



Netra™ High Availability Suite 3.0 1/08 Foundation Services NMA Programming Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 819-5239-13
March 2008, Revision A

Submit comments about this document at: <http://www.sun.com/hwdocs/feedback>

Copyright 2008 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents>, and one or more additional patents or pending patent applications in the U.S. and in other countries.

This document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of the product or of this document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Java, docs.sun.com, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2008 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats-Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Java, docs.sun.com, et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciées de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Adobe PostScript

Contents

Preface xiii

- 1. Introduction to the Node Management Agent 1**
 - Accessing the NMA 2
 - Master and Node Views 3
 - MBean Instances on the Master Node 5
 - MBean Instances on Peer Nodes 5
 - Floating External Address 6
- 2. Configuration Files, Dependencies, and Requirements 7**
 - Configuration Files 7
 - Dependencies 8
 - Software Requirements 8
- 3. Developing an External Java Manager 9**
 - Configuring an External Java Manager Using HTTP 9
 - Connecting to the NMA 10
 - Using the Floating Address 10
 - Using a Physical Node Address 10
 - Using Proxy MBeans 11

4. Developing a Remote SNMP Manager	13
Configuring an SNMP Agent	13
IP-Based Access Control Lists	15
Format of the <code>acl</code> Group	16
Format of the Trap Group	17
SNMPv3 User-Based Access Control	18
Configuring SNMPv3 Security	19
Engine ID	19
Context Name	19
Managing Users in Security Files	20
SNMP Manager Configuration Examples	21
SNMPv2 Configuration Example	22
SNMPv2 and SNMPv3 Hybrid Configuration Example	23
SNMPv3 Configuration Example	25
5. Manipulating the Cluster Using the NMA	29
Using the NMA to Initiate a Switchover	29
Checking Whether the Netra HA Suite Software Is Ready for Switchover	30
Initiating a Switchover	30
Example of Switchover Using an HTTP Connector Client	30
Getting the CMM Status of All Cluster Nodes	34
Manipulating Daemon Monitor Retry Settings	34
6. Carrier Grade Transport Protocol Statistics	35
Introducing CGTP Statistics	35
CGTP Master Statistics	36
<code>CgtpMasterMBean</code>	36
Getting All Local CGTP Addresses for Which Statistics Are Available	36
Getting All Remote CGTP Addresses for Which Statistics Are Available	36

CGTP Node Statistics	36
CgtpMBean	36
Getting All Local CGTP Addresses for Which Statistics Are Available	37
Getting All Remote CGTP Addresses for Which Statistics Are Available	37
CgtpEmitterStatisticsMBean	37
Getting the Number of Packets Sent Through Each Subinterface	37
CgtpFilterMBean	37
Getting the Number of Packets Not Received in Duplicate	38
Getting the Amount of Memory Currently Used by the Filter Module	38
Getting the Number of Packets Successfully Received	38
Getting the Number of Hash Table Collisions	38
Getting the Number of Direct Hash Table Entries	38
Getting the Number of Hash Table Entries	38
Getting the Maximum Amount of Memory Used by the Filter Module	39
Getting the Number of Packets Waiting for Duplicate Reception	39
Getting the Number of Packet Hash Collisions	39
Getting the Maximum Number of Packet Hash Collisions	39
Getting the Maximum Number of Packet Hash Collisions in One Row	39
CgtpReceiverStatisticsMBean	39
Getting the Number of Packets Not Successfully Filtered	40
Getting the Number of Packets Successfully Filtered	40
Getting the Number of Packets Received Through Each Subinterface	40
CgtpReliableLinkStatisticsMBean	40
Getting the Remote Subinterface Addresses	40
Getting Local End Reliable Link CGTP Addresses	41
Getting Remote End Reliable Link CGTP Addresses	41
Getting Local Subinterface Addresses	41

7. Daemon Monitor Statistics	43
Example of Accessing Statistics Using an HTTP Client	43
Introducing Daemon Monitor Statistics	47
Daemon Monitor Master Statistics	48
PmdMasterStatisticsMBean	48
Getting All Nametags	48
Daemon Monitor Node Statistics	48
PmdStatisticsMBean	48
Getting All Nametags	48
PmdNameTagStatisticsMBean	49
Getting the Daemon Monitor Nametag	49
Getting the PIDs Associated With a Nametag	49
Getting the Daemon Monitor Maximum Retries	49
Getting the Number of Retries for a Nametag	49
8. Reliable NFS Statistics	51
Introducing Reliable NFS Statistics	51
Reliable NFS Using SNDR	51
Reliable NFS Master Statistics for SNDR	52
RnfsMasterReplicatedSliceMBean	52
Reliable NFS Peer Node Statistics for SNDR	52
RnfsStatisticsMBean	52
RnfsMasterStatisticsMBean	53
RnfsReplicatedSliceMBean	53
Reliable NFS Using Shared Disk Configuration	55
Reliable NFS Master Statistics for Shared Disk	55
SDMasterStatisticsMBean	55
9. Cluster Membership Manager Statistics	59

Introducing CMM Statistics	59
CMM Master Statistics	60
CmmMasterStatisticsMBean	60
Getting the Average Time Between Node Starts	60
Getting the Number of CMM Clients	60
Getting the CMM Lifetime	60
Getting the Number of Node Elections	60
Getting the Longest Interval Between Node Starts	61
Getting the Shortest Interval Between Node Starts	61
Getting the Number of Nodes in the Cluster	61
Getting the Number of Outstanding CMM Requests	61
Getting the Switchover Count	61
CMM Node Statistics	62
ClusterNodeMBean	62
Getting a Node's CGTP Address	62
Getting the Domain ID of the Cluster That a Node Is Eligible to Join	62
Getting the Time Since Node Was Last Rebooted	63
Getting the CMM Membership Role of a Node	63
Getting the Node ID	63
Getting the Node Name	63
Getting the Node Boot Image ID	64
Getting the CMM State Flags of a Node	64
CmmStatisticsMBean	64
Getting the Average Time Taken to Start CMM Services	64
Getting the Number of Master Elections Performed on a Node	64
Getting the Maximum Time Taken to Elect a Master Node	65
Getting the Minimum Time Taken to Elect a Master Node	65
Getting the Number of Nodes Present	65

Getting the Number of Switchovers Performed	65
Getting the Number of Currently Connected CMM Clients	65
Getting the Number of Outstanding Requests	65
Getting the Lifetime of the CMM on a Node	66
10. Receiving Notifications	67
Registering to Receive Notifications	67
NhasCmmNotification	67
NhasPmdMaxRetriesNotification	68
NhasPmdAttributeChangeNotification	68
NhasPmdNewNameTagNotification	69
NhasPmdRemoveNameTagNotification	69
Registering to Receive SNMP Traps	69
A. MBean Naming Conventions	71
Nodes and Services	72
Cluster Membership Manager	72
Reliable NFS Using SNDR	72
Reliable NFS Using a Shared-Disk Configuration	72
Daemon Monitor	73
CGTP	73
Index	75

Figures

- [FIGURE 1-1](#) Remote Manager Communication 3
- [FIGURE 1-2](#) Cascading Information From Peer Nodes to the Master Node 4

Code Examples

CODE EXAMPLE 4-1	Typical <code>nma.acl</code> File	15
CODE EXAMPLE 4-2	Example <code>nma.uacl.template</code> File	18
CODE EXAMPLE 4-3	Using the <code>SnmpV3AppliMibRegistration</code> API	21
CODE EXAMPLE 4-4	Example Entries in <code>nma.properties</code> for SNMPv2	22
CODE EXAMPLE 4-5	Example Entries in <code>nma.acl</code> for SNMPv2	23
CODE EXAMPLE 4-6	Example Entry in <code>nma.targets.txt</code> for SNMPv2	23
CODE EXAMPLE 4-7	Example Entry in <code>nma.params.txt</code> for SNMPv2	23
CODE EXAMPLE 4-8	Example Entry in <code>nma.notifs.txt</code> for SNMPv2	23
CODE EXAMPLE 4-9	Example Entries in <code>nma.properties</code> for Hybrid Configuration	24
CODE EXAMPLE 4-10	Example Entries in <code>nma.security</code> for Hybrid Configuration	24
CODE EXAMPLE 4-11	Example Entries in <code>nma.acl</code> for Hybrid Configuration	24
CODE EXAMPLE 4-12	Example Entries in <code>nma.uacl</code> for Hybrid Configuration	24
CODE EXAMPLE 4-13	Example Entries in <code>nma.targets.txt</code> for Hybrid Configuration	25
CODE EXAMPLE 4-14	Example Entries in <code>nma.params.txt</code> for Hybrid Configuration	25
CODE EXAMPLE 4-15	Example Entry in <code>nma.notifs.txt</code> for Hybrid Configuration	25
CODE EXAMPLE 4-16	Example Entries in <code>nma.security</code> for Hybrid Configuration	25
CODE EXAMPLE 4-17	Example Entries in <code>nma.properties</code> for SNMPv3 Configuration	26
CODE EXAMPLE 4-18	Example Entries in <code>nma.security</code> for SNMPv3 Configuration	26
CODE EXAMPLE 4-19	Example Entries in <code>nma.acl</code> for SNMPv3 Configuration	26
CODE EXAMPLE 4-20	Example Entries in <code>nma.uacl</code> for SNMPv3 Configuration	27

CODE EXAMPLE 5-1	NmaSwitchover.java	30
CODE EXAMPLE 7-1	NmaMasterNametags.java	43
CODE EXAMPLE 10-1	Implementation of the SnmpTrapListener Class	69
CODE EXAMPLE 10-2	Registering the Trap Listener	70

Preface

This book describes how to use the Node Management Agent (NMA) Java™ application programming interfaces (APIs) of the Netra™ High Availability (HA) Suite 3.0 1/08 software.

The information in this book can help you perform the following tasks:

- Access cluster information from a remote Java or Simple Network Management Protocol (SNMP) management and monitoring application
- Receive SNMP traps or Java Management Extensions (JMX™) notifications about changes occurring in the cluster
- Provoke a cluster mastership switchover
- Manipulate Daemon Monitor parameters

Who Should Use This Book

This book is for application developers who want to develop applications that use the NMA.

Before You Read This Book

To program the NMA you must have working knowledge of the Java language. Knowledge of the Java Dynamic Management Kit (DMK) 5.0 and the Solaris™ Operating System (Solaris OS) is an advantage.

Before reading this book, read the *Netra High Availability Suite 3.0 1/08 Foundation Services Overview*.

How This Book Is Organized

[Chapter 1](#) outlines the features of the NMA.

[Chapter 2](#) introduces and defines the requirements and dependencies of the NMA.

[Chapter 3](#) describes how to access the NMA from a Java manager.

[Chapter 4](#) describes how to access the NMA from a SNMP manager.

[Chapter 5](#) explains how to use the NMA to provoke a master node switchover and set Daemon Monitor parameters.

[Chapter 6](#) describes the Carrier Grade Transfer Protocol (CGTP) statistics that can be accessed from the NMA.

[Chapter 7](#) describes the Daemon Monitor statistics that can be accessed from the NMA.

[Chapter 8](#) describes the Reliable NFS statistics that can be accessed from the NMA.

[Chapter 9](#) describes the Cluster Membership Manager (CMM) statistics that can be accessed from the NMA.

[Chapter 10](#) explains the NMA notification mechanism and the meaning of the NMA notification types.

[Appendix A](#) describes the syntax of the MBean naming conventions.

Using UNIX Commands

This document might not contain information on basic UNIX[®] commands and procedures such as shutting down the system, booting the system, and configuring devices. Refer to the following for this information:

- Software documentation that you received with your system
- Solaris Operating System documentation, which is at

<http://docs.sun.com>

Shell Prompts

Shell	Prompt
C shell	<i>machine-name%</i>
C shell superuser	<i>machine-name#</i>
Bourne shell and Korn shell	\$
Bourne shell and Korn shell superuser	#

Typographic Conventions

Typeface*	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output	% su Password:
<i>AaBbCc123</i>	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the <i>User's Guide</i> . These are called <i>class</i> options. To delete a file, type rm <i>filename</i> .

* The settings on your browser might differ from these settings.

Related Documentation

The following table lists the documentation for this product. The online documentation is available at:

<http://docs.sun.com/app/docs/prod/netra.ha30>

Application	Title	Part Number
Late-breaking news	<i>Netra High Availability Suite 3.0 1/08 Release Notes</i>	819-5249-14
Introduction to concepts	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Overview</i>	819-5240-13
Basic setup, supported hardware, and configurations	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Getting Started Guide</i>	819-5241-13
Automated installation methods	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Installation Guide</i>	819-5242-13
Detailed installation methods	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Manual Installation Guide for the Solaris OS</i>	819-5237-13
Cluster administration	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Cluster Administration Guide</i>	819-5235-13
Using the Cluster Membership Manager	<i>Netra High Availability Suite 3.0 1/08 Foundation Services CMM Programming Guide</i>	819-5236-13
Using the SAF CMM API	<i>Netra High Availability Suite 3.0 1/08 Foundation Services SA Forum Programming Guide</i>	819-5246-13
Using the Node Management Agent	<i>Netra High Availability Suite 3.0 1/08 Foundation Services NMA Programming Guide</i>	819-5239-13
Configuring outside the cluster using CGTP	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Standalone CGTP Guide</i>	819-5247-13
Man pages for Foundation Services features and APIs using the Solaris OS	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Solaris Reference Manual</i>	819-5244-13
Man pages for Foundation Services features and APIs using Linux	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Linux Reference Manual</i>	819-5245-12
Definitions and acronyms	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Glossary</i>	819-5238-13
Common problems	<i>Netra High Availability Suite 3.0 1/08 Foundation Services Troubleshooting Guide</i>	819-5248-13

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (<http://www.sun.com/documentation>)
- Support (<http://www.sun.com/support>)
- Training (<http://www.sun.com/training>)

Third-Party Web Sites

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

<http://www.sun.com/hwdocs/feedback>

Please include the title and part number of your document with your feedback:

Netra™ High Availability Suite 3.0 1/08 Foundation Services NMA Programming Guide,
part number 819-5239-13

Introduction to the Node Management Agent

This chapter describes how to access the Node Management Agent (NMA) and introduces the master node view and the floating external address.

