HP OpenView Operations

HTTPS Agent
Concepts and Configuration Guide

Software Version: A.08.10 & A.08.20
Edition 7

For the HP-UX and Sun Solaris Management Server Operating Systems

Manufacturing Part Number: B7491-90068

August 2006

Legal Notices

Warranty.

Hewlett-Packard makes no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

This product includes software written by Info-ZIP (http://www.info-zip.org/license.html)

This product includes software written by Tim Hudson (tjh@cryptsoft.com)

Trademark Notices.

Adobe® is a trademark of Adobe Systems Incorporated.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

UNIX® is a registered trademark of the Open Group.

Windows® and MS Windows® are U.S. registered trademarks of Microsoft Corporation.

1. OVO HTTPS Agent Overview

2. Concepts of HTTPS Communication

3. Security Concepts

5. Working with HTTPS Managed Nodes

6. Working with Certificates

7. Virtual Nodes in OVO

8. Proxies

9. Managing HTTPS Agents on DHCP Client Systems

10. MOM Environments

F. OVO 8 Quick Start Guide

Printing History

The printing date and part number of the manual indicate the edition of the manual. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The part number of the manual will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

Table 1

First Edition:	June 2004
Second Edition:	December 2004
Third Edition:	March 2005
Fourth Edition:	May 2005
Fifth Edition:	June 2005
Sixth Edition:	October 2005
Seventh Edition:	August 2006

Conventions

The following typographical conventions are used in this manual.

Table 2 Typographical Conventions

Font	Meaning	Example
Italic	Book or manual titles, and man page names	Refer to the *OVO Administrator's Reference* and the *opc(1M)* manpage for more information.
	Emphasis	You *must* follow these steps.
	Variable that you must supply when entering a command	At the prompt, enter **`rlogin`** ***`username`***.
	Parameters to a function	The *`oper_name`* parameter returns an integer response.
Bold	New terms	The **HTTPS agent** observes...
`Computer`	Text and other items on the computer screen	The following system message appears: `Are you sure you want to remove current group?`
	Command names	Use the `grep` command ...
	Function names	Use the `opc_connect()` function to connect ...
	File and directory names	`/opt/OV/bin/OpC/`
	Process names	Check to see if `opcmona` is running.
	Window/dialog box names	In the `Add Logfile` window ...
	Menu name followed by a colon (`:`) means that you select the menu, then the item. When the item is followed by an arrow (`->`), a cascading menu follows.	Select `Actions: Filtering -> All Active Messages` from the menu bar.

Table 2 Typographical Conventions (Continued)

Font	Meaning	Example
Computer Bold	Text that you enter	At the prompt, enter **ls -l**
Keycap	Keyboard keys	Press **Return**.
[Button]	Buttons in the user interface	Click [OK].

OVO Documentation Map

HP OpenView Operations (OVO) provides a set of manuals and online help that help you to use the product and to understand the concepts underlying the product. This section describes what information is available and where you can find it.

Electronic Versions of the Manuals

All the manuals are available as Adobe Portable Document Format (PDF) files in the documentation directory on the OVO product CD-ROM.

With the exception of the *OVO Software Release Notes*, all the manuals are also available in the following OVO web-server directory:

`http://<management_server>:3443/ITO_DOC/<lang>/manuals/*.pdf`

In this URL, <*management_server*> is the fully-qualified hostname of your management server, and <*lang*> stands for your system language, for example, `C` for the English environment and `japanese` for the Japanese environment.

Alternatively, you can download the manuals from the following website:

`http://ovweb.external.hp.com/lpe/doc_serv`

Watch this website regularly for the latest edition of the OVO Software Release Notes, which gets updated every 2-3 months with the latest news such as additionally supported OS versions, latest patches and so on.

OVO Manuals

This section provides an overview of the OVO manuals and their contents.

Table 3 OVO Manuals

Manual	Description	Media
OVO Installation Guide for the Management Server	Designed for administrators who install OVO software on the management server and perform the initial configuration. This manual describes: • Software and hardware requirements • Software installation and de-installation instructions • Configuration defaults	Hardcopy PDF
OVO Concepts Guide	Provides you with an understanding of OVO on two levels. As an operator, you learn about the basic structure of OVO. As an administrator, you gain an insight into the setup and configuration of OVO in your own environment.	Hardcopy PDF
OVO Administrator's Reference	Designed for administrators who install OVO on the managed nodes and are responsible for OVO administration and troubleshooting. Contains conceptual and general information about the OVO DCE/NCS-based managed nodes.	PDF only
OVO DCE Agent Concepts and Configuration Guide	Provides platform-specific information about each DCE/NCS-based managed-node platform.	PDF only
OVO HTTPS Agent Concepts and Configuration Guide	Provides platform-specific information about each HTTPS-based managed-node platform.	PDF only
OVO Reporting and Database Schema	Provides a detailed description of the OVO database tables, as well as examples for generating reports from the OVO database.	PDF only
OVO Entity Relationship Diagrams	Provides you with an overview of the relationships between the tables and the OVO database.	PDF only

Table 3 OVO Manuals (Continued)

Manual	Description	Media
OVO Java GUI Operator's Guide	Provides you with a detailed description of the OVO Java-based operator GUI and the Service Navigator. This manual contains detailed information about general OVO and Service Navigator concepts and tasks for OVO operators, as well as reference and troubleshooting information.	PDF only
Service Navigator Concepts and Configuration Guide	Provides information for administrators who are responsible for installing, configuring, maintaining, and troubleshooting the HP OpenView Service Navigator. This manual also contains a high-level overview of the concepts behind service management.	Hardcopy PDF
OVO Software Release Notes	Describes new features and helps you: • Compare features of the current software with features of previous versions. • Determine system and software compatibility. • Solve known problems.	PDF only
OVO Supplementary Guide to MPE/iX Templates	Describes the message source templates that are available for the MPE/iX managed nodes. This guide is not available for OVO on Solaris.	PDF only
Managing Your Network with HP OpenView Network Node Manager	Designed for administrators and operators. This manual describes the basic functionality of the HP OpenView Network Node Manager, which is an embedded part of OVO.	Hardcopy PDF
OVO Database Tuning	This ASCII file is located on the OVO management server at the following location: `/opt/OV/ReleaseNotes/opc_db.tuning`	ASCII

Additional OVO-related Products

This section provides an overview of the OVO-related manuals and their contents.

Table 4 Additional OVO-related Manuals

Manual	Description	Media
HP OpenView Operations for UNIX Developer's Toolkit If you purchase the HP OpenView Operations for UNIX Developer's Toolkit, you receive the full OVO documentation set, as well as the following manuals:		
OVO Application Integration Guide	Suggests several ways in which external applications can be integrated into OVO.	Hardcopy PDF
OVO Developer's Reference	Provides an overview of all the available application programming interfaces (APIs).	Hardcopy PDF
HP OpenView Event Correlation Designer for NNM and OVO If you purchase HP OpenView Event Correlation Designer for NNM and OVO, you receive the following additional documentation. Note that HP OpenView Event Correlation Composer is an integral part of NNM and OVO. OV Composer usage in the OVO context is described in the OS-SPI documentation.		
HP OpenView ECS Configuring Circuits for NNM and OVO	Explains how to use the ECS Designer product in the NNM and OVO environments.	Hardcopy PDF

OVO Online Information

The following information is available online.

Table 5 OVO Online Information

Online Information	Description
HP OpenView Operations Administrator's Guide to Online Information	Context-sensitive help system contains detailed help for each window of the OVO administrator Motif GUI, as well as step-by-step instructions for performing administrative tasks.
HP OpenView Operations Operator's Guide to Online Information	Context-sensitive help system contains detailed help for each window of the OVO operator Motif GUI, as well as step-by-step instructions for operator tasks.
HP OpenView Operations Java GUI Online Information	HTML-based help system for the OVO Java-based operator GUI and Service Navigator. This help system contains detailed information about general OVO and Service Navigator concepts and tasks for OVO operators, as well as reference and troubleshooting information.
HP OpenView Operations Man Pages	Manual pages available online for OVO. These manual pages are also available in HTML format. To access these pages, go to the following location (URL) with your web browser: `http://<management_server>:3443/ITO_MAN` In this URL, the variable <*management_server*> is the fully-qualified hostname of your management server. Note that the man pages for the OVO HTTPS-agent are installed on each managed node.

About OVO Online Help

This preface describes online documentation for the HP OpenView Operations (OVO) Motif and the Java operator graphical user interfaces (GUIs).

Online Help for the Motif GUI

Online information for the HP OpenView Operations (OVO) Motif graphical user interface (GUI) consists of two separate volumes, one for operators and one for administrators. In the operator's volume you will find the HP OpenView OVO Quick Start, describing the main operator windows.

Types of Online Help

The operator and administrator volumes include the following types of online help:

- **Task Information**

  Information you need to perform tasks, whether you are an operator or an administrator.

- **Icon Information**

  Popup menus and reference information about OVO icons. You access this information with a right-click of your mouse button.

- **Error Information**

  Information about errors displayed in the `OVO Error Information` window. You can access context-sensitive help when an error occurs. Or you can use the number provided in an error message to perform a keyword search within the help system.

- **Search Utility**

  Index search utility that takes you directly to topics by name.

- **Glossary**

  Glossary of OVO terminology.

- **Help Instructions**

  Instructions about the online help system itself for new users.

- **Printing Facility**

  Printing facility, which enables you to print any or all topics in the help system. (An HP LaserJet printer or a compatible printer device is required to print graphics.)

To Access Online Help

You can access the help system in any of the following ways:

- **F1 Key**

  Press **F1** while the cursor is in any active text field or on any active button.

- **Help Button**

  Click `[Help]` at the bottom of any window.

- **Help Menu**

  Open the drop-down `Help` menu from the menu bar.

- **Right Mouse Click**

  Click a symbol, then right-click the mouse button to access the `Help` menu.

You can then select task lists, which are arranged by activity, or window and field lists. You can access any topic in the help volume from every help screen. Hyperlinks provide related information on other help topics.

You can also access context-sensitive help in the `Message Browser` and `Message Source Templates` window. After selecting `Help: On Context` from the menu, the cursor changes into a question mark, which you can then position over the area about which you want help. When you click the mouse button, the corresponding help page is displayed in its help window.

Online Help for the Java GUI and Service Navigator

The online help for the HP OpenView Operations (OVO) Java graphical user interface (GUI), including Service Navigator, helps operators to become familiar with and use the OVO product.

Types of Online Help

The online help for the OVO Java GUI includes the following information:

- **Tasks**

  Step-by-step instructions.

- **Concepts**

  Introduction to the key concepts and features.

- **References**

  Detailed information about the product.

- **Troubleshooting**

  Solutions to common problems you might encounter while using the product.

- **Index**

  Alphabetized list of topics to help you find the information you need, quickly and easily.

Viewing a Topic

To view any topic, open a folder in the left frame of the online documentation window, then click the topic title. Hyperlinks provide access to related help topics.

Accessing the Online Help

To access the help system, select `Help: Contents` from the menu bar of the Java GUI. A web browser opens and displays the help contents.

NOTE To access online help for the Java GUI, you must first configure OVO to use your preferred browser.

Support

Please visit the HP OpenView support web site at:

http://www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valuable support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and log in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

http://www.managementsoftware.hp.com/passport-registration.html

1 OVO HTTPS Agent Overview

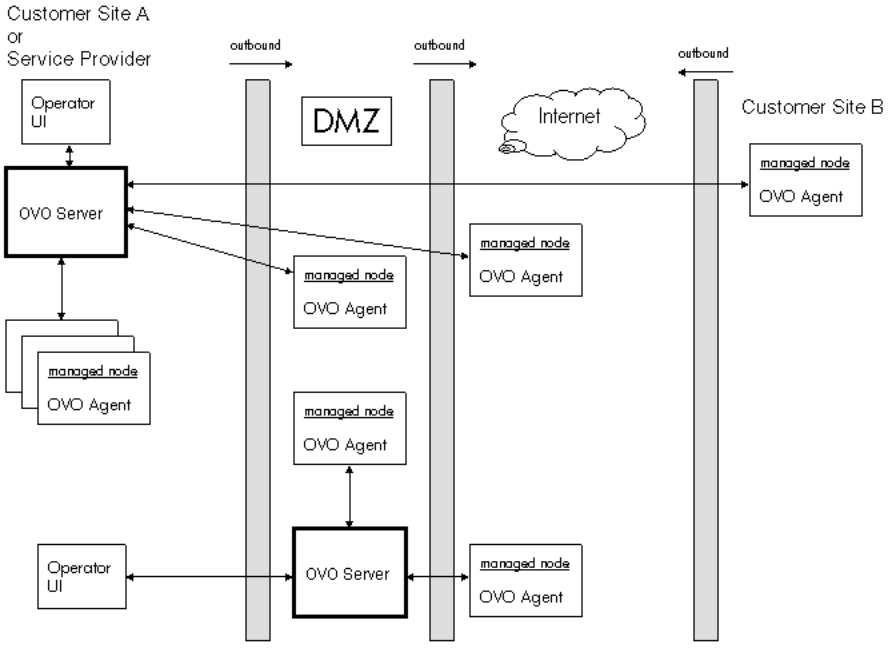
Introduction

HTTPS agent software provides highly secure communication between OVO 8.x management servers and their managed nodes. HTTPS agents are generally used and administered in the same way as DCE-based agents. Applications are launched in the same way. Command line interfaces, such as `opcragt`, can be used for all managed nodes. All functionality that is available with DCE-based agents is also available with HTTPS agents unless explicitly stated otherwise.

Policies for HTTPS agents are created, assigned and deployed in a similar way as templates for DCE-based agents. For example, heartbeat polling of nodes results in the same type of status messages and are displayed in a very similar way in the message browser. Figure 1-1 illustrates a typical environment managed by HP OpenView Operations.

The HTTPS agents have many advantages and benefits over DCE-based agents. These are described in the following chapters.

