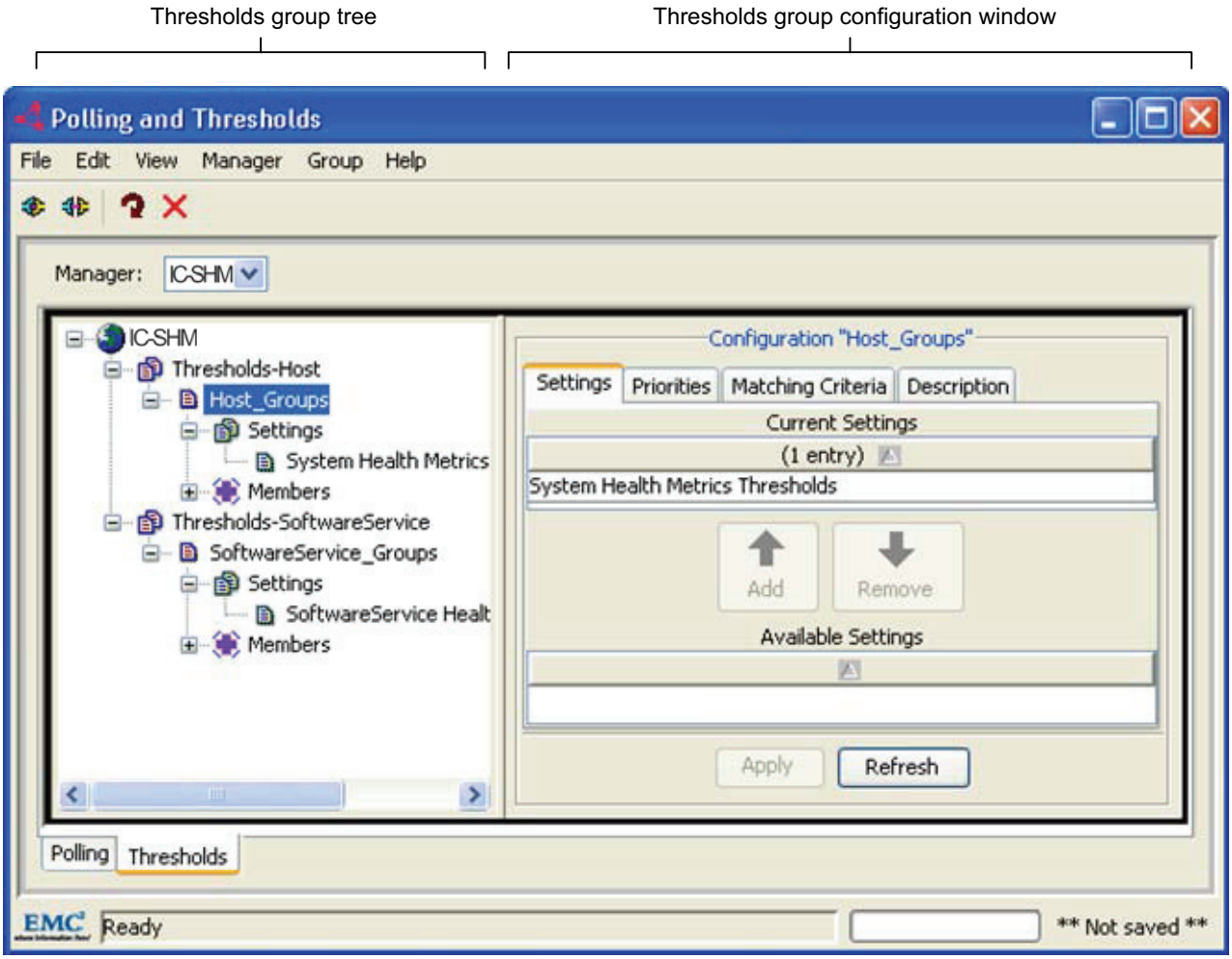


Figure 9 Thresholds tab for the Health Monitor

Opening the Polling and Thresholds Console

Accessing the Polling and Thresholds Console requires attaching the Global Console to the Health Monitor (IC-SHM, for example) in your deployment.

User accounts and passwords

Attaching the Global Console to the Health Monitor requires an account with administrative privileges. The default administrative account is username *admin* and password *changeme*.

Procedure for opening the Polling and Thresholds Console

To open the Polling and Thresholds Console:

1. Start the Global Console.
   - On a Windows system, select **Start › Programs › InCharge › EMC Ionix Global Console**.
   - On a UNIX system, go to the BASEDIR/smarts/bin directory in the Service Assurance Manager (Global Manager) installation area and enter:

     sm_gui

   The Attach Manager dialog box opens.
2. In the dialog box:
   a. Ensure that the EMC Smarts Broker for your deployment appears in the **Broker** text box.
   b. Click the **Manager** list box or the **Browse** button to display a list of active (running) Managers, and from that list select the Health Monitor (**IC-SHM**, for example) in your deployment as the Manager to which you want to connect.
   c. Type your login username and password.
   d. Click **OK**.

   The Topology Browser Console opens.
3. In the Topology Browser Console, select **Configure › Domain Manager Administration Console** to open the Domain Manager Administration Console.

   The Domain Manager Administration Console opens.
4. In the Domain Manager Administration Console, select **Configure › Polling and Thresholds** to open the Polling and Thresholds Console.

   The Polling and Thresholds Console opens.

Understanding polling and thresholds

As indicated in Figure 10 on page 60, the Health Monitor uses EMC Smarts remote API to monitor the managed environment.

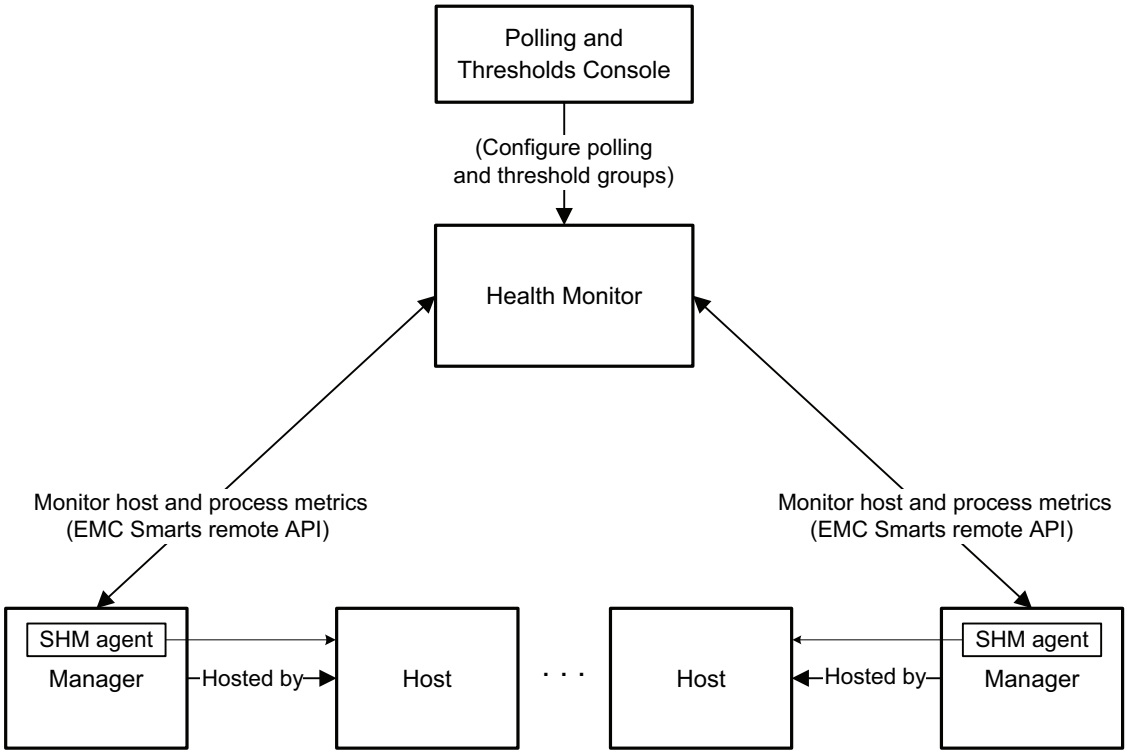


Figure 10 Polling configuration and operation

The monitoring is accomplished by periodically calling the SHM agent of each discovered Manager for host and process data. The SHM agent returns the following data:

- Host CPU, physical-memory, and virtual-memory performance metrics
- Process queue, CPU, physical-memory, virtual-memory, heap, file-handle, and other performance metrics

NOTICE

"Process," as used here, refers to instances of the InChargeService class, which is described in Chapter 5, "Topology Objects and Their Failures."

The Health Monitor also subscribes to process heap-headroom metric updates on the discovered Managers.

The monitoring is controlled by polling-related and threshold-related objects that are created for the Health Monitor through the Polling and Thresholds Console. The monitoring is also controlled by certain attributes of the SHM_InChargeServerInterface object, as explained in Chapter 2, "Configuring the Health Monitor."

Understanding polling groups, threshold groups, and settings

Using the Polling and Thresholds Console, you can customize the polling for the Health Monitor by modifying the default polling group, or by creating a new polling group. The polling group periodically polls the Managers to collect performance metrics.

Using the Polling and Thresholds Console, you can also customize the polling thresholds for the Health Monitor by modifying default performance threshold groups, or by creating new threshold groups.

The Health Monitor uses the *settings* in a polling or threshold group to assign parameters to the group.

Modifying polling and threshold groups

Procedures for modifying existing groups and creating new ones are given in "Working with groups and settings" on page 68.

When you finish editing the properties of a group, perform these steps:

1. Click the **Apply** button to save your configuration changes to an existing group object or to a newly created group object in the repository of the Health Monitor.
2. Click the **Reconfigure** toolbar button ( ) to make your configuration changes take effect.

When the reconfiguration completes, the Health Monitor imports the group object and saves the group object to its repository. Then, for a polling group object, the Health Monitor starts the polling that is configured for the object; for a threshold group object, the Health Monitor starts using the polling thresholds that are configured for the object.

Default polling groups and settings

The Health Monitor provides one default polling group. Table 5 on page 62 lists the default polling group, its target class, and its setting.

Table 5 Default Health Monitor polling group

Polling category	Polling group	Target class	Setting
Polling Groups - Queues	Queue_Groups	UnitaryComputerSystem (no specified attributes)	Queues Polling Setting

Two things to note about Table 5:

- Only discovered objects that match the type and attribute values in the "Target class" column can become members of the group.
- By default, all discovered objects that match the type and attribute values in the "Target class" column become members of the group.

  You can specify matching criteria for a polling group to limit its membership, as explained in "Edit matching criteria" on page 70.

In addition to periodically gathering process queue performance metrics in accordance to the polling group, the Health Monitor also periodically gathers host performance metrics and other process performance metrics, and subscribes to process heap-headroom metric updates on the discovered Managers. Through these subscriptions, the Health Monitor receives once per minute heap-headroom metrics from each discovered Manager.

All host and process performance metrics, including the heap-headroom metrics, serve as input to the Health Monitor's correlation analysis function.

Queue_Groups polling group

Through the Queues Polling Setting of the Queue_Groups polling group, you control the remote API polling of process queue performance metrics.

Queues Polling Setting

Table 6 on page 63 lists the Queues Polling Setting parameters.

Table 6 Queues Polling Setting parameters and their values

Parameter	Values	Description
AnalysisMode	ENABLED DISABLED Default: ENABLED	Enables or disables: 1. The Health Monitor polling of process queue performance metrics. 2. The subsequent analysis of the polled data to determine the status of the process queue performance metrics. **Note:** AnalyisMode is valid only when the disable_polling parameter for the SHM_InChargeServerInterface object is set to its default (FALSE) value. See "Disabling polling" on page 44 for details.
DynamicThresholding	ENABLED DISABLED Default: ENABLED	Enables or disables the automatic dynamic thresholding feature for the threshold parameters that are identified in this table. The automatic dynamic thresholding feature is described in "Automatic dynamic thresholding" on page 23.
PollingCyclesBacklogged-Threshold [1]	Unsigned integer Default: 60	Acceptable number of consecutive polling cycles for which the number of entries on this queue exceeds the QueueSizeThreshold.
PollingCyclesNotServiced-Threshold [1]	Unsigned integer Default: 30	Acceptable number of consecutive polling cycles for which no entries on this queue are processed.
QueueSizeThreshold [1]	Unsigned integer Default: 100	Number-of-entries threshold for this queue. See the PollingCyclesBackloggedThreshold in this table for an understanding of how this threshold is used.

[1] Dynamically calculated when automatic dynamic thresholding is enabled (default).

If you determine that the default threshold values are not appropriate for particular queues in your environment, change the default values, or create another queue polling group and specify matching criteria that allow only the hosts of the Managers with those particular queues to become members of the group. Give the new queue polling group a higher priority than the default queue polling group.

Procedures for performing these tasks are given in "Working with groups and settings" on page 68. Do a reconfigure ( ) upon completing these tasks.

Default threshold groups and settings

The Health Monitor provides two default threshold groups. Table 7 on page 64 lists the default threshold groups, their target classes, and their settings.

Table 7 Default Health Monitor threshold groups

Threshold category	Threshold group	Target class	Setting
Thresholds-Host	Host_Groups	UnitaryComputerSystem (no specified attributes)	System Health Metrics Thresholds
Thresholds-SoftwareService	SoftwareService_Groups	SoftwareService (no specified attributes)	SoftwareService Health Metrics Thresholds

Two things to note about Table 7:

- Only discovered objects that match the type and attribute values in the “Target class” column can become members of the group.
- By default, all discovered objects that match the type and attribute values in the “Target class” column become members of the group.

  You can specify matching criteria for a threshold group to limit its membership, as explained in “Edit matching criteria” on page 70.

Host_Groups threshold group

Through the System Health Metrics Thresholds setting of the Host_Groups threshold group, you define the thresholds to which the polled host CPU and memory performance metrics will be compared, to determine acceptable and unacceptable levels of performance.

System Health Metrics Thresholds setting

Table 8 on page 65 lists the System Health Metrics Thresholds setting parameters.