The NMA is a management agent that conforms to the Java Management Extensions (JMX) v1.1 Maintenance Release. The NMA is based on the Java Dynamic Management Kit (DMK) 5.0 APIs.

The NMA monitors the performance and the status of the following Netra HA Suite software components:

- Cluster Membership Manager
- Carrier Grade Transport Protocol (CGTP)
- Daemon Monitor
- Reliable NFS

The NMA exposes this information through a Simple Network Management Protocol (SNMP) management information base (MIB) and a JMX-compliant interface. The JMX specification defines a three-level management architecture:

- The *instrumentation* level makes resources manageable as Java objects called MBeans.
- The *agent* level exposes these objects for management.
- The *distributed* services level allow remote access and security.

By programmatically accessing these MBeans, management applications can be used to help perform tuning operations, diagnostic operations, and troubleshooting.

The NMA also provides a method for provoking a master node switchover, and emits notifications which can be used to keep up to date with the current state of certain aspects of the cluster.

The Java DMK implements the JMX specification. The NMA requires the Java DMK runtime libraries. You can use the Java DMK to develop a remote Java manager to access the NMA. Alternately, access the NMA MIB by using an off-the-shelf or custom SNMP manager, or any JMX-compliant Java manager.

This chapter contains the following sections:

- [“Accessing the NMA” on page 2](#)
- [“Master and Node Views” on page 3](#)
- [“Floating External Address” on page 6](#)

Accessing the NMA

An external manager can communicate with the NMA using any of the following protocols:

- HTTP
- SNMP version 1 (SNMPv1)
- SNMP version 2 (SNMPv2)
- SNMP version 3 (SNMPv3)

Note – The floating external address cannot be used when you are using the SNMP protocol. See [“Floating External Address” on page 6](#) for more information on the floating external address.

Use SNMPv3 to take advantage of the enhanced security mechanism introduced in SNMPv3. Note that not all protocol interfaces are enabled by default. See the `nma.properties` man page for information on how to enable, disable, and configure protocol interfaces.

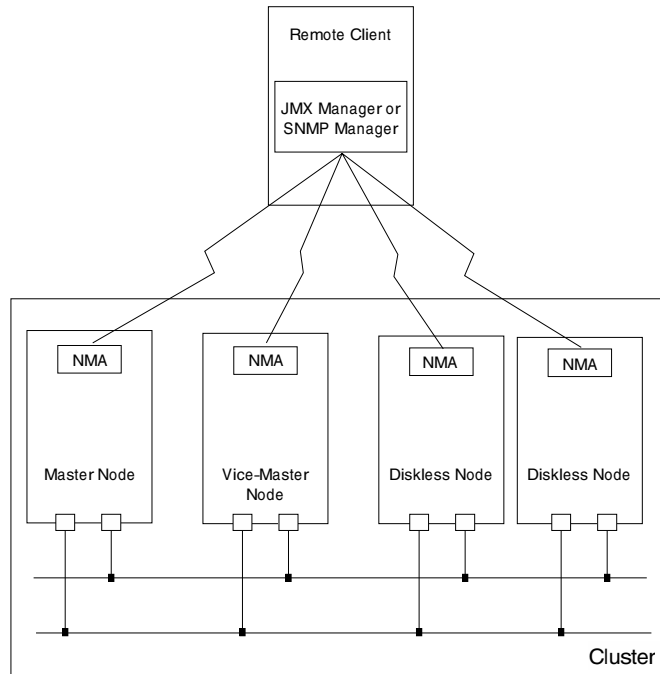
HTML can be used to view the NMAs in a running cluster through an HTML browser. By default port 8082 exports this view. To interact with the NMA on a node, access the URL of the form `http://nodeIPAddress:portNumber`

See HTML Protocol Adaptor in the *Java Dynamic Management Kit 5.0 Tutorial* for more information.

Note that if you provoke a switchover using the HTML Protocol Adaptor connected to the floating external address, the connection to the NMA might be broken prematurely and the information transfer will not finish. If this happens, stop the transfer and reload the page to get the correct node information.

FIGURE 1-1 represents the communication paths between an external manager and the cluster:

FIGURE 1-1 Remote Manager Communication

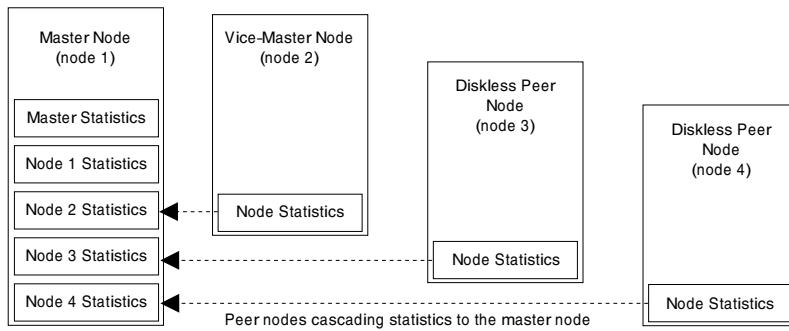


Master and Node Views

The NMA running on the master node makes information from the NMAs running on all nodes visible in the NMA running on the master node through a cascading connection. The MBeans listed in [“MBean Instances on the Master Node” on page 5](#) are available on the master node only.

FIGURE 1-2 shows how information is cascaded from all NMAs to the NMA on the master node.

FIGURE 1-2 Cascading Information From Peer Nodes to the Master Node



Note – The master view is only available to a Java manager that communicates using the HTTP protocol.

All of the agents must use the same port number for the service used to implement cascading. If this is not the case, the master agent will start the cascading service but will not enable the cascading connections to NMA.

Note – After failover or switchover, there is a short period of time during which the NMA information of other nodes is made available on the new master. If you query this information from the master during this period, an exception will be thrown.

Cascading is controlled by the following properties, which are defined in the `nma.properties` file:

<code>com.sun.nhas.ma.cascading.enabled</code>	Set to <code>true</code> to enable cascading.
<code>com.sun.nhas.ma.cascading.retries.max</code>	The maximum number of times that the master node tries to create a cascading connection to an NMA on another node.
<code>com.sun.nhas.ma.cascading.retries.delay</code>	The time in milliseconds between each attempt to establish a cascading connection.

<code>com.sun.nhas.ma.cascading.comm.protocol</code>	The protocol can be http or rmi.
<code>com.sun.nhas.ma.cascading.socket.timeout.wait</code>	The timeout in milliseconds for a cascading connection to terminate cleanly in the case of a communication fault. After this time has elapsed, the cascading connection attempt will be forcibly aborted and then restarted.
<code>com.sun.nhas.ma.eam.polling</code>	The time interval between checks of the floating addresses and IP Network Multipathing (IPMP) groups. If NMA notices a difference between the floating addresses and IPMP groups, NMA will send an SMTP trap to inform you of the difference. If the property is not present in this file, which is the default, the value is 113000 milliseconds.

MBean Instances on the Master Node

One instance of each of the following MBeans is instantiated on the master node of the cluster.

- `CmmMasterNodeMBean`
- `CmmMasterStatisticsMBean`
- `RnfsMasterStatisticsMBean` on a configuration using StorEdge™ Network Data Replicator (SNDR)
- `SDMasterStatisticsMBean` on a configuration using shared disks
- `PmdMasterStatisticsMBean`
- `CgtpMasterMBean`

An `RnfsReplicatedSliceMBean` is instantiated on the master node and on the vice-master node for each Reliable NFS partition.

MBean Instances on Peer Nodes

One instance of each of the following MBeans is instantiated on each peer node of the cluster:

- `ClusterNodeMBean`
- `CmmStatisticsMBean`
- `RnfsStatisticsMBean` on a configuration using SNDR
- `SDStatisticsMBean` on a configuration using shared disks
- `PmdStatisticsMBean`

- CgtpMBean
- CgtpFilterMBean

Multiple instances of the following MBeans might be instantiated:

CgtpReliableLinkStatisticsMBean	One instance for each reliable link
PmdNameTagStatisticsMBean	One instance for each group of processes monitored by the Daemon Monitor

Floating External Address

An external Java manager can use the floating external address (for example, 10.250.10.1) to communicate with the master node. After failover and switchover, the floating address is reassigned to the new master node. The external Java manager must then connect to the new master node, but can connect to the same IP address as before. SNMP cannot use the floating external address to communicate with the master node.

Configuration Files, Dependencies, and Requirements

This chapter defines the NMA software requirements and dependencies.

This chapter contains the following sections:

- “Configuration Files” on page 7
- “Dependencies” on page 8
- “Software Requirements” on page 8

Configuration Files

The following table summarizes the NMA configuration files.

TABLE 2-1 NMA Configuration Files

File	Description
<code>/etc/opt/SUNWcgha/nma.properties</code>	NMA properties file. See the <code>nma.properties4</code> man page for more information.
<code>/etc/opt/SUNWcgha/nma.security</code>	NMA SNMPv3 security parameter configuration file. See the <code>nma.security4</code> man page for more information.
<code>/etc/opt/SUNWcgha/nma.notifs.txt</code>	SNMP trap identification configuration file. See Chapter 4 for more information.
<code>/etc/opt/SUNWcgha/nma.params.txt</code>	SNMP trap parameter configuration file. See Chapter 4 for more information.

TABLE 2-1 NMA Configuration Files *(Continued)*

File	Description
/etc/opt/SUNWcggha/nma.targets.txt	SNMP trap target configuration file. See Chapter 4 for more information.
/etc/opt/SUNWcggha/nma.uacl.template	Template for SNMPv3 user access configuration file. See Chapter 4 for more information.
/etc/opt/SUNWcggha/nma.acl.template	Template for SNMPv1, SNMPv2, and SNMPv3 IP configuration file. See Chapter 4 for more information.

SNMP applications can also manipulate SNMPv3 configuration by using the `com.sun.jdmk.snmp.rfc2573.managerSnmPV3AppliMibRegistration` class.

Dependencies

To enable an external Java manager to access NMA statistics, the class path of the external Java manager must contain the following Java Archive (JAR) files:

- `installDir/SUNWcggha/lib/jcmm.jar`
- `installDir/SUNWcggha/lib/ma.jar`
- `installDir/SUNWcggha/lib/cghautil.jar`
- `installDir/SUNWjdmk/jdmk5.0/lib/jdmkrt.jar`
- `/usr/sadm/lib/snmp/jsnmpapi.jar`
- `installDir/SUNWcggha/lib/rfc2573mgr.jar`
- `installDir/SUNWcggha/lib/rfc2573.jar`

These JAR files are contained in the `SUNWnhmaj` package, the `SUNWjdrt` package, and the `SUNWjsnmp` package.

Software Requirements

The NMA requires the Java™ Runtime Environment (JRE™) 1.4.2 for the Solaris 9 OS and the Solaris 10 OS.

A Java DMK client requires the Java DMK 5.0 runtime libraries, but the usage of Java DMK communication interfaces is configurable. Both Java DMK 4.2 clients and Java DMK 5.0 clients are supported. To interface with Java DMK 4.2 clients, set the value of the `jmx.serial.form` property to `1.0`.

Developing an External Java Manager

For information about how to develop an external Java manager, see the following sections:

- [“Configuring an External Java Manager Using HTTP” on page 9](#)
- [“Connecting to the NMA” on page 10](#)
- [“Using Proxy MBeans” on page 11](#)

Configuring an External Java Manager Using HTTP

To use the HTTP protocol adaptor, edit the following NMA properties in the `nma.properties` file:

<code>com.sun.nhas.ma.connectors.http.enabled</code>	Set to <code>true</code> to enable the HTTP protocol adaptor
<code>com.sun.nhas.ma.connectors.http.port</code>	Set to the number of the port to be used for HTTP communication, for example, <code>8081</code>

These properties are by default `true` and `8081` respectively.

Connecting to the NMA

The procedure for connecting to the NMA, or reconnecting to the NMA in case of a change of mastership, depends on your addressing scheme.

Using the Floating Address

If you are using the floating address to connect to the NMA running on the master node, perform the following steps.

To Manage a Failover or Switchover

1. Use the Java DMK heartbeat mechanism to detect the loss of contact with the master node.

Reduce the timeouts on requests, if necessary, to guarantee the timely detection of a master node crashing abruptly. See Heartbeat Mechanism in the *Java Dynamic Management Kit 5.0 Tutorial* for information about how to use the Java DMK heartbeat mechanism.

2. Reconnect to the NMA.

The master floating address will be assigned to the new master node of the cluster. Reconnect to this NMA at this address.