Figure 1-1 A Typical OVO Managed Environment



HTTPS-based communication provides you with the following major advantages:

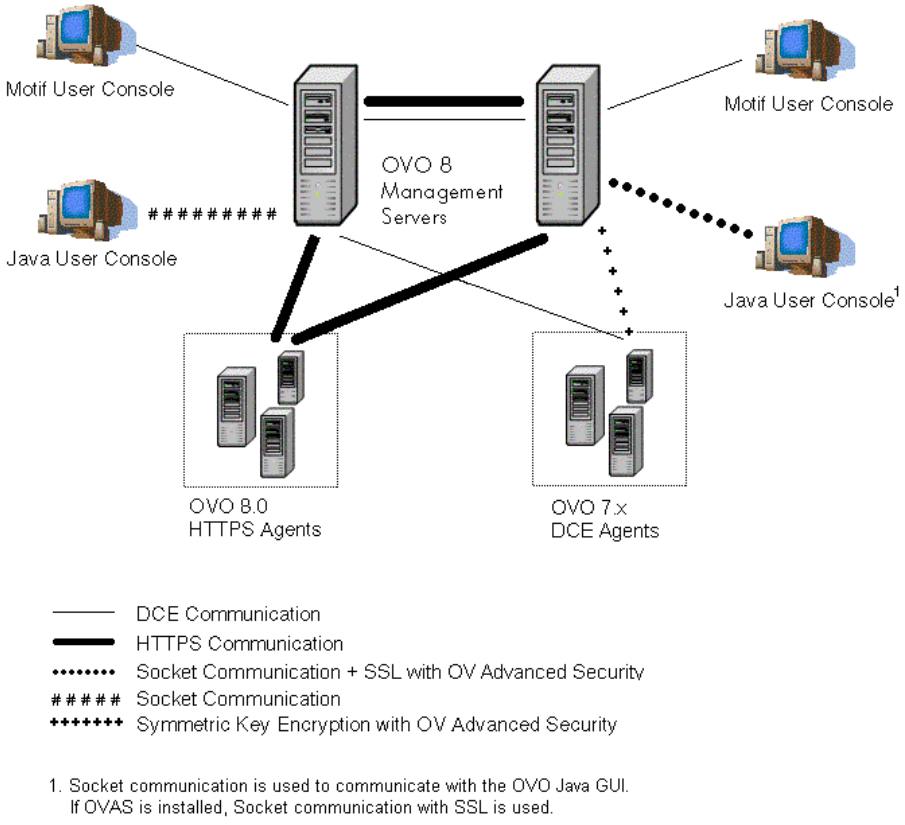
- Simple management through firewalls with configurable, single-port, secure communication using, open, HTTPS-based communication techniques. Restrict outside access to dedicated HTTP proxies and reduce port usage by multiplexing over HTTP proxies.
- Out-of-the-box Internet Secure Communication using SSL/PKI encryption with server and client certificates for authentication.
- Communication is based on standard Web technologies (HTTP, SOAP, Proxies, SSL, …), available in every environment today, and familiar to every IT administrator.
- OVO message format based on XML and SOAP used for message security from the HTTPS agent to the OVO management server.
- IP independence/dynamic IP (DHCP). Managed nodes can be identified by their unique `OvCoreID` and not necessarily by their IP addresses.
- No need for additional investments (training, additional software such as DCE).
- OpenView standard control and deployment mechanism.
- OpenView standard logging capability.
- OpenView standard tracing capability.

Additional advantages available with OVO 8 include:

- Duplicate IP support will be available for both HTTPS and DCE agents.

Figure 1-2 illustrates an example of the different communication types in OVO.

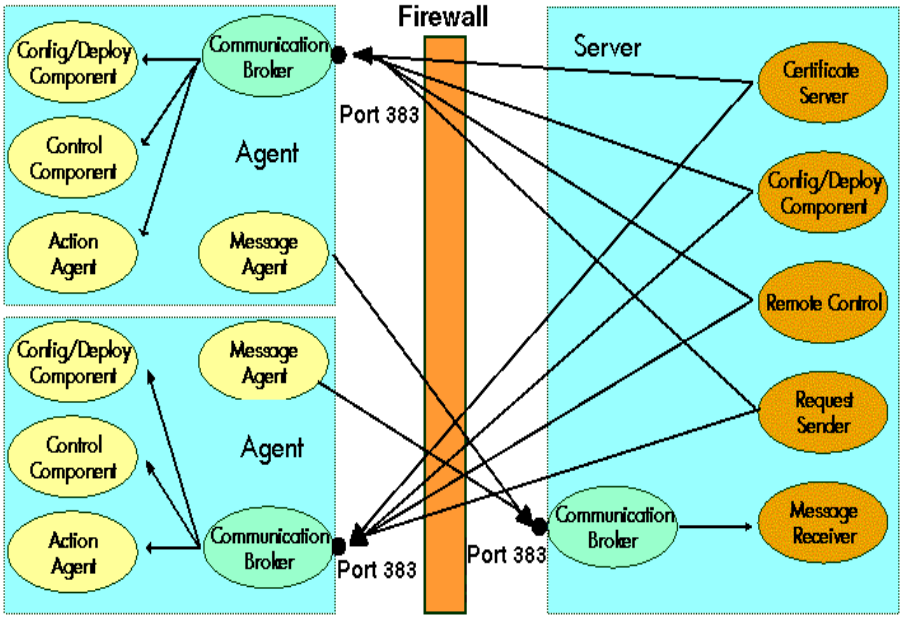
Figure 1-2 Communication Overview in HP OpenView Operations



HP OpenView Operations HTTPS Agent Architecture

The following graphics illustrate the architecture of the HTTPS communication in OVO.

Figure 1-3 **HTTPS Agent Components and Responsibilities**



HTTPS Agent Platforms Supported with OVO 8[1]

- AIX
- HP-UX (PA-RISC)

  HP-UX (Itanium IA64)
- Linux (Intel x86)
  - Debian
  - Mandrake
  - RedFlag Professional Server
  - Red Hat
  - SuSE and SuSE Enterprise Server
  - Turbolinux Server and Turbolinux Enterprise Server (Japanese Environments only)
- Microsoft Windows (Intel x86)
- Sun Solaris (SPARC)

1. For the most current list of supported managed node platforms, refer to the latest version of the OVO Release Notes. This document is available in pdf format from: `http://ovweb.external.hp.com/lpe/doc_serv/` under *operations for UNIX*, version *8.x*. Select the operating system of your management server and all the related documentation will be listed.

Organization of HTTPS Managed Nodes

Generic Directory Structure on OVO Managed Nodes

The files associated with the HTTPS agent are found in the following directory structures:

- **<OVInstallDir>**

HP-UX, Solaris, Linux	`/opt/OV`
AIX	`/usr/lpp/OV`
Windows	`<ProgramFilesDir>\HP OpenView`

  This directory contains static files that are installed from the product media and never change, for example, executables. Since these files never change, you can mount `<OVInstallDir>` as “read-only” for increased security in highly sensitive environments. It is not necessary to back up these files as they can be re-installed from the product media.

  All other files change during operation and must be backed up regularly.

- **<OVDataDir>**

HP-UX, Solaris, Linux, AIX	`/var/opt/OV`
Windows	`<ProgramFiles>\HP OpenView\data`

  This directory contains configuration and runtime data files that are used only on the local system. The most important directory contains the instrumentation files such as actions, commands and monitors:

  `<OvDataDir>``/bin/instrumentation`

The *`<OvInstallDir>`*`/newconfig/inventory/*.xml` files contain a list of all the directories and files that are created and installed with the agent software.

OVO Agent User and the opc_op Accounts

The OVO agent runs by default as `root` on UNIX, and as `system` on Windows. It is assumed that the OVO agent account already exists on a managed node, when the agent is installed. At OVO agent installation time, an additional minimal-rights account is created - the `opc_op` account. Its main purpose is to execute actions with minimal rights.

Table 1-1 shows the OVO agent accounts on UNIX managed nodes.

Table 1-1 OVO Accounts on UNIX Managed Nodes

Account Characteristics	OVO Agent Account	Additional Minimal-rights Account
User Name	`root`	`opc_op`[a]
Password	Defined for user `root`	Defined during installation
Group	`sys`	`opcgrp`
Login Shell	Korn Shell (`/bin/ksh`)	Korn Shell (`/bin/ksh`)
home directory	`/.root`	`/home/opc_op`

a. It is not possible to log into the system directly using the `opc_op` account (enter `*` in `/etc/passwd`).

NOTE OVO software on UNIX managed nodes systems can be configured to run under a user that does not have full root permissions, often referred to as “running as non-root”. For details, refer“Agents Running Under Alternative Users” on page 76.

If the managed node is a Network Information Service (NIS or NIS+) client, you must add the `opc_op` account as a member of the `opcgrp` group on the NIS server before installing the OVO software on a managed node. This ensures that the `opc_op` account is used by OVO and is consistent on all systems.

If you do not add the opc_op account on the NIS server, the installation creates a user opc_op with the group opcgrp locally on the managed node.

Table 1-2 shows the OVO agent accounts on Windows managed nodes.

Table 1-2 OVO Agent Accounts on Windows Managed Nodes

Account Characteristics	OVO Agent Account
User Name	Built-in system
Password	N.A.
Rights	Local Administrator

UNIX System Resources

OVO applies changes in the following system resource files:

/etc/passwd	Default OVO operator entry.
/etc/group	Default OVO operator group entry.
<*BootDir*>/OVCtrl	OVO startup and shutdown.
<*BootDir*>/TrcSrv	OpenView Tracing start and stop.

<*BootDir*>:

AIX	/etc/rc.d
HP-UX	/sbin/init.d
Linux	/etc/rc.d/init.d
Solaris	/etc/init.d

NOTE If you are working with Network Information Services (NIS or "yellow pages"), you should adapt the user registration accordingly.

Symbolic links <*BootDir*>/OVCtrl and <*BootDir*>OVTrcSrv to define the start and stop sequences at boot time of the https agent:

HP-UX

Start Trace Daemon.	/sbin/rc3.d/S900VTrcSrv
Start OVO Agent.	/sbin/rc3.d/S920OVCtrl

Stop Agent.	`/sbin/rc2.d/K010VCtrl`
Stop Trace Server.	`/sbin/rc2.d/K020VTrcSrv`
AIX	
Start Trace Daemon	`/etc/rc.d/rc2.d/S90OVTrcSrv`
Start OVO Agent	`/etc/rc.d/rc2.d/S92OVCtrl`
Stop Agent	`/etc/rc.d/rc<num>.d/K02OVTrcSrv`
Stop Trace Server	`/etc/rc.d/rc<num>.d/K01OVCtrl`
	Where <num> = 3, 4, 5, ...8, 9.
Solaris	
Start Trace Daemon	`/etc/rc3.d/S90OVTrcSrv`
Start OVO Agent	`/etc/rc3.d/S92OVCtrl`
Stop Agent	`/etc/rc<key>.d/K01OVCtrl`
Stop Trace Server	`/etc/rc<key>.d/K01OVTrcSrv`
	`/etc/rc<key>.d/K02OVTrcSrv`
	Where <key> = 0 | 1 | 2 | S.
Linux	
Start Trace Daemon	`/etc/rc.d/rc<num>.d/S90OVTrcSrv`
Start OVO Agent	`/etc/rc.d/rc<num>.d/S92OVCtrl`
	<num> = 3 | 4 | 5
Stop OVO Agent	`/etc/rc.d/rc<num>.d/K01OVCtrl`
Stop Trace Server	/etc/rc.d/rc<num>.d/K02OVTrcSrv
	<num> = 0 | 1 | 2 | 6

Windows System Resources

User Environment Variables

No System level environment variables are changed. Only the following following user level variables are modified.

OVO sets the following user environment variables, which can be used in scripts, for example, when setting up automatic actions in policies:

Table 1-3 Windows User Environment Variables

Variable	Location and Explanation
`OvAgentDir`	The installation directory for the Windows agent.
	`C:\Program Files\HP OpenView\Installed Packages\{790c06b4-844e-11d2-972b-080009ef8c2a}`
`OvDataDir`	The directory for HP OpenView configuration and runtime data files.
	`C:\Program Files\HP OpenView\Data\`
`OvInstallDir`	The installation directory for the HP OpenView.
	`C:\Program Files\HP OpenView\`
`OvPerlBin`	The absolute path to the Perl interpreter.
	`C:\Program Files\HP OpenView\Installed Packages\{790c06b4-844e-11d2-972b-080009ef8c2a}\bin\perl.exe`

Starting and Stopping the Windows Agent

`ovcd` starts the components after reboot if the value of `START_ON_BOOT` is set to `true` in the `[ctrl]` namespace (common to both Windows and UNIX platforms). To set this value, enter the following command:

```
ovconfchg -ns ctrl -set START_ON_BOOT true
```

The `Control` service startup should be configured to startup automatically after reboot. To configure a service, open the `Services` window:

Click **Start** →**Control Panel** →**Administrative Tools** →**Services**

Open the properties window of the `Control` service, and select **Automatic** from the **Startup type** drop down menu.

Registry Keys

OVO inserts several keys in the Windows Registry.

The keys and their associated values can be viewed with the Registry Editor, using the following command:

`%SystemRoot%\System32\regedt32.exe`

There are many registry changes that happen during the installation of the agent. They can be generally classified as follows:

- Registry keys which contain configuration settings for the agent software that is installed added under:

  `HKLM\SOFTWARE\HEWLETT-PACKARD`

- Registry keys added when OpenView registers a Windows service under the name `HP ITO Agent`.
- Keys added to register `.dll` and `.exe` files.
- Keys related to de-installing agent software.
- Keys added by OpenView core components.

For example, the Windows Registry includes the following keys for OVO:

❑ ***<OvInstallDir>***

`HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\OpenView`

Value Name: `InstallDir`

Value Type: `string`

❑ ***<OvDataDir>***

`HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\OpenView\data`

Value Name: `DataDir`

Value Type: `string`

If on a domain controller, the Windows Registry Editor also shows:

```
HKEY_LOCAL_MACHINE\SYSTEM\Current ControlSet\Services\
HP ITO Installation Server
```

Path Variables

Following values are added to the PATH variable.

```
C:\Program Files\HP OpenView\Installed Packages\
{790c06b4-844e-11d2-972b-080009ef8c2a}\bin;
C:\Program Files\HP OpenView\Installed Packages\
{790c06b4-844e-11d2-972b-080009ef8c2a}\bin\OpC
```

Libraries

HTTPS agent libraries can be identified by the extensions, such as `.dll`, `.sl`, and `.so`.

The library names for shared Openview components include:

```
libOvXpl
libOvBbc
libOvSecCore
libOvSecCm
libOvCtrl
libOvCtrlUtils
libOvConf
libOvDepl
libjopcagtbase
libjopcagtmsg
```

The OVO agent-specific libraries include (on UNIX and Windows):

```
libopc_r
libjopcagtbase
libjopcagtmsg
```

The OVO-agent-specific libraries on UNIX include:

```
libnsp
```

The OVO-agent-specific libraries on Windows include:

```
opcapi.dll
opcauth.dll
OpCWbemInterceptor.dll
pdh.dll
```

As an example, the platform-specific filenames for the `libopc_r` library are listed below:

AIX	`libopc_r.so`
HP-UX 11.0 and 11.11	`libopc_r.sl`
HP-UX Itanium	`libopc_r.so`
Linux	`libopc_r.so`
Solaris	`libopc_r.so`
Windows	`libopc.dll` `pdh.dll`

Include Files

On supported managed node platforms, use the appropriate include file:

AIX	`/usr/lpp/OV/include/opcapi.h`
HP-UX	`/opt/OV/include/opcapi.h`
Linux	`/opt/OV/include/opcapi.h`
Solaris	`/opt/OV/include/opcapi.h`
Windows	`\usr\OV\include\opcapi.h`

An example of how the API functions are used is available in the following file on the management server:

`/opt/OV/OpC/examples/progs/opcapitest.c`

Makefiles

The following directory on the management server contains the makefiles for building executables:

```
/opt/OV/OpC/examples/progs
```

The build an executable with correct compile and link options, use the following makefiles:

AIX	`Makef.aix`
HP-UX	`Makef.hpux11`
	`Makef.hpuxIA32`
Linux	`Makef.linux`
Solaris	`Makef.solaris`
Windows	To built an executable, use Microsoft Developer Studio 6.0 or higher.