Table 8 System Health Metrics Thresholds parameters and their values

Parameter	Values	Description
CpuUtilizationThreshold [1]	Unsigned integer Default: 90	Threshold that is used to determine whether the host is overutilizing its CPU. The threshold is expressed as a percentage of host CPU usage.
DynamicThresholding	ENABLED DISABLED Default: ENABLED	Enables or disables the automatic dynamic thresholding feature for the threshold parameters that are identified in this table. The automatic dynamic thresholding feature is described in "Automatic dynamic thresholding" on page 23.
TotalPhysicalMemory-UtilizationThreshold [1]	Unsigned integer Default: 90	Threshold that is used to determine whether the host is overutilizing its physical memory. The threshold is expressed as a percentage of total physical-memory usage for the host.
TotalVirtualMemory-UtilizationThreshold [1]	Unsigned integer Default: 70	Threshold that is used to determine whether the host is overutilizing its virtual memory. The threshold is expressed as a percentage of total virtual-memory usage for the host.

[1] Dynamically calculated when automatic dynamic thresholding is enabled (default).

If you determine that the default threshold values are not appropriate for particular hosts in your environment, change the default values, or create another host threshold group and specify matching criteria that allow only those particular hosts to become members of the group. Give the new host threshold group a higher priority than the default host threshold group.

Procedures for performing these tasks are given in "Working with groups and settings" on page 68. Do a reconfigure ( ) upon completing these tasks.

SoftwareService_Groups threshold group

Through the SoftwareService Health Metrics Thresholds setting of the SoftwareService_Groups threshold group, you define the thresholds to which the polled or subscribed-to process CPU, memory, heap, file-handle, and other performance metrics will be compared, to determine acceptable and unacceptable levels of performance.

SoftwareService Health Metrics Thresholds setting

Table 9 on page 66 lists the SoftwareService Health Metrics Thresholds setting parameters.

Table 9 SoftwareService Health Metrics Thresholds parameters and their values (page 1 of 2)

Parameter	Values	Description
CorrelationTimeThreshold [1]	Unsigned integer Default: 200	Threshold, in seconds, that is used to determine whether the last correlation time for the software service took too long.
CpuUtilizationThreshold [1]	Floating point number Default: 90	Threshold that is used to determine whether the software service is consuming excessive CPU resources. The threshold is expressed as a percentage of host CPU usage.
DebugDataCollection-Interval	Unsigned long integer Default: 20	Upon an event notification, time, in seconds, that the software service waits before collecting another debug-data sample.
DebugDataCollection-Iterations	Unsigned long integer Default: 5	Upon an event notification, number of times that the software service collects a debug-data sample.
DynamicThresholding	ENABLED DISABLED Default: ENABLED	Enables or disables the automatic dynamic thresholding feature for the threshold parameters that are identified in this table. The automatic dynamic thresholding feature is described in "Automatic dynamic thresholding" on page 23.
FullDiscoveryTime-Threshold [1]	Unsigned integer Default: 600	Threshold, in seconds, that is used to determine whether the last full discovery time for the software service took too long.
HeapSizeThreshold	Unsigned integer Default: 10	Threshold that is used to determine when the heap headroom for the software service shrinks to an unacceptable level. The threshold is expressed as a percentage of the total heap size that is allowed for the software service.
PendingDiscoveryTime-Threshold [1]	Unsigned integer Default: 600	Threshold, in seconds, that is used to determine whether the last pending discovery time for the software service took too long.

[1] Dynamically calculated when automatic dynamic thresholding is enabled (default).

Table 9 SoftwareService Health Metrics Thresholds parameters and their values (page 2 of 2)

Parameter	Values	Description
PhysicalMemoryUtilization-Threshold [1]	Floating point number Default: 70	Threshold that is used to determine whether all of the software modules that realize the software service, when taken together, are consuming excessive physical-memory resources. The threshold is expressed as a percentage of physical-memory usage for the software service.
ProcessFileHandles-Threshold	Unsigned integer Default: 50	Threshold that is used to determine when the number of file handles for the software service increases to an unacceptable level. The threshold is expressed as a percentage of the maximum file handles that are allowed for the software service.
QueuePollingIntervalsToSkip	Unsigned integer Default: 1 (no skipping)	Number of queue monitoring polling cycles that the Health Monitor will skip during the peak load periods of the software service.
RecomputeTimeThreshold [1]	Unsigned integer Default: 200	Threshold, in seconds, that is used to determine whether the last recompute time for the software service took too long.
ReconfigureTimeThreshold [1]	Unsigned integer Default: 200	Threshold, in seconds, that is used to determine whether the last reconfigure time for the software service took too long.
VirtualMemoryUtilization-Threshold [1]	Floating point number Default: 70	Threshold that is used to determine whether all of the software modules that realize the software service, when taken together, are consuming excessive virtual-memory resources. The threshold is expressed as a percentage of virtual-memory usage for the software service.

[1] Dynamically calculated when automatic dynamic thresholding is enabled (default).

Working with groups and settings

You can use the Polling and Thresholds Console to perform any of the following tasks:

- "Assign managed objects to groups"
- "Modify the properties of a group"
- "Add or remove settings"
- "Modify the parameters of a setting"
- "Modify the priority of groups"
- "Edit matching criteria"
- "Create new groups"

Assign managed objects to groups

When the Health Monitor performs discovery, it automatically assigns each managed object to one or more groups based on the:

- Matching criteria for a group.
- Priority of a group, which determines membership when an object meets the matching criteria for more than one group within a particular group category.

A managed object may be a member of one and only one group within a particular group category.

Modify the properties of a group

A group's matching criteria and priority determine which managed objects become members of the group.

When a group is selected in the left panel of the Polling and Thresholds Console, four tabs appear in the right panel of the console:

- Settings
- Priorities
- Matching Criteria
- Description

Modifying the properties in the Settings, Priorities, or Matching Criteria tab changes the configuration of the group.

When you finish editing the properties of a group, perform these steps:

1. Click the **Apply** button to save the changes.
2. Click the **Reconfigure** toolbar button, or select **Group › Reconfigure**, to make the configuration changes take effect.

Add or remove settings

A group's settings determine what parameters are applied to the managed objects that are members of the group.

The Settings tab is divided into two sections:

- Current Settings

  Lists the settings that are applied to the group.

- Available Settings

  Lists additional available settings.

To add or remove a setting:

1. In the **Settings** tab, select a setting in the **Available Settings** list or in the **Current Settings** list.
2. Click **Add** to move an available setting to the **Current Settings** list, or click **Remove** to move a current setting to the **Available Settings** list.
3. Click the **Apply** button to save your changes.
4. Click the **Reconfigure** toolbar button to make your changes take effect.

Modify the parameters of a setting

The parameters of a setting are changed in one of two ways:

- By choosing a value from a drop-down menu.
- By entering a value in a Value field or adjusting a slider bar that represents a range of values.

Changing the parameters of a setting

To change the parameters of a setting:

1. In the left panel of the **Polling and Thresholds Console**, select the setting to display the parameters of the setting in the right panel of the console.
2. Change the value of a parameter by using one of the following methods:

   For a drop-down menu, click the menu and select a value.

   For a slider bar presentation, do one of the following:

   - Type a value into the **Value** field and press **Enter**.
   - Select the slider bar and drag its handle with the mouse to change the value or select the slider bar and use the arrow keys to move its handle to change the value.
3. Click the **Apply** button to save your changes.
4. Click the **Reconfigure** toolbar button to make your changes take effect.

Restoring the default values of a setting

To restore the default values of a setting:

1. In the left panel of the **Polling and Thresholds Console**, select the setting to display the parameters of the setting in the right panel of the console.
2. Click **Restore Defaults**.
3. Click the **Reconfigure** toolbar button to make your changes take effect.

Modify the priority of groups

In any group category, you can arrange the groups in a high-to-low priority. A candidate member object is compared against the highest priority group first and becomes a member of the first group for which there is a criteria match.

The Priorities tab shows groups in the order of their priority, from highest to lowest.

To change the priority of a group:

1. In the **Priorities** tab, select the group for which you want to change the priority.
2. Click the up or down arrow to change its position relative to the other groups.
3. Click the **Apply** button to save your changes.
4. Click the **Reconfigure** toolbar button to make your changes take effect.

Edit matching criteria

A group's matching criteria, which appear at the top of the Matching Criteria tab, are defined by using the attributes of the target class that is associated with the group. Each matching criterion has three fields: Name, Description, and Value.

- Name identifies the name of the target class attribute.
- Description is the description of the target class attribute.
- Value is the value of the target class attribute, which can be any combination of text, integers, and wildcards. The Value field for a matching criterion is not case-sensitive.

As an example of a matching criterion for target class UnitaryComputerSystem:

- If you specify "Router" for the Type attribute, only routers become members of the group.
- If you specify "R1*" for the Name attribute, only routers with names that begin with the string R1 become members of the group.

Appendix B, "Wildcard Patterns," discusses wildcard syntax.

When a managed object matches a group's target class and matches all of the attributes that are defined for the target class, the managed object becomes a member of the group.

Adding or removing matching criteria

To add or remove matching criteria:

1. In the **Disabled Criteria** area of the **Matching Criteria** tab, select a matching criterion.
2. Click **Enable** to make the criterion active, moving it to the top of the **Matching Criteria** tab.

   Use **Disable** to deactivate the criterion, moving it to the **Disabled Criteria** area of the **Matching Criteria** tab.
3. If you are adding a matching criterion, type a matching pattern in the **Value** field.
4. Click the **Apply** button to save your changes.
5. Click the **Reconfigure** toolbar button to make your changes take effect.

Changing the value of a matching criterion

To change the value of a matching criterion:

1. In the **Matching Criteria** tab, select the string in the **Value** field or double-click the **Value** field to highlight the current value.
2. Type the new value.
3. Click the **Apply** button to save your changes.
4. Click the **Reconfigure** toolbar button to make your changes take effect.

The Health Monitor compares a managed object against the matching criteria of the group with the highest priority. If an object matches all the criteria, the Health Monitor adds the object as a member of the group. Otherwise, the Health Monitor compares the object against the matching criteria of the group with the second highest priority, and so on.

Create new groups

You can create additional groups within any group category.

Creating a new group enables you to customize the settings for a group of managed objects. You can use two methods to create a new group:

- Copy an existing group. The new group contains the same settings and thresholds as the original group. Matching criteria are not copied.
- Create an empty group. The new group does not contain any settings or members. You must add settings and matching criteria, and set the priority of the new group.

After you create a new group, use the procedures described in this chapter to add settings and matching criteria and to set the priority of the new group.

Copying an existing group

To copy an existing group:

1. In the left panel of the **Polling and Thresholds Console**, right-click the group that you want to copy.
2. Select **Copy** from the pop-up menu to display the **Copy Group** dialog box.
3. In the dialog box, type a name and an optional description for the new group and click **OK**.

   The new group contains the same settings and thresholds as the group you copied.
4. Edit the settings, matching criteria, and priority of the new group. Change the value of any parameters as necessary.
5. Click the **Reconfigure** toolbar button to make your changes take effect.

Creating an empty group

To create an empty group:

1. In the left panel of the **Polling and Thresholds Console**, right-click the group for which you want to create a new group.
2. Select **New Group** from the pop-up menu to display the **New Group** dialog box.
3. In the dialog box, type a name and an optional description for the new group and click **OK**.
4. Add settings and matching criteria, and set the priority of the new group. Change the values of any thresholds or parameters as necessary.
5. Click the **Reconfigure** toolbar button to make your changes take effect.

CHAPTER 5
Topology Objects and Their Failures

This chapter identifies the topology objects that are discovered and monitored by the Health Monitor, and describes the root-cause problems that are diagnosed for the objects. It contains the following sections:

- Root-cause analysis 74
- Data model 74
- Summary of root-cause problems and events 75
- Topology object attributes, problems, and events 78

Root-cause analysis

The Health Monitor performs availability- and performance-based root-cause analysis of the discovered Managers and hosts in the managed deployment. It also relates diagnosed problems to the Managers, along the affected data path, that are no longer accessible or suffer from lack of response or performance as a result of the problems.

Data model

The Health Monitor builds a data model of the discovered Managers and hosts in its domain. The model, shown in Figure 11 on page 74, represents the Manager and host objects, their relationships, their internal components, and their logical connections.

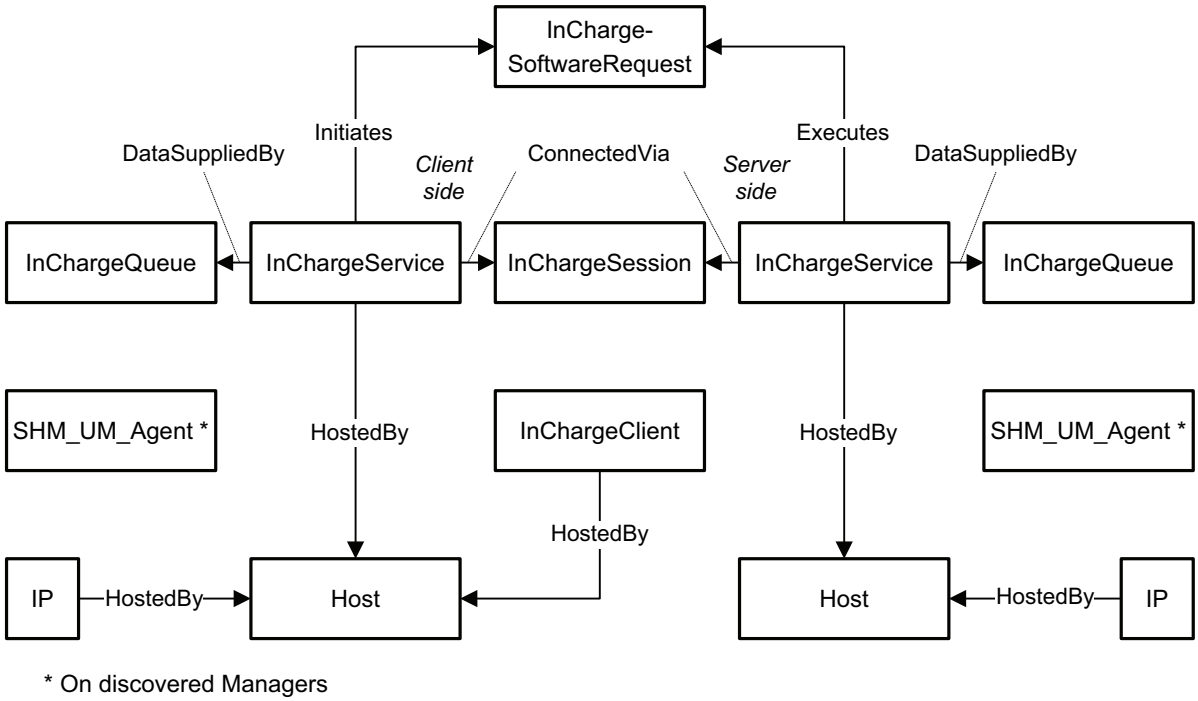


Figure 11 Health Monitor data model

The objects are represented as instances of the following EMC Smarts classes:

- Host
- InChargeClient
- InChargeQueue
- InChargeService
- InChargeSession
- InChargeSoftwareRequest
- IP
- SHM_UM_Agent

InChargeClient and InChargeService are both software services, and InChargeClient is a child of InChargeService. Here is the class hierarchy:

InChargeClient : InChargeService : MgmtService : SoftwareService

Also, InChargeSession is a child of Session, and InChargeSoftwareRequest is a child of SoftwareRequest.