3. Wait until the cascading service has finished restarting and the master view has restarted.

The cascading service queries all of the NMAs running in the cluster and makes this information available from the NMA on the master node. During this service restart period, not all MBeans cascading from other nodes will be available. If you attempt to manipulate an MBean on the master node that has not yet cascaded, an `InstanceNotFoundException` is thrown.

4. Reregister all notification listeners.

Notifications might be lost between when mastership changes and listeners are reregistered. Query the NMAs in the cluster to discover the current cluster state.

Using a Physical Node Address

If you are connecting to the NMA on each node using the node IP address, no connections will fail after failover or switchover.

Using Proxy MBeans

The statistics providers in the NMA are implemented as MBeans. A set of generated proxy classes for the statistics MBeans is supplied with the NMA. A remote manager can access these statistics through the exposed MBean interfaces. The Java DMK enables predefined proxy classes of these MBeans to be instantiated in an external Java manager, and the objects to be manipulated as if they were present locally. Communication with the proxied MBeans is handled automatically.

For more information about using proxy MBeans, see MBean Proxies in the *Java Dynamic Management Kit 5.0 Tutorial*.

To use the supplied proxy classes, the Java DMK Remote Manager class path must contain the path to `proxies_42.jar` or `proxies_50.jar`, depending on the version of the Java DMK runtime that you are using. The Java DMK toolkit can be used to regenerate the proxy classes. See [Appendix A](#) for more information.

Developing a Remote SNMP Manager

NMA information can be accessed using the Simple Network Management Protocol (SNMP). This chapter explains how to configure an external SNMP manager, and provides examples of the configuration files required for three types of SNMP configurations.

The Java DMK can be used to develop a remote manager that communicates with the NMA using SNMP. For information on how to use the Java DMK to develop a manager that communicates using SNMP, see [“SNMP Manager Configuration Examples” on page 21](#). Alternatively, any SNMP manager can be used.

This chapter contains the following sections:

- [“Configuring an SNMP Agent” on page 13](#)
- [“IP-Based Access Control Lists” on page 15](#)
- [“SNMPv3 User-Based Access Control” on page 18](#)
- [“Configuring SNMPv3 Security” on page 19](#)
- [“SNMP Manager Configuration Examples” on page 21](#)

Configuring an SNMP Agent

The NMA offers SNMPv1, SNMPv2, and SNMPv3 interfaces via the SNMP protocol adaptor. Edit the following values in the `nma.properties` file to configure the SNMP protocol adaptor:

```
com.sun.nhas.ma.adaptors.snmp.enabled
```

Set to true to enable the SNMP protocol adaptor

```
com.sun.nhas.ma.adaptors.snmp.port
```

Set to the number of the port to be used for SNMP communication, for example, 8085

```
com.sun.nhas.ma.adaptors.snmp.trap.port
```

Set to the number of the port to be used to send SNMP traps, for example, 8086

By default the NMA uses the standard Java DMK access control configuration files. The following templates are available for use in a default installation:

```
installDir/etc/opt/SUNWcggha/nma.acl.template
```

Used for SNMPv1, SNMPv2, and SNMPv3 IP access. For SNMPv3, IP access is only relevant to SNMP traps.

```
installDir/etc/opt/SUNWcggha/nma.uacl.template
```

Used for SNMPv3 user access only.

Use these templates to create configuration files for customized access control configuration. Edit the `jdmk.acl.file` and `jdmk.uacl.file` properties in the `nma.properties` file to reflect the paths to your access control configuration files.

The following three files are included for SNMP traps and security configuration, in accordance with the Internet Engineering Task Force RFC 2573.

```
installDir/etc/opt/SUNWcggha/nma.targets.txt
```

SNMP trap target configuration file

```
installDir/etc/opt/SUNWcggha/nma.params.txt
```

SNMP trap security parameter configuration file

`installDir/etc/opt/SUNWcgha/nma.notifs.txt`

SNMP trap identification configuration file

The NMA MIB is located at:

`/SUNWcgha/services/SUNWcgha/doc/ma/nhasmib.txt` in a default installation.

Note – SNMPv1 does not support 64-bit counters. Retrieval of CGTP statistics that use 64-bit counters is not possible when using SNMPv1.

IP-Based Access Control Lists

In SNMPv1 and SNMPv2, access control is provided on the basis of the IP address and community of the manager's host machine.

CODE EXAMPLE 4-1 Typical `nma.ac1` File

```
acl = {
  {
    communities = public
    access = read-only
    managers = yourmanager
  }
  {
    communities = private
    access = read-write
    managers = yourmanager
  }
}

trap = {
  {
    trap-community = public
    hosts = yourmanager
  }
}
```

Format of the `acl` Group

The `acl` group contains one or more access configurations.

```
access1  access2      ...  
acl = {accessN}
```

Each access configuration has the following format:

```
{  
  communities = communityList  access = accessRights  managers = hostList}
```

The *communityList* is a list of SNMP community names to which this access control applies. The community names in this list are separated by commas.

The *accessRights* specifies the rights to be granted to all managers connecting from the hosts specified in the *hostList*. There are two possible values: either `read-write` or `read-only`.

The *hostList* specifies the hosts of the managers to be granted the access rights. The *hostList* is a comma-separated list of hosts, each of which can be expressed as any one of the following:

- A host name
- An IP address
- A subnet mask

The set of all access configurations defines the access policy of the SNMP agent. A manager whose host is specified in a *hostList* and that identifies itself in one of the communities of the same configuration will be granted the permissions defined by the corresponding *accessRights*. A manager's host can appear in several access configurations, provided it is associated with a different community list. This will define different access communities with different rights from the same manager.

A manager whose host-community identification pair does not appear in any of the access configurations will be denied all access. This means that protocol data units (PDUs) from this manager will be dropped without being processed.

Format of the Trap Group

The trap group specifies the hosts to which the agent will send traps if the `InetAddressAcl` mechanism is used. This group contains one or more trap community definitions.

```
community1 community2 ...  
trap = {communityN}
```

Each community definition defines the association between a set of hosts and the SNMP community string in the traps to be sent to them. Each trap definition has the following format:

```
{  
  trap-community = trapCommunityName  hosts = trapHostList}
```

The *trapCommunityName* item specifies a single SNMP community string. It will be included in the traps sent to the hosts specified in the *hosts* item. SNMPv3 does not use the community string, so use IP addresses or the context name instead.

The *trapHostList* item specifies a comma-separated list of hosts. Each host must be identified by its name or complete IP address.

When the SNMP protocol adaptor is instructed to send a trap using the `InetAddressAcl` mechanism, it will send a trap to every host listed in the trap community definitions. If a host is present in more than one list, it will receive more than one trap, each one identified by its corresponding trap community.

SNMPv3 User-Based Access Control

The user-based access control implemented by SNMPv3 is based on contexts and user names. The users, contexts, and associated security information controlling access to the agents in an SNMP session are defined in the `nma.uacl` file.

CODE EXAMPLE 4-2 Example `nma.uacl.template` File

```
acl = {  
  # {  
  # context-names = TEST-CONTEXT  
  # access = read-write  
  # security-level = authNoPriv  
  # users = defaultUser  
  # }  
}
```

In the `nma.uacl` file, you define the following:

- A list of context names, separated by commas. You can define a null context by declaring `context-names = null`
- The access level, which can be either `read-write` or `read-only`
- The security level, as follows:

<code>noAuthNoPriv</code>	No security mechanisms activated
<code>authNoPriv</code>	Authentication activated, with no privacy
<code>authPriv</code>	Both authentication and privacy activated

- A list of authorized users, separated by commas; an asterisk (*) opens access to all users.

By uncommenting the `acl` block in [CODE EXAMPLE 4-2](#), you would limit access to MIBs in the `TEST-CONTEXT` context only, and grant read-write access to the user `defaultUser`. The security level in the file must also match that of user `defaultUser`. Therefore, any non-authenticated requests, any request with different security levels, or any requests from a user other than `defaultUser`, would be rejected.

Configuring SNMPv3 Security

Under SNMPv1 and SNMPv2, agents act as information servers, and IP-based access control is used to protect this information from unauthorized access. The SNMPv3 protocol provides much more sophisticated security mechanisms, implementing a user-based security model (USM). This model allows both authentication and encryption of the requests sent between agents and their managers, as well as user-based access control.

Note – The default NMA configuration is an example of an SNMPv3 configuration. Modify the security parameters to fit your security requirements.

You can add and remove users in the `nma.security` file as specified in [“Managing Users in Security Files” on page 20](#).

Engine ID

Secure SNMPv3 communication requires that the SNMP engine ID, which is generated by the NMA for each node, is used to communicate with the NMA. The SNMP engine ID is unique for the SNMP domain. It is a hexadecimal string calculated from a concatenation of the following properties of the NMA on each node:

- Node CGTP address
- Communication port number
- IANA number. By default this is 42.

The engine ID is stored in the `nma.security` file of each NMA. The engine ID may be substituted for another engine ID.

Context Name

The NMA MIB is not registered under the scope of any context.

Managing Users in Security Files

Every user that has access to an agent is represented by a `userEntry` line in each of the agent's security files.

You configure the `userEntry` as follows, with the parameters separated commas:

`userEntry=engine ID , user name , security name , authentication
algorithm , authentication key , privacy algorithm , privacy key , storage type , template`

The only mandatory parameters are the engine ID and the user name. All the other parameters are optional.

The possible values for the parameters are as follows:

Engine ID	A local or remote SNMP engine, defined in one of the following ways: <ul style="list-style-type: none">• The string <code>localEngineID</code>, to denote the local engine• A hexadecimal string, for example, <code>0x8000002a05819dcb6e00001f95</code>• A human-readable string used to generate an engine ID, providing any or all of the host name, port, and IANA number
User name	Any human-readable string
Security name	Any human-readable string
Authentication algorithm	The following algorithms are permitted: <ul style="list-style-type: none">• <code>usmHMACMD5AuthProtocol</code>• <code>usmHMACSHAAuthProtocol</code>• <code>usmNoAuthProtocol</code>
Authentication key	Any text password or any hexadecimal key starting with <code>0x</code> ; for example, <code>0x0098768905AB67EFAA855A453B665B12</code> , of size: <ul style="list-style-type: none">• 0 to 32 inclusive for HMACMD5• 0 to 40 inclusive for HMACSHA
Privacy algorithm	The following algorithms are permitted: <ul style="list-style-type: none">• <code>usmDESPrivProtocol</code>• <code>usmNoPrivProtocol</code> If no algorithm is specified, the default is <code>usmNoPrivProtocol</code> .

	Any text password or any hexadecimal key starting with 0x; for example, 0x0098768905AB67EF8A855A453B665B12, of size 0 to 32 inclusive If a hexadecimal string is provided, it must be a localized key.
Storage type	A value of 3 denotes <i>non-volatile</i> , meaning that the user entry is flushed in the security file; any value other than 3 will be rejected, throwing an <code>IllegalArgumentException</code> .
template	Can be either <code>true</code> or <code>false</code> : If <code>true</code> , the row is a template, not seen from USM MIB. This kind of user is used when cloning users. The default is <code>false</code> .

Users can also be managed through USM MIB access.

SNMP Manager Configuration Examples

This section contains three examples of SNMP configurations. The NMA implements the Notification MIB module specified by the Internet Engineering Task Force in RFC 2573, which is accessible from <http://www.ietf.org>.

By default the NMA authorizes `localhost` to access its MIB using SNMPv1 or SNMPv2 on port 8085. SNMP traps are sent using the mechanism described in the RFC 2573. Traps are sent by default to `localhost` on port 8086 using SNMPv2 parameters, as defined in the default RFC 2573 configuration files:

- `nma.params.txt`
- `nma.notifs.txt`
- `nma.targets.txt`

The RFC 2573 configuration files can be manually edited. Alternately, use the `com.sun.jdmk.snmp.rfc2573.manager.SnmpV3AppliMibRegistration` class, found in the `rfc2573mgr.jar` file. Use this class to dynamically register or unregister SNMP managers at runtime. [CODE EXAMPLE 4-3](#) is a code snippet that uses this class to register a trap target on trap port `trapPort` of the host `localhost`. Traps are received using SNMPv3 parameters.

CODE EXAMPLE 4-3 Using the `SnmpV3AppliMibRegistration` API

```
//Register the manager/params to the NMA
try {
```

CODE EXAMPLE 4-3 Using the SnmpV3AppliMibRegistration API (Continued)

```
System.out.println("Register the Manager to receive Traps using
SNMPv3 " +
                    parameters");

// Register the SNMP Parameters V3
SnmpV3AppliMibRegistration.registerParams(session,
                                           "manager_paramsv3",
                                           3,
                                           3,
                                           "defaultUser",
                                           2);

// Register the Manager to receive traps with SNMPv3 parameters
SnmpV3AppliMibRegistration.registerTarget(session,
                                           "manager_targetv3",
                                           "1.3.6.1.6.1.1",
                                           localhost + "/" + trapPort,
                                           10000,
                                           2,
                                           "trap",
                                           "manager_paramsv3");

}
catch(SnmpStatusException e) {
    System.out.println("ERROR in registration " + e.getMessage());
}
```

SNMPv2 Configuration Example

In this configuration, the NMA MIB is accessed using SNMPv2 on port number 8085. The SNMP manager is authorized to access the MIB located on host 10.8.1.253. Traps are sent to the manager on port 8086 using SNMPv2, using the Notification MIB described in RFC 2573.