For more information about the managed node makefile, see the ReadMe file:

```
/opt/OV/OpC/examples/progs/README
```

HTTPS Communication Administration Commands in OVO

HTTPS Communication can be controlled using the following commands.

On the OVO Management Server and Managed Nodes:

- **ovcoreid** (OpenView Unique System Identifier)

  The `ovcoreid` command is used to display existing `OvCoreId` value and, in addition, create and set new `OvCoreId` values on the local system.

  For details of how to use this tool, refer to the `ovcoreid(1)` man page.

- **ovc** (OpenView Process Control)

  `ovc` controls starting and stopping, event notification, and status reporting of all components registered with the OpenView Control service, `ovcd`. A component can be a server process, an agent (for example, the Performance Agent or the Discovery Agent), an event interceptor, or an application delivered by an integrator.

  For details of how to use this tool, refer to the `ovc(1)` man page.

- **bbcutil**

  The `bbcutil` command is used to control the OV Communication Broker.

  For syntax information and details of how to use this tool, refer to the `bbcutil(1)` man page.

- **ovconfget**

  Installed OpenView components have associated configuration settings files that contain one or more namespaces and apply system wide or for a specified High Availability Resource Group. A namespace is a group of configuration settings that belong to a component. All configurations specified in the settings files are duplicated in the `settings.dat` configuration database.

For each specified namespace, `ovconfget` returns the specified attribute or attributes and writes them to `stdout`. Used without arguments, `ovconfget` writes all attributes in all namespaces to `stdout`.

For details of how to use this tool, refer to the `ovconfget(1)` man page.

- **ovconfchg**

  Installed OpenView components have associated configuration settings files that contain one or more namespaces. A namespace is a group of configuration settings that belong to a component.

  `ovconfchg` manipulates the settings in either the system-wide configuration file or the configuration file for the specified `High Availability Resource Group`, updates the configuration database, and triggers notification scripts.

  For details of how to use this tool, refer to the `ovconfchg(1)` man page.

- **ovpolicy**

  `ovpolicy` manages local policies and templates. A policy or template is a set of one or more specifications, rules and other information that help automate network, system, service, and process management. Policies and templates can be deployed to managed systems, providing consistent, automated administration across the network. Policies and templates can be grouped into categories. Each category can have one or more policies. Each category can also have one or more attributes, an attribute being a name value pair.

  You use `ovpolicy` to install, remove, enable, and disable local policies and templates. For details of how to use this tool, refer to the `ovpolicy(1)` man page.

On Managed Nodes:

- **ovcert**

  The `ovcert` command is used to manage certificates on an HTTPS node through the Certificate Client. You can execute tasks such as initiating a new certificate request to the Certificate Server, adding managed node certificates and importing the private keys, adding certificates to the trusted root certificates, and checking the certificate status.

  For details of how to use this tool, refer to the `ovcert(1)` man page.

On the OVO Management Server:

- **opccsacm** (Certificate Server Adapter Control Manager)

  The `opccsacm` command is used to issue new node certificates and installation keys manually on the HP OpenView server. It also modifies the OVO database to reflect the changes made by certificate management actions.

  For details of how to use this tool, refer to the `opccsacm(1m)` man page.

- **opccsa** (Certificate Server Adapter)

  The `opccsa` command is used to list the pending certificate requests, map certificate requests to target nodes from the OVO database, grant, deny and delete specified certificate requests.

  For details of how to use this tool, refer to the `opccsa(1m)` man page.

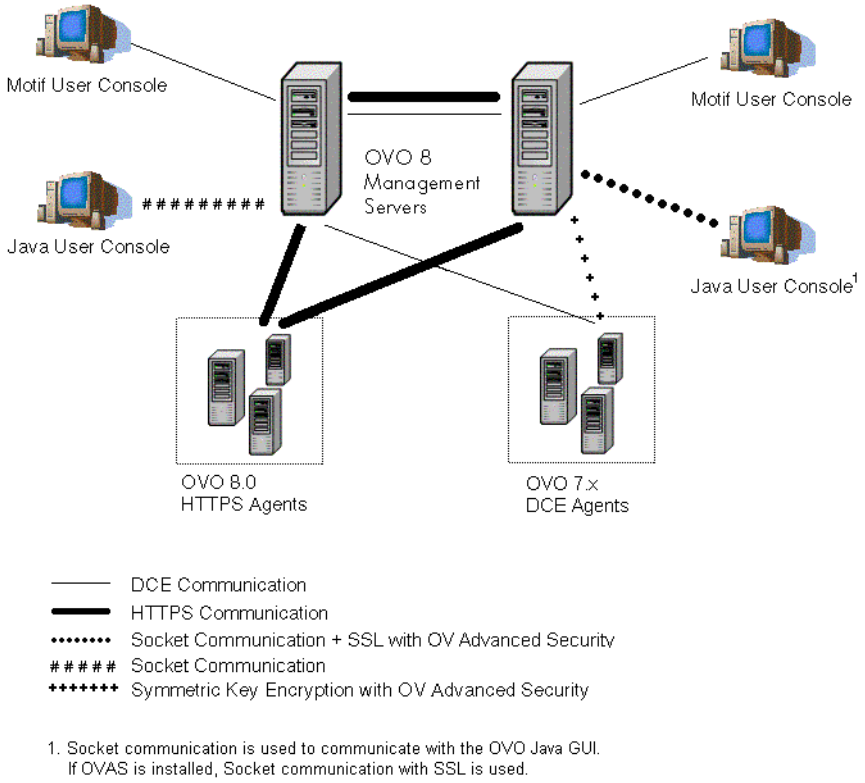
2 Concepts of HTTPS Communication

HTTPS Communication in OVO

HTTPS 1.1 based communications is the latest communication technology used by HP OpenView products and allows applications to exchange data between heterogeneous systems.

OpenView products using HTTPS communication can easily communicate with each other, as well as with other industry-standard products. It is also now easier to create new products that can communicate with existing products on your network and easily integrate with your firewalls and HTTP-proxies. Figure 2-1 illustrates an example of HTTPS communication.

Figure 2-1 Communication Overview in HP OpenView Operations



1. Socket communication is used to communicate with the OVO Java GUI. If OVAS is installed, Socket communication with SSL is used.

Advantages

HTTPS communication provides the following major advantages:

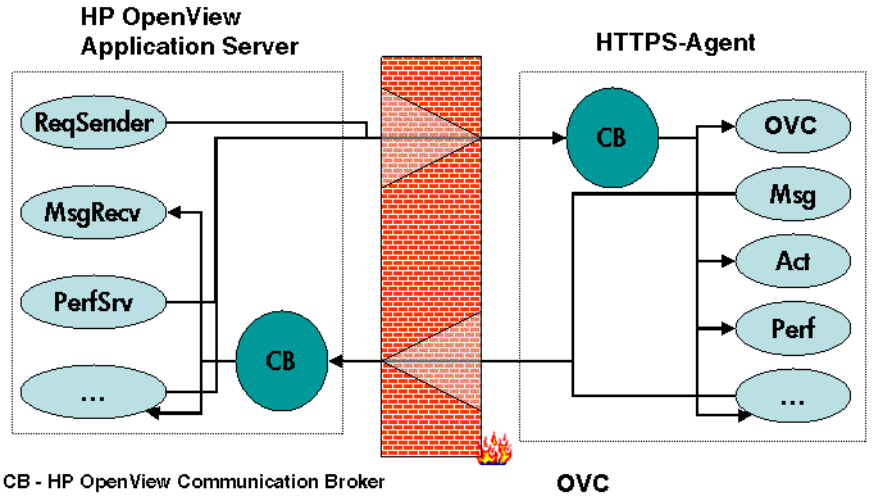
- Firewall Friendly
- Secure
- Open
- Scalable

Firewall Friendly

More and more organizations need to cross firewalls in a safe, secure, and easily manageable way. Most of these organizations are very familiar and comfortable with HTTP, HTTP proxies, and firewalls. Their IT environments are already configured to allow communication through HTTP proxies and firewalls. By focusing on technology that is already a part of most IT infrastructures, it helps you to be more efficient and effective, without the need for new training. The end result reduces support and maintenance costs, while simultaneously creating a highly secure environment without significant effort.

Figure 2-2 illustrates crossing a firewall using HTTPS-communication.

Figure 2-2 Crossing a Firewall with HTTPS Communication



Secure

HP OpenView's HTTPS communication is based on the TCP/IP protocol, the industry standard for reliable networking. Using the Secure Socket Layer (SSL) protocol, HTTPS communication uses authentication to validate who can access data, and encryption to secure data exchange. Now that businesses are sending and receiving more transactions across the Internet and private intranets than ever before, security and authentication assume an especially important role.

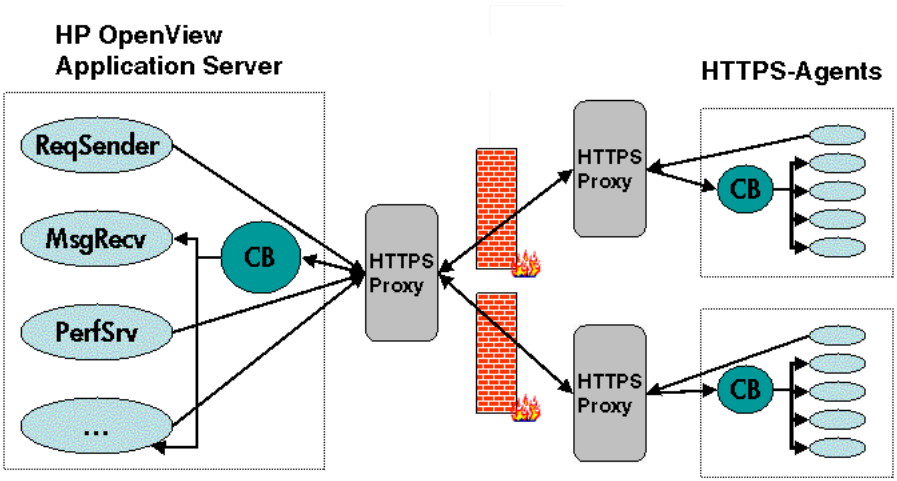
HP OpenView's HTTPS communication meets this goal through established industry standards. HTTP protocol and SSL encryption and authentication insure data integrity and privacy. By default, data is compressed, ensuring that data is not transmitted in clear text format, even for non-SSL connections.

In addition:

- All remote messages and requests arrive through the Communication Broker, providing a single port entry to the node.
- Restricted bind port range can be used when configuring firewalls.
- Configure one or more standard HTTP proxies to cross a firewall or reach a remote system when sending messages, files or objects.

Figure 2-3 illustrates crossing firewalls using standard HTTP proxies.

Figure 2-3 Crossing a Firewall using External HTTPS Proxies



To work with HTTPS communication and proxies, you will need to:

- Configure HTTP proxy servers.
- Implement SSL encryption.
- Establish server side authentication with server certificates.
- Establish client side authentication with client certificates.

How you do this in HP OpenView is described in the following sections.

Open

HP OpenView's HTTPS communication is built on the industry standard HTTP 1.1 protocol and SSL sockets. HP OpenView's adherence to open standards, such as HTTP, SSL and SOAP, allows you to maximize the use of your current HTTP infrastructure.

NOTE Content filtering for OVO agents is not supported.

HTTP proxies are widely used in today's networks. They are workhorses to help safely bridge private networks to the Internet. The use of HTTP allows HP OpenView to slot into and take advantage of current infrastructures.

Scalable

HP OpenView's HTTPS communication is designed to perform well, independent of the size of the environment and the number of messages sent and received. HP OpenView's HTTPS communication can be configured to suit the environment within which it is to work. Large applications are able to handle many simultaneous connections while consuming the minimum of resources. If the maximum number of configured connections is exceeded, an entry in a logfile is created from which a warning message can also be raised.

3 Security Concepts

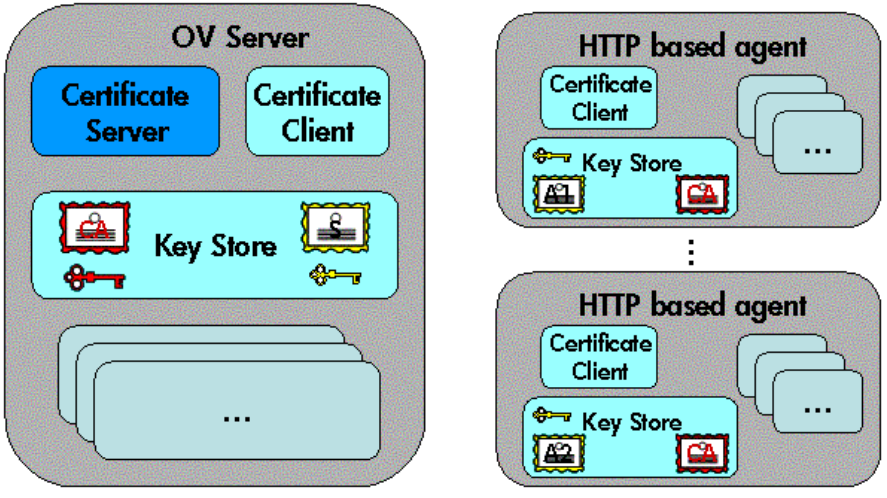
HTTPS-Based Security Components

Managed nodes must have a valid, industry standard, X509 certificate issued by the HP OpenView Certificate Server to be able to communicate with HP OpenView management servers. Certificates, signed by 1024 bit keys, are required to identify managed nodes in a managed environment using the Secure Socket Layer (SSL) protocol. The “SSL handshake” between two managed nodes only succeeds if the issuing authority of the certificate presented by the incoming managed node is a trusted authority of the receiving managed node. The main communication security components responsible for creating and managing certificates are:

- HP OpenView Certificate Server
- HP OpenView Key Store
- HP OpenView Certificate Client

Figure 3-1 illustrates these components:

Figure 3-1 Components of Authenticated Communication



Each system hosting an HTTPS agent is allocated a unique identifier value for the parameter, `OvCoreId`, created during installation of the HP OpenView software on that system.