The Global Manager imports InChargeClient, InChargeService, InChargeSession, and InChargeSoftwareRequest objects from the Health Monitor. It imports InChargeClient objects as SoftwareService objects, InChargeService objects as InChargeService objects, InChargeSession objects as Session objects, and InChargeSoftwareRequest objects as SoftwareRequest objects.

The *EMC Smarts ICIM Reference* provides descriptions of EMC Smarts classes, relationships, connections, and hierarchies.

Summary of root-cause problems and events

The Health Monitor diagnoses root-cause problems and computes (detects) events from metric updates that are acquired in the following ways:

- By using EMC Smarts remote API to periodically gather the following performance metrics from the SHM agents on the discovered Managers:
  - Host CPU, physical-memory, and virtual-memory performance metrics
  - Process queue, CPU, physical-memory, virtual-memory, heap, file-handle, and other performance metrics
- By subscribing to heap-headroom metric updates on the discovered Managers

The Health Monitor correlates these events to diagnose root-cause problems.

NOTICE

Events that are used as symptoms to diagnose a root-cause problem can also be perceived as impacts that are caused by the problem.

The Global Manager imports:

- All problems that are diagnosed by the Health Monitor.
- Key events that are computed by the Health Monitor.

The root-cause problem notifications (one problem per notification) and the event notifications (one event per notification) are displayed in the Notification Log Console view of the Global Console.

Root-cause problems

Table 10 on page 76 lists the root-cause problems that are diagnosed by the Health Monitor. For each problem, Table 10 identifies the managed object for which the problem is diagnosed, the condition for the problem, and the events that are correlated by the Health Monitor to diagnose the problem.

Table 10 Root-cause problems diagnosed by the Health Monitor

Managed object	Root cause	Condition	Events (symptoms) of root cause [1]
Host	AllServicesDown	All InChargeServices on this host are down. The host, itself, might be down.	Host::AllServicesDown_Event
	ApproachingHost-ResourceLimit	This host is becoming unresponsive due to a low level of available system resources.	Host::HighCpuUtilization Host::HighPhysicalMemoryUtilization Host::HighVirtualMemoryUtilization
InChargeService	Down	This InChargeService is down.	InChargeService::ProcessDown
	Malfunction	This InChargeService is not functioning correctly.	InChargeService::ApproachingMax-FileHandlesLimit InChargeService::CorrelationIsSlow InChargeService::FullDiscovery-TimeExceeded InChargeService::HighCPUProcess-Utilization InChargeService::HighPhysical-MemoryUtilization InChargeService::HighVirtual-MemoryUtilization InChargeService::PendingDiscovery-TimeExceeded InChargeService::RecomputeIsSlow InChargeService::ReconfigureIsSlow InChargeService::RunningOutOf-Memory

[1] Interpret event entries as ‹object name›::‹event name›.

A problem indicates that one or more events are occurring in a particular host (Host) or Manager (InChargeService). As an example, when the physical-memory usage of a particular Manager exceeds the PhysicalMemoryUtilizationThreshold, the Health Monitor computes a HighPhysicalMemoryUtilization event and diagnoses a Malfunction problem for the Manager.

Problem notifications appear in the Notification Log Console. For a performance-type problem, the notification automatically clears when the polled value of the associated metric drops below its threshold value.

Events

Table 11 on page 77 lists the events that are computed by the Health Monitor and used as symptoms to diagnose root-cause problems. For each event, Table 11 identifies the managed object for the event, the condition or threshold that triggers the event, and whether the event is imported by the Global Manager.

Table 11 Events computed by the Health Monitor

Managed object	Event (symptom)	Condition or threshold
Host	AllServicesDown_Event	All InChargeServices on this host are down.
	HighCpuUtilization	CpuUtilizationThreshold
	HighPhysicalMemoryUtilization [1]	TotalPhysicalMemoryUtilizationThreshold
	HighVirtualMemoryUtilization [1]	TotalVirtualMemoryUtilizationThreshold
InChargeQueue	QueueBacklogged [1]	PollingCyclesBackloggedThreshold, QueueSizeThreshold
	QueueNotServiced [1]	PollingCyclesNotServicedThreshold
InChargeService	ApproachingMaxFileHandlesLimit [1]	ProcessFileHandlesThreshold
	CorrelationIsSlow	CorrelationTimeThreshold
	FullDiscoveryTimeExceeded [1]	FullDiscoveryTimeThreshold
	HighCPUProcessUtilization	CpuUtilizationThreshold
	HighPhysicalMemoryUtilization [1]	PhysicalMemoryUtilizationThreshold
	HighVirtualMemoryUtilization [1]	VirtualMemoryUtilizationThreshold
	PendingDiscoveryTimeExceeded [1]	PendingDiscoveryTimeThreshold
	PrimaryMissingOrDown [1]	The primary software service that is configured for this standby software service could not be discovered, or was discovered but is now down.
	ProcessDown [1]	This software service is down.
	RecomputeIsSlow	RecomputeTimeThreshold
	ReconfigureIsSlow	ReconfigureTimeThreshold
	RunningOutOfMemory [1, 2]	HeapSizeThreshold
	StdbyMissingOrDown [1]	The standby software service that is configured for this primary software service could not be discovered, or was discovered but is now down.
InChargeSession	Disconnected [1]	The InChargeSession is not established.

[1] Imported by the Global Manager.
[2] Applicable only to an InChargeService object that is hosted by a Solaris- or Linux-based host.

In Table 11, Availability events are AllServicesDown_Event, PrimaryMissingOrDown, ProcessDown, StdbyMissingOrDown, and Disconnected. All other events are performance events. A performance event occurs when some aspect of a host (Host), queue (InChargeQueue), or Manager (InChargeService) exceeds a performance threshold.

Event notifications appear in the Notification Log Console. For a performance-type event, the notification automatically clears when the polled value of the associated metric drops below its threshold value.

Topology object attributes, problems, and events

The sections that follow describe key attributes, diagnosed problems, and computed events for the following categories of Manager and host object types:

- "Host"
- "InChargeClient"
- "InChargeQueue"
- "InChargeService"
- "InChargeSession"
- "InChargeSoftwareRequest"
- "IP"
- "SHM_UM_Agent"

Host

A Host is a workstation or server. For each discovered Manager, the Health Monitor creates a Host object for the host on which the Manager runs.

Attributes for Host

Table 12 on page 79 lists some key attributes for Host.

Table 12 Attributes for Host

Attribute	Description	Allowed values
CpuUtilization	Percentage of CPU usage for this host.	Floating point number
CpuUtilizationThreshold	Threshold that is used to determine whether this host is overutilizing its CPU. The threshold is for CpuUtilization.	Unsigned integer Default: 90
NumberOfServices	Total number of discovered software services (excluding InChargeClients) on this host.	Unsigned integer
ServiceProcessorCycles	During the last polling cycle, total number of CPU cycles that were used by all of the discovered software services (InChargeServices) on this host.	Unsigned long integer
TotalFreePhysicalMemory	Total bytes of free physical memory for this host.	Unsigned long integer
TotalPhysicalMemory	Total bytes of physical memory for this host.	Unsigned long integer
TotalPhysicalMemory-Utilization	Percentage of total physical-memory usage for this host.	Floating point number
TotalPhysicalMemory-UtilizationThreshold	Threshold that is used to determine whether this host is overutilizing its physical memory. The threshold is for TotalPhysicalMemoryUtilization.	Unsigned integer Default: 90
TotalProcessorCycles	During the last polling cycle, total number of CPU cycles that were used by this host. The number is the sum of the CPU cycles for each of the CPU processors on this host.	Unsigned long integer
TotalProcessors	Total number of CPU processors on this host.	Unsigned integer
TotalVirtualMemoryUtilization	Percentage of total virtual-memory usage for this host. This metric is not supported for Windows.	Floating point number
TotalVirtualMemory-UtilizationThreshold	Threshold that is used to determine whether this host is overutilizing its virtual memory. The threshold is for TotalVirtualMemoryUtilization.	Unsigned integer Default: 70

Root-cause problems for Host

Table 13 on page 80 lists the root-cause problems that are diagnosed for Host.

Table 13 Root-cause problems for Host

Problem	Description
AllServicesDown	All InChargeServices on this host are down. The host, itself, might be down.
ApproachingHostResource-Limit	This host is becoming unresponsive due to a low level of available system resources.

Events for Host

Table 14 on page 80 lists the events that are computed for Host.

Table 14 Events for Host

Event	Description
AllServicesDown_Event	All InChargeServices on this host are down.
HighCpuUtilization	The CPU usage for this host has exceeded the CpuUtilizationThreshold.
HighPhysicalMemory-Utilization	The physical-memory usage for this host has exceeded the TotalPhysicalMemoryUtilizationThreshold.
HighVirtualMemoryUtilization	The virtual-memory usage for this host has exceeded the TotalVirtualMemoryUtilizationThreshold.

InChargeClient

An InChargeClient is any of the following:

- Console

  A Global Console that connects to one or more of the discovered Managers. The Health Monitor creates an InChargeClient object for each discovered Global Console.

  Because a Global Console does not register with the Broker, the Health Monitor discovers a Global Console through its connection to a discovered Manager. Only if the Global Console is running on a host on which a discovered Manager is running can the Health Monitor obtain metrics for the console.

- ASL driver

  A driver that is running within a Manager, such as an internally running adapter, and acts as a client to another Manager. The Health Monitor creates an InChargeClient object for each ASL driver that connects to a discovered Manager.

- Remote repository accessor handle

  A handle for a remote repository accessor that is running within a Manager and acts as a client to another Manager. The Health Monitor creates an InChargeClient object for each remote repository accessor that connects to a discovered Manager.

Attributes for InChargeClient

Table 15 on page 81 lists some key attributes for InChargeClient.

Table 15 Attributes for InChargeClient

Attribute	Description	Allowed values
ClientIP	IP address of this software service.	String
ClientPort	Port to which this software service is bound on the host where the software service is running.	Integer
ClientType	Client type of this software service: • ASL • Console • dmctl • Java • Perl • Remote Accessor • ‹custom client type› "Console" represents a Global Console, a Web Console, or a Business Dashboard, all of which are Java-based client applications that are delivered with the Service Assurance Manager. "Java" represents a non-console, Java-based client application.	String

The remaining attributes for InChargeClient are a subset of the InChargeService attributes, which are listed in Table 18 on page 83. The Health Monitor does not diagnose any problems or compute any events for InChargeClient objects.

InChargeQueue

An InChargeQueue is a queue within a Manager. The Health Monitor monitors various queue metric to compute events that are defined for InChargeQueue objects.

Attributes for InChargeQueue

Table 16 on page 81 lists some key attributes for InChargeQueue.

Table 16 Attributes for InChargeQueue (page 1 of 2)

Attribute	Description	Allowed values
CurrentQueueSize	Current number of entries on this queue.	Unsigned integer
MaxSize	Maximum number of entries that this queue has ever had.	Unsigned integer
PollingCyclesBacklogged-Threshold	Acceptable number of consecutive polling cycles for which the number of entries on this queue exceeds the QueueSizeThreshold.	Unsigned integer Default: 60

Table 16 Attributes for InChargeQueue (page 2 of 2)

Attribute	Description	Allowed values
PollingCyclesNotServiced-Threshold	Acceptable number of consecutive polling cycles for which no entries on this queue are processed.	Unsigned integer Default: 30
PollingCycles_NoProcessing	Number of consecutive polling cycles that have occurred without any entries on this queue being processed.	Unsigned integer
PollingCycles_SlowProcessing	Number of consecutive polling cycles that have occurred with the number of entries on this queue being greater than the QueueSizeThreshold.	Unsigned integer
ProcessedEntries	Total number of entries that have been processed for this queue.	Double floating point number
QueueSizeThreshold	Number-of-entries threshold for this queue. See the PollingCyclesBackloggedThreshold in this table for an understanding of how this threshold is used.	Unsigned integer Default: 100

Events for InChargeQueue

Table 17 on page 82 lists the events that are computed for InChargeQueue.

Table 17 Events for InChargeQueue

Event	Description
QueueBacklogged	The PollingCyclesBackloggedThreshold and QueueSizeThreshold have been exceeded: The acceptable number of consecutive polling cycles for which the number of entries on this queue are greater than the QueueSizeThreshold, has been exceeded. When a QueueBacklogged event occurs, the queuing service of the affected Manager is deemed to be working but at an unacceptable level, possibly due to a busy CPU or slow connections.
QueueNotServiced	The PollingCyclesNotServicedThreshold has been exceeded: The acceptable number of consecutive polling cycles for which no entries on this queue have been processed, has been exceeded. When a QueueNotServiced event occurs, the queuing service of the affected Manager is deemed to be inoperative, possibly due to a hanging thread or a broken connection.

InChargeService

An InChargeService is a Manager, that is, a Domain Manager, a Global Manager, an Adapter Platform, an Adapter, or the Broker. The Health Monitor creates an InChargeService object for each Manager that is registered with the Broker, for the Broker itself, and for each standby Manager, as explained in "Specifying a list of standby Managers for discovery" on page 39. It also creates an InChargeService object for itself.