[CODE EXAMPLE 4-4](#) through [CODE EXAMPLE 4-8](#) list the entries in the NMA configuration files that support this SNMP configuration.

CODE EXAMPLE 4-4 Example Entries in nma.properties for SNMPv2

```
com.sun.nhas.ma.adaptors.snmp.enabled=true
com.sun.nhas.ma.adaptors.snmp.port=8085
com.sun.nhas.ma.adaptors.snmp.rfc2573.enabled=true
com.sun.nhas.ma.adaptors.snmp.rfc2573.v1v2set.enabled=true
com.sun.nhas.ma.adaptors.snmp.rfc2573.target.addr.file=\\
/etc/opt/SUNWcgha/nma.targets.txt
```


CODE EXAMPLE 4-4 Example Entries in `nma.properties` for SNMPv2

```
com.sun.nhas.ma.adaptors.snmp.rfc2573.target.params.file=\
/etc/opt/SUNWcgha/nma.params.txt
com.sun.nhas.ma.adaptors.snmp.rfc2573.notification.file=\
/etc/opt/SUNWcgha/nma.notifs.txt
jdkm.ac1.file=/etc/opt/SUNWcgha/nma.ac1
```

CODE EXAMPLE 4-5 Example Entries in `nma.ac1` for SNMPv2

```
ac1 = {
{
communities = public, private
access = read-only
managers = 10.8.1.253
}
{
communities = public, private
access = read-write
managers = 10.8.1.253
}
}
```

CODE EXAMPLE 4-6 Example Entry in `nma.targets.txt` for SNMPv2

```
targetsEntry=
managerV2,snmpUDPDomain,10.8.1.253/8086,10000,2,trap,snmpV2,3
```

CODE EXAMPLE 4-7 Example Entry in `nma.params.txt` for SNMPv2

```
paramsEntry=snmpV2,1,2,public,1,3
```

CODE EXAMPLE 4-8 Example Entry in `nma.notifs.txt` for SNMPv2

```
notificationEntry=notif1,trap,1,3
```

SNMPv2 and SNMPv3 Hybrid Configuration Example

In this configuration, the NMA is located at the CGTP address 10.8.3.18. The NMA MIB can be accessed through SNMPv2 and SNMPv3 using port number 8085. The manager that authorizes access to the MIB in SNMPv2 is located on host 10.8.1.253. The user `defaultUser` is authorized to access the MIB through SNMPv3 using the security parameters described in the `nma.security` file. Traps are sent to the manager on port 8086 using SNMPv2 and on port 8095 using SNMPv3. The notification MIB described in the RFC 2573 is used.

[CODE EXAMPLE 4-9](#) through [CODE EXAMPLE 4-16](#) list the entries in the NMA configuration files that support this SNMP configuration.

CODE EXAMPLE 4-9 Example Entries in `nma.properties` for Hybrid Configuration

```
com.sun.nhas.ma.adaptors.snmp.enabled=true
com.sun.nhas.ma.adaptors.snmp.port=8085
com.sun.nhas.ma.adaptors.snmp.rfc2573.enabled=true
com.sun.nhas.ma.adaptors.snmp.rfc2573.v1v2set.enabled=true
com.sun.nhas.ma.adaptors.snmp.rfc2573.target.addr.file=\
/etc/opt/SUNWcgha/nma.targets.txt
com.sun.nhas.ma.adaptors.snmp.rfc2573.target.params.file=\
/etc/opt/SUNWcgha/nma.params.txt
com.sun.nhas.ma.adaptors.snmp.rfc2573.notification.file=\
/etc/opt/SUNWcgha/nma.notifs.txt
jdmk.acl.file=/etc/opt/SUNWcgha/nma.acl
jdmk.uacl.file=/etc/opt/SUNWcgha/nma.uacl
```

CODE EXAMPLE 4-10 Example Entries in `nma.security` for Hybrid Configuration

```
userEntry=
localEngineID,defaultUser,null,usmHMACMD5AuthProtocol,mypasswd
localEngineBoots=23
localEngineID=0x8000002a050a08031200001f95
```

CODE EXAMPLE 4-11 Example Entries in `nma.acl` for Hybrid Configuration

```
acl = {
{
communities = public, private
access = read-only
managers = 10.8.1.253
}
{
communities = public, private
access = read-write
managers = 10.8.1.253
}
}
```

CODE EXAMPLE 4-12 Example Entries in `nma.uacl` for Hybrid Configuration

```
acl = {
{
context-names = null
access = read-write
security-level=authNoPriv
}
```

CODE EXAMPLE 4-12 Example Entries in `nma.uac1` for Hybrid Configuration

```
users = defaultUser
}
}
```

CODE EXAMPLE 4-13 Example Entries in `nma.targets.txt` for Hybrid Configuration

```
targetsEntry=
managerV2,snmpUDPDomain,10.8.1.253/8086,10000,2,trap,snmpV2,3
targetsEntry=
managerV3,snmpUDPDomain,10.8.1.253/8095,10000,2,trap,snmpV3,3
```

CODE EXAMPLE 4-14 Example Entries in `nma.params.txt` for Hybrid Configuration

```
paramsEntry=snmpV2,1,2,public,1,3
paramsEntry=snmpV3,3,3,defaultUser,2,3
```

CODE EXAMPLE 4-15 Example Entry in `nma.notifs.txt` for Hybrid Configuration

```
notificationEntry=notif1,trap,1,3
```

CODE EXAMPLE 4-16 Example Entries in `nma.security` for Hybrid Configuration

```
userEntry=
10.8.3.18:8085,defaultUser,defaultUser,usmHMACMD5AuthProtocol,\
mypasswd
userEntry=
localEngineID,defaultUser,defaultUser,usmHMACMD5AuthProtocol,\
mypasswd
localEngineBoots=26
localEngineID=0x8000002a05000000ef6540c3f9
```

SNMPv3 Configuration Example

In this configuration, the NMA MIB is accessed using SNMPv3 on port number 8085. The manager authorized to access the MIB is located on host 10.8.1.253. Traps are sent to the manager on trap port 8086. In this case, the notification MIB is not used. Traps are always sent to trap port 8086 as defined in the `nma.properties` file and use only SNMPv2. The `nma.targets.txt`, `nma.params.txt`, and `nma.notifs.txt` files are not used in this configuration.

CODE EXAMPLE 4-17 through CODE EXAMPLE 4-20 list the entries in the NMA configuration files that support this SNMP configuration.

CODE EXAMPLE 4-17 Example Entries in `nma.properties` for SNMPv3 Configuration

```
com.sun.nhas.ma.adaptors.snmp.enabled=true
com.sun.nhas.ma.adaptors.snmp.port=8085
com.sun.nhas.ma.adaptors.snmp.trap.port=8086
com.sun.nhas.ma.adaptors.snmp.rfc2573.enabled=false
com.sun.nhas.ma.adaptors.snmp.rfc2573.v1v2set.enabled=false
com.sun.nhas.ma.adaptors.snmp.rfc2573.target.addr.file=\
/etc/opt/SUNWcgha/nma.targets.txt
com.sun.nhas.ma.adaptors.snmp.rfc2573.target.params.file=\
/etc/opt/SUNWcgha/nma.params.txt
com.sun.nhas.ma.adaptors.snmp.rfc2573.notification.file=\
/etc/opt/SUNWcgha/nma.notifs.txt
jdmk.acl.file=/etc/opt/SUNWcgha/nma.acl
jdmk.uacl.file=/etc/opt/SUNWcgha/nma.uacl
```

CODE EXAMPLE 4-18 Example Entries in `nma.security` for SNMPv3 Configuration

```
userEntry=
localEngineID,defaultUser,null,usmHMACMD5AuthProtocol,mypasswd
localEngineBoots=23
localEngineID=0x8000002a050a08031200001f95
```

CODE EXAMPLE 4-19 Example Entries in `nma.acl` for SNMPv3 Configuration

```
acl = {
{
communities = public, private
access = read-only
managers = 10.8.1.253
}
{
communities = public, private
access = read-write
managers = 10.8.1.253
}
}

trap = {
{
trap-community = public
hosts = 10.8.1.253
}
{
trap-community = private
```

CODE EXAMPLE 4-19 Example Entries in `nma.ac1` for SNMPv3 Configuration

```
    hosts = 10.8.1.253
  }
}
```

CODE EXAMPLE 4-20 Example Entries in `nma.uac1` for SNMPv3 Configuration

```
acl = {
  {
    context-names = null
    access = read-write
    security-level=authNoPriv
    users = defaultUser
  }
}
```


Manipulating the Cluster Using the NMA

The Node Management Agent (NMA) exposes methods for causing a switchover and manipulating the retry count for processes and process groups monitored by the Daemon Monitor.

This chapter contains the following sections:

- [“Using the NMA to Initiate a Switchover” on page 29](#)
- [“Manipulating Daemon Monitor Retry Settings” on page 34](#)

Using the NMA to Initiate a Switchover

Methods of the `CmmMasterNodeMBean` can be used to check whether a switchover is possible, initiate the switchover, and then gauge the success of the switchover. The switchover is performed by the Cluster Membership Manager (CMM).

Because it is possible to set a timeout value for CMM operations, it is also possible that CMM operations might not be completed during the time allowed. If the timeout value is too short, some or all CMM operations will fail. For more information about this CMM behavior, see the `cmm_connect3CMM` man page.

Note – To disable the ability to perform remote operations on a cluster, set the `com.sun.nhas.ma.operation.flag` property in `nma.properties` to `false`.

Checking Whether the Netra HA Suite Software Is Ready for Switchover

To check whether a switchover is possible, invoke the `isSwitchOverReady` method. The `isSwitchOverReady` method takes no parameters, and returns a Boolean value.

Note – Even if the `isSwitchOverReady` method returns `true`, this does not guarantee that switchover will succeed. Switchover might not succeed, for example, if cluster readiness changes between the time when the `isSwitchOverReady` is invoked and the `switchOver` method is invoked.

Initiating a Switchover

To initiate a switchover, invoke the `switchOver` method. The `switchOver` method takes no parameters, and returns a `void`. Note that if this method is invoked using the floating external address the connection to the NMA might be broken prematurely and the `switchOver` method will never finish executing. Write code to detect and handle this eventuality.

Example of Switchover Using an HTTP Connector Client

The `NmaSwitchover` class, the code of which is listed below, can be used to provoke a switchover of the current master node. The mechanism used below (the `invoke()` method of the `HTTPConnectorClient` class) can be used to invoke the methods of the NMA MBeans.

CODE EXAMPLE 5-1 `NmaSwitchover.java`

```
// java import
//
import java.net.InetAddress;

// jmx import
//
import javax.management.ObjectName;
import javax.management.MBeanException;

// jdk import
//
```


CODE EXAMPLE 5-1 NmaSwitchover.java (Continued)

```
import com.sun.jdmk.TraceManager;
import com.sun.jdmk.comm.HttpConnectorClient;
import com.sun.jdmk.comm.HttpConnectorAddress;
import com.sun.jdmk.comm.CommunicationException;
import com.sun.jdmk.comm.UnauthorizedSecurityException;

/**
 * This java client uses an HTTP connector client to establish a
 * connection
 * to the NMA and perform a mastership switchover.
 *
 * To compile the client:
 *
 * javac NmaSwitchover.java
 *
 * Note: First ensure that the jar files specified in the chapter
 * 'Configuration Files, Dependencies and Requirements' of the
 * "Netra High Availability Suite Foundation Services 2.1 6/03
 * NMA Programming Guide" are in your CLASSPATH.
 *
 * To run the client:
 *
 * java NmaSwitchover <domain_name> <master_IP_address>
 * <HttpConnectorServer_port>
 *
 * For example: java NmaSwitchover cluster_8 10.8.1.18 8081
 *
 * Note: This example must be run on a machine with access to the
 * cluster, for example, the cluster install server.
 */
public class NmaSwitchover {

    public static void main(String argv[]) {

        try {

            /**
             * Debug
             * To activate the debug or trace mechanism from the
             * command line, use the syntax:
             * java -DLEVEL_DEBUG NmaSwitchover <arguments>
             * or:
             * java -DLEVEL_TRACE NmaSwitchover <arguments>
             *
             * For example:
             * java -DLEVEL_DEBUG NmaSwitchover cluster_6 10.6.1.1 8081
```

CODE EXAMPLE 5-1 NmaSwitchover.java (*Continued*)

```
*
*/

TraceManager.parseTraceProperties();

// Set the domain name of the cluster
//
String domain = "DefaultDomain";
if (argv.length >= 1) domain = argv[0];

// Set the host name of the remote MBean server.
//
String agentHost = InetAddress.getLocalHost().getHostName();
if (argv.length >= 2) agentHost = argv[1];

// Set the port number of the remote connector server.
//
int agentPort = 8081;
if (argv.length >= 3)
    agentPort = Integer.decode(argv[2]).intValue();

System.out.println(">>> Connecting to " + agentHost +
    " using port number " + agentPort);

// Set up the HTTP Connector Client.
//
HttpConnectorClient connector = new HttpConnectorClient();
try {
    // Initialize communication with the remote MBean server.
    //
    HttpConnectorAddress hca =
        new HttpConnectorAddress(agentHost, agentPort);
    connector.connect(hca);
} catch (IllegalArgumentException e)
{
    System.out.println("Connection exception! " +
        e.getMessage());
} catch (CommunicationException e)
{
    System.out.println("Connection exception! " +
        e.getMessage());
} catch (UnauthorizedSecurityException e)
{
    System.out.println("Connection exception! " +
        e.getMessage());
}
```

CODE EXAMPLE 5-1 NmaSwitchover.java (*Continued*)

```
String[] iargs = {};  
String[] isig = {};  
  
String instanceName = domain +  
    ".master:nhas-object=cluster_node";  
ObjectName node = new ObjectName(instanceName);  
// Invoke the isSwitchOverReady() method to check that the  
// cluster is in a condition to support switchover  
successfully.  
// If this method returns true, invoke the switchover method.  
// Note that this does not guarantee that the switchover  
operation  
// will be successfully invoked, because the readiness of the  
// cluster may change before the switchover can be performed.  
//  
try {  
    if (connector.invoke(node, "isSwitchOverReady",  
        iargs, isig).equals(new Boolean(true)))  
    {  
        System.out.println("Performing switchover");  
        // Switchover  
        //  
        connector.invoke(node, "switchOver", iargs, isig);  
    } else  
    {  
        System.out.println("Cluster not ready for switchover");  
    }  
} catch (MBeanException e)  
{  
    System.out.println("Got an exception invoking switchover!"  
    +  
        e.getTargetException().getMessage());  
}  
  
// Terminate communication with the remote MBean server.  
//  
connector.disconnect();  
  
// Exit program  
//  
System.exit(0);  
  
} catch (Exception e) {  
    System.out.println("Got an exception !" + e.getMessage());  
    e.printStackTrace();  
    System.exit(1);  
}
```

```
    }  
  }  
}
```

Getting the CMM Status of All Cluster Nodes

To get the CMM status of all cluster nodes, invoke the `getAllNodeInfo` method. The `getAllNodeInfo` method takes no parameters, and returns a `CmmMemberInfo[]`.