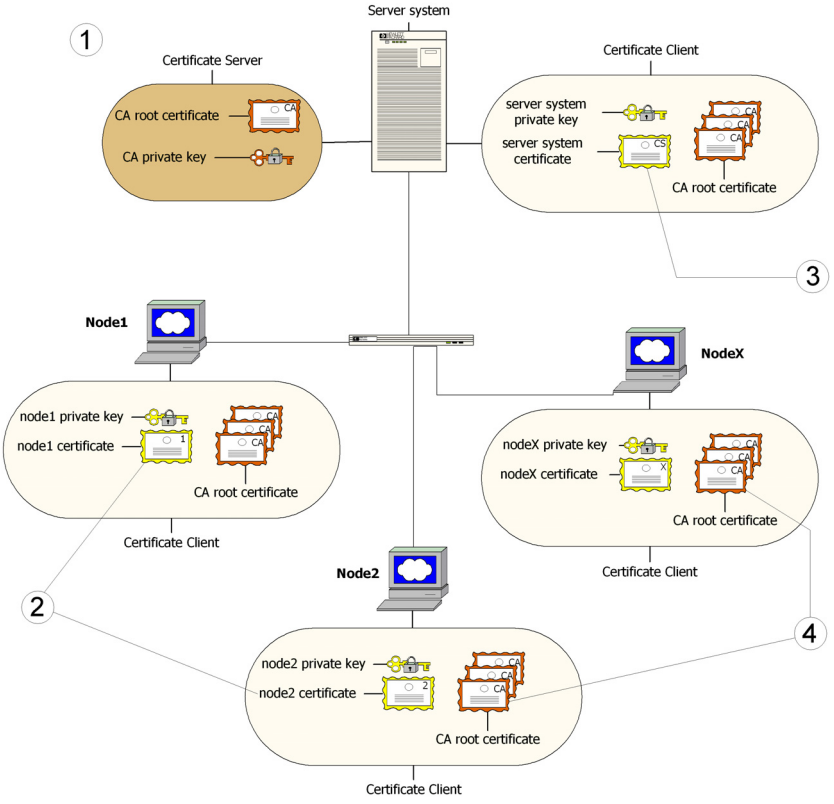
NOTE

After the `OvCoreId` for an HTTPS managed node has been created, it does not change, even if the hostname or the IP address, for example through DHCP, of the system is changed.

For each OpenView system (managed node or server) `OvCoreId` is used as a unique identifier and is contained in the corresponding managed node certificate. `OvCoreId` is allocated its value during installation.

Figure 3-2 illustrates an environment for authenticated communication:

Figure 3-2 Environment for Authenticated Communication



1. A server system hosts the Certificate Server, which contains the needed certification authority (CA) functionality.
2. Every system has a certificate that was signed by the Certificate Server with the certification authority private key.
3. The server system also needs a certificate to prove its identity.
4. Every system has a list of trusted root certificates, which must contain at least one certificate. The trusted root (CA) certificates are used to verify the identity of the communication partners; a communication partner is only trusted if the presented certificate can be validated using the list of trusted certificates.

   A list of trusted root certificates is required, when the certificate client is being managed by more than one HP OpenView management server. For instance, when a managed node is managed simultaneously by multiple OVO management servers.

Certificates

There are two types of certificates:

- Root certificates
- Managed node certificates

A root certificate is a self-signed certificate, containing the identity of the certification authority of the certificate server. The private key belonging to the root certificate is stored on the certificate server system and protected from unauthorized access. The certification authority uses its root certificate to digitally sign all certificates.

Every HTTPS managed node in the managed environment receives a managed node certificate issued by a certificate server, a corresponding private key stored in the file system and the root certificates valid in its environment. The certificate client running on the managed node ensures this.

NOTE

A managed node certificate contains the unique identity `OvCoreId`. The following is an example of an `OvCoreId`:

```
d498f286-aa97-4a31-b5c3-806e384fcf6e
```

Each managed node can be securely authenticated through its managed node certificate. The managed node certificate can be verified by all other managed nodes in the environment using the root certificate(s) to verify the signature.

Managed node certificates are used to establish SSL-based connections between two HTTPS managed nodes that use client and server authentication, and can be configured to encrypt all communication.

The `ovcert` tool provided by the certificate client can be used to list the contents of the Key Store or to show information about an installed certificate. The `ovcert` tool is described in the `ovcert` man page.

HP OpenView Certificate Server

The certificate server is responsible for the following:

- Creating and installing self-signed root certificates.
- Importing self-signed root certificates from the file system.
- Storing the private keys of root certificates.
- Granting or denying certification requests.
- Creating a new certificate and a corresponding private key or creating an installation key for manual certificate installation.
- Offering a service for clients to automatically retrieve trusted root certificates.

Certification Authority

NOTE Every OVO management server is automatically configured as a Certificate Authority. The default setting for `sec.cm.client:CERTIFICATE_SERVER` for every agent is its own OVO management server.

The certification authority is part of the certificate server and is the center of trust in certificate management. Certificates signed by this certification authority will be regarded as valid certificates and therefore be trustworthy. The certification authority must be hosted in a highly secure location. By default, it is installed on the system hosting the HP OpenView management server, for example the OVO management server system.

Since the certification authority is the root of trust, it operates with a self-signed root certificate. This root certificate and the corresponding private key are created and stored on the file system with the level of protection to allow the certification authority to operate. After the certification authority is successfully initialized, it is responsible for signing granted certificate requests using its root certificate.

Certificate Client

The certificate client runs on a managed node and acts as the counterpart of the certificate server's certificate request handler.

The certificate client operates as follows:

- The certificate client checks whether the managed node has a valid certificate.
- If the managed node has no certificate, the certificate client generates a new public and private key pair and creates a certificate request based on the unique identity (`OvCoreId` value) of the managed node. This certificate request is sent to the certificate server together with any additional managed node properties and the certificate client waits for a response.

  The additional managed node properties, for example DNS name and IP address of the managed node are intended to be used as additional information that, on the certificate server, should help to determine from which system in the environment a certificate request comes and to decide whether this request should be granted.
- After receiving the new certificate, it is installed on the managed node. After being installed, the certificate client can ensure that all HTTPS-based communication uses this certificate.

  If the request is not successfully processed, a descriptive error is logged and the associated status is set.

In addition, the certificate client does the following:

- It can be triggered to contact a certificate server to update its trusted root certificates, for example, using the command line tool `ovcert`. Refer to the `ovcert` man page for details.
- It supports the import of a managed node certificate and the corresponding private key from the file system with its command line interface `ovcert`. For more details see "Certificate Generation for Manual Certificate Deployment" on page 149 and "Manual Certificate Deployment with Installation Key" on page 154. Manual certificate installation is used to improve security on sensitive systems.
- It supports the import of trusted root certificates.

- It provides status information. Status includes `OK`, `valid certificate`, `no certificate`, `certificate requested`, and `certificate request denied`.

Root Certificate Update and Deployment

It may be necessary to update the trusted root certificates of one or more managed nodes, for example, in environments hosting several HP OpenView certificate servers.

It is possible to supply all currently trusted root certificates to certificate clients in a secured way. It is usually sufficient to supply the root certificate of the certification authority. However, it may be necessary to deploy one or more additional root certificates to selected certificate clients, for example when there is more than one certification authority in the environment.

The certificate client allows triggering the “trusted root certificates update” through the command line tool `ovcert`. Refer to the `ovcert` man page.

Security in Manager of Manager (MoM) Environments

The use of certificate servers in Manager of Manager environments can be divided into the following two types:

- “Environments Hosting Several Certificate Servers”
- “Establish a Shared CA in MoM Environments”

Environments Hosting Several Certificate Servers

It is possible that a managed environment has more than one certificate server. This situation would arise if two existing managed environments, both having an operating certificate server are joined to form a single environment. This is termed **merge**.

Both certificate servers are each using a self-signed root certificate. As a result, all clients belonging to one certificate server do not trust any client belonging to the other. This is solved by adding the root certificate of each certificate server to the trusted root certificates of the other certificate server. Finally, all clients in the managed environment are triggered to receive the updated root certificate list from their certificate server.

If an agent is managed by multiple management servers some certificate management configuration must be made. By default, every OVO server has its own Certificate Authority and the agent trusts only certificates subscribed by this authority. For MoM environments, you must establish a trust between two or more managers so that their environments are able to communicate with each other.

The common scenarios are:

- “Merge Two Existing MoM Environments”
- “Certificate Handling for a Second OVO Management Server”
- “Establish a Shared CA in MoM Environments”

These scenarios are discussed in greater detail in the following sections.

Merge Two Existing MoM Environments

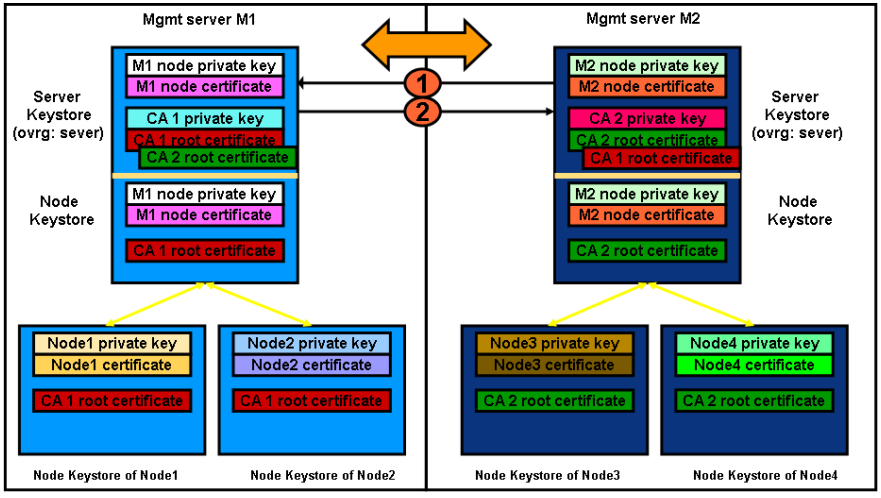
Assume you have an environment belonging to server M1 with the agents AM1 and the second of M2 with AM2. Assume that each server has its own Certificate Authority.

Complete the following steps to merge the environments:

NOTE HA environments and non-HA environments are handled in the same way. The following steps are valid for both types of installations.

1. Synchronize the trusted certificates on the management servers: M1 gets the root certificates of M2 and M2 the root certificate of M1.

Figure 3-3 Synchronizing the Trusted Certificates on the Management Servers



a. On OVO management server M1, enter the command:

```
ovcert -exporttrusted -ovrg server -file <my_file>
```

b. Copy *<my_file>* to the management server M2, for example using `ftp`.

c. Enter the following command on M2:

ovcert -importtrusted -ovrg server -file *<my_file>*

d. Repeat the procedure for management server M2.

e. To verify that M1 and M2 have the root certificate of the other, on both management server systems, execute the command:

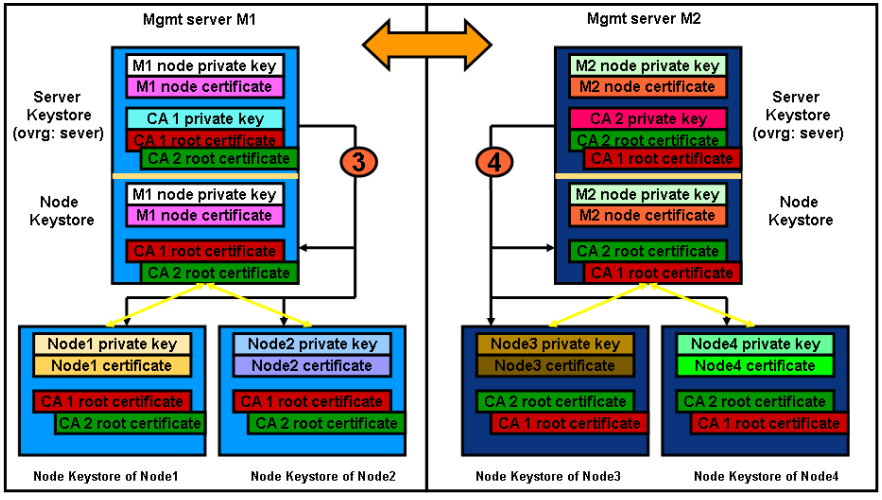
ovcert -list

Two trusted certificates should be listed.

2. Go to the `OVO Application Bank` and call the `Update Trusts` application to update the local root certificates on each managed node:

`Certificate Tools` → `Update Trusts`

Figure 3-4 Updating the Local Root Certificates on Each Managed Node



NOTE In cluster installations, the local certificates for the agents and the server are not the same.

On each management server (M1 and M2), select all required managed nodes and execute the application. The agents contact their certificate server and ask for new root certificates.

You can verify this on all managed nodes by executing command:

```
ovcert -list
```

Two trust certificates should be displayed.

NOTE

You can also trigger this action on each managed node. Log on to each managed node system in turn and execute the command:

```
ovcert -updatetrusted
```

NOTE

The certificate server is identical to the management server in this scenario.

3. Configure other management server as regular nodes in the OVO node bank. M1 must be added to the node bank of M2 with its `OvCoreId` and M2 must be added to the node bank of M1 with its `OvCoreId`.

   a. Add node M1 in the node bank of M2 and M2 in the node bank of M1 as follows:

      In the Administrator's GUI, select:

      `Action` → `Node` → `Add`

      Note: You can also use the command line tool:

      On node M1, enter the command:

      ```
      opcnode -add_node node_name=<M2> \
      net_type=<network_type> mach_type=<machine_type> \
      group_name=<node_group_name>
      ```

      On M2, enter the command:

      ```
      opcnode -add_node node_name=<M1> \
      net_type=<network_type> mach_type=<machine_type> \
      group_name=<node_group_name>
      ```

b. M1's OvCoreId must be stored in M2's database:

On M1, call the ovcoreid command to display its OvCoreId:

```
ovcoreid -ovrg server
```

Note down the displayed value.

On M2, call the opcnode command to add M1's OvCoreId into M2's database:

```
opcnode -chg_id node_name=<M1> id=<core_id_of_M1>
```

c. M2's OvCoreId must be stored in M1's database:

On M2, call the ovcoreid command to get OvCoreId of M2:

```
ovcoreid -ovrg server
```

Note down the displayed value.

On M1, call the opcnode command to add M2's OvCoreId into M1's database:

```
opcnode -chg_id node_name=<M2> id=<core_id_of_M2>
```

You can verify that the nodes have been correctly added to the databases by executing the following commands:

a. On M1, enter the command:

```
opcnode -list_id node_list=<M2>
```

The OvCoreId of node M2 should be displayed.

b. On M2, enter the command:

```
opcnode -list_id node_list=<M1>
```

The OvCoreId of node M1 should be displayed.

NOTE Do not forget to add uploaded nodes to Node Group so that you are able to see messages.