Attributes for InChargeService

Table 18 on page 83 lists some key attributes for InChargeService.

Table 18 Attributes for InChargeService (page 1 of 6)

Attribute	Description	Allowed values
AvailableHeapHeadRoom	For a software service running as a 32- or 64-bit application, heap headroom, in kilobytes, that is available for the software service to expand in heap. Derived by subtracting CurrentHeapUsage from TotalHeap or TotalHeapSize. This metric is not supported for Windows.	Unsigned long integer
CorrelationTime	Time, in seconds, of the last correlation time for this software service.	Unsigned integer
CorrelationTimeThreshold	Threshold, in seconds, that is used to determine whether the last correlation time for this software service took too long.	Unsigned integer Default: 200
CpuUtilization	Percentage of host CPU usage for this software service.	Floating point number
CpuUtilizationThreshold	Threshold that is used to determine whether this software service is consuming excessive CPU resources. The threshold is for CpuUtilization.	Floating point number Default: 90
CurrentHeapUsage	Current heap usage, in kilobytes, for this software service.	Unsigned long integer
DebugDataCollectionInterval	Upon an event notification, time, in seconds, that this software service waits before collecting another debug-data sample.	Unsigned long integer Default: 20
DebugDataCollection-Iterations	Upon an event notification, number of times that this software service collects a debug-data sample.	Unsigned long integer Default: 5
DisconnectedFromRemote-ICServer	True when this software service has disconnected from the remote software service.	Boolean: true or false

Table 18 Attributes for InChargeService (page 2 of 6)

Attribute	Description	Allowed values
ErrCodeGettingUptime	Error code in fetching the "uptime" attribute for this software service; the uptime attribute is used to compute the InChargeSession::Disconnected event.	Unsigned integer
FullDiscoveryTimeThreshold	Threshold, in seconds, that is used to determine whether the last full discovery time for the software service took too long.	Unsigned integer Default: 600
HeapHeadRoom	For a software service running as a 32-bit application, heap headroom, in kilobytes, that is available for the software service to expand in heap. Derived by subtracting CurrentHeapUsage from TotalHeapSize. For a 64-bit application, see AvailableHeapHeadRoom. HeapHeadRoom is not supported for Windows. **Note:** For a 32-bit application, HeapHeadRoom and AvailableHeapHeadRoom have the same value.	Unsigned long integer
HeapSizeThreshold	Threshold that is used to determine when the heap headroom for this software service shrinks to an unacceptable level: AvailableHeapHeadRoom or HeapHeadRoom becomes less than HeapSizeThreshold x TotalHeap or TotalHeapSize. The threshold is expressed as a percentage of the TotalHeap or TotalHeapSize for the software service.	Unsigned integer Default: 10
IsStdby	True if this software service is configured as a standby Manager, and False if this software service is configured as a primary Manager. This setting is valid only when IsStdbyConfgd is True.	Boolean: true or false

Table 18 Attributes for InChargeService (page 3 of 6)

Attribute	Description	Allowed values
IsStdbyConfgd	True if either of the following conditions is in effect: • This software service is configured as a standby Manager *and* its primary Manager is discovered. • This software service is configured as a primary Manager *and* its standby Manager is discovered. False if one of the following conditions is in effect: • This software service is not configured as a standby Manager or a primary Manager. • This software service is configured as a standby Manager *but* its primary Manager is not discovered, or was discovered but is now down. • This software service is configured as a primary Manager *but* its standby Manager is not discovered, or was discovered but is now down.	Boolean: true or false
lastFullDiscovery-Duration_long	Time, in seconds, of the last full discovery time for this software service.	Unsigned long integer
lastPendingDiscovery-Duration_long	Time, in seconds, of the last pending discovery time for this software service.	Unsigned long integer
MaxFileHandles	Maximum number of file handles for this software service. For a software service on a Solaris- or Linux-based host, the limitation is imposed on the software service by the OS. For a software service on a Windows-based host, the limitation is imposed (set to 2048) by the Health Monitor. Reason: Windows does not impose a limit on the maximum number of file handles that a process may use.	Unsigned long integer
PendingDiscoveryTime-Threshold	Threshold, in seconds, that is used to determine whether the last pending discovery time for the software service took too long.	Unsigned integer Default: 600
PhysicalMemoryUtilization	Percentage of physical-memory usage for all of the software modules that realize this software service. Derived by dividing ResidentSetSize by TotalPhysicalMemory.	Floating point number

Table 18 Attributes for InChargeService (page 4 of 6)

Attribute	Description	Allowed values
PhysicalMemoryUtilization-Threshold	Threshold that is used to determine whether all of the software modules that realize this software service, when taken together, are consuming excessive physical-memory resources. The threshold is for PhysicalMemoryUtilization.	Floating point number Default: 70
PrimaryOrStdbyName	If this software service is configured as a primary Manager, name of the standby Manager that is configured for this software service. If this software service is configured as a standby Manager, name of the primary Manager that is configured for this software service. This attribute is valid only when IsStdbyConfgd is True.	String
ProcessFileHandlesThreshold	Threshold that is used to determine when the number of file handles for this software service increases to an unacceptable level: TotalFileHandles exceeds ProcessFileHandlesThreshold x MaxFileHandles. The threshold is expressed as a percentage of the MaxFileHandles for the software service.	Unsigned integer Default: 50
ProcessMaxVirtualMemory-Size	Maximum bytes of virtual memory that are allowed for this software service. For a software service running as a 32-bit application, the Health Monitor sets this value to 4294967296, which corresponds to the maximum number (2^{32}) of possible memory addresses on a 32-bit computer. For a software service running as a 64-bit application, the Health Monitor sets the value to the total bytes of physical memory for the host.	Unsigned long integer
ProcessorCycles	During the last polling cycle, total number of CPU cycles that were used by all of the software modules that realize this software service.	Unsigned long integer
ProcessVirtualMemorySize	Total virtual memory, in bytes, that is used by all of the software modules that realize this software service.	Unsigned long integer
QueuePollingIntervalsToSkip	Number of queue monitoring polling cycles that the Health Monitor will skip during the peak load periods of this software service.	Unsigned integer Default: 1 (no skipping)

Table 18 Attributes for InChargeService (page 5 of 6)

Attribute	Description	Allowed values
RecomputeTime	Time, in seconds, of the last recompute time for this software service.	Unsigned integer
RecomputeTimeThreshold	Threshold, in seconds, that is used to determine whether the last recompute time for this software service took too long.	Unsigned integer Default: 200
ReconfigureTime	Time, in seconds, of the last reconfigure time for this software service.	Unsigned long integer
ReconfigureTimeThreshold	Threshold, in seconds, that is used to determine whether the last reconfigure time for this software service took too long.	Unsigned integer Default: 200
ResidentSetSize	Total physical memory, in bytes, that is used by all of the software modules that realize this software service.	Unsigned long integer
TotalFileHandles	Total number of file handles that are used by this software service.	Unsigned long integer
TotalHeap	For a software service running as a 32-application, total heap size, in kilobytes, that is imposed on the software service by the OS. For a software service running as a 64-application, total heap size, in kilobytes, that is imposed on the software service by the Health Monitor. The Health Monitor sets the value to the total kilobytes of physical memory for the host. TotalHeap is not supported for Windows.	Unsigned long integer
TotalHeapSize	For a software service running as a 32-bit application, total heap size, in kilobytes, that is imposed on the software service by the OS. For a 64-bit application, see TotalHeap. TotalHeapSize is not supported for Windows. **Note:** For a 32-bit application, TotalHeapSize and TotalHeap have the same value.	Unsigned long integer
TotalPhysicalMemory	Total bytes of physical memory for the host of this software service.	Unsigned long integer

Table 18 Attributes for InChargeService (page 6 of 6)

Attribute	Description	Allowed values
TotalProcessorCycles	During the last polling period, total number of CPU cycles that were used by the host of this software service. The number is the sum of the CPU cycles for each of the host's CPU processors.	Unsigned long integer
TotalProcessors	Total number of CPU processors for the host of this software service.	Unsigned integer
uptime	The length of time that the remote software service to which this software service is connected has been running.	Unsigned integer
VirtualMemoryUtilization	Percentage of virtual-memory usage for all of the software modules that realize this software service. Derived by dividing ProcessVirtualMemorySize by ProcessMaxVirtualMemorySize.	Floating point number
VirtualMemoryUtilization-Threshold	Threshold that is used to determine whether all of the software modules that realize this software service, when taken together, are consuming excessive virtual-memory resources. The threshold is for VirtualMemoryUtilization.	Floating point number Default: 70

Root-cause problems for InChargeService

Table 19 on page 88 lists the root-cause problems that are diagnosed for InChargeService.

Table 19 Root-cause problems for InChargeService

Problem	Description
Down	This InChargeService is down.
Malfunction	This InChargeService is not functioning correctly.

Events for InChargeService

Table 20 on page 89 lists the events that are computed for InChargeService.

Table 20 Events for InChargeService

Event	Description
ApproachingMaxFileHandles-Limit	The number of file handles that are used by this software service has exceeded the ProcessFileHandlesThreshold.
CorrelationIsSlow	The correlation time for this software service has exceeded the CorrelationTimeThreshold.
FullDiscoveryTimeExceeded	The full discovery time for this software service has exceeded the FullDiscoveryTimeThreshold.
HighCPUProcessUtilization	The CPU usage for this software service has exceeded the CpuUtilizationThreshold.
HighPhysicalMemory-Utilization	The physical-memory usage for this software service has exceeded the PhysicalMemoryUtilizationThreshold.
HighVirtualMemoryUtilization	The virtual-memory usage for this software service has exceeded the VirtualMemoryUtilizationThreshold.
PendingDiscoveryTime-Exceeded	The pending discovery time for this software service has exceeded the PendingDiscoveryTimeThreshold.
PrimaryMissingOrDown	The primary software service that is configured for this standby software service could not be discovered, or was discovered but is now down. See "Specifying a list of standby Managers for discovery" on page 39 for details.
ProcessDown	This software service is down.
RecomputeIsSlow	The recompute-codebook time for this software service has exceeded the RecomputeTimeThreshold.
ReconfigureIsSlow	The reconfigure time for this software service has exceeded the ReconfigureTimeThreshold.
RunningOutOfMemory [1]	The heap headroom for this software service has dropped below the HeapSizeThreshold.
StdbyMissingOrDown	The standby software service that is configured for this primary software service could not be discovered, or was discovered but is now down. See "Specifying a list of standby Managers for discovery" on page 39 for details.

[1] Applicable only to an InChargeService object that is hosted by a Solaris- or Linux-based host.

InChargeSession

An InChargeSession is a session connection between two InChargeService objects. Each complete InChargeSession connects two InChargeServices.

The Health Monitor creates an InChargeSession object for each Manager-to-Manager session connection in the managed environment, including the connection between itself and each of the Managers. For a Manager that acts as both a server and a client to another Manager, the Health Monitor creates two InChargeSession objects between the two Managers.

Attributes for InChargeSession

Table 21 on page 90 lists some key attributes for InChargeSession.

Table 21 Attributes for InChargeSession

Attribute	Description	Allowed values
DisconnectedFromRemote-ICServer	True when the session connectivity with the remote Manager no longer exists. Possibly one of the Managers is no longer responding or is down.	Boolean: true or false
IsDisconnected	True when the session connectivity is no longer established between the two Managers. Possibly one of the Managers is no longer responding or is down.	Boolean: true or false

Events for InChargeSession

The following event is computed for InChargeSession:

Disconnected: This InChargeSession is not established.

InChargeSoftwareRequest

An InChargeSoftwareRequest is a service connection between two InChargeService objects. Each complete InChargeSoftwareRequest connects two InChargeServices.

The Health Monitor creates an InChargeSoftwareRequest object for each Manager-to-Manager service connection in the managed environment. The client-side Manager initiates the software request, and the server-side Manager executes the software request.

Attributes for InChargeSoftwareRequest

Table 22 on page 91 lists a key attribute for InChargeSoftwareRequest.

Table 22 Attributes for InChargeSoftwareRequest

Attribute	Description	Allowed values
CurrentQueueSizeThreshold	Threshold for the current queue size above which a failed or degraded situation is deemed to exist between the client-side Manager and the server-side Manager.	Unsigned integer Default: 100

IP

An IP is a type of service access point that describes the IP layer characteristics of a network-attached interface. An IP endpoint is designated by a unique IPv4 address.

Attributes for IP

Table 23 on page 91 lists some key attributes for IP.

Table 23 Attributes for IP

Attribute	Description	Allowed values
Address	IP address of this IP endpoint; for example: 1.1.1.1	String
Netmask	Network mask for this IP endpoint; for example: 255.255.255.252	String

SHM_UM_Agent

An SHM_UM_Agent is an agent on a discovered Manager. The “UM” stands for utilization metrics.

An instance of SHM_UM_Agent, named SHM-UM-Agent and commonly referred to as the “SHM agent,” is created and started on a Manager by the Health Monitor during the discovery of the Manager. By default, the SHM-UM-Agent object is not viewable through a Global Console that is attached to the discovered Manager.

When requested to do so by the Health Monitor, the SHM agent on the discovered Manager establishes a direct connection to the operating system of the Manager’s host, collects host and process performance metrics for the Manager, and returns the metrics to the Health Monitor. By default, the Health Monitor sends a metrics request to the SHM agent every four minutes.

An SHM agent object has no relationships with any other discovered objects. It has three common-type, non-distinguishing attributes.