Manipulating Daemon Monitor Retry Settings

The Daemon Monitor controls groups of processes and attempts to restart these processes if they fail. The number of times that the Daemon Monitor attempts to restart a group of failed processes can be set using the `updateMaxRetryCount` method of the `PmdNameTagStatisticsMBean`. When this method is invoked, the current count of the number of times that the Daemon Monitor has attempted to restart the process group is reset. When a group of processes has been successfully restarted, invoke the `resetRetryCount` method to ensure that the stipulated number of retries are attempted if the process group fails again.

Carrier Grade Transport Protocol Statistics

For information about the Carrier Grade Transfer Protocol (CGTP) statistics that can be accessed from the NMA, see the following sections:

- [“Introducing CGTP Statistics” on page 35](#)
- [“CGTP Master Statistics” on page 36](#)
- [“CGTP Node Statistics” on page 36](#)

Introducing CGTP Statistics

The CGTP statistics collected by the NMA can be used to ascertain the degree of success with which CGTP is operating in a running cluster. Network traffic conditions might slow down or prohibit the arrival of duplicate packets, or a bottleneck might occur at the local or remote end of the CGTP link. CGTP statistics provide a measure of the success with which CGTP is operating. These statistics are also useful for diagnosing the source of bad performance or failure. Using the CGTP statistics collected by the NMA it is possible to measure the success of packet duplication and filtering, redundant link by redundant link, and pinpoint the source of slow performance or packet loss.

For more information about CGTP, see the *Netra High Availability Suite 3.0 1/08 Foundation Services Overview*.

Set the `com.sun.nhas.ma.cgtp.polling` property to specify the interval in milliseconds between sequential updates of CGTP information in the NMA.

CGTP Master Statistics

CgtpMasterMBean

The `CgtpMasterMBean` MBean exposes the CGTP master node view. This MBean provides lists of local and remote CGTP addresses. One MBean implementing the `CgtpMasterMBean` interface is instantiated on the cluster master node.

Getting All Local CGTP Addresses for Which Statistics Are Available

To return the list of all local CGTP addresses for which statistics are available, invoke the `getLocalCgtpAddresses` method. The `getLocalCgtpAddresses` method takes no parameters, and returns a `String[]`. If statistics are not available, the method returns `null`.

Getting All Remote CGTP Addresses for Which Statistics Are Available

To return the list of remote CGTP addresses for which statistics are available, invoke the `getRemoteCgtpAddresses` method. The `getRemoteCgtpAddresses` method takes no parameters, and returns a `String[]`. If statistics are not available, the method returns `null`.

CGTP Node Statistics

CgtpMBean

One instance of the `CgtpMBean` is instantiated on each node.

Getting All Local CGTP Addresses for Which Statistics Are Available

To return the list of all local CGTP addresses for which statistics are available, invoke the `getLocalCgtpAddresses` method. The `getLocalCgtpAddresses` method takes no parameters, and returns a `String[]`. If statistics are not available, the method returns `null`.

Getting All Remote CGTP Addresses for Which Statistics Are Available

To return the list of remote CGTP addresses for which statistics are available, invoke the `getRemoteCgtpAddresses` method. The `getRemoteCgtpAddresses` method takes no parameters, and returns a `String[]`. If statistics are not available, the method returns `null`.

CgtpEmitterStatisticsMBean

The `CgtpEmitterStatisticsMBean` MBean provides statistics about the packets a node has sent through the local subinterface, in the reliable link operated by CGTP from a local CGTP address to a remote CGTP address.

Getting the Number of Packets Sent Through Each Subinterface

To get the number of packets sent through each local subinterface taking part in the reliable link, invoke the `getSubInterfaceSentCount` method. The subinterfaces are placed in the same order as that used in `CgtpReliableLinkStatisticsMBean.getSubInterfaceAddresses()`. The `getSubInterfaceSentCount` method takes no parameters, and returns an `int[]`.

CgtpFilterMBean

This MBean interface exposes the statistics available on the CGTP filter. There is one MBean per node which provides information on the CGTP filter.

Getting the Number of Packets Not Received in Duplicate

To get the number of packets that have not been duplicated, invoke the `getFilterFailure` method. The `getFilterFailure` method takes no parameters, and returns a `long`.

Getting the Amount of Memory Currently Used by the Filter Module

To get the amount of memory used by the filter module, invoke the `getFilterMemory` method. The `getFilterMemory` method takes no parameters, and returns a `long`.

Getting the Number of Packets Successfully Received

To get the number of packets successfully received and successfully filtered, invoke the `getFilterSuccess` method. The `getFilterSuccess` method takes no parameters, and returns a `long`.

Getting the Number of Hash Table Collisions

To get the number of collisions that have occurred in the hash table, invoke the `getHostCollisions` method. The `getHostCollisions` method takes no parameters, and returns an `int`.

Getting the Number of Direct Hash Table Entries

To get the number of direct entries in the hash table, invoke the `getHostDirect` method. The `getHostDirect` method takes no parameters, and returns an `int`.

Getting the Number of Hash Table Entries

To get the number of entries in the hash table, invoke the `getHostEntries` method. The `getHostEntries` method takes no parameters, and returns an `int`.

Getting the Maximum Amount of Memory Used by the Filter Module

To get the highest amount of memory used by the filter, invoke the `getMaxFilterMemory` method. The `getMaxFilterMemory` method takes no parameters, and returns a `long`.

Getting the Number of Packets Waiting for Duplicate Reception

To get the number of packets for which no duplicate has yet been received, invoke the `getPremiumPacket` method. The `getPremiumPacket` method takes no parameters, and returns an `int`.

Getting the Number of Packet Hash Collisions

To get the current number of packet hash collisions, invoke the `getCollisions` method. The `getCollisions` method takes no parameters, and returns an `int`.

Getting the Maximum Number of Packet Hash Collisions

To get the maximum number of packet hash collisions, invoke the `getMaxCollisions` method. The `getMaxCollisions` method takes no parameters, and returns an `int`.

Getting the Maximum Number of Packet Hash Collisions in One Row

To get the maximum number of packet hash collisions found in one of the rows of the packet database, invoke the `getMaxLineCollisions` method. The `getMaxLineCollisions` method takes no parameters, and returns an `int`.

CgtpReceiverStatisticsMBean

The `CgtpReceiverStatisticsMBean` MBean provides statistics about the packets received by this node through the reliable link operated by CGTP from a remote CGTP address to a local CGTP address.

Getting the Number of Packets Not Successfully Filtered

To return the number of packets not successfully filtered on reception, invoke the `getFilterFailureCount` method. The `getFilterFailureCount` method takes no parameters, and returns an `int`.

Getting the Number of Packets Successfully Filtered

To return the number of packets successfully filtered on reception, invoke the `getFilterSuccessCount` method. The `getFilterSuccessCount` method takes no parameters, and returns an `int`.

Getting the Number of Packets Received Through Each Subinterface

To return the number of packets received through each local subinterface taking part in the reliable link, invoke the `getSubInterfaceReceivedCount` method. The subinterfaces are placed in the same order as that used in `CgtpReliableLinkStatisticsMBean.getSubInterfaceAddresses()`. The `getSubInterfaceReceivedCount` method takes no parameters, and returns an `int[]`.

CgtpReliableLinkStatisticsMBean

The `CgtpReliableLinkStatisticsMBean` MBean lists the addresses of the subinterfaces, and the reliable link addresses currently in use by the Reliable Transport Service.

Getting the Remote Subinterface Addresses

To get the remote interface addresses used by CGTP to send and receive packets, invoke the `getGatewayAddresses` method. The number of redundant links making up the reliable link is limited to two. The array elements are ordered identically to the subinterface. The `getGatewayAddresses` method takes no parameters, and returns a `String[]`.

Getting Local End Reliable Link CGTP Addresses

To get the local CGTP address at the local end of the reliable link for which these statistics are provided, invoke the `getLocalCgtpAddress` method. The `getLocalCgtpAddress` method takes no parameters, and returns a `String`.

Getting Remote End Reliable Link CGTP Addresses

To return the remote CGTP address at the remote end of the reliable link for which these statistics are provided, invoke the `getRemoteCgtpAddress` method. The `getRemoteCgtpAddress` method takes no parameters, and returns a `String`.

Getting Local Subinterface Addresses

To return the underlying local subinterface addresses used by CGTP to send and receive packets, invoke the `getSubInterfaceAddresses` method. The `getSubInterfaceAddresses` method takes no parameters, and returns a `String[]`. The number of redundant links making up the reliable link is limited to two. The array elements are ordered identically to those of the gateway.

Daemon Monitor Statistics

This chapter describes the Daemon Monitor statistics that can be accessed from the NMA.

This chapter contains the following sections:

- “Example of Accessing Statistics Using an HTTP Client” on page 43
- “Introducing Daemon Monitor Statistics” on page 47
- “Daemon Monitor Master Statistics” on page 48
- “Daemon Monitor Node Statistics” on page 48

Example of Accessing Statistics Using an HTTP Client

The `NmaMasterNametags` example, the code of which is listed in Example 7-1, queries the `PmdMasterStatisticsMBean` for the list of daemon monitor nametags active on the master node. The mechanism used below (the `invoke()` method of the `HTTPConnectorClient` class) can be used to invoke the methods of the NMA MBeans and query the NMA for statistics and information.

CODE EXAMPLE 7-1 `NmaMasterNametags.java`

```
/*
 * @(#)file      NmaMasterNametags.java
 * @(#)author    Sun Microsystems, Inc.
 * @(#)version    1.2
 * @(#)date      02/06/06
 *
 * Copyright 2002 Sun Microsystems, Inc. All rights reserved.
 * This software is the proprietary information of Sun Microsystems, Inc.
```

CODE EXAMPLE 7-1 NmaMasterNametags.java (Continued)

```
* Use is subject to license terms.
*
* Copyright 2002 Sun Microsystems, Inc. Tous droits rservs.
* Ce logiciel est propri  t   de Sun Microsystems, Inc.
* Distribu   par des licences qui en restreignent l'utilisation.
*/

// java import
//
import java.net.InetAddress;

// jmx import
//
import javax.management.ObjectName;
import javax.management.MBeanException;

// jdmk import
//
import com.sun.jdmk.TraceManager;
import com.sun.jdmk.comm.HttpConnectorClient;
import com.sun.jdmk.comm.HttpConnectorAddress;
import com.sun.jdmk.comm.CommunicationException;
import com.sun.jdmk.comm.UnauthorizedSecurityException;

/**
 * This java client uses an HTTP connector client to establish a connection
 * to the Master NMA and retrieve all Nametags.
 *
 * To compile the client:
 *
 * javac NmaMasterNametags.java
 *
 * Note: First ensure that the jar files specified in the chapter
 * 'Configuration Files, Dependencies and Requirements' of the
 * "Netra High Availability Suite 3.0
 * NMA Programming Guide" are in your CLASSPATH.
 *
 * To run the client:
 *
 * java NmaMasterNametags domain_name> master_IP_address>
 * HttpConnectorServer_port>
 *
 * For example: java NmaMasterNametags cluster_8 10.8.1.18 8081
 *
 * Notes:
 * 1) This example must be run on a machine with access to the
```