4. Create or enhance the responsible manager policy on both servers and deploy it to their own agents.
5. Synchronize the Node Banks using opccfgupld and opccfgdwn. M1 gets the entries of M2, M2 gets the entries of M1 including their OvCoreIds.

Certificate Handling for a Second OVO Management Server

Assume the second OVO management server has its own Certificate Authority and is used as a backup management server or competence center. Assume that server M1 owns the agents AM1 and that the server M2 initially has no agents.

1. Synchronize the trusted certificates on the management servers: M1 gets the root certificates of M2 and M2 the root certificate of M1.

   a. On OVO management server M1, enter the command:

   ovcert -exporttrusted -ovrg server -file *<my_file>*

   b. Copy *<my_file>* to the management server M2, for example using `ftp`.

   c. Enter the following command on M2:

   ovcert -importtrusted -ovrg server -file *<my_file>*

   d. Repeat the procedure for management server M2.

   e. To verify that M1 and M2 have the root certificate of the other, on both management server systems, execute the command:

   ovcert -list

   Two trusted certificates should be listed.

2. Go to the `Application Desktop` and call the `Update Trusts` application to update the root certificate on M1.

   `Certificate Tools` → `Update Trusts`

   On M1 select AM1, and execute the application. The agent contacts its certificate server and ask for a new root certificate.

NOTE You can also trigger this action on the managed node by executing:

ovcert -updatetrusted

NOTE The certificate server is identical to the management server in this scenario.

3. Configure other management server as regular nodes in the OVO node bank. M1 must be added to the node bank of M2 with its `OvCoreId` and M2 must be added to the node bank of M1 with its `OvCoreId`.

   a. Add node M1 in the node bank of M2 and M2 in the node bank of M1 as follows:

      In the Motif Administrator's GUI, select:

      `Action` → `Node` → `Add`

      Note: You can also use the command line tool:

      On node M1, enter the command:

      ```
      opcnode -add_node node_name=<M2> \
      net_type=<network_type> mach_type=<machine_type> \
      group_name=<node_group_name>
      ```

      On M2, enter the command:

      ```
      opcnode -add_node node_name=<M1> \
      net_type=<network_type> mach_type=<machine_type> \
      group_name=<node_group_name>
      ```

   b. M1's `OvCoreId` must be stored in M2's database:

      On M1, call the `ovcoreid` command to display the `OvCoreId` of M1:

      ```
      ovcoreid -ovrg server
      ```

      Note down the displayed value.

      On M2, call the `opcnode` command to add M1's `OvCoreId` into M2's database:

      ```
      opcnode -chg_id node_name=<M1> id=<core_id_of_M1>
      ```

   c. M2's `OvCoreId` must be stored in M1's database:

      On M2, call the `ovcoreid` command to get `OvCoreId` of M2:

      ```
      ovcoreid -ovrg server
      ```

      Note down the displayed value.

      On M1, call the `opcnode` command to add M2's `OvCoreId` into M1's database:

      ```
      opcnode -chg_id node_name=<M2> id=<core_id_of_M2>
      ```

You can verify that the nodes have been correctly added to the databases by executing the following commands:

a. On M1, enter the command:

```
opcnode -list_id node_list=<M2>
```

The `OvCoreId` of node M2 should be displayed.

b. On M2, enter the command:

```
opcnode -list_id node_list=<M1>
```

The `OvCoreId` of node M1 should be displayed.

NOTE Do not forget to add uploaded nodes to `Node Group` so that you are able to see messages.

4. Create or enhance the responsible manager policy on both servers and deploy it to their own agents. M1 must deploy a responsible manager policy to all its managed nodes, in this case, they are M1 and AM1. M2 must deploy a responsible manager policy to its local agent if it was not already a part of M1's environment.
5. Synchronize the Node Banks using `opccfgupld` and `opccfgdwn`. Now M2 receives all agents of M1 and M1 loads the local agent of M2, if not already present in the database.

Switch CAs in MoM Environments

Assume that a MoM environment is already established. For example:

- Management server A is active CA.
- Management server B is alternative CA.
- Systems are managed as OVO nodes.
- Responsible manager template has been created and distributed to all managed nodes.

To change the CA in the above MoM environment:

1. Set management server B as a primary manager for a managed node by executing the following command on management server B system:

   opcragt -primmgr *<node hostname>*

2. Remove all templates from this managed node that were distributed from management server A.
3. Stop the agent software on the managed node:

   ovc -kill

4. Remove the agent certificate from the managed node:

   ovcert -remove *<alias>*

5. Remove the trusted management server A certificate from the managed node:

   ovcert -remove *<alias>*

NOTE Management server B must present on node (**ovcert -list**).

6. Issue a new node certificate manually from management server B:

   opccsacm -issue -name *<nodename>* -file *<filename>* -coreid *<OvCoreId>*

7. Transfer the newly created certificate to the managed node.
8. Import the new certificate to the managed node:

   ovcert -importcert -file *<filename>*

9. Change the configuration settings on the managed node to reflect the change to the new CA (management server B):

   ```
   ..
   [sec.cm.client]
   CERTIFICATE_SERVER=<management server B hostname>
   [sec.core.auth]
   MANAGER=<management server B hostname>
   MANAGER_ID=<management server B OvCoreId>
   ...
   ```

   The configuration settings are changed with the following commands:

   ovconfchg -ns sec.cm.client -set CERTIFICATE_SERVER *<management server B hostname>*

   ovconfchg -ns sec.core.auth -set MANAGER *<management server B hostname>*

   ovconfchg -ns sec.core.auth -set MANAGER_ID *<management server B OvCoreId>*

10. Start the agent software on the managed node:

    ovc -start

11. Distribute templates to the managed node.

Establish a Shared CA in MoM Environments

The scenarios described above show how to merge environments with separate Certificate Authorities. It is also possible to work with only one Certificate Authority. However, this should be considered before setting up an OVO MoM Managed environment.

NOTE If you have an existing environment with two certificate authorities, it is not recommended to use the shared CA scenario, as this would require you to replace all certificate that have been granted by one of the CAs.

In addition, consider that all OVO management servers and their managed nodes are dependent on one Certificate Authority.

Assume that server M1 has a Certificate Authority and M2 should not have one.

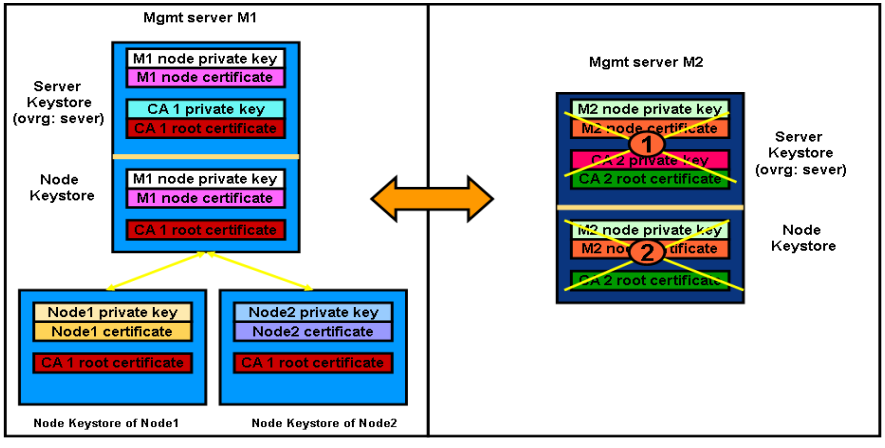
Execute the following steps:

1. Immediately after the installation of M2, remove the local certificates with the following commands:

   ```
   ovcert -remove <cert_id>

   ovcert -remove -ovrg server <cert_id>
   ```

Figure 3-5 Removing Certificates from the M2 Management Server



2. Add M2 to the node bank of M1:

   On node M1, using the Administrator's GUI:

   `Action` → `Node` → `Add`

   Note: You can also use the command line tool:

   On node M1, enter the command:

   ```
   opcnode -add_node node_name=<M2> \
   net_type=<network_type> mach_type=<machine_type> \
   group_name=<node_group_name>
   ```

3. Create a certificate for M2 on M1 with the following commands:

   ```
   opccsacm -issue -name <M2> -coreid <core_ID_M2> \
   -file <M2_cert> -pass <password>
   ```

NOTE

To display the `OvCoreId` of M2, on the M2 system, enter the command:

```
ovcoreid -ovrg server
```

`opccsacm` also adds the `OvCoreId` of M2 to the database.

4. Copy the certificate to M2 (HA server) and install it as the server certificate:

   ```
   ovcert -importcert -ovrg server -file <my_cert> \
   -pass <password>
   ```

   If M2 is not an OVO HA cluster server, call the same command as above but without the resource group `server` option to install a node certificate:

   ```
   ovcert -importcert -file <my_cert> -pass <password>
   ```

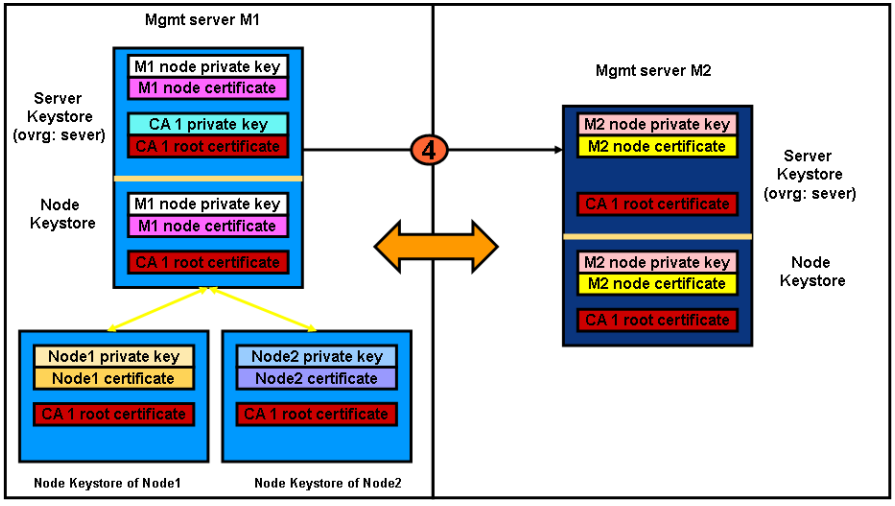
   If M2 is an HA system, create an extra node certificate for each physical node. On M1 call:

   ```
   opccsacm -issue -name <hostname_M2_cluster_node> \
   -coreid <OvCoreId_M2_cluster_node> -file <my_cert> \
   -pass <password>
   ```

   Copy the node certificates to the M2 cluster nodes and install using the command:

```
ovcert -importcert -file <my_cert> -pass <password>
```

Figure 3-6 Issuing Certificate for M2 on M1 and Installing it on M2



5. Instruct every managed node which will be installed by M2 that its certificate server is M1 by placing an entry into the `bbc_inst_defaults` file. This file is used to automatically generate profiles for the agent installation. The location of the file is:

   `/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults`

NOTE

If this file does not exist, create it now using the following sample file as a template:

`/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults.sampl`

Add the namespace and certificate server specifications to your `bbc_inst_defaults` file as follows:

```
[sec.cm.client]
CERTIFICATE_SERVER <hostname_M1>
```

For the local agent on M2 call:

```
ovconfchg -ns sec.cm.client -set \
CERTIFICATE_SERVER <hostname_M1>
```

6. On M1, specify the `OvCoreId` of M2 as a trusted `OvCoreId`:

   ```
   ovconfchg -ovrg server -ns opc -set \
   OPC_TRUSTED_SERVER_COREIDS <M2's OvCoreId>
   ```

   If you have more than two management servers in your Shared-CA environment, you need to complete some additional steps. Let us assume that you have three management servers: M1, M2 and M3. On M1, the CA root certificate is installed. M2 and M3 receive a shared CA which is issued by M1. The followings steps must also be be completed:

   a. On management server M1, which has a Certificate Authority, specify a trusted `OvCoreId` list which contains a comma separated list of all management server OvCoreIds in your MoM environment, excluding the `OvCoreId` from management server M1.

      On M1, specify the OvCoreIds of M2 and M3 as trusted OvCoreIds:

      ```
      ovconfchg -ovrg server -ns opc -set \
      OPC_TRUSTED_SERVER_COREIDS
      <OVCoreID>_M2>,<OVCoreID_M3>
      ```

   b. On all other management servers (for example, M2 and M3) which have a shared CA, specify a trusted `OvCoreId` list which contains a comma separated list of management server OvCoreIds. This list contains all management server OvCoreIds in this MoM environment, excluding the OvCoreIds from:

      - Local management server where the following command is executed.
      - Management server (M1) which has the Certificate Authority.

      On M2, specify the `OvCoreId` of M3 as a trusted `OvCoreId`:

      ```
      ovconfchg -ovrg server -ns opc -set \
      OPC_TRUSTED_SERVER_COREIDS <OVCoreID_M3>
      ```

On M3, specify the `OvCoreId` of M2 as a trusted `OvCoreId`:

```
ovconfchg -ovrg server -ns opc -set \
OPC_TRUSTED_SERVER_COREIDS <OVCoreID_M2>
```

7. Unregister the Certificate Server (`ovcs`) component from system M2 using the command:

   ```
   ovcreg -del ovcs
   ```

8. Create or enhance the responsible manager policy on both servers and deploy it to their own agents. M1 must deploy a responsible manager policy to all of its agents which are to be managed by M2. M2 must deploy a responsible manager policy to its local agent, if it was not already a part of M1's environment.

9. Download the node bank configuration on M1 and upload to M2 by using the `opccfgupld` and `opccfgdwn` tools.

Remote Action Authorization

From the point of view of security, remote actions are a very special case in OVO managed environments. It must be ensured that it is not possible to send a faked remote action to a management server that is then executed on the specified remote system in the environment. In particular, this is sensitive since it is not possible to regard any managed system as a secure system. It is assumed that root access to a managed node is available to unauthorized users.

In addition, one OVO management server of a service provider must be able to manage the environments of several of its customers, while ensuring that no system located in one customer segment is allowed to trigger any actions in any other customer segment.

OVO ensures that action strings, for example, a specific command, cannot be tampered with by a malicious user. On the OVO management server, it is possible to configure:

- On which systems the OVO management server is allowed to execute an action.
- Whether only "signed actions" originating from an HTTPS agent are accepted.

Action requests contained in OVO messages which specify a target system for the action other than the sender of the message are remote actions and must be handled securely. These remote actions are subjected to additional security checks describe in the following section. Remote actions are only be executed if they pass these security checks.

The following general rules apply:

- A remote action is defined as an automatic action or operator-initiated action which is defined within an OVO message sent by Managed Node A and configured to run on Managed Node B. The execution of such actions can be controlled with the file

  `/etc/opt/OV/share/conf/OpC/mgmt_sv/remactconf.xml`.