APPENDIX A
Terminology

This chapter defines EMC Smarts and computer-related terms and concepts. It consists of the following sections:

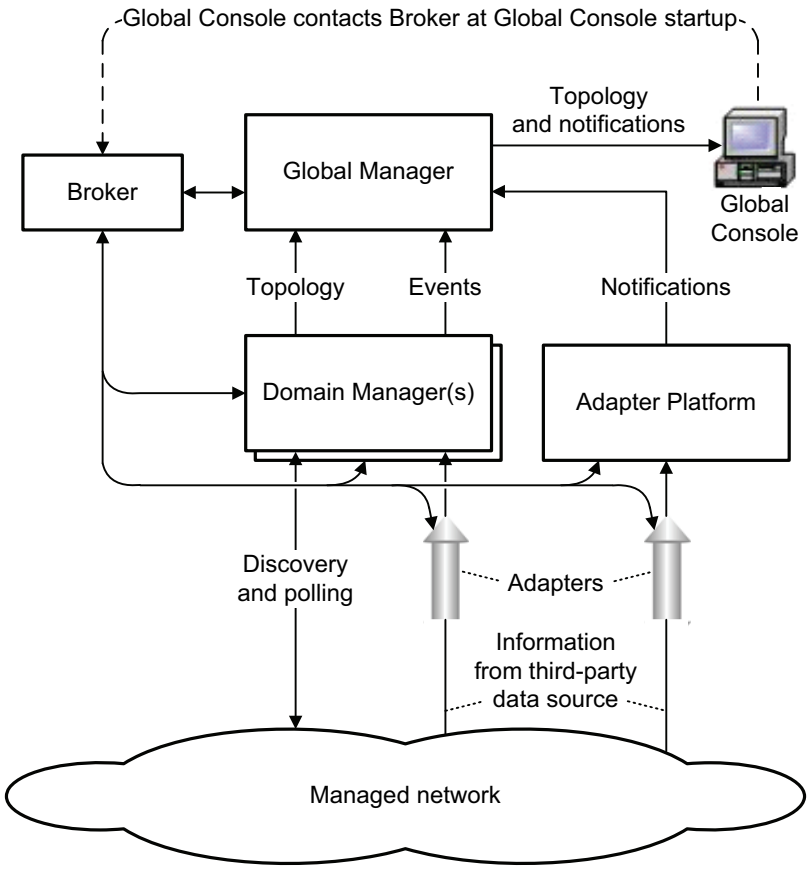
- Overview 94
- EMC Smarts terms and concepts 97
- Computer-related terms and concepts 109

Overview

The terms and concepts that are presented in this appendix should prove helpful in understanding how the applications in an EMC Smarts deployment work together to determine and display the health of the managed network.

The terms and concepts should also prove helpful in understanding how the Health Monitor determines the health of the applications and host computers in an EMC Smarts deployment.

Begin by examining Figure 12 on page 94 through Figure 14 on page 96.



Note: A Global Manager supports one or more Domain Managers.

Figure 12 Basic components and their connections in an EMC Smarts deployment

In client/server terminology, the Global Console takes on a client role, and the Global Manager, Domain Manager, Adapter Platform, Adapter, and Broker take on a server role. The EMC Smarts architecture supports server-to-server communications: The Global Manager operates as a client when importing topology/event/notification information from a Domain Manager or the Adapter Platform.

In addition, a Domain Manager operates as a client when importing topology and topology status update information from another Domain Manager, or from an Adapter.

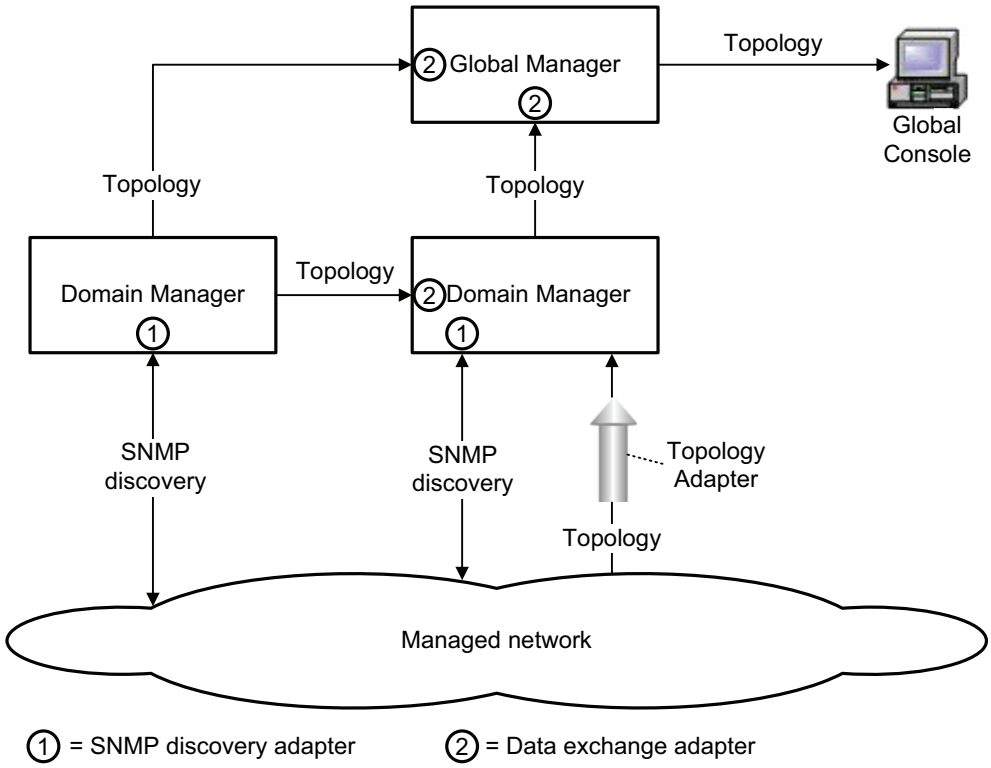


Figure 13 EMC Smarts topology discovery and import

Topology discovery, or just discovery, is the process of creating within the repository of a Domain Manager an EMC Smarts object representation of the managed topology. A Domain Manager repository is an in-memory, object-oriented database that is based on EMC Smarts object classes.

Topology information is collected by topology *adapters*, which run as threads within a Domain Manager, and by topology *Adapters*, which run as external applications. The SNMP discovery adapter is an example of the former, and a topology Adapter *that delivers a set of systems that were already discovered by a third-party manager* is an example of the later.

When an SNMP discovery adapter discovers a system, say a router, it instantiates the router object, retrieves a wealth of information from the router (router type, router vendor, what cards the router contains, what other systems connect to the router), and stores the data in the Domain Manager repository as EMC Smarts objects.

The Global Manager and Domain Manager use data exchange adapters to import topology from other managers. Topology objects in the source repository are replicated in the destination repository.

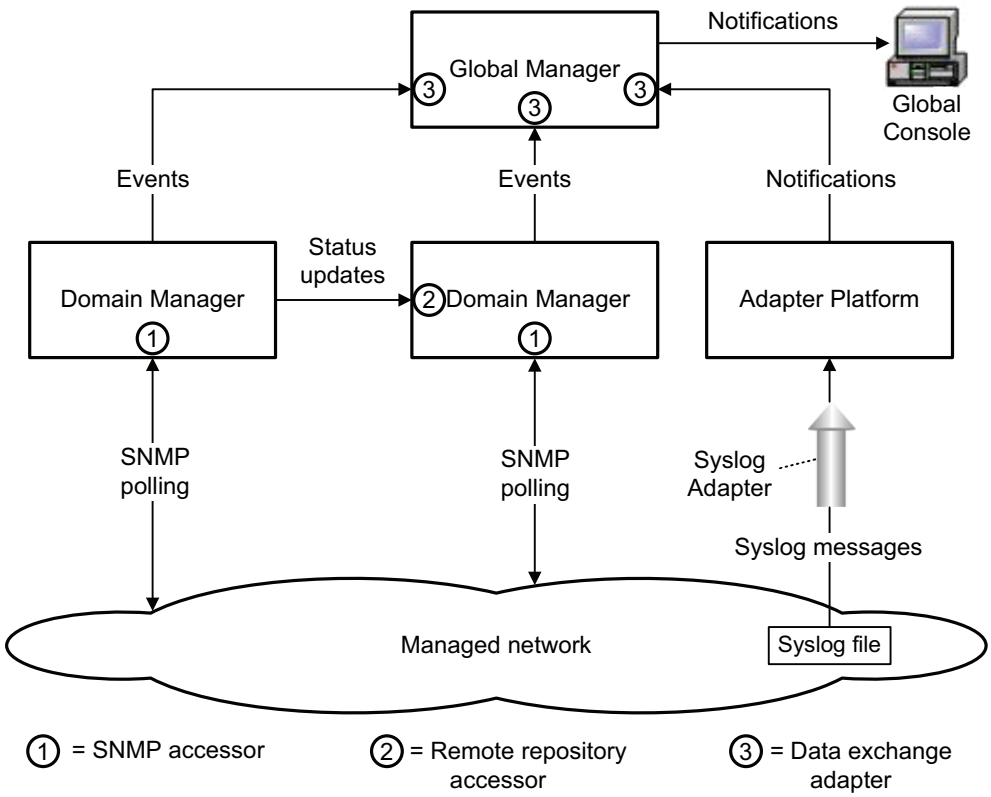


Figure 14 EMC Smarts monitoring and events and notifications

In the EMC Smarts environment, an event is a stateful condition that is applied to a topology class in a Domain Manager. An event for a discovered topology object, at any given time, is either notified (active) or clear (inactive). Events are generated by the Domain Manager.

The Global Manager becomes aware of a generated event by subscribing to the event. The Global Manager will receive a NOTIFY message when the event becomes active, and a CLEAR message when the event clears.

Events are associated with topology objects, but are not objects themselves. Notifications, on the other hand, are objects that represent events. The Global Manager maps the data in an imported event to the attributes of a notification object.

Topology status information is collected by accessors, which run as threads within a Domain Manager, and by event Adapters, which run as external applications. The SNMP accessor is an example of an accessor, and the Syslog Adapter is an example of an event Adapter.

Domain Managers use remote repository accessors to synchronize their topology status with other Domain Managers.

The Global Manager uses data exchange adapters to import events and notifications from other managers.

EMC Smarts terms and concepts

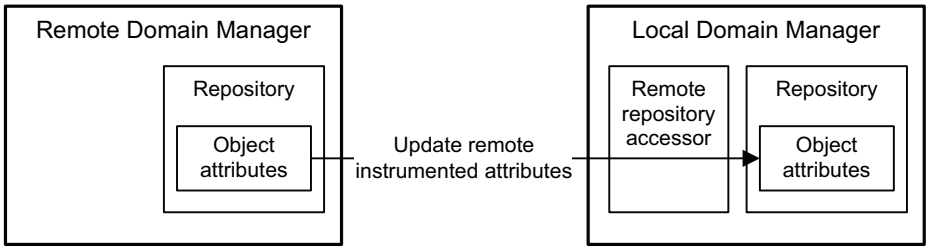
The EMC Smarts terms and concepts in this section are listed in alphabetical order.

- Accessor

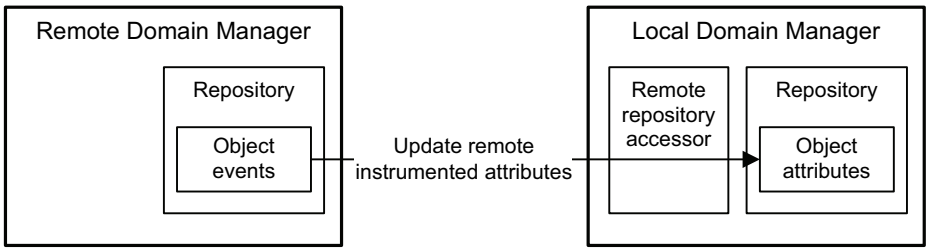
  Part of an EMC Smarts application and responsible for providing periodic updates to the instrumented attributes of topology objects. Several types of accessors are available, including ICMP accessor, SNMP accessor, remote repository accessor, and integration accessor.

  An accessor uses a mode of information exchange that matches the name of the accessor. One noticeable exception is the integration accessor, which is a wrapper for user-provided Java/Perl instrumentation code.

  A remote repository accessor is a subscription-based mechanism that synchronizes specific attributes or events between Domain Managers. Its operation is shown in the following figure.



A. Synchronizing topology object attribute status



B. Synchronizing topology object event status

  A remote repository accessor is established by implementing the *proxy* keyword in a local DXA configuration file. (See DXA in this list.) The proxy keyword is implemented by using the set_instrumentation() operation in the MR_RemoteReposInterface class on the local Domain Manager.

  For any class in the DXA configuration file that is preceded with the proxy keyword, the set_instrumentation() operation attaches the class to the remote repository accessor that is pointing to the remote Domain Manager. The set_instrumentation() operation maps the remote instrumented attributes of the class to attributes or events of the same class in the remote Domain Manager. The model for the class on the local Domain Manager defines whether the mapping is to remote attributes or to remote events.

When the mapping is complete, all instances of the remote class for a proxy-specified class will be instrumented locally. That is, the accessor will continuously monitor the instrumented attributes or events of the class instances on the remote Domain Manager, and reflect changes in the local objects as changes occur in the remote objects.

- Adapter

  Has two meanings:

  1. A process that is started with the EMC Smarts **sm_server**, **sm_adapter**, **sm_ehealth**, **sm_sdi**, or **sm_trapd** command

     A process that enables an EMC Smarts application to interface with third-party managers. Adapters are included in several EMC Smarts products.

     Only Adapters that are started with the **sm_server** command have repositories. For example, the Adapter for Alcatel-Lucent 5620 SAM EMS is started with the **sm_server** command and therefore has a repository.

     NOTICE

     Any Adapter that is started with the **sm_server** command has all of the basic capabilities of a Domain Manager.

     With the following two exceptions, all Adapters register with the Broker:

     – Any Adapter that is started with an **sm_server** command that includes the --noregister option will not register with the Broker.

     – Any Adapter that is started with an **sm_adapter** command that does *not* include the --name option will not register with the Broker.

     Adapters are organized into three groups:

     – Topology Adapters, which gather and normalize topology or provisioning information from third-party managers for input to EMC Smarts applications.

     – Event Adapters, which gather and normalize SNMP trap, Syslog message, or alarm information from third-party managers for input to EMC Smarts applications.

     – Notification Adapters, which notify third-party managers about EMC Smarts analysis results.

     Many Adapters can be configured to gather both topology and event information.

  2. A thread that runs within an EMC Smarts application

     A thread that enables an EMC Smarts application to interface with the managed devices, themselves, or with other EMC Smarts applications. Examples of internally running adapters are the SNMP topology adapter and the data exchange adapter.