CODE EXAMPLE 7-1 NmaMasterNametags.java (*Continued*)

```
* cluster, for example, the cluster install server.
* 2) The second parameter can also be the master floating address, for
* example, 10.8.1.1
*
*/

public class NmaMasterNametags {

    public static void main(String argv[]) {

        try {

            /**
             * Debug
             * To activate the debug or trace mechanism from the command
             * line, use the syntax:
             * java -DLEVEL_DEBUG NmaMasterNametags arguments> or
             * java -DLEVEL_TRACE NmaMasterNametags arguments>
             *
             * For example:
             * java -DLEVEL_DEBUG NmaMasterNametags cluster_6 10.6.1.1 8081
             *
             */

            TraceManager.parseTraceProperties();

            // Set the domain name of the cluster
            //
            String domain = "DefaultDomain";
            if (argv.length >= 1) domain = argv[0];

            // Set the host name of the remote MBean server.
            //
            String agentHost = InetAddress.getLocalHost().getHostName();
            if (argv.length >= 2) agentHost = argv[1];

            // Set the port number of the remote connector server.
            //
            int agentPort = 8081;
            if (argv.length >= 3)
                agentPort = Integer.decode(argv[2]).intValue();

            System.out.println(">>> Connecting to " + agentHost +
                " using port number " + agentPort);

            // Set up the HTTP Connector Client.
            //
        }
    }
}
```

```

HttpConnectorClient connector = new HttpConnectorClient();

try {
    // Initialize communication with the remote MBean server.
    //
    HttpConnectorAddress hca =
        new HttpConnectorAddress(agentHost,agentPort);
    connector.connect(hca);
} catch (IllegalArgumentException e) {
    System.out.println("Connection exception! " +
        e.getMessage());
} catch (CommunicationException e) {
    System.out.println("Connection exception! " +
        e.getMessage());
} catch (UnauthorizedSecurityException e) {
    System.out.println("Connection exception! " +
        e.getMessage());
}

// Get Nametags
//

String[] iargs = {};
String[] isig = {};

String instanceName = domain + ".master:nhas-object=pmd_stats";
ObjectName node =
    new ObjectName(instanceName);
try {
    // Attempt to invoke getNameTags()
    //
    String[] nt = (String[])
        connector.invoke(node, "getNameTags", iargs, isig);

    System.out.println("Node " + argv[0] +
        " is running process groups:");
    // Print each element of the array returned by getNameTags()
    // to the standard output. Each element is a nametag
    // managed by the daemon monitor
    //
    for (int i = 0; i < nt.length; i++) {
        System.out.println(nt[i]);
    }
} catch (MBeanException e) {
    System.out.println("Got an exception invoking " +
        "getNameTags()! " + e.getTargetException().getMessage());
}

```



```
        // Terminate communication with the remote MBean server.
        //
        connector.disconnect();

        // Exit program
        //
        System.exit(0);

    } catch (Exception e) {
        System.out.println("Got an exception !" + e.getMessage());
        e.printStackTrace();
        System.exit(1);
    }
}
```

Introducing Daemon Monitor Statistics

The Daemon Monitor statistics are useful in maintaining awareness of processes that fail, and processes that are unable to restart within the allowed number of retries. Access to the process IDs (PIDs) of the processes allows for the monitoring of these processes using standard Solaris Operating System commands.

Note – Daemon Monitor statistics are cached. The `com.sun.nhas.ma.pmd.cache.validity` and `com.sun.nhas.ma.pmd.polling` properties in the `nma.properties` file control the Daemon Monitor polling interval and cache data validity period. If the values of these properties are set too low, the cache might be refreshed before all statistics cached in the previous polling period are read. The default values should be sufficient in most cases.

See Daemon Monitor in *Netra High Availability Suite 3.0 1/08 Foundation Services Overview* for more information about the Daemon Monitoring service.

Daemon Monitor Master Statistics

This section describes the Daemon Monitor statistics available from the NMA on the master node.

`PmdMasterStatisticsMBean`

The `PmdMasterStatisticsMBean` MBean provides the nametags of all daemons currently being monitored.

Getting All Nametags

To return all the nametags managed by the NMA, invoke the `getNameTags` method. The `getNameTags` method takes no parameters, and returns a `String[]`.

Daemon Monitor Node Statistics

This section describes the Daemon Monitor statistics collected by the NMA on each peer node.

`PmdStatisticsMBean`

The `PmdStatisticsMBean` provides a list of all the nametags monitored by the Daemon Monitor.

Getting All Nametags

To return all the nametags managed by the Daemon Monitor, invoke the `getNameTags` method. The `getNameTags` method takes no parameters, and returns a `String[]`.

PmdNameTagStatisticsMBean

The `PmdNameTagStatisticsMBean` MBean provides information about the number of attempts that can be made to restart a daemon, and the number of attempts that have already been made. This MBean is the source of:

- A `NhasPmdMaxRetriesNotification`, which is sent whenever the maximum allowed number of retry attempts is exceeded
- A `AttributeValueChangeNotification`, which is sent whenever the number of allowed retry attempts is changed
- A `NhasPmdNewNameTagNotification`, which is sent whenever the Daemon Monitor creates a new nametag
- A `NhasPmdNewNameTagNotification`, which is sent whenever the Daemon Monitor removes a nametag from the collection

One instance of this MBean is instantiated for each Daemon Monitor by the Daemon Monitor service.

Getting the Daemon Monitor Nametag

To get the nametag that the `PmdNameTagStatisticsMBean` MBean is providing data on, invoke the `getNameTag` method. The `getNameTag` method takes no parameters, and returns a `String`.

Getting the PIDs Associated With a Nametag

To get the list of process IDs associated with this nametag, invoke the `getPidList` method. The `getPidList` method takes no parameters, and returns an `int[]`.

Getting the Daemon Monitor Maximum Retries

To get the maximum number of restart retries allowed for this nametag, invoke the `getMaxRetryCount` method. The `getMaxRetryCount` method takes no parameters, and returns an `int`.

Getting the Number of Retries for a Nametag

To number of restart retries already attempted for this nametag, invoke the `getRetryCount` method. The `getRetryCount` method takes no parameters, and returns an `int`.

Reliable NFS Statistics

This chapter describes the Reliable NFS statistics that can be accessed from the NMA.

This chapter contains the following sections:

- [“Introducing Reliable NFS Statistics” on page 51](#)
- [“Reliable NFS Using SNDR” on page 51](#)
- [“Reliable NFS Using Shared Disk Configuration” on page 55](#)

Introducing Reliable NFS Statistics

The Reliable NFS statistics collected by the NMA provide a view on the current state of replication in the cluster, node by node, reliable link by reliable link. Reliable NFS statistics are available only on the master node and the vice-master node.

See *File Sharing and Data Replication in the Netra High Availability Suite 3.0 1/08 Foundation Services Overview* for more information.

Reliable NFS Using SNDR

Reliable NFS can be used with SNDR or shared-disk configurations. This section describes the statistics that are generated for an SNDR configuration.

Reliable NFS Master Statistics for SNDR

This section describes the Reliable NFS statistics collected by NMA running on the master node.

RnfsMasterReplicatedSliceMBean

The `RnfsMasterReplicatedSliceMBean` MBean models a Reliable NFS replicated slice. Each slice is composed of a primary partition and a secondary partition. One instance of this MBean is instantiated for each replicated slice mounted on either the master or the vice-master node.

Getting the Completed Recovery Percentage

To get the percentage of segments of the slice that has been resynchronized, invoke the `getCompletedRecoveryPercentage` method, which takes no parameters and returns a `float`. This information is meaningful if the primary slice of this MBean is mounted on the host running the agent. No statistics are provided for a distant primary MBean.

Getting the Percentage of Segments Requiring Recovery

To get the percentage of segments of the slice that require recovery, invoke the `getNeededRecoveryPercentage` method, which takes no parameters and returns a `float` that indicates the percentage of segments that require recovery. This information is meaningful if the primary slice of this MBean is mounted on the host running the agent. No statistics are provided for distant primary MBean.

Reliable NFS Peer Node Statistics for SNDR

This section describes the Reliable NFS statistics collected by NMA running on each peer node.

RnfsStatisticsMBean

The `RnfsStatisticsMBean` MBean provides global Reliable NFS statistics. One instance of this MBean is instantiated on each master eligible node in the cluster. This MBean only provides statistics about Reliable NFS on the node on which it is running.

Getting the Primary Slice

To return the primary dual copy slice file name and slice name, invoke the `getPrimarySlice` method, which takes no parameters and returns a `Slice`.

Getting the Secondary Slice

To return the secondary dual copy slice file name and slice name, invoke the `getSecondarySlice` method, which takes no parameters and returns a `Slice`.

`RnfsMasterStatisticsMBean`

The `RnfsMasterStatisticsMBean` MBean provides Reliable NFS statistics on the master node.

Getting the Names of All Primary Files on the Local Host

To get an array of the names of the primary slices mounted on the local host, invoke the `getPrimaryFiles` method, which takes no parameters and returns a `String[]`.

Getting the Names of All Secondary Files on the Local Host

To return an array of the names of the secondary slices mounted on the local host, invoke the `getSecondaryFiles` method, which takes no parameters and returns a `String[]`.

`RnfsReplicatedSliceMBean`

The `RnfsReplicatedSliceMBean` MBean models a Reliable NFS slice. One instance of this MBean is instantiated for each replicated slice mounted on each MEN.

Getting the Completed Recovery Percentage

To get the percentage of segments of the partition that has been resynchronized, invoke the `getCompletedRecoveryPercentage` method, which takes no parameters and returns a `float`. This information is meaningful if the primary slice of this MBean is mounted on the host running the agent. No statistics are provided for a distant primary MBean.

Getting the Dual Copy Status

To get the current status of the dual copy as a `DualCopyStatusEnum` value, invoke the `getDualCopyStatus` method, which takes no parameters and returns a `DualCopyStatusEnum`. This information is meaningful if the primary slice of this MBean is mounted on the host running the agent. No statistics are provided for a distant primary MBean.

Getting the Link Status

To find out if replication is enabled, disabled, or in progress, invoke the `getLinkStatus` method, which takes no parameters and returns a `LinkStatusEnum`.

Getting the Percentage of Segments Requiring Recovery

To indicate the percentage of segments of the partition that require recovery, invoke the `getNeededRecoveryPercentage` method, which takes no parameters and returns a `float`. This information is meaningful if the primary slice of this MBean is mounted on the host running the agent. No statistics are provided for a distant primary MBean.

Getting the Primary Slice

To return the primary dual copy slice file name and slice name, invoke the `getPrimarySlice` method, which takes no parameters and returns a `Slice`.

Getting the Secondary Slice

To return the secondary dual copy slice file name and slice name, invoke the `getSecondarySlice` method, which takes no parameters and returns a `Slice`.

Getting the Names of All Primary Files on the Local Host

To get an array of the names of the primary slices mounted on the local host, invoke the `getPrimaryFiles` method, which takes no parameters and returns a `String[]`.

Getting the Names of All Secondary Files on the Local Host

To return an array of the names of the secondary slices mounted on the local host, invoke the `getSecondaryFiles` method, which takes no parameters and returns a `String[]`.

Reliable NFS Using Shared Disk Configuration

Reliable NFS can be used with SNDR or shared-disk configurations. This section describes the statistics that are generated for a shared-disk configuration.

Reliable NFS Master Statistics for Shared Disk

This section describes the Reliable NFS statistics collected by the NMA running on shared disk configuration.

SDMasterStatisticsMBean

The `SDMasterStatisticsMBean` MBean for shared disk models the Reliable NFS statistics.

Displaying the Shared Disk Device Name and ID

To display the device name and the device ID for the shared disk on the master node, invoke the `getDevices` method. The `getDevices` method takes no parameters and returns a `String[]`.

Displaying the List of Drives Managed by the Shared Disk

To display the list of drives that are managed by the shared disk, invoke the `getDrives` method. The `getDrives` method takes no parameters and returns a `String[]`.

Getting Disk Fencing Information

To display the list of disks taking part in the fencing provided by Reliable NFS (RNFS), invoke the `getFencingDisks` method. This list is pulled directly from the `nhfs.conf` file.

Displaying the Disk Fencing Type

To display the type of disk fencing used to guarantee data integrity for the shared disk, invoke the `getFencingType` method. Properties are `NONE`, `SCSI2`, and `SCSI3`. The default is `NONE`. For more information about this method, refer to the `nhfs.conf4` man page.

Displaying the List of Hosts That Access the Shared Disk

To display the names of the hosts that access the shared disk device, invoke the `getHosts` method. The `getHosts` method takes no parameters and returns a `String[]`.

Displaying the Details of the Shared Devices Configuration

To display information about the configuration of the shared devices, invoke the `getMetadevicesConfig` method. The `getMetadevicesConfig` method takes no parameters and returns a `String[]`.

Displaying the Mirror and Submirror Information

To display the names of the mirrors, their submirrors, and the status of the submirrors, invoke the `getMirrors` method. The `getMirrors` method takes no parameters and returns a `String[]`.

Displaying the Name of the Host that Owns the Shared Disk

To display the name of the host that is identified as the owner of the shared disk, invoke the `getOwner` method. The `getOwner` method takes no parameters and returns a `String[]`.

Displaying the Disk Partitions Managed by RNFS

To display the disk partitions that must be managed by Reliable NFS, invoke the `getRNFSslices` method. Properties are `metadevice` and `mountflag`. There is no default value for this property. For more information about this method, refer to the `nhfs.conf4` man page.

Displaying the Name of the Set of Disks Managed by the Shared Disk

To display the name of the set of disks that are managed by shared disk and used by the Netra HA Suite software, invoke the `getSetName` method. The `getSetName` method takes no parameters and returns a `String[]`.