- Whenever a message containing a remote action arrives on the OVO Management Server, this file is re-loaded if modified and the message will be processed against the rules contained in the remote action configuration file.

- If the remote action configuration file does not exist, is empty, unreadable or does not contain rules, all remote actions are disabled.
- A message containing remote actions will be matched against the rules in the same sequence as configured. The first match determines the result - a `deny` clause disables the remote actions within the message and adds an appropriate annotation to the message. Other than this, the actual message is processed normally. An `allow` clause leaves the message unmodified.
- If the message does not match any rule, the remote actions will be disabled in the same way as if it had matched a `deny` rule.
- A rule matches if all rule elements match in an AND-logic fashion. If a possible rule element is omitted, for example no `<target>` tag is specified, any appropriate message value matches. However, this does not apply to the `<certified>` tag - if this is not specified, a default of `true` applies (*).
- If the remote action configuration file contains syntax errors or other logical errors, such as a non-existing node group, parsing stops and all subsequent rules are ignored (*).
- The `trust` section is not supported (*).
- The `certified` tag can have the values `true` (default) or `false`. The meaning is whether the message originates from a certified source and the message certificate has been verified. Effectively a rule containing the clause `<certified>false</certified>` matches messages from DCE nodes. `<certified>true</certified>` matches messages from HTTPS nodes.

Server Configuration of Remote Action Authorization

The message manager uses a file-based configuration on the OVO management server to specify authorization of remote actions. The configuration contains a trust section that defines which systems are trusted as action signers, and a list of rules, each of which consist of a condition and an action. Each action request is checked against all condition in the order of their definition. If a condition matches, processing of the action request the action is stopped.

The conditions allow checking properties on an action request, such as source node, target node, or signature. There are only two possible actions: `allow` and `deny`. An `allow` action means that the action request is authorized. A `deny` action means that the action request is rejected.

Authorization data is logged with the reason for denying authorization. If an action is unauthorized, it is automatically deleted from the message and details about the match and the signature status are added as an annotation to the message. Unauthorized messages never appear in the GUI and therefore cannot be accidentally executed.

Source and target nodes are matched against node groups or single nodes. A dedicated keyword can be used for the management server.

If the new configuration file is missing or contains no rules, all remote actions are disabled. A default configuration file that contains the `OvCoreId` of the management server is installed with the product. The default configuration file also contains some examples in comments.

During startup, the message manager reads the file:

`/etc/opt/OV/share/conf/OpC/mgmt_sv/remactconf.xml`

It may also be triggered at runtime to re-read the file.

The syntax of the configuration file is XML based, and according with the following schema:

Figure 3-7 Remote Action Configuration File Syntax

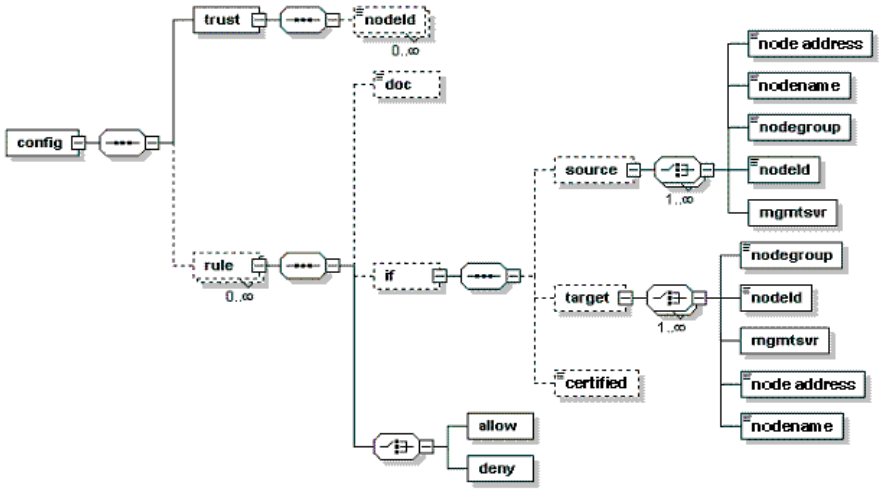


Table 3-1 Remote Action Configuration File Components

Elements	Description
`config`	`config` consists of a trust element and of a list of rule elements.
`trust`	The `trust` element consists of a list of `nodeId` element, each containing the `OvCoreId` of a trusted node.
`rule`	Each `rule` consists of the following components: • `doc` (optional) containing a description. string • `if` (optional) containing a `condition`. • An `allow` or a `deny` action. The `allow` and `deny` actions are empty and define if action execution is allowed or denied.
`condition`	A condition consists of a sequence of optional checks. A condition matches only if all contained checks match. If no check is defined, or if no condition is defined, a match is always successful. The checks are: • `source` • `target` • `certified`

Table 3-1 Remote Action Configuration File Components (Continued)

Elements	Description

`source`
 	
`target`
	Used to check the source node of an action request.
	
	Used to check against the target node of an action request.
	
	Both source and target consist of a set of choices. These checks match if any of the elements match.
	
	• `nodegroup`
	
	The `nodegroup` element contains the name of a node group from the OVO database. It matches if the request's node is a member of that node group.
	
	• `nodeId`
	
	The `nodeId` element contains an `OvCoreId`. It will mach if this `OvCoreId` is the ID of the request's node.
	
	• `mgmtsrv`
	
	The `mgmtsrv` element is empty. It matches if the request's node is the management server.
	
	• `nodeAddress`
	
	• `nodename`

`certified`	The certified check allows the values `valid` and `invalid`. `Valid` matches only if a signature and a certificate are provided, with the signature being signed by the certificate's owner, and when the `OvCoreId` of the certificate's subject is listed in the trust element. `Invalid` matches all other cases.

The following is an example of a remote action configuration:

```
<?xml version="1.0"?>
<config xmlns="http://openview.hp.com/xmlns/Act/Config/2002/08">
  <rule>
    <doc>Actions from Group2 to Group1 are always allowed</doc>
    <if>
      <source>
        <nodegroup>Group2</nodegroup>
      </source>
      <target>
      <nodegroup>Group1</nodegroup>
      </target>
    </if>
    <allow/>
  </rule>
  <rule>
    <doc>No actions from Group3 are allowed</doc>
    <if>
      <source>
        <nodegroup>Group3</nodegroup>
      </source>
    </if>
    <deny/>
  </rule>
  <rule>
    <doc>Actions to Group3 are allowed if certified</doc>
    <if>
      <target>
        <nodegroup>Group3</nodegroup>
      </target>
      <certified>true</certified>
    </if>
    <allow/>
  </rule>
</config>
```

Remote Action Authorization Rules for DCE Nodes

By default all remote actions from DCE nodes are denied on the OVO 8 management server. This also includes operator-initiated actions where the target node and the node which sends the message are the same. If needed, this restriction can be weakened by using the "self-rule". It allows in particular "local" operator-initiated actions.

In general: the security check is successful, if message sender and action target are identical. The "self-rule" is only useful for DCE nodes. The rule is sufficient, to run most of the SPIs for DCE agents (all except the ones which require an action to be executed on the management server).

Here an example for such a rule:

```
<rule>
  <doc>allow DCE local operator action</doc>
  <if>
    <target>
      <self/>
    </target>
    <certified>false</certified>
  </if>
  <allow/>
</rule>
```

Agents Running Under Alternative Users

OVO processes normally run under user `root` on UNIX systems and under the `System` account on Windows systems. The root/administrative privileges enable the processes to:

- Access OpenView resources. OpenView files are normally also restricted to privileged access only.
- Allow a switch user for application specific access rights.
- Directly access operating system resources such as log files and configuration files.
- Start application or operating system specific commands and executables.

There may be systems within IT environments that are highly security sensitive and it is necessary to limit the number of processes that have full root permissions to a small, well defined and tested group. In addition, it is desirable to be able to identify the precise process that manipulated critical system resources. This is not possible if many applications are running under the privileged user.

NOTE `ovswitchuser` is not supported by the OVO HTTPS agent on Windows platforms.

OVO software on UNIX managed node systems can be configured to run under a user that does not have full `root` permissions, often referred to as “running as non-root”. To run an agent as non-root, access to non-OpenView files and executables must be specifically given to the OVO processes on the managed node.

All OVO HTTPS agents on UNIX systems can be configured to run under a user other than root using the `ovswitchuser` tool.

The `ovswitchuser` tool allows the UNIX HTTPS agent on an OVO managed node to run under a user other than the privileged root user. The `ovswitchuser` tool makes the following changes:

- Perform change group ownership on:
  - All registered files of all installed component packages.
  - All files and directories of *<OVDataDir>* recursively.
- Change operating system daemon/service registration to start OVO processes under the new user.

Limitations of Running OVO Agents Under Alternative Users

Agents running under alternative users have the following limitations:

WARNING **The OVO management server processes must always run under the user `root`. The `ovswitchuser` tool must not be called on the OVO management server system.**

NOTE The OVO HTTPS agent on Windows platforms does not support `ovswitchuser`. Windows systems must run under the `System` user and cannot be switched to any other user.

- Actions can only be executed if the account under which the agent runs has suitable privileges.
- It is not possible to access files or any other operating system resources unless the agent account has suitable privileges.

NOTE It is possible to circumvent access restrictions by implementing a sudo program, which gives the agent user additional capabilities for specific operations. For further details, refer to “Working with Sudo Programs on UNIX Agents” on page 92.

Configure an Agent to Run Under an Alternative User

Prepare the System Environment

WARNING **Do not use ovswitchuser.sh on the OVO management server system. The OVO agent on the OVO management server must run under the user root.**

NOTE After the change of user has been made using the `ovswitchuser` command, the agent processes must be run under this newly assigned user and no longer under the user `root`.

For HTTPS agents, you must select a UNIX group for the agent. All users under which the agent is to run must belong to this group.

If you are migrating from a non-root DCE agent to a non-root HTTPS agent there are some issues to consider. For example, if the DCE non-root agent is run as user `OVO_Agent` of group `Security`. No-one except user `OVO_Agent` or the super-user is able to read runtime files of this agent. With the HTTPS agent, permissions are defined and granted at the group level and all users belonging to the group `Security` can access the runtime data of the agent. Therefore, it might be necessary to create a new group `Security2` and put the user `OVO_Agent` into the group `Security2`. Otherwise all other users in the group `Security` could access the runtime data of the agent, including private keys.

NOTE The users and groups used in the above scenario are only examples. You are free to choose your own user and group names.

As long as the DCE agent user belongs to a group containing only trusted users, when the DCE agent is replaced by an HTTPS agent which should also run as non-root, no migration step is needed. The HTTPS agent can run under the same user that was used for the DCE agent.

If you are migrating from a non-root DCE agent to a non-root HTTPS agent there are some issues to consider. For example, if the DCE non-root agent is run as user `OVO_Agent` of group `Security`. No-one except user `OVO_Agent` or the super-user is able to read runtime files of this agent. With the HTTPS agent, permissions are defined and granted at the group level and all users belonging to the group `Security` can access the runtime data of the agent. Therefore, it might be necessary to create a new group `Security2` and put the user `OVO_Agent` into the group `Security2`. Otherwise all other users in the group `Security` could access the runtime data of the agent, including private keys.

umask Setting on UNIX The non-root concept relies on the user under which the agent runs belonging to a specific UNIX group. Therefore the group bits of any files that are created by OV applications must be set. This allows OV applications to be run under dedicated users if required, while sharing the same resources, for example log files. Therefore, it is recommended to set the `umask` to suit the users that are used to run OV applications.

A `umask` setting of 02 is preferable. 022 would cause problems when multiple applications are run under different users.

If only the OVO agent is installed or if all applications run under the same user, the `umask` does not need to be set.

Install an Agent Using an Alternative User on UNIX Managed Nodes

Complete the following steps to run a managed node under an alternative account to `root`:

1. Install the OVO software on the desired managed node as usual.
2. Stop the agent with the command:

   `ovc -kill`

NOTE

Do not use the command:

`ovc -stop`

This stops the agent processes but not the core OpenView processes. When you later start the agent processes with the command:

`ovc -start`

as the core processes are already running under the `root` user, all other process are also started under the `root` user.

3. Set the `umask` of the user to grant `Group Permissions`.
4. Call the `ovswitchuser` command:

   `/opt/OV/bin/ovswitchuser.sh -existinguser <my_user> \`
   `-existinggroup <my_trusted_group>`

5. By default the OVO HTTPS agent uses port 383 for network communication. This is a privileged port which can only be opened by user `root`.

   To configure the non-root agent to communicate over the network, you must select one of the following port configuration alternatives.

   If you want to continue using the reserved, privileged port `383`, set the `SUID` bit as described in the first point below. However, if you wish to use an alternative port, reset it using the following `ovconfchg` command as described in the second point.

WARNING **Only apply one of the following approaches: `setuid` OR change the `PORTS` setting.**

- It is possible to continue using the reserved, privileged port `383` by setting the `SUID` bit on the communication broker executable. Then, the communication broker only uses root privileges to open up the port and then switches back to the agent user for all other activities.

  Set the `setuid` bit of the `ovbbccb` binary with the following command:

  ```
  chmod 4550 /opt/OV/bin/ovbbccb
  ```

  Enter the following configuration command so that the root directory can be changed:

  ```
  ovconfchg -ns bbc.cb -set CHROOT_PATH /
  ```

- Select a non-privileged `ovbbccb` port. Change the port from `383` to a desired port with a value greater than `1024`.

  For HTTPS agents, the communication broker port on a system where the HTTPS agent is not running under user `root` is changed to a non-privileged port. As a result, all other applications using the communication broker on this managed node experience the same limitation. If you want to use an alternative port, refer to “Configure the OVO Management Server For Agents Running Under Alternative Users”.

  On a managed node, use the commands:

  ```
  ovconfchg -ns bbc.cb -set SERVER_PORT <NEW_PORT_NUMBER>
  ```

  ```
  ovconfchg -ns bbc.cb.ports -set PORTS \
  <FULL_DNS_NODE_NAME>:<NEW_PORT_NUMBER>
  ```

6. Restart the agent using the command:

   ```
   ovc -start
   ```

Configure the OVO Management Server For Agents Running Under Alternative Users

If you use a different port than the default 383 on a managed node, you must also configure this on the OVO management server. In addition, the port to be used for a particular managed node must be known to all OVO management servers that need to contact that managed node. This is done by setting the `bbc.cb.ports PORTS` variable on OVO management servers.

For example, let us assume that we have a managed node with hostname `ovo_node.sales.mycom.com`, the OVO management server hostname is `ovo_srv.sales.mycom.com`. The new `ovbbccb` port on `ovo_node.sales.mycom.com` is `8001`.

This port value must be set on the managed node and the OVO management server.