  In this document, the term “Adapter” is used to mean an EMC Smarts Adapter product, and the term “adapter” is used to mean an internally running adapter.

- Adapter Platform

  An EMC Smarts application that receives events of interest from the EMC Smarts Adapter sources and prepares the events for input to the Global Manager. The Adapter Platform is included in the Service Assurance Manager.

  The Adapter Platform imports and normalizes topology from Domain Managers (specifically, IP Availability Managers), and receives and normalizes events of interest from Adapters. Once normalized, the information is transferred as EMC Smarts notifications to the Global Manager.

- ASL

  Adapter Scripting Language. An EMC Smarts proprietary scripting language.

- BASEDIR

  Used in documentation to represent the top-level directory structure of an EMC Smarts product installation. BASEDIR is not an environment variable.

  Here is a list of some BASEDIR representations:

  - ‹root_dir›/SAM

    for a Service Assurance Manager installation

  - ‹root_dir›/IP

    for an IP Manager installation

  - ‹root_dir›/MPLS

    for an MPLS Manager installation

  - ‹root_dir›/NPM

    for a Network Protocol Manager installation

  - ‹root_dir›/VoIP

    for a VoIP Manager installation

  - ‹root_dir›/MCAST

    for a Multicast Manager installation

  - ‹root_dir›/ESM

    for a Server Manager installation

  - ‹root_dir›/ASAM

    for an Adapter for Alcatel-Lucent 5620 SAM EMS installation

- Broker

  An EMC Smarts application that facilitates Global Console connections to Global Managers, Domain Managers, Adapter Platforms, and Adapters.

  Although the Broker is included in all EMC Smarts products, it is typically installed during the installation of the Service Assurance Manager. As in any EMC Smarts deployment, the Broker is started before any other application is started.

  The Broker maintains a registry of active Global Managers, Domain Managers, Adapter Platforms, and Adapters for the deployment. Each application registers its name, its process ID, the host name of the computer system on which the application is running, and the TCP port at which the application is listening for client connections, such as Global Console, dmctl, and other types of client connections.

  When a user starts a Global Console process, the process connects to the Broker, and launches a dialog box through which the user views and selects any application that is registered with the Broker. After the user selects an application, the Global Console connects to the application and disconnects from the Broker.

  Other interesting facts about the Broker:

  - Only one Broker process runs in an EMC Smarts deployment. It typically runs on the host that is running the Global Manager.
  - At startup, an EMC Smarts application contacts the Broker in the following manner:
    - Checks whether the Broker's location is specified as an option in the startup command line: --broker=‹host or IP›:‹port›. If specified, contacts and registers with the Broker at that location. If not specified, continues to the next step.
    - Checks whether the Broker's location is set in the SM_BROKER environment variable: SM_BROKER=‹host or IP›:‹port›. If set, contacts and registers with the Broker at that location. If not set, continues to the next step.
    - Checks whether the Broker's location is set in the SM_BROKER_DEFAULT environment variable: SM_BROKER_DEFAULT=‹host or IP›:‹port›. If set, contacts and registers with the Broker at that location. If not set, continues to the next step.
    - Checks whether the Broker is running on the local host and listening on TCP port 426: *localhost:426*. If yes, contacts the Broker at that location. If no, throws an error and does not start.
  - At five minute intervals, the Broker pings each registered application to determine its reachability. If the application responds, the Broker sets the application's state to Running; otherwise, the Broker sets the application's state to Dead.
  - Just before shutting down, an application notifies the Broker about the shutdown, and the Broker removes the application from the registry.

- Client

  A software program that is used to contact and obtain data from a server software program on another computer. Each client program is designed to work with one or more specific kinds of server programs, and each server requires a specific kind of client. A Global Console is a specific kind of client.

  Data exchange adapters and remote repository accessors are clients to remote EMC Smarts applications.

- Codebook

  Contains problem signatures that are unique to a specific managed infrastructure.

  All Domain Managers are build on the EMC Smarts codebook technology. The codebook leverages the fact that each problem has a unique signature in the symptoms that it causes. Problems are diagnosed by matching real-time symptoms (polled data or asynchronous events) to signatures, as in a medical diagnosis, where a patient's symptoms are compared to signatures of known diseases.

  A Domain Manager that manages a specific infrastructure contains problem signatures that are unique to that infrastructure.

- Data model

  Instantiated EMC Smarts behavior models that represent (1) the physical and logical objects in the managed environment and (2) their relationships to one another.

  As a Domain Manager discovers the logical and physical objects and their relationships, it instantiates corresponding EMC Smarts behavior models to build a data model that represents that information. It then uses the data model to compute problem signatures for the codebook.

- Device

  See System in this list.

- Domain Manager

  An EMC Smarts application that manages a particular type of information-technology domain: networks, systems, applications, or application services. A Domain Manager discovers, models, monitors, and analyzes the managed objects in its domain.

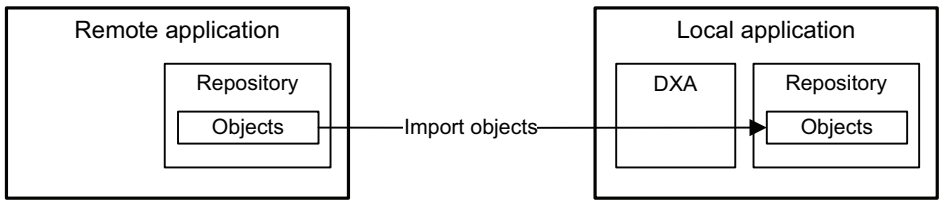
  A Domain Manager discovers the logical and physical objects in its domain and their relationships, builds a data model that represents this information in its repository by instantiating the corresponding EMC Smarts object classes, and forwards a streamlined copy of the discovered topology to the Global Manager. It monitors and analyzes the status of the discovered topology and exports the results of the analysis to the Global Manager in the form of events.

  Each Domain Manager is autonomous in the sense that it maintains its own data models, repository, and problem signatures; and monitors and analyzes the objects in its own domain.

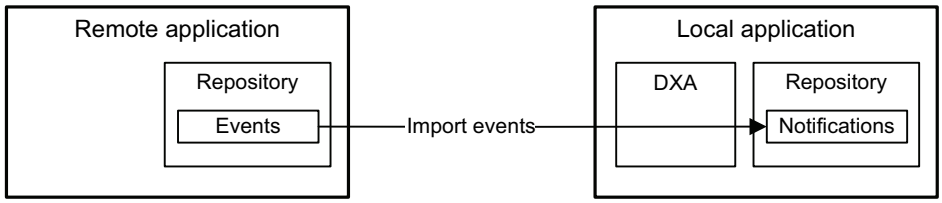
- DXA

  Data exchange adapter. The underlying data transfer mechanism between two EMC Smarts applications. The DXA establishes a connection to a remote application and imports objects or events in accordance to a local DXA configuration file.

  DXA operation is shown in the following figure.



A. Transferring objects



B. Transferring events

  Here are some examples of DXA operation:

  - For a connection between two Domain Managers, the DXA on the destination Domain Manager imports (subscribes to) topology objects. A DXA connection between two Domain Managers eliminates double discovery.
  - For a connection between a Domain Manager and the Global Manager, the DXA on the Global Manager imports (subscribes to) both topology objects and events. The Global Manager stores the events as ICS_Notification objects.
  - For a connection between the Adapter Platform and the Global Manager, the DXA on the Global Manager imports (subscribes to) ICS_Notification objects.

  The DXA provides a set of configuration commands that a user specifies as keywords in the DXA configuration file. The keywords are class, proxy, attr, rel, Rattr, Uattr, oper, peek, in, eq, struct, lookup, probe, conditionalprobe, server, hook, prehook, posthook, ignoreclass, and filter. Keywords that are valid only for the Global Manager are group, notification, sub, sev, topoUpdate, and heartbeat.

  Detailed descriptions of the keywords are available at the following EMC website:

  https://community.emc.com/docs/DOC-1213

- EMC Smarts application

  Synonymous with EMC Smarts product or EMC Smarts component. Also, see Manager in this list.

- EMC Smarts bootstrap files

  Files from which EMC Smarts products are started.

  The --config option (or just --c) in a startup command points to the subdirectory under BASEDIR/smarts/conf where the bootstrap file (bootstrap.conf by default) and the configuration files reside for the EMC Smarts product that is to be started.

  If more than one bootstrap file exists in the subdirectory, a user can use the --bootstrap option to point to a particular bootstrap file. For example, a user can start an instance of IP Availability Manager and IP Performance Manager that run as the same process by using the following options:

  --config icf --bootstrap=bootstrap-am-pm.conf

  Here are the bootstrap and configuration file locations for some EMC Smarts products:

  - For the Service Assurance Manager, the bootstrap and configuration files for the:
    - Global Manager reside in the BASEDIR/smarts/conf/ics directory.
    - Adapter Platform reside in the BASEDIR/smarts/conf/icoi directory.
    - Global Console reside in the BASEDIR/smarts/conf/console directory.
    - Health Monitor server reside in the BASEDIR/smarts/conf/serverhealth directory.
  - For the IP Manager, the bootstrap and configuration files reside in the BASEDIR/smarts/conf/icf directory.
  - For the MPLS Manager, the bootstrap and configuration files for the:
    - MPLS Topology Server reside in the BASEDIR/smarts/conf/mpls-t directory.
    - MPLS Monitoring Server reside in the BASEDIR/smarts/conf/mpls-m directory.
    - MPLS Analysis Server reside in the BASEDIR/smarts/conf/mpls-a directory.
    - VPN-Tagging Server reside in the BASEDIR/smarts/conf/vpn-tagging directory.
    - ISC Adapter reside in the BASEDIR/smarts/conf/isc directory.
  - For the Network Protocol Manager, the bootstrap and configuration files for the:
    - Network Protocol Manager for BGP reside in the BASEDIR/smarts/conf/bgp directory.
    - Network Protocol Manager for EIGRP reside in the BASEDIR/smarts/conf/eigrp directory.
    - Network Protocol Manager for IS-IS reside in the BASEDIR/smarts/conf/isis directory.
    - Network Protocol Manager for OSPF reside in the BASEDIR/smarts/conf/ospf directory.

- For the VoIP Manager, the bootstrap and configuration files reside in the BASEDIR/smarts/conf/voip directory.
- For the Multicast Manager, the bootstrap and configuration files reside in the BASEDIR/smarts/conf/mcast directory.
- For the Server Manager, the bootstrap and configuration files reside in the BASEDIR/smarts/conf/esm directory.
- For the Adapter for Alcatel-Lucent 5620 SAM EMS, the bootstrap and configuration files reside in the BASEDIR/smarts/conf/alcatel-sam directory.

◆ EMC Smarts deployment

A group of interconnected EMC Smarts applications that work with one another to determine and display the health of the networks, systems, applications, or application services in the managed environment.

Customers work with the EMC Smarts Professional Services group to develop a plan for their EMC Smarts deployment. The number of computer systems that are involved in a deployment, and on which computer systems the various applications run, depend on several factors, including the processing power of the computer systems, the size of the managed environment, and the degree of reliability required.

◆ EMC Smarts event

Represents either a root-cause problem that is diagnosed by a Domain Manager or a symptom that is computed by a Domain Manager. A Domain Manager correlates one or more symptoms to diagnose a root-cause problem. Events that are used as symptoms to diagnose a problem can also be perceived as impacts that are caused by the problem.

NOTICE

In this document, the term “problem” is used to mean a “root-cause problem” event, and the term “event” is used to mean a “symptom” or an “impact” event.

An event is uniquely identified by three properties:

- Model class name (unique within a repository)
- Instance name (unique within a repository)
- Event name (unique within a class)

Events are associated with the topology objects in a Domain Manager’s repository. Events are not objects.

The Global Manager becomes aware of an event that is generated by a Domain Manager by subscribing to the event. Once the Global Manager has subscribed to a given event, it will receive from the Domain Manager a NOTIFY message when the event becomes active, and a CLEAR message when the event clears.

- EMC Smarts licensing

  EMC Smarts products use the FLEXnet license host software that is packaged with the distribution to acquire licenses. The licenses are packaged as a file, referred to as the license file, that the EMC Smarts licensing coordinator creates and e-mails to customers.

- EMC Smarts management model

  A management model that is based on the industry-standard Common Information Model that is defined by the Distributed Management Task Force. The management model uses an object-oriented design to present a conceptual view of the managed environment. The object-oriented design promotes the use of inheritance, relationships, abstraction, and encapsulation to improve the quality and consistency of management data.

  The EMC Smarts management model is an information model that captures notions that are applicable to all areas of management. It is a set of object classes, common characteristics and features (properties), relationships (associations), and behavior (methods) that provide a basic vocabulary for describing and analyzing managed objects. The object classes represent network equipment, computer equipment, and application services.

  The EMC Smarts classes, or models, are written in MODEL, which is an object-oriented programming language that defines the types of entities that an EMC Smarts application will manage, the potential relationships between instances of these entities, the problems to be diagnosed, and the impacts (symptoms) that are caused by these problems.

- EMC Smarts notification

  An EMC Smarts event, a normalized third-party SNMP trap, or a normalized third-party Syslog message that is represented as an instance of the ICS_Notification class.

  The Global Manager subscribes to EMC Smarts events. For an imported event, the Global Manager produces the standard class-name/instance-name/event-name triplet that uniquely identifies the notification object to which the incoming data should be mapped; maps the values in the incoming data to the attributes of the notification object; and stores the object in its repository.

- EMC Smarts products

  A family of service assurance products that can analyze and pinpoint network, system, application, and application-service failures, to ensure high availability and peak performance. The key word here is pinpoint.

  Unlike traditional service assurance products that overwhelm the operator with a flood of seemingly unrelated alarms and events, an EMC Smarts application intelligently analyzes and correlates the alarm and event information to identify root-cause problems, calculates their impact on services and customers, and automates problem resolutions.

- EMC Smarts products

  Here is a list of some EMC Smarts products (formerly known as “product suites”):

  - Service Assurance Manager

    Includes the Global Manager, Adapter Platform, Global Console, Health Monitor server, and other applications.

  - IP Manager

    Includes the IP Availability Manager, IP Performance Manager, and IP Server Performance Manager. The term “IP Manager” is used to collectively represent all three of these Domain Managers.