Displaying the Numeric Identifier of the Disks Managed by the Shared Disk

To display the number assigned to the set of disks that are managed by shared disk and used by the Netra HA Suite software, invoke the `getSetNumber` method. The `getSetNumber` method takes no parameters and returns a `String[]`.

Cluster Membership Manager Statistics

This chapter describes the Cluster Membership Manager (CMM) statistics that can be accessed from the NMA.

This chapter contains the following sections:

- [“Introducing CMM Statistics” on page 59](#)
- [“CMM Master Statistics” on page 60](#)
- [“CMM Node Statistics” on page 62](#)

Introducing CMM Statistics

The CMM statistics collected by the NMA provide the role and status of each node in the cluster.

When a direct link is configured between the master-eligible nodes, the NMA can monitor the following statistics:

- The number of times that the vice-master node has requested to become the master node.
- The state of the direct link. The state can be *up* or *down*.

For information about the direct link, see the *Netra High Availability Suite 3.0 1/08 Foundation Services Overview*.

Because it is possible to set a timeout value for CMM operations, it is also possible that CMM operations cannot be completed during the time allowed. If the timeout value is too short, some or all CMM operations will fail. For more information about this CMM behavior, see the `cmm_connect3CMM` man page.

CMM Master Statistics

`CmmMasterStatisticsMBean`

The `CmmMasterStatisticsMBean` MBean interface makes master state information available. One MBean implementing the `CmmMasterStatisticsMBean` interface is instantiated on the CMM cluster master node.

Getting the Average Time Between Node Starts

To get the average time in seconds between nodes when starting the CMM, invoke the `getAverageElectionDelay` method. This information can be used for tuning the master election mechanism. The `getAverageElectionDelay` method takes no parameters, and returns an `int`.

Getting the Number of CMM Clients

To get the number of CMM clients currently connected, invoke the `getClientCount` method. The `getClientCount` method takes no parameters, and returns an `int`. This information is available on all nodes.

Getting the CMM Lifetime

To get the lifetime of the CMM on this node, expressed in the number of seconds since boot, invoke the `getCmmUpTime` method. The `getCmmUpTime` method takes no parameters, and returns an `int`. This information is available on all nodes.

Getting the Number of Node Elections

To get the number of elections processed on the platform, invoke the `getElectionCount` method. The `getElectionCount` method takes no parameters, and returns an `int`.

Getting the Longest Interval Between Node Starts

To get the maximum time in seconds between nodes when starting the CMM, invoke the `getMaxElectionDelay` method. The `getMaxElectionDelay` method takes no parameters, and returns an `int`. This information can be used for tuning the election mechanism.

Getting the Shortest Interval Between Node Starts

To get the minimum time in seconds between nodes when starting the CMM, invoke the `getMinElectionDelay` method. The `getMinElectionDelay` method takes no parameters, and returns an `int`. This information can be used for tuning the election mechanism.

Getting the Number of Nodes in the Cluster

To get the number of nodes acknowledged by the master node as being present in the cluster, invoke the `getPresentNodeCount` method. The `getPresentNodeCount` method takes no parameters, and returns an `int`.

Getting the Number of Outstanding CMM Requests

To get the number of requests currently outstanding, invoke the `getRequestCount` method. The `getRequestCount` method takes no parameters, and returns an `int`. This information is available on all nodes.

Getting the Switchover Count

To get the number of switchovers performed, invoke the `getSwitchOverCount` method. The `getSwitchOverCount` method takes no parameters, and returns an `int`.

CMM Node Statistics

ClusterNodeMBean

The `ClusterNodeMBean` MBean interface exposes the CMM view of the local node on which the agent runs. This MBean interface makes the CMM state information of the current node available. One MBean implementing the `ClusterNodeMBean` interface is instantiated in each management agent in the cluster. The MBean that implements this interface is the emitter of the `java.com.sun.nhas.ma.cmm.NhasCmmNotification`.

Getting a Node's CGTP Address

To return the CGTP address of a node, invoke the `getCgtpAddress` method. The `getCgtpAddress` method takes no parameters, and returns a `String`. This is not a symbolic name for the node. Having the IP address in dot-notation makes it possible to avoid translation into IP format.

Getting the Domain ID of the Cluster That a Node Is Eligible to Join

To return the domain ID of the cluster that a node is eligible to join, invoke the `getDomainId` method. The `getDomainId` method takes no parameters, and returns an `int`. The domain ID identifies the cluster that the node can join. A cluster is composed of nodes that have the same domain ID. Two nodes running incompatible versions of a software package must have different domain IDs. Nodes can only belong to one cluster. The ID of that cluster is the domain ID of the current node as defined in the CMM configuration. Because the CMM is only aware of what occurs in its own cluster, the domain ID field will be the same for all nodes reachable from this node. This information is useful when interpreting CMM debug traces, which will refer to this domain ID when necessary.

Getting the Time Since Node Was Last Rebooted

To return the incarnation number, invoke the `getIncarnationNumber` method. The incarnation number is computed locally by each node. The value of the incarnation number is the time of the last reboot expressed in the number of seconds since epoch (01/01/1970). The `getIncarnationNumber` method takes no parameters, and returns a `long`.

Getting the CMM Membership Role of a Node

To get the membership role of this node, invoke the `getMembershipRole` method. This information is extracted from the `CmmStateFlag`. The `getMembershipRole` method takes no parameters, and returns a `com.sun.nhas.ma.cmm.CmmMembershipRoleEnum`, which is one of the following values:

MASTER	The node is the current cluster master node.
VICEMASTER	The node is the current vice-master node.
IN_CLUSTER	The node is a regular node in the cluster.
OUT_OF_CLUSTER	The node is down.

Getting the Node ID

To return the unique ID that identifies this node within the cluster, invoke the `getNodeId` method. This information is useful when interpreting CMM debug traces, which refer to this ID. The `getNodeId` method takes no parameters, and returns an `int`.

Getting the Node Name

To get a human-readable `String` that uniquely identifies this node within the cluster, invoke the `getNodeName` method. The name is not intended to be a parameter to be passed to system calls. It is intended to be used to format display messages. For instance, this name could be formatted to refer to the position of the card on a shelf to make it easy for an operator to locate and replace it in case of failure. The `getNodeName` method takes no parameters, and returns a `String`.

Getting the Node Boot Image ID

To return the ID of the current boot image used by this node, invoke the `getSoftwareLoadId` method. Since CMM is only aware of the domain ID, if two nodes run two incompatible boot images they must have different domain IDs and different software load IDs. The `getSoftwareLoadId` method takes no parameters, and returns a `String`.

Getting the CMM State Flags of a Node

To return the membership state flags of a node, invoke the `getStateFlags` method. These flags are a concatenation of the administrative attributes, the membership roles, and the qualification as seen from the perspective of the CMM. The `getStateFlags` method takes no parameters, and returns a `CmmStateFlag`.

CmmStatisticsMBean

The `CmmStatisticsMBean` MBean provides statistics about the service performed by the CMM. Some of these statistics will only be available on the master node. Others will be available on each node in the cluster. One MBean instance is instantiated for each NMA on the cluster.

Getting the Average Time Taken to Start CMM Services

To get the average time in seconds between nodes when starting the CMM, invoke the `getAverageElectionDelay` method. The `getAverageElectionDelay` method takes no parameters, and returns an `int`. This information can be used for tuning the election mechanism. This information is available on the master node only.

Getting the Number of Master Elections Performed on a Node

To get the number of elections processed on the platform, invoke the `getElectionCount` method. The `getElectionCount` method takes no parameters, and returns an `int`. This information is available on the master node only.

Getting the Maximum Time Taken to Elect a Master Node

To get the maximum time in seconds between nodes when starting the CMM, invoke the `getMaxElectionDelay` method. The `getMaxElectionDelay` method takes no parameters, and returns an `int`. This information can be used for tuning the election mechanism. This information is available on the master node only.

Getting the Minimum Time Taken to Elect a Master Node

To get the minimum time in seconds between nodes when starting the CMM, invoke the `getMinElectionDelay` method. The `getMinElectionDelay` method takes no parameters, and returns an `int`. This information can be used for tuning the election mechanism. This information is available on the master node only.

Getting the Number of Nodes Present

To get the number of nodes acknowledged by the master node as being present in the cluster, invoke the `getPresentNodeCount` method. The `getPresentNodeCount` method takes no parameters, and returns an `int`. This information is available on the master node only.

Getting the Number of Switchovers Performed

To get the number of switchovers performed, invoke the `getSwitchOverCount` method. The `getSwitchOverCount` method takes no parameters, and returns an `int`. This information is available on the master node only.

Getting the Number of Currently Connected CMM Clients

To get the number of CMM clients currently connected, invoke the `getClientCount` method. The `getClientCount` method takes no parameters, and returns an `int`. This information is available on all nodes.

Getting the Number of Outstanding Requests

To get the number of requests currently outstanding, invoke the `getRequestCount` method. The `getRequestCount` method takes no parameters, and returns an `int`. This information is available on all nodes.

Getting the Lifetime of the CMM on a Node

To get the lifetime of the CMM on this node, expressed in the number of seconds since boot, invoke the `getCmmUpTime` method. The `getCmmUpTime` method takes no parameters, and returns an `int`. This information is available on all nodes.

Receiving Notifications

This chapter explains the NMA notification mechanism and describes the NMA notifications in detail.

This chapter contains the following sections:

- “Registering to Receive Notifications” on page 67
- “Registering to Receive SNMP Traps” on page 69

Registering to Receive Notifications

For information and instructions about writing and registering a notification listener, see the *Java Dynamic Management Kit 5.0 Tutorial*. Note that this information applies to the NMA only and is separate from the process of registering for notifications sent by the CMM. For information on these, see *Receiving and Handling Change Notifications in the Netra High Availability Suite 3.0 1/08 Foundation Services CMM Programming Guide*.

NhasCmmNotification

A `NhasCmmNotification` notification is sent by the `ClusterNodeMBean` when a node leaves or joins the cluster, or when a failover or switchover occurs. In addition to the standard notification information, this notification contains the following information:

type	The possible types for this notification are listed below.
source	The <code>ObjectName</code> of the node.

The notification type is one of the following:

MASTER	The node is now the cluster master node.
VICEMASTER	The node is now the cluster vice-master node.
IN_CLUSTER	The node is now part of the cluster.
OUT_OF_CLUSTER	The node is no longer part of the cluster.

NhasPmdMaxRetriesNotification

A `NhasPmdMaxRetriesNotification` notification is sent by the `PmdStatisticsMBean` when the maximum number of retries has been reached for a nametag. In addition to the standard notification information, this notification contains the following information:

source	The <code>PmdStatisticsMBean</code> <code>ObjectName</code>
maxRetry	The maximum retry number that was exceeded

The `MAX_RETRIES` field of this notification contains the name of the nametag that reached its maximum number of retries limit.

NhasPmdAttributeChangeNotification

A `javax.management.AttributeChangeNotification` is sent by the `PmdNameTagStatisticsMBean` when either of the following conditions is true:

- The number of allowed retries changes
- The retry counter is reset

In addition to the standard notification information, this notification contains the following information:

type	ATTRIBUTE_CHANGE
source	<code>PmdNameTagStatisticsMBean</code>
attributeName	Either <code>RetryCount</code> if the retry counter has been reset, or <code>MaxRetryCount</code> if the maximum number of retries allowed has changed
oldValue	The old number of retries allowed
newValue	The new number of retries allowed

The `nametag` field contains the name of the nametag that has been affected.

NhasPmdNewNameTagNotification

A `NhasPmdNewNameTagNotification` is sent whenever the Daemon Monitor creates a new nametag. This notification contains the field `NEW_NAMETAG`, which contains the name of the new nametag.

NhasPmdRemoveNameTagNotification

A `NhasPmdRemoveNameTagNotification` is sent whenever the Daemon Monitor removes a nametag from the collection. This notification contains the field `REMOVE_NAMETAG`, which contains the name of the nametag that was removed.

Registering to Receive SNMP Traps

For a Java DMK SNMP manager to receive SNMP traps, an implementation of the `SnmpTrapListener` class must be registered on the SNMP trap port.

[CODE EXAMPLE 10-1](#) shows an implementation of the `SnmpTrapListener` class. `SnmpTrapListener` is a code snippet that registers the `TrapListenerImpl` class as a trap listener on trap port `trapPort`. The trap listener listens for SNMPv1, SNMPv2 and SNMPv3 traps. The `TrapListenerImpl` class prints the details of all the traps it receives to the standard output.

CODE EXAMPLE 10-1 Implementation of the `SnmpTrapListener` Class

```
class TrapListenerImpl implements SnmpTrapListener {

    public void processSnmpTrapV1(SnmpPduTrap trap) {
        System.out.println("NOTE: TrapListenerImpl received trap V1:");
        System.out.println("\tGeneric " + trap.genericTrap);
        System.out.println("\tSpecific " + trap.specificTrap);
        System.out.println("\tTimeStamp " + trap.timeStamp);
        System.out.println("\tAgent adress " +
            trap.agentAddr.stringValue());
    }

    public void processSnmpTrapV2(SnmpPduRequest trap) {
        System.out.println("NOTE: TrapListenerImpl received trap V2:");
```

CODE EXAMPLE 10-1 Implementation of the `SnmpTrapListener` Class *(Continued)*

```
        SnmpPdu pdu = trap.getResponsePdu();
        System.out.println("\tFrom Address" +
            pdu.address.getHostAddress());
    }

    public void processSnmpTrapV3(SnmpScopedPduRequest trap) {
        System.out.println("NOTE: TrapListenerImpl received trap V3:");
        System.out.println("\tContextEngineId : " +
            SnmpEngineId.createEngineId(trap.contextEngineId));

        System.out.println("\tContextName : " + new
String(trap.contextName));
        System.out.println("\tVarBind list :");
        for (int i = 0; i < trap.varBindList.length; i++) {
            System.out.println("oid : " + trap.varBindList[i].getOid() +
                " val : " + trap.varBindList[i].getSnmpValue());
        }
    }
}
```

The following code snippet registers the `TrapListenerImpl` class as a trap listener on trap port `trapPort`.