To set an alternative value for the `ovbbccb` port, enter the following command on both OVO management server and the managed node:

```
ovconfchg -ns bbc.cb.ports -set PORTS \
"ovo_node.sales.mycom.com:8001"
```

Individually setting the new port values for each managed node is inefficient and error-prone. Wildcards are recognized and should be used to specify groups of managed nodesss as used in the following examples.

Let us now assume that all managed nodes of domain `sales.mycom.com` should use port `8001`. To set this port for all systems in this domain, enter the following command on both OVO management server and the managed nodes:

```
ovconfchg -ns bbc.cb.ports -set PORTS \
"*.sales.mycom.com:8001"
```

However, it is recommended that OVO management servers always use port 383. So we should modify the previous step and enter the following command on both OVO management server and the managed nodes:

```
ovconfchg -ns bbc.cb.ports -set PORTS \
"ovo_srv.sales.mycom.com:383,*.sales.mycom.com:8001"
```

It is important that the `bbc.cb.ports:PORTS` entries on OVO management servers is always up-to-date. It is not normally important for a managed node to know which port is used by another managed

node. Therefore, only the setting on the OVO management server and the setting on a newly installed managed node agent must be considered. No update of the PORTS setting on existing agents is needed.

Changing the Default Port

It is recommended that you maintain the PORTS setting in a central place on the OVO management server system and use wildcards to reduce the need to make changes on the management server.

A sample configuration file with examples of how to set up parameters is available:

/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults.sampl

Take a copy of the bbc_inst_defaults.sampl, rename it bbc_inst_defaults, and modify it as follows:

Make a bbc_inst_defaults file entry of the form:

```
[bbc.cb.ports]
PORTS = ovo_srv.sales.mycom.com:383,*.sales.mycom.com:8001
```

As a result, all newly installed agents are automatically provided with the information that ovo_srv.sales.mycom.com uses port 383, while all agents matching *.sales.mycom.com use port 8001. The bbc_inst_defaults file is the basis for the “Agent Profile”, which is installed with every new managed node. The “Agent Profile” is explained in more detail on page 89.

If a new managed node system belongs to the domain *.sales.mycom.com, the OVO management server is correctly configured and port 8001 is used. You can check this by entering the following command on the OVO management server:

```
ovconfget bbc.cb.ports
```

If the OVO management server does not have the correct settings, take the value from the bbc_inst_defaults file and call ovconfchg to update the OVO server with a command of the following form:

```
ovconfchg -ns bbc.cb.ports -set PORTS \
"<ovo_server>:383,<system1>:<port1>,<system2>:<port2>, \
*.<domain1>:<port3>,*.<domain2>:<port4>"
```

Agent Profile

An agent profile maintained on the OVO is a list of configuration settings which is copied to the agent at install time. The profile contains some default values do not need to be configured in the `bbc_inst_defaults` file. Any settings defined in the `bbc_inst_defaults` file are also added to the agent profile.

The profile is concerned in ALL types of agent initial installations.

Use of the `bbc_inst_defaults` file is optional. If it exists, it is processed and the agent profile is enriched with data from the file.

In case of manual agent installation, you can create the agent profile using the command:

`/opt/OV/bin/OpC/opcsw -create_inst_info <node>`

The profile is located at:

`/var/opt/OV/share/tmp/OpC/distrib/<hex_IP_addr_of_node>.i`

NOTE When `opcsw` is called, it prints the `<hex_IP_addr_of_node>` to stdout.

Copy the profile together with the software packages to the managed node and enter a command of the following form:

`opc_inst -configure <profile_name> ...`

The utility `opcsw` includes the option:

`create_inst_info`

If you call **`opcsw -create_inst_info <node_specifier>`**

For each managed node specified in `<node_specifier>`, a file is created at:

`/var/opt/OV/share/tmp/OpC/distrib/<hex_IP_addr>.i`

This file contains the installation defaults for the managed node with IP address `<hex_IP_addr>`. The file is automatically copied to the target managed node during remote agent installation using `inst.sh`, or you can use it for manual agent installation.

The `opcsw -create_inst_info` command creates agent profiles using configuration data from the file:

`/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults`

on the management server and the following additional information from the OVO database:

- **`CORE_ID`**: `OvCoreId` of managed node. An optional parameter which is added to the profile if a value for `CORE_ID` is available in the OVO database under the namespace `sec.core`. If the CORE_ID parameter is not present in the database nor on the managed node, one is automatically created on the agent.
- **`MANAGER`**: Long hostname of primary OVO management server in namespace `sec.auth`.

  Only managed node `MANAGER` is authorized to perform config-, deployment-, message-, or action-execution related tasks after initial installation.
- **`MANAGER_ID`**: `OvCoreId` of `MANAGER` in namespace `sec.auth`.

  `MANAGER_ID` corresponds to `MANAGER` and is needed to perform the authorization checks.
- **`CERTIFICATE_SERVER`**: Long hostname of the system where a certificate request is issued (certificate authority) in namespace `sec.cm.client`.

  If no valid managed node certificate is present on the managed node, one is requested from `CERTIFICATE_SERVER` using the `CORE_ID` as the identifier.
- **`PROXY`**

  Defines which proxy and port to use for a specified hostname.

These five parameters are the minimum initial settings required on a managed node. It is possible to overwrite them in the `bbc_inst_defaults` file, for example, if you have one dedicated certificate authority for several OVO management servers.

Upgrading and Patching an Agent Running Under an Alternative User

Upgrading and patching DCE/NCS agents requires you to call `opcswitchuser` after each agent software installation, including upgrades and patch installations. This modifies the ownership of all OV files and directories to the customer defined owner. Additionally it changes the startup script to start the OVO processes under this specific user. `opcswitchuser` must be run every time you install additional OpenView modules to a specific system so that the ownership of the new files is changed to match the non-root user.

Running `ovswitchuser` is not required for upgrading and patching HTTPS agents. How to handle upgrading and patching HTTPS agents is described in the following sections.

Copy To Managed Node and Manually Install Later

NOTE The “Copy To Managed Node and Manually Install Later” concept is only valid for HTTPS nodes.

It is possible that an OVO administrator does not have root access to a system and the OVO agent is running under a non-root user. However, for HTTPS agents, if the communication broker is running on a system, you do not need to enter passwords, as data transfer works without them. Without root access, the complete remote installation of the agent, as described in the section “Installing HTTPS Managed Nodes Manually” on page 124, cannot be performed. It is only possible to copy the agent packages to the managed node system and a manual installation must be done at the managed node system itself. Native installer calls, such as `pkgadd` on Solaris, `rpm` on Linux, `swinstall` on HP-UX, need superuser privileges. This HTTPS node concept can be viewed as “copy to managed node and manually install later”.

If you run a non-root agent and you want to deploy a sub agent, a patch or a complete upgrade package which requires native installer access, the following is done automatically:

1. The bits are copied to `/tmp/<pkg_name>`.
2. The installation cannot proceed further, because the deployer is not able to call a native installer as this requires root capabilities.

   It finishes with `OK` but generates a warning message.
3. Inform an authorized person on the target managed node that the packages are locally available. This administrator can then continue with the installation by calling the `opc_inst` script in the same way as for a manual agent installation.

NOTE HTTPS-transfer is preferred to bootstrap transport methods. This means that a remote sub-agent, patch or upgrade installation of a non-root agent will not ask for passwords but on the other hand it will terminate after copying the bits. You are not prompted for the root password and the installation must be triggered explicitly. However, the additional manual installation step respects the current agent user.

Working with Sudo Programs on UNIX Agents

NOTE The “Copy To Node and Manually Install Later” concept and the use of sudo programs is only valid for HTTPS nodes.

One way to get the required rights is to configure a tool like sudo and configure the `OV_SUDO` setting. Sudo allows a permitted user to execute a command as the superuser or another user, as specified in the `sudoers` file. The real and effective `uid` and `gid` are set to match those of the target user as specified in the `passwd` file. The group vector is also initialized when the target user is not `root`. By default, sudo requires that users authenticate themselves with a password. By default this is the user's password, and not the root password. After a user has been authenticated, a timestamp is updated and the user may then use sudo without a password for a short period of time. By default, 15 minutes unless overridden in the `sudoers` file.

TIP

Sudo is free software and it is distributed under BSD-style licence. It can be obtained from `http://www.sudo.ws`.

Sudo software is not packaged as part of the OVO software.

Let us take an HTTPS agent running on a Solaris managed node as a non-root user, `ovo_user`.

The procedure is as follows:

1. Open the `/etc/sudoers` file.
2. Add the following line into `/etc/sudoers` file. Use `vi /etc/sudoers` or `visudo` command.

   ```
   ovo_user ALL =(root) = NOPASSWD: /var/opt/OV/\
   installation/incoming/bundles/OVO-Client/opc_inst
   ```

   Only the installation script `opc_inst` is called under a superuser, `root`.

NOTE

This command is valid for remote installation using the Administrator IU or using `opc_inst`. In all other cases, the actual path for `opc_inst` must be substituted.

If `NOPASSWD` is not specified, you should enter your own password, for example for the user `ovo_user`, and not superuser (`root`) password.

How to Setup a Sudo Program

NOTE

The bootstrap installation does not support `OV_SUDO`.

OpenView installation utilities that make native installer calls contain code of the form:

```
${OV_SUDO} opc_init
```

If the `OV_SUDO` variable is not set, it is interpreted as an empty string and ignored.

If the OV_SUDO variable is set, the variable is either exported from the non-root user's login shell, or it is read using `ovconfget ctrl.sudo` and then added to the environment by the install scripts.

NOTE Reading the OV_SUDO variable using `ovconfget ctrl.sudo` has higher priority than exporting its value from the non-root user's login shell.

A typical bootstrap installation of a non-root agent with sudo requires the following steps:

- Install agent as root.
- Call `/opt/OV/bin/ovswitchuser` to set the preferred user and group.
- Set preferred sudo program using the command:

  ```
  ovconfchg -ns ctrl.sudo -set OV_SUDO \
  <my_sudo_with_full_path>
  ```

- Set preferred sudo user using the command:

  ```
  ovconfchg -ns ctrl.sudo -set OV_SUDO_USER <my_sudo_user>
  ```

- Set preferred sudo group using the command:

  ```
  ovconfchg -ns ctrl.sudo -set OV_SUDO_GROUP <my_sudo_group>
  ```

NOTE The benefit of setting a sudo allows automatic sub-agent, patch and upgrade installation of non-root environments without entering passwords. Conversely, a remote bootstrap installation requires that an OVO administrator knows a super-user password of the managed node.

The remote agent installation first checks as which user an agent is running and whether OV_SUDO is setup. It decides then, whether “copy to managed node and manual install later” is needed. Depending on this bootstrap installation with password prompting or automatic installation is chosen.

A Comparison of DCE and HTTPS Alternative User Concepts

All OVO agents on UNIX systems can be configured to run under a user other than root. This is done using the `opcswitchuser` tool for DCE- and NCS-based UNIX agents, and the `ovswitchuser` tool for the HTTPS agent.

For a DCE/NCS agent, all files and directories of any OV application are set to the same user and group by the `opcswitchuser` tool.

For DCE/NCS nodes:

```
/opt/OV/bin/utils/opcswitchuser.sh <my_trusted_user> \
<my_group>
```

NOTE opcswitchuser.sh is not located in `/opt/OV/` on all platforms. Check the actual value of `OVInstallDir` and `OVDataDir`.

- You must select a UNIX group for the for the user under which the HTTPS agent is to run. This is not necessary for the DCE/NCS agent. For more details, refer to “Prepare the System Environment” on page 83.
- The HTTPS agent has file access rights opened for the assigned user and all other users which belong to the same group as the user of the HTTPS agent. The DCE/NCS agent can only be run under the assigned user. Example: OVO queue files: HTTPS `0660`, DCE `0600`.

NOTE Before changing the user under which the agent processes are to be run, set the `umask` of the user to grant `Group Permissions` and shutdown the agent.

- The HTTPS agent has the `group-id` bit set on its base directories.

  The `group-id` bit guarantees that all files created under such directories will belong to the agent's group. This also works if the primary group of the user under which the agent is running is different from the group of the agent files and directories.

For example, the primary group of user `OVO_Agent` is `Security`, agent files and directories belong to group `Security2`. Now also add `OVO_Agent` to group `Security2` (`Security` remains the primary group of `OVO_Agent`) and run the agent under user `OVO_Agent`. All files created by the agent running under the user `OVO_Agent` will belong to `Security2`. This mechanism allows OV components to run under different users but share common files.

- The `set group-id` bit may cause warnings of security check tools like medusa, which can be safely ignored. On DCE/NCS agents, no such warnings occur.
- No “copy to node and manual install later” concept for DCE/NCS nodes.
- No sudo concept for DCE/NCS nodes.
- For DCE/NCS nodes it is necessary to call `opcswitchuser` after each patch/upgrade installation on non-root agent. On HTTPS agent this is not required. You call `ovswitchuser` only once after bootstrap installation. Later you call `ovswitchuser` only, when you want to change the group/user of the agent, for example, back to root.

4 Concepts of Managing HTTPS Nodes

Controlling HTTPS Nodes

The OVO management server can perform the following functions on HTTPS nodes:

- Remote control of HTTPS agents.
- Remote and manual installation of HTTPS agents.
- Remote and manual patch installation and agent upgrade.
- Remote and manual configuration deployment.
- Support of multiple parallel configuration servers for HTTPS agents.
- Heartbeat polling.
- Security management of HTTPS nodes.
- Support of HTTPS nodes through the OVO management server APIs and utilities.

The following sections explain some new concepts for HTTPS nodes.

- “Configuration Deployment to HTTPS Nodes” on page 99
- “Heartbeat Polling of HTTPS Nodes” on page 104
- “Remote Control of HTTPS Nodes” on page 106
- “OVO Server Components and Processes” on page 338

Configuration Deployment to HTTPS Nodes

Configuration deployment to HTTPS agents differs slightly from that of DCE-based nodes:

- Policies are used by HTTPS agents in place of Templates.
- Instrumentation is the single term used by HTTPS agents for Actions, Commands, and Monitors.
- A configuration parameter schema with a name-value pair policy type for HTTPS agents replaces `nodeinfo` and `opcinfo` files.
- `mgrconf` file is enhanced for HTTPS agents by a role model-based security authorization mechanism.

The following sections explain the new configuration management concepts introduced with the HTTPS agents.

Policy Management

A policy is a template in XML format, with the strict separation of data and meta information. The header contains attributes such as name, type, version, and state. Five operations are possible on policies: install, remove, enable, disable and list. Template files contain all individual templates of a certain source type in one file, a policy file contains only the content of one template and this information is referred to as the policy data.

It is possible to manually install and remove policies using the `ovpolicy` tool, provided that you adhere to some guidelines.

Existing OVO templates can also be used with HTTPS agents as these are converted into policies at distribution time by the `opcbbcdist` process. The `mgrconf` and the `nodeinfo` configuration types are now treated as policies. Only one `mgrconf` file and one `nodeinfo` file are required, and a unique policy id is used.