  - MPLS Manager

    Includes the MPLS Manager, VPN-Tagging Server, and ISC Adapter. The MPLS Manager is split into three components: MPLS Topology Server, MPLS Monitoring Server, and MPLS Analysis Server.

  - Network Protocol Manager

    Includes the Network Protocol Manager for BGP, Network Protocol Manager for EIGRP, Network Protocol Manager for IS-IS, and Network Protocol Manager for OSPF.

  - VoIP Manager

    Includes the VoIP Availability Manager, VoIP Performance Manager, VoIP Performance Reporter, and other applications.

  - Multicast Manager
  - Server Manager
  - Adapter for Alcatel-Lucent 5620 SAM EMS

- Event

  An event is either on or off. See EMC Smarts event in this list for a description of EMC Smarts events.

- Global Console

  An EMC Smarts application that provides a graphical user interface to Global Managers, Domain Managers, Adapter Platforms, and Adapters. The Global Console is included in the Service Assurance Manager.

  The Global Console, unlike other EMC Smarts applications, does not register with the Broker. It does not contain a repository.

  The Global Console is one of three EMC Smarts GUI types: Global Console, Web Console, and Business Dashboard. All three are Java-based client applications that are delivered with the Service Assurance Manager. The Global Console is a standalone application, and the Web Console and Business Dashboard are Web-browser applications.

  Using the Global Console or Web Console, users can connect to the Global Manager or any other Manager in an EMC Smarts deployment, to configure and administer the Manager.

For a Global Console connection to the Global Manager, users can monitor the results of a Domain Manager's analysis. Users are able to navigate the network and to view cause-and-effect relationships between root-cause problems, their symptoms, their impacts on other parts of the managed environment, and their impacts on services and customers. Users can also initiate automated actions such as pages or e-mail in response to problems.

- Global Manager

  An EMC Smarts application that integrates and correlates management data from the connected Domain Managers and Adapter Platform, and provides back-end support for the Global Console. The Global Manager is included in the Service Assurance Manager.

  The Global Manager relates the management data to services and customers. Through correlation with the root-cause problems that are diagnosed by the Domain Managers, the Global Manager provides the information that is needed for the EMC Smarts administrator to manage the environment in the context of services and customers.

  The Global Manager provides this information to the Global Console, through which the administrator or an operator browses the topology and receives notifications about events that impact network and application performance.

- Host

  A workstation or server. A server host provides services that are available to other computers on the network.

- Manager

  In this document, refers to any of the following EMC Smarts applications:

  - Domain Manager
  - Global Manager
  - Adapter Platform
  - Adapter
  - Broker

- Modeled topology

  A Domain Manager uses EMC Smarts object class models to create within its repository instances of topology objects, their relationships, and their connections. The "modeled topology" mirrors the real topology in the managed environment.

  Also, see Data model in this list.

- Normalize

  To convert topology or event information from a third-party manager to a common form that is understood by EMC Smarts applications.

- Notification

  See EMC Smarts notification in this list.

- Object

  Has a dual meaning: To simultaneously represent both (1) an EMC Smarts object in the modeled topology and (2) a physical or logical entity in the real topology. An EMC Smarts object corresponds to a physical or logical entity in the real topology.

- Process

  A specific instance of a software program in execution. A program can exist in active memory in any number of individual instances. Each instance has a unique Process ID.

- Remote repository accessor

  See Accessor in this list.

- Repository

  An in-memory, object-oriented database that is maintained by an EMC Smarts application. A repository holds a data model representation of the topology in the managed network. It can also hold event data and notification objects.

- Server

  A software package, or a computer machine, that provides a specific kind of service to client software that is running locally or on another computer. Thus, the term “server” can refer to a particular piece of software, or to the computer machine on which the software is running.

  Moreover, a single server machine may have several different server software packages running on it, thereby providing many different services to clients on the network. For example, a single server machine might be running the Global Manager, the Health Monitor, and the Adapter Platform.

  Of course, a software package can act as a server for one application and as a client for another. For example the Global Manager acts as a server for the Global Console and as a client for a Domain Manager.

- Service Assurance Manager

  Better known as the Global Manager. See Global Manager in this list.

- Software service

  In this document, a process that is represented by an instance of the InChargeClient or InChargeService class. Also, see Process in this list.

- System

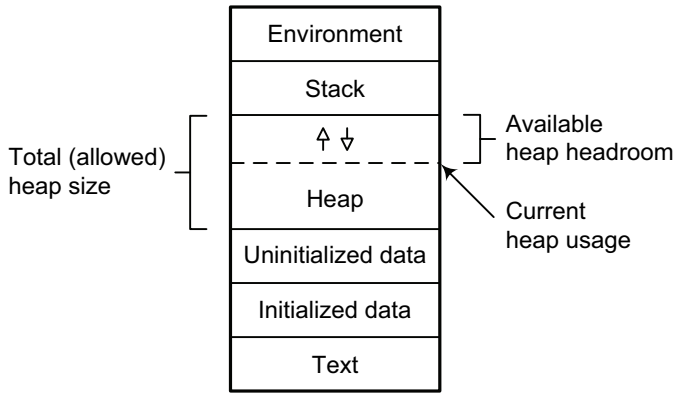
  A generic term that represents a computer-based network entity, such as a host, router, or switch. The term “device” has essentially the same meaning as system except that, in some cases, “device” also conveys the sense of specific model, such as a specific model of host, router, or switch.

Computer-related terms and concepts

The computer-related terms and concepts in this section are listed in alphabetical order.

- Address space

  The virtual memory that is available to an executing software program, or process. The address space for a program is created by the operating system when the program is executed. Typically, a program's address space is organized into six sections, as shown in the following figure.



  - Environment is used to store environment variables and command-line arguments.
  - Stack is used to store memory for function arguments, return values, and automatic variables.
  - Heap, or free store, is used for dynamic memory allocation.
  - Uninitialized data and Initialized data are used to store initialized and uninitialized static and global variables.
  - Text is used to store the actual program code.

  On a 32-bit computer, memory addresses can be a maximum of 32 bits long, which results in a maximum of 2^{32} (approximately 4,000,000,000) possible memory addresses. Thus, on a 32-bit computer, a program's address space is limited to 4 gigabytes.

  On a 64-bit computer, memory addresses can be a maximum of 64 bits long, which results in a maximum number of 2^{64} possible memory addresses, a number that far exceeds the amount of memory that is available to any modern-day computer. Thus, on a 64-bit computer, a program's address space is limited by the amount of memory that is available on the computer.

  Also, see Dynamic memory allocation, Heap, and Virtual memory in this list.

- API

  Application programming interface. A set of formalized software calls and routines that can be referenced by an application program to access underlying services.

- CPU

  Central processing unit, or just “processor” for short. The part of a computer that handles the majority of the calculations and data processing. The CPU fetches instructions from the computer’s memory, decodes them, and executes them to produce signals that control all aspects of the computer.

  Also, see Multi-core processor in this list.

- CPU cycle

  Fundamental sequence of steps that a CPU performs. A CPU cycle consists of fetching an instruction from memory, decoding it, and executing it.

- CPU time

  Amount of time that the CPU is actually executing instructions. The CPU time of an executing software program is generally much less than the total execution time of the program.

- CPU usage

  The percentage of a CPU’s total capacity that a computer or software program is using at any given time. If a computer is using 30 percent of a CPU’s total capacity, the computer’s CPU usage is 30 percent. If an executing software program is using 5 percent of a CPU’s total capacity, the program’s CPU usage is 5 percent.

- Dynamic memory allocation

  Allocation of a computer’s memory to operating-system processes and user processes by a dynamic memory allocator, such as malloc, dlmalloc, or ptmalloc. An allocator relies on the computer’s runtime library and operating-system calls to allocate and deallocate memory for a process.

  When a process requires a free memory block to execute a request, it calls the allocator to find the free memory. The allocator traverses the process’s heap, or free store, to find the smallest free block of memory that is bigger than the size of the request. It then returns a pointer to the found memory block, which the process uses to access the memory.

  When the memory is no longer needed, the process calls the allocator to free the memory. The allocator responds by passing the pointer to free.

  An allocator adds a deallocated memory block to a list, called a free list. For any future allocation request, the allocator attempts to fulfill the request by checking the free list first. If unsuccessful, the allocator allocates the memory from the main heap memory.

  An allocator will expand and contract the heap to fulfill allocation requests. However, if no free memory is available, or if the process has exceeded the amount of memory that it is allowed, the allocator will return a null pointer.

- File handle

  A value, typically a number, that the operating system temporarily assigns to a file that a process has asked the operating system to open. A special area of physical memory (RAM) is reserved for file handles, and the size of this area determines how many files can be open at any given time.

- File-handle usage

  The percentage of the maximum number of allowed file handlers that a process is using at any given time. If a process is using 40 percent of the maximum number of allowed file handlers, the process's file-handle usage is 40 percent.

  The maximum number of allowed file handles for a process depends on the operating system, amount of computer physical memory (RAM), kernel tunables, and user-configured file-handle limits. Unlike other operating systems, the Windows operating system does not set any limit on the maximum number of file handles that a process may use.

- HDD

  Hard disk drive. The mechanism that reads and writes data on a hard disk. Because the hard disk drive and hard disk are packaged as a unit, the term "hard disk drive" and "hard drive" are often used interchangeably.

- Heap

  A physical memory (RAM) area where a process gets its blocks of memory for allocation. A heap is a data structure, similar to binary trees, that maintains free and reserved areas of memory for a process.

  The heap has two significant attributes: base and length. Base is the beginning of the heap in virtual memory space, and can never change. Length is the size of the heap. The heap requires enough memory from the system to fill its entire length.

  Like any method of memory allocation, the heap will become fragmented: Blocks of free (unused) memory and blocks of reserved (used) memory will develop in the allocated memory space on the heap. The fragmentation can cause a process to run out of memory: Free memory is available on the heap, but no contiguous block of memory is large enough to hold all of the process's variables.

  Also, see Dynamic memory allocation in this list.

- Heap headroom

  Amount of memory space that is left for a process's heap to expand. Stated another way, the amount of memory space between the maximum size that is allowed for the heap and the current size of the heap.

- Multi-core processor

  A physical CPU package that contains two or more independent processors, called "execution cores" or just "cores." Manufacturers integrate the cores and their caches and cache controllers onto a single integrated circuit.

  A dual-core processor is a physical CPU package that contains two processors, a quad-core processor is a physical CPU package that contains four processors, and a hexa-core processor is a physical CPU package that contains six processors.

- OS

  Operating system. A computer software program on which other software programs, called applications, run.

- Persistent storage

  Refers to data that is written to disk and retrieved from disk when needed.

- Physical memory

  Hardware that a computer uses to load the operating system and to run software programs. Physical memory, also known as main memory, primary memory, or fixed memory, consists of one or more RAM chips. The amount of physical memory in a computer is determined by the amount of installed RAM.

- Physical-memory usage

  The percentage of total physical-memory capacity that a computer or software program is using at any given time. If a computer is using 25 percent of its total physical-memory capacity, the computer's physical-memory usage is 25 percent. If an executing software program is using 10 percent of the computer's total physical-memory capacity, the program's physical-memory usage is 10 percent.

- Process

  A specific instance of a software program in execution. A program can exist in active memory in any number of individual instances. Each instance has a unique Process ID.

- RAM

  Random access memory. A type of computer memory that can be accessed randomly; that is, any byte of memory can be accessed without touching the preceding bytes. Types of RAM include dynamic RAM, extended data output dynamic RAM, and static RAM.

- Virtual memory

  The use of space on a hard disk drive to emulate physical memory (RAM). Virtual memory enables software programs to run in a memory space that is greater than the memory space of the computer's physical memory.

  Virtual address space is the set of addresses that identify locations in virtual memory. A program is allocated a virtual address space, and uses the virtual addresses to store instructions and data. When the program is actually executed, the virtual addresses are converted into real memory addresses.

  The size of virtual memory is the maximum size that a computer can address by using pointers. On a 32-bit computer, a program can address up to 4 gigabytes of memory. On a 64-bit computer, a program can address up to more memory than any computer on the market today can possibly provide.

  Thus, on a 64-bit computer, the size of virtual memory is limited by the amount of *swap space* that is available on the computer. Swap space is the space on a hard disk drive that is used to store the overflow from main memory. In general, the size of the swap space should be about twice the amount of physical memory.

The swap space is divided into segments called pages, each of which is associated with a specific address in memory. (A page is typically 4 to 8 kilobytes in size.) When an address is referenced, the page is swapped into main memory. The page is returned to the disk when it is no longer needed.

- Virtual-memory usage

  The percentage of total virtual-memory capacity that a computer or software program is using at any given time. If a computer is using 35 percent of its total virtual-memory capacity, the computer's virtual-memory usage is 35 percent. If an executing software program is using 20 percent of the computer's total virtual-memory capacity, the program's virtual-memory usage is 20 percent.

APPENDIX B
Wildcard Patterns

This appendix shows you how to use a wildcard pattern in a text field to match a number of objects instead of specifying each object by name. It consists of the following sections:

- Types of wildcard patterns 116

Types of wildcard patterns

A wildcard pattern is a series of characters that are matched against incoming character strings. You can use these patterns when you define pattern matching criteria.

Matching is done strictly from left to right, one character or basic wildcard pattern at a time. Basic wildcard patterns are defined in Table 24 on page 116. Characters that are not part of match constructs match themselves. The pattern and the incoming string must match completely. For example, the pattern abcd does not match the input abcde or abc.

A compound wildcard pattern consists of one or more basic wildcard patterns separated by ampersand (&) or tilde (~) characters. A compound wildcard pattern is matched by attempting to match each of its component basic wildcard patterns against the entire input string. Compound wildcard patterns are listed in Table 25 on page 117.

If the first character of a compound wildcard pattern is an ampersand (&) or tilde (~) character, the compound is interpreted as if an asterisk (*) appeared at the beginning of the pattern. For example, the pattern ~*[0-9]* matches any string that does not contain any digits. A trailing instance of an ampersand character (&) can only match the empty string. A trailing instance of a tilde character (~) can be read as “except for the empty string.”