CODE EXAMPLE 10-2 Registering the Trap Listener

```
System.out.println("Creating the trap listener on trapPort = " +
    trapPort);

// Create the Trap listener
TrapListenerImpl trapListener = new TrapListenerImpl();
SnmpEventReportDispatcher trapAgent = null;

try{
    trapAgent = new SnmpEventReportDispatcher(trapPort);
} catch (SocketException e) {
    System.out.println("ERROR Creating the trapListener " +
        e.getMessage());
}

// Start the Event Report dispatcher
new Thread(trapAgent).start();

// Add the trap listener on the Event report dispatcher
trapAgent.addTrapListener(trapListener);

System.out.println("Created the trap listener");
```


MBean Naming Conventions

This appendix describes the syntax of the MBean naming conventions.

All NMA MBeans are named according to conventions to allow easy location and manipulation. To identify an MBean, use an `ObjectName` in the format *domainID:MBeanName*. The *NodeID* is the unique ID given to each node in the cluster.

This appendix contains the following sections:

- [“Nodes and Services” on page 72](#)
- [“Cluster Membership Manager” on page 72](#)
- [“Reliable NFS Using a Shared-Disk Configuration” on page 72](#)
- [“Daemon Monitor” on page 73](#)
- [“CGTP” on page 73](#)

Nodes and Services

ClusterNodeMBean	nhas-object=cluster_node,node= <i>NodeID</i>
CmmMasterNodeMBean	nhas-object=cluster_node
NhasSwitchOverService	nhas-object=switchover,node= <i>NodeID</i>

Cluster Membership Manager

CmmMasterStatisticsMBean	nhas-object=cmm_stats
CmmStatisticsMBean	nhas-object=cmm_stats,node= <i>NodeID</i>

Reliable NFS Using SNDR

RnfsStatisticsMBean	nhas-object=rnfs_sndr_stats,node= <i>NodeID</i>
RnfsMasterStatisticsMBean	nhas-object=rnfs_sndr_stats
RnfsReplicatedSliceMBean	nhas-object=rnfs_sndr_stats,node= <i>NodeID</i> ,file= <i>SliceName</i>

Reliable NFS Using a Shared-Disk Configuration

SDMasterStatisticsMBean	nhas-object=rnfs_sd_stats,node= <i>NodeID</i>
also on the master:	nhas-object=rnfs_sd_stats

Daemon Monitor

PmdMasterStatisticsMBean	nhas-object=pmd_stats
PmdStatisticsMBean	nhas-object=pmd_stats,node= <i>NodeID</i>
PmdNameTagStatisticsMBean	nhas-object=pmd_stats,node= <i>NodeID</i> ,nametag= <i>tag</i>

CGTP

CgtpMasterMBean	nhas-object=cgtp_stats
CgtpMBean	nhas-object=cgtp_stats,node= <i>NodeID</i>
CgtpFilterMBean	nhas-object=cgtp_stats,node= <i>NodeID</i> ,cgtp=filtering
CgtpReliableLinkStatisticsMBean	nhas-object=cgtp_stats,node= <i>NodeID</i> ,alias= <i>AliasNumber0</i> ,address= <i>IPAddress</i>

Index

A

- access control
 - IP-based, 15
 - SNMPv1, 15
 - SNMPv2, 15
 - SNMPv3, 18
 - template configuration files, 14
- accessing statistics
 - CGTP, 35
 - CMM, 59
 - Daemon Monitor, 43
 - Reliable NFS, 51
- acl group, 16
- ACL *See* access control, 15
- authentication
 - SNMPv3, 19

B

- browsers, web
 - using to view the NMA, 2

C

- cascading service, 3
 - properties, 4
 - requirement to use same port, 4
- cghautil.jar file, 8
- CGTP
 - addresses, accessing, 36
 - master node statistics, accessing, 36
 - MBean naming conventions, 73
 - peer node statistics, accessing, 36
 - statistics, 35

- CgtpEmitterStatisticsMBean MBean, 37
- CgtpFilterMBean MBean, 37
- CgtpMasterMBean MBean, 36
- CgtpMBean MBean, 36
- CgtpReceiverStatisticsMBean MBean, 39
- CgtpReliableLinkStatisticsMBean MBean, 40
- class path
 - NMA, 8
 - remote managers, 11
- Cluster Membership Manager *See* CMM, 1
- ClusterNodeMBean MBean, 62
- CMM
 - master node statistics, accessing, 60
 - MBean naming conventions, 72
 - peer node statistics, accessing, 62
 - statistics, 59
 - status of peer nodes, 34
- CmmMasterNodeMBean MBean, 29
- CmmMasterStatisticsMBean MBean, 60
- CmmStatisticsMBean MBean, 64
- configuration files
 - access control, 14
 - paths, 7
 - SNMP manager examples, 21
- configuring
 - cascading service, 4
 - IP-based access control, 15
 - nma.acl file, 15
 - nma.security file, 19
 - nma.uacl file, 18
 - RFC 2573 configuration files, 21
 - SNMP agents, 13

- SNMP engine ID, 19
- SNMP managers, examples, 21
- SNMPv1 and SNMPv2 access control, 15
- SNMPv2 and SNMPv3 managers, 23
- SNMPv2 managers, 22
- SNMPv3 access control, 18
- SNMPv3 managers, 25
- SNMPv3 security, 19
 - user-based access control, 18
- connecting Java manager, 10

D

Daemon Monitor

- accessing nametags, 48
- master node statistics, accessing, 48
- MBean naming conventions, 73
- nametag change notifications, 69
- peer node statistics, accessing, 48
- processes, restarting, 34
- statistics, 43

daemons

- nametag change notifications, 69
- restart statistics, 49

dependencies, NMA, 8

E

encryption, SNMPv3, 19

external addresses, floating *See* floating external addresses

external managers

- Java managers, 9
- protocols used, 2
- SNMP managers
 - configuring, 21
 - overview, 13

F

failover

- notification, 67
- reconnecting Java manager, 10

files, configuration *See* configuration files

floating external addresses

- failover procedure with Java manager, 10
- Java manager, using with, 6
- SNMP, warning not to use with, 2
- switchover procedure with Java manager, 10
- using with switchOver method, 30

H

HTTP adaptor, configuring NMA for, 9

I

InetAddressAcl mechanism, 17

initiating a switchover, 29, 30

IP addresses, using for access control, 15

J

JAR files, 8

Java Archive files *See* JAR files, 8

Java Dynamic Management Kit

- external Java managers, 9
- external SNMP managers, 13
- heartbeat mechanism, using, 10

Java managers, external, 9

- class path, 8
- connecting and reconnecting, 10
- external floating addresses, using, 10
- HTTP, using, 9
- physical addresses, using, 10
- proxy MBeans, using, 11

jcmn.jar file, 8

jdmkrt.jar file, 8

JMX specification

- implementation by Java DMK, 9
- overview, 1

jmx.serial.form property, 8

jsnmpapi.jar file, 8

M

ma.jar file, 8

managers, external

- Java managers, 9
- protocols used, 2
- SNMP managers
 - configuring, 21
 - overview, 13

managing a cluster, 29

manipulating a cluster, 29

master view, 3, 4

MBeans

- naming conventions
 - CGTP, 73
 - Cluster Membership Manager, 72
 - Daemon Monitor, 73

- overview, 71
 - Reliable NFS, 72
- proxies, 11
- MIB file, default location of, 15

N

- nametags daemon, accessing, 48
- Netra HA Suite, checking if switchover is possible, 30
- nhasmib.txt file, 15
- nma.acl file, 15
 - SNMPv2 and SNMPv3 manager, configuring, 24
 - SNMPv2 manager, configuring, 22
 - SNMPv3 manager, configuring, 26
 - template, 8, 14
- nma.notifs.txt file, 7, 15
 - SNMPv2 and SNMPv3 manager, configuring, 24
 - SNMPv2 manager, configuring, 22
- nma.params.txt file, 7, 15
 - SNMPv2 and SNMPv3 manager, configuring, 24
 - SNMPv2 manager, configuring, 22
- nma.properties file, 7
 - access control properties, 14
 - cascading, specifying properties for, 4
 - configuring for HTTP adaptor, 9
 - disabling remote operations, 29
 - SNMP agents, configuring, 13
 - SNMP managers, configuring, 21
 - SNMPv2 and SNMPv3 manager, configuring, 24
 - SNMPv2 manager, configuring, 22
 - SNMPv3 manager, configuring, 26
- nma.security file, 7, 19
 - SNMPv2 and SNMPv3 manager, configuring, 24
 - SNMPv3 manager, configuring, 26
 - userEntry line, 20
- nma.targets.txt file, 8, 14
 - SNMPv2 and SNMPv3 manager, configuring, 24
 - SNMPv2 manager, configuring, 22
- nma.uacl file
 - SNMPv2 and SNMPv3 manager, configuring, 24
 - SNMPv3 manager, configuring, 26
 - template, 8, 14, 18
- node view
 - defined, 3
 - using HTTP protocol, 4
- notifications, 67
 - losses during failover or switchover, 10

- maximum number of retries, 68
- nametag changes, 69
- NhasCmmNotification, 67
- NhasPmdAttributeChangeNotification, 68
- NhasPmdMaxRetriesNotification, 68
- NhasPmdNewNameTagNotification, 69
- NhasPmdRemoveNameTagNotification, 69
- nodes joining or leaving cluster, 67
- retry changes, 68

P

- packages
 - SUNWjdr, 8
 - SUNWjsnmp, 8
 - SUNWnhmaj, 8
- packet duplication and filtering, measuring success of, 35
- performance, monitoring with CGTP statistics, 35
- physical addresses, switchover or failover with a Java manager, 10
- PmdMasterStatisticsMBean MBean, 48
- PmdNameTagStatisticsMBean MBean, 49
- PmdStatisticsMBean MBean, 48
- prerequisites for NMA, 8
- processes
 - monitoring, 47
 - restarting with Daemon Monitor, 34
- properties, configuring *See* nma.properties file
- protocols for communicating with the NMA, 2
- proxies, MBeans, 11
- proxies.jar file, 11

R

- reconnecting Java manager, 10
- Reliable NFS
 - master node statistics, accessing, 55
 - MBean naming conventions, 72
 - peer node statistics, accessing, 52
 - statistics, 51
- remote managers
 - class path, 11
 - Java managers, 9
 - SNMP managers, 13
- remote operations, disabling, 29
- restarting processes, Daemon Monitor, 34
- RFC standards

- RFC 2573
 - configuration files, editing, 21
 - defined, 14
- `rfc2573.jar` file, 8
- `rfc2573mgr.jar` file, 8
- `RnfsMasterReplicatedSliceMBean` MBean, 55
- `RnfsMasterStatisticsMBean` MBean, 53
- `RnfsReplicatedSliceMBean` MBean, 53
- `RnfsStatisticsMBean` MBean, 52

S

- security parameters, configuring *See* `nma.security` file
- security, configuring for SNMPv3, 19
- services, monitored by the NMA, 1
- Simple Network Management Protocol *See* SNMP, 2
- SNMP
 - access
 - policy, 16
 - rights, 16
 - access control lists, 15
 - acl group, 16
 - community names, 16
 - configuration files, 7, 14, 15
 - configuring
 - access control, 14
 - an agent, 13
 - an SNMPv2 and SNMPv3 manager, 23
 - an SNMPv2 manager, 22
 - an SNMPv3 manager, 25
 - context names, 19
 - engine ID, 19
 - floating external addresses, incompatibility with, 2
 - host list, 16
 - `InetAddressAcl` mechanism, 17
 - protocol adaptor, 13
 - registering trap listeners, 69
 - remote managers
 - overview, 13
 - registering at runtime, 21
 - trap group, 17
 - traps, receiving, 69
 - user-based
 - access control, 18
 - security model, 19
- `SnmpTrapListener` class, 69

- `SnmpV3AppliMibRegistration` class, 21
- software requirements, 8
- statistics, 66
 - CGTP, 35
 - CMM, 59
 - Daemon Monitor, 43
 - Reliable NFS, 51
- SUNWjdrt package, 8
- SUNWjsnmp package, 8
- SUNWnhmaj package, 8
- switchover, 65
 - checking
 - if switchover is possible, 30
 - success, 34
 - initiating, 29, 30
 - notification, 67
 - reconnecting Java manager, 10

T

- trap group, 17

U

- user-based
 - access control, 18
 - security model, 19
- USM *See* user-based security model

V

- views
 - master view
 - overview, 3
 - using HTTP protocol, 4
 - node view, 3

W

- web browsers, using to view the NMA, 2