NOTICE

Spaces are interpreted as characters and are subject to matching even if they are adjacent to operators like “&”.

Table 24 Basic wildcard patterns (page 1 of 2)

Character	Description
?	Matches any single character. For example, server?.example.com matches server3.example.com and serverB.example.com, but not server10.example.com.
*	Matches an arbitrary string of characters. The string can be empty. For example, server*.example.com matches server-ny.example.com and server.example.com (an empty match).
[set]	Matches any single character that appears within [set]; or, if the first character of [set] is (^), any single character that is *not* in the set. A hyphen (-) within [set] indicates a range, so that [a-d] is equivalent to [abcd]. The character before the hyphen (-) must precede the character after it or the range will be empty. The character (^) in any position except the first, or a hyphen (-) at the first or last position, has no special meaning. For example, server[789-].example.com matches server7.example.com through server9.example.com, but not server6.example.com. It also matches server-.example.com. For example, server[^12].example.com does not match server1.example.com or server2.example.com, but will match server8.example.com.

Table 24 Basic wildcard patterns (page 2 of 2)

Character	Description
‹n1-n2›	Matches numbers in a given range. Both *n1* and *n2* must be strings of digits, which represent nonnegative integer values. The matching characters are a non-empty string of digits whose value, as a nonnegative integer, is greater than or equal to *n1* and less than or equal to *n2*. If either end of the range is omitted, no limitation is placed on the accepted number. For example, 98.49.‹1-100›.10 matches a range of IPv4 addresses from 98.49.1.10 through 98.49.100.10. Example of an omitted high end of the range: ‹50-› matches any string of digits with a value greater than or equal to 50. Example of an omitted low end of the range: ‹-150› matches any value between zero and 150. For a more subtle example: The pattern ‹1-10›* matches 1, 2, up through 10, with * matching no characters. Similarly, it matches strings like 9x, with * matching the trailing x. However, it does not match 11, because ‹1-10› always extracts the longest possible string of digits (11) and then matches only if the number it represents is in range.
|	Matches alternatives. For example, ”ab|bc|cd” without spaces matches exactly the three following strings: “ab”, “bc”, and “cd”. A | as the first or last character of a pattern accepts an empty string as a match. Example with spaces “ab | bc” matches the strings “ab” and “bc”.
\	Removes the special status, if any, of the following character. Backslash (\) has no special meaning within a set ([set]) or range (‹n1-n2›) construct.

Special characters for compound wildcard patterns are summarized in Table 25 on page 117.

Table 25 Compound wildcard patterns

Character	Description
&	“And Also” for a compound wildcard pattern. If a component basic wildcard pattern is preceded by & (or is the first basic wildcard pattern in the compound wildcard pattern), it *must* successfully match. Example: *NY*&*Router* matches all strings that contain NY and also contain Router. Example: ‹1-100›&*[02468] matches even numbers between 1 and 100 inclusive. The ‹1-100› component only passes numbers in the correct range and the *[02468] component only passes numbers that end in an even digit. Example: *A*|*B*&*C* matches strings that contain either an A or a B, and also contain a C.
~	“Except” for a compound wildcard pattern (opposite function of &).If a component basic wildcard pattern is preceded by ~, it *must not* match. Example: 10.20.30.*~10.20.30.50 matches all devices on network 10.20.30 except 10.20.30.50. Example: *Router*~*Cisco*&*10.20.30.*~10.20.30.‹10-20›* matches a Router, except a Cisco router, with an address on network 10.20.30, except not 10.20.30.10 through 10.20.30.20.

APPENDIX C
Topology Naming Conventions

This appendix describes the naming conventions that are used by the Health Monitor to name the discovered topology objects. It consists of the following sections:

- DisplayName and Name attributes 120
- Naming convention for topology objects 121

DisplayName and Name attributes

Each topology object has a unique name, which is defined by the DisplayName and Name attributes for the object, as shown in Figure 15 on page 120.

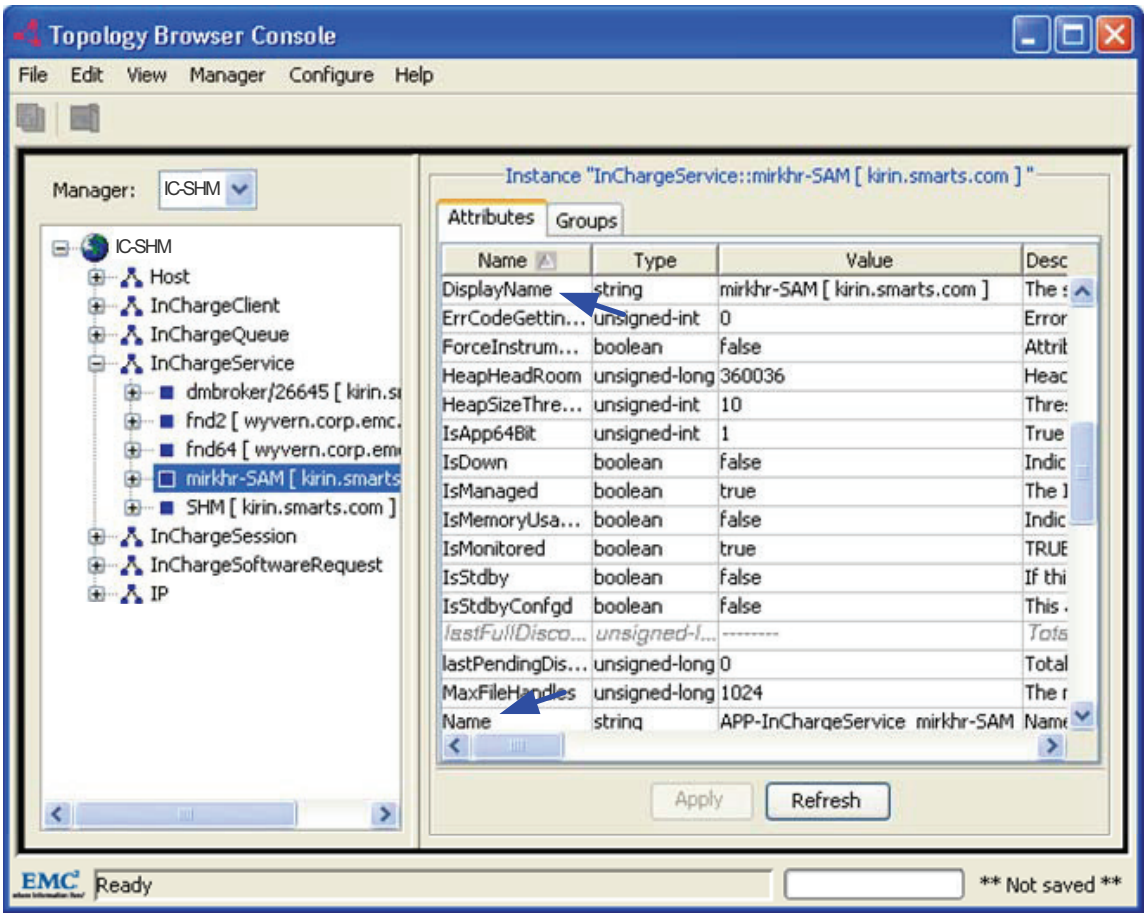


Figure 15 DisplayName and Name attributes for a topology object

The DisplayName attribute value is the name of the object that is displayed in the Global Console. The Name attribute value is the name of the object as known in the Domain Manager's repository.

The DisplayName and Name attributes of a topology object are viewable through a Topology Browser Console that is attached to the Global Manager or to the Health Monitor.

NOTICE

To launch a Topology Browser Console from the Global Console, select **File › New › Topology Browser Console**.

An object's DisplayName attribute value also appears in the Name attribute of a notification for that object. Notifications are imported by the Global Manager and displayed in the Notification Log Console.

Naming convention for topology objects

To name a discovered topology object, the Health Monitor uses the appropriate naming format in Table 26 on page 121 to assign the object a unique name.

Table 26 Naming formats for discovered topology objects (page 1 of 2)

Class	Naming format
Host	Name: ‹fully qualified domain name *or* IP address› Example: zach.emc.com DisplayName: ‹empty, which means that DisplayName defaults to Name›
InChargeClient [1]	Name: App-InChargeClient-‹client type›[‹client host device name›] [[clientID=]‹client identifier›/‹client PID›]‹serving software service name› Examples: App-InChargeClient-Console[USITSYLVESL1C][clientID=1/1]IC-SHM App-InChargeClient-Java[vernon][clientID=13471/13471]zach.emc.com:3953/dmbroker DisplayName: IC_CLIENT_‹client type› [‹client host device name›] [[clientID=]‹client identifier›/‹client PID›] -› ‹serving software service name› Examples: IC_CLIENT_Console [USITSYLVESL1C][clientID=1/1]-›IC-SHM IC_CLIENT_Java [vernon][clientID=13471/13471]-›zach.emc.com:3953/dmbroker **Note:** "Console" represents a Global Console, a Web Console, or a Business Dashboard, all of which are Java-based client applications that are delivered with the Service Assurance Manager. "Java" represents a non-console, Java-based client application.
InChargeQueue	Name: IC_Queue-‹queue name›-‹servicing software service name›/‹host device name› Example: IC_Queue-SubscriberFE-INCHARGE-AM_Event-Driver-EventTimer- INCHARGE-SA/zach.emc.com DisplayName: ‹empty, which means that DisplayName defaults to Name›
InChargeService [1]	Name: APP-InChargeService_‹software service name› Example: APP-InChargeService_INCHARGE-SA DisplayName: ‹software service name› [‹host device name›] Example: INCHARGE-SA [zach.emc.com]

[1] This object type is imported by the Global Manager.

Table 26 Naming formats for discovered topology objects (page 2 of 2)

Class	Naming format
InChargeSession [1]	Name: SESSION-APP-InChargeService_‹name of software service that initiates session› -> APP-InChargeService_‹name of software service that terminates session› Example: SESSION-APP-InChargeService_INCHARGE-SA->APP-InChargeService_IC-SHM DisplayName: SESSION [‹name of software service that initiates session› [‹host device name›] -> ‹name of software service that terminates session› [‹host device name›]] Example: SESSION [INCHARGE-SA [zach.emc.com] -> IC-SHM [zach.emc.com]]
InChargeSoftwareRequest [1]	Name: IC_SW_REQ-APP-InChargeService_‹name of software service that initiates request› -> APP-InChargeService_‹name of software service that executes request› Example: IC_SW_REQ-APP-InChargeService_INCHARGE-SA->APP-InChargeService_IC-SHM DisplayName: IC_SW_REQ [‹name of software service that initiates request› [‹host device name›] -> ‹name of software service that executes request› [‹host device name›]] Example: IC_SW_REQ [INCHARGE-SA [zach.emc.com] -> IC-SHM [zach.emc.com]]
IP	Name: IP-‹IP address› Example: IP-10.1.8.51 Display Name: ‹IP address› Example: 10.1.8.51

[1] This object type is imported by the Global Manager.

INDEX

A
Adding or removing matching criteria 71
Analysis 74
Attributes
    DisplayName 120
    Name 120
Automatic dynamic thresholding
    Description 23

B
Broker 100

C
Changing matching criteria 71
Changing priority of a group 70
Changing setting parameters 69
Codebook 30, 101
Copying a group 72
Creating a group 72

D
Data exchange adapter
    See DXA
Default polling interval
    Changing 45
Device
    Definition of 108
Disconnected
    InChargeSession 90
Discovery
    Discover All command 29
DisplayName attribute 120
dmctl utility
    dmctl get command 36
    dmctl put command 36
Domain Manager Administration Console
    Discover All command 29
DXA 102
Dynamic thresholding
    Description 23

E
Event 31
    Summary table 77

G
Global Console 31, 50, 59
    Type of
        Global Manager Administration Console 50
        Notification Log Console 31
        Polling and Thresholds Console 59
        Topology Browser Console 120
Global Manager 31
    Configuration tasks 28
Global Manager Administration Console 50
    Opening 50
Group
    Changing priority 70
    Copying 72
    Creating 72
    Properties 68
Groups
    Polling 62
        Queue_Groups 62
    Threshold 64
        Host_Groups 64
        SoftwareService_Groups 64, 65

H
Health Monitor
    Analysis 74
    Configuration tasks 28
    Notifications 31
    Repository file 37, 38
Host 79
    AllServicesDown 76, 80
    ApproachingHostResourceLimit 76, 80
Host_Groups threshold group 64

I
InChargeClient 80
InChargeQueue 81
InChargeService 83
    Down 76, 88
    Malfunction 76, 88
InChargeSession 90
    Disconnected 90
InChargeSoftwareRequest 91
IP 91

M
Matching
    Pattern 116
Matching criteria
    Adding or removing 71
    Changing 71

N
Name attribute 120
Naming convention 121
Notification
    Naming convention 120
    Problem 75
    Unique name 120

Notification Log Console 31
Notifications 31
    Display of 31
    Types 31
    Event
        see Event
    Root cause
        see Root cause

O

Object
    Definition of 108
    Naming convention 121
    Unique name 120
Operator
    Wildcard 116

P

Parameter
    Override 36, 38
Pattern 116
Pattern matching 116
Polling 44
    Disabling 44
Polling and Thresholds Console 59
    Opening 59
Priority
    Changing 70

Q

Queue_Groups polling group 62
Queues Polling Setting 63

R

Removing or adding matching criteria 71
Repository
    Location 37, 38
Restoring default values of a setting 70
Root cause 31
    Host
        AllServicesDown 76, 80
        ApproachingHostResourceLimit 76, 80
    InChargeService
        Down 76, 88
        Malfunction 76, 88
    Summary table 76

S

Security 48
    clientConnect.conf 48
    serverConnect.conf 48
Service Assurance Manager
    See Global Manager
Setting
    Changing parameters 69
    Restoring default values 70
SHM agent support 19
SHM_UM_Agent 92
SoftwareService Health Metrics Thresholds setting 66
SoftwareService_Groups threshold group 64, 65
Support
    SHM agent 19
System
    Definition of 108
System Health Metrics Thresholds setting 65

T

Topology
    File location 37, 38
Topology Browser 120
Topology object
    Naming convention 121
    Unique name 120

W

Wildcard 116
    Chart of operators 116