In addition to the unique policy id, the header contains the policy name, policy type name, policy version, policy type version, and status. These attributes are generated by `opcbbcdist` as the data is being deployed.

Only one version of a policy can be installed on a node. A policy is identified by its id, but also the name plus policy type must be unique.

All policies that are deployed from the OVO server are allocated the version number `1` as OVO does not support policy versioning.

The status of a policy deployed for the first time is set to `enabled`. If the policy is already present on the system, a newly deployed policy assumes the status of the policy it replaces.

There is a utility called `opctemplate` for HTTPS nodes, which is wrapper for `ovpolicy` and allows common definitions with DCE nodes, for example in the application desktop.

Instrumentation Management

On HTTPS nodes, the actions-, commands-, and monitor directories are replaced with:

```
$OVDataDir/bin/instrumentation
```

which can have one level of sub directories. All instrumentation programs are installed at this location.

NOTE

The directory for executables on the OVO management server is located under:

```
/var/opt/OV/share/databases
```

No `instrumentation` directory is created and the directories actions, commands, and monitors are used.

Typically, action, command, and monitor executables are referenced in OVO templates. As long as these executables are not referred with their full path in policies, this change is transparent, because the new locations of the binaries is also added to the path variables of utilities like the OVO action agent, monitor agent and logfile encapsulator.

Files from the monitor directory on the OVO management server are installed on the agent with the rights `744`, all others with the rights `755`. This is identical to the settings on DCE-based nodes.

The configuration management process can also update running executables. Scripts and binaries of running executables are renamed and allowed to complete their tasks. Subsequent execution of these programs use the newly installed files.

Manual Installation of Policies and Instrumentation

It is not possible to copy policy data directly to a managed node because the agent must receive the configuration data in a secured format. This is required to avoid illegal manipulation of configuration data by unauthorized persons on the managed nodes.

The `opctmpldwn` tool is used to prepare the manual installation of policies on the OVO management server. The output data is stored in a directory on the management server system dedicated to the managed node.

There are minor differences between how `opctmpldwn` handles DCE and HTTPS nodes:

- For HTTPS nodes, the `nodeinfo` and `mgrconf` data are regarded as policies and therefore contained in the directory mentioned above. For DCE-based node, the `nodeinfo` data is disregarded.
- Templates and policies are secured using different methods. A template is encrypted with a node-specific key. The policy data is signed through a management server specific certificate while a policy header is only secured through file rights.

HTTPS Agent Distribution Manager

`opcbbcdist` is the configuration management adapter between the OVO management server and the HTTPS agents. Its main function are:

- Convert templates into policies.
- Create instrumentation from existing actions, commands, and monitors.
- Convert ECS templates into policies and their associated circuits.
- Switch `nodeinfo` settings into the XPL format used on HTTPS nodes.

`Opcbbcdist` is the counter part of opcdistm, the distribution manager for all other communication types. Just like `opcdistm`, it uses the internal file system interface:

```
/var/opt/OV/share/tmp/OpC/distrib
```

to get the information about what data should be deployed. `Opcbbcdist` also distinguishes between the four configuration categories:

- Policies/templates
- Instrumentation actions/commands/monitors
- nodeinfo
- mgrconf

Unlike `opcdistm`, `opcbbcdist` only accepts requests from other OVO management server components of the form `deploy configuration types xyz to node abc`. These requests may be issued by the GUI, by a configuration API or by `opcragt -update` and `opcragt -distrib`.

`opcbbcdist` possesses an automatic retry mechanism which is started if it was not possible to reach a node and new data is present for it. You can also manually trigger a retry by calling `opcragt -update`.

When `opcbbcdist` or `opcdistm` complete a task for a certain node, you get a message in the browser confirming correct distribution of configuration data. If tasks are not completed, messages, such as `Node Unreachable`, are displayed.

`Opcbbcdist` transfers instrumentation data first, then policies. This is done to avoid synchronization issues when an executable is referenced in a template. In addition `opcbbcdist` follows a simple transaction model: only if all data of a certain configuration type is successfully deployed, is the next category processed. The distribution of one configuration type is regarded as one transaction. If a transaction fails, it is rolled back and retried later. This schema is also applied when `opcbbcdist` is stopped due to OVO server shutdown.

Configuration Push

The OVO management server triggers all configuration deployment tasks to HTTPS nodes. The OVO server pushes configuration data down to the agent and there is only out-bound communication. The more secure OVO management server triggers the managed nodes.

A disadvantage is that a managed node must run with old data in the case of the system not being reachable when new configuration was distributed. The OVO management server must poll all nodes for which configuration is present but could not be delivered. The OVO management server does this task:

- at least once an hour per pending node.
- when the server is restarted.

- when the configuration push is explicitly triggered by `opcragt -update`, `opcragt -distrib`, or within the GUI by pressing the `Distribute` button, or by directly calling the API associated with the command.

NOTE In addition, DCE-based agents ask the OVO distribution manager `opcdistm` for new configuration data after system reboot or agent restart.

A monitor called `dist_mon.sh` checks for pending distributions. If any data in the configuration transfer directory:

`/var/opt/OV/share/tmp/OpC/distrib`

is older than 30 minutes, a message is displayed that specifies the managed node where a distribution is pending.

Delta Distribution

By default in OVO, the distribution process, known as delta-distribution, only deploys data which has been modified or added since the last configuration transfer. This minimizes the amount of data transferred and reduces the number of reconfiguration requests for interceptors and other sub agents. If required, the complete configuration can be re-deployed to the managed node.

In the delta-distribution mode, the OVO management server requests the policy inventory of the managed node and time stamps of the last instrumentation distribution. The policy inventory is compared with the policy assignment list and `opcbbcdist` computes and executes the required policy removal and installation tasks for the node. For instrumentation deployment, the time stamp of the last deployment is compared with the time stamps in the management server instrumentation directories. All files on the OVO management server that are newer than the corresponding file on the managed node are distributed. No instrumentation data is ever removed from the managed node, except if the `opcragt -purge` command line command and option is applied. This cannot be executed from the Administrator UI.

Heartbeat Polling of HTTPS Nodes

Heartbeat polling of managed nodes checks for following things:

- Does the managed node respond to ping.
- Is `ovbbccb` (HTTPS) or `rpcd` (DCE) or `llbd` (NCS) is reachable.
- Is `ovcd` (HTTPS) or `opcctla` (DCE) is reachable.
- Is the message agent (`opcmsga`) reachable.

NOTE

Other agent processes, such as `opcmona`, `opcle`, and `opcacta`, are not checked by the heartbeat polling but are monitored by the agent's health check.

If any of these processes dies and is not disabled, `ovcd` issues a message and automatically re-start the process.

Heartbeat polling of HTTPS nodes and DCE-based nodes is very similar. Heartbeat polling of OVO managed nodes is driven by the OVO request sender process `ovoareqsdr` and is divided into three phases:

- The request sender `ovoareqsdr` sends ping packages to check whether the node is reachable.
- The HTTPS agent communication broker is polled.
- OV Control RPC server is requested.

TIP

You can use the `RPC_only` mode, where the ping phase is omitted, to get through firewalls which have the ICMP filter enabled. In `RPC_only` mode, less checks are executed. Should a problem arise, the detail available from the error messages is reduced.

You can set different polling intervals per node.

HBP error messages of HTTPS nodes and DCE-based nodes are also very similar. For example, the message `DCE rpcd is down` for DCE-based agents corresponds to `communication broker is down` for HTTPS agents and is allocated the same error number.

Reduce Network and CPU Load

To reduce CPU load, HTTPS node heartbeat-polling does not use SSL.

Heartbeat polling includes the option `agent_sends_alive_packages`. When enabled, the agent regularly informs the OVO management server that it is working correctly by sending ping packages. The OVO management server only starts polling when it has not received an alive package from one or more managed nodes in the last period.

The server plays an active role only in failure cases and the alive packages are very small. This results in an extreme reduction of network and CPU load. This feature is of great benefit when large environments are managed with no firewalls between managed nodes and the OVO management server.

Remote Control of HTTPS Nodes

The `opcragt` utility is used to control agents from the OVO management server. All supported operations can be simultaneously executed on HTTPS nodes and non-HTTPS nodes. These operations includes start, stop, get status, primary manager switch, get and set configuration variables, as well as configuration distribution.

There is a wrapper called `opcagt` on HTTPS nodes. This utility can be used to perform remote control tasks by application launch from the operator's desktop. It allows to setup a common action definition for any kind of OVO managed nodes.

The output format of `opcragt -status` as well as for other `opcragt` operations looks identical for HTTPS nodes and DCE-based nodes. Error messages are also very similar.

Subagents are identified by names on HTTPS nodes and by numbers on DCE nodes. Therefore, you can specify aliases of the form:

<alias> <maps_to>

in the configuration file:

`/etc/opt/OV/share/conf/OpC/mgmt_sv/subagt_aliases`

The entries `1 EA` and `12 CODA` are pre-defined. To automatically transform the `-id 1` into `-id EA` for HTTPS managed nodes, enter the command:

opcragt -status -id 1 *<BBC_nodes_and_DCE_nodes_list>*

5 Working with HTTPS Managed Nodes

Configuring HTTPS Nodes

HTTPS nodes are configured in the same way as DCE-RPC- and NCS-RPC-based nodes and configured through the `Add`, `Modify`, and `Copy` Node windows in the OVO Administrator´s user interface or using the `opcnode(1m)` and the `Node Communication Options` and `Node Advanced Options` windows.

As OVO administrator, do the following for HTTPS nodes:

- Specify a new communication type `HTTP-Based` for supported platforms.
- Specify whether a node's IP address is static or dynamically assigned using DHCP. See “Managing HTTPS Agents on DHCP Client Systems” on page 219.

NOTE When changing the communication type between DCE and HTTPS, the DCE agent software is automatically removed. Local configuration or runtime data, including `opcinfo` file settings, ECS data and fact stores, Embedded Performance Component database files, are converted and re-used by the HTTPS agent.

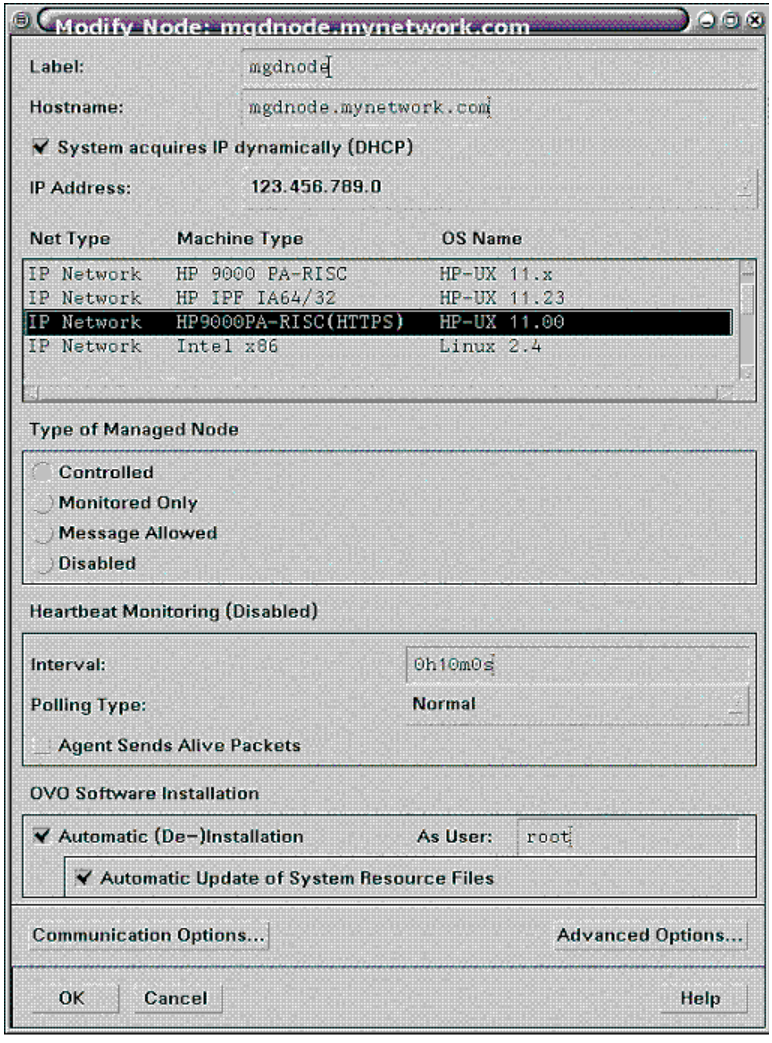
Security of HTTPS communication is achieved using certificates which results in some new steps being required to install HTTPS agents. The steps that you must complete are:

1. Install the OVO HTTPS agent software on the managed node through the `Add Node` window. The node automatically sends a certificate request to the OVO certificate server which is automatically granted. If auto-grant is disabled, the next two steps are also required.
2. Select the nodes to which you want to grant certificates from the `OVO Node Certificate Requests` window.
3. Grant the certificate requests to the selected nodes.
   The nodes for which certificates have been granted are added to the `Holding Area` (default) or in the configured layout group as specified in the configuration setting `OPC_CSA_LAYOUT_GROUP` in the namespace `opc`.

Install OVO Software Automatically on HTTPS Nodes

OVO software installation is controlled from the `Add Node` window, illustrated in Figure 5-1.

Figure 5-1 Add/Modify Node Window For an HTTPS Node



To install the OVO software automatically:

1. Open the `Add Node` window by selecting:

   `Actions: Node -> Add`

   from the menu bar of the `OVO Node Bank` window (see Figure 5-1) and enter the following information:

2. Enter a label used to identify the system.
3. Enter the hostname of the system.
4. Select a system/operating system combination.

NOTE

In a NAT environment (management server IP address is translated on the managed node side) the Windows HTTPS agent installation may hang. This is caused by ftp which is used during installation. The ftp connection to Windows hangs.

Install the HTTPS Agent software manually. FTP is unlikely to work. therefore, another file transport mechanism must be used.

5. Use the `System acquires IP dynamically (DHCP)` checkbox next to the IP address if you want specify that the IP address of the selected HTTPS node is dynamic. This is most useful when the node uses DHCP to get its IP address. Similarly, if the IP address of a node is changed manually and `Dynamic IP` is selected, the change is also updated in OVO. If DHCP is selected, OVO automatically deals with managed node IP address changes without causing any problems, without losing any messages or without creating an inconsistent or undefined state.

NOTE

`Dynamic IP` is only supported on HTTPS nodes. Dynamic change of hostname is not supported.

6. Select the type of managed node. `Controlled` is the default.

   `Type of managed node` is also accessible from the `OVO Node Defaults` window.

NOTE

On HTTPS managed node set to `Monitored Only`, automatic actions will execute, however, operator-initiated action will not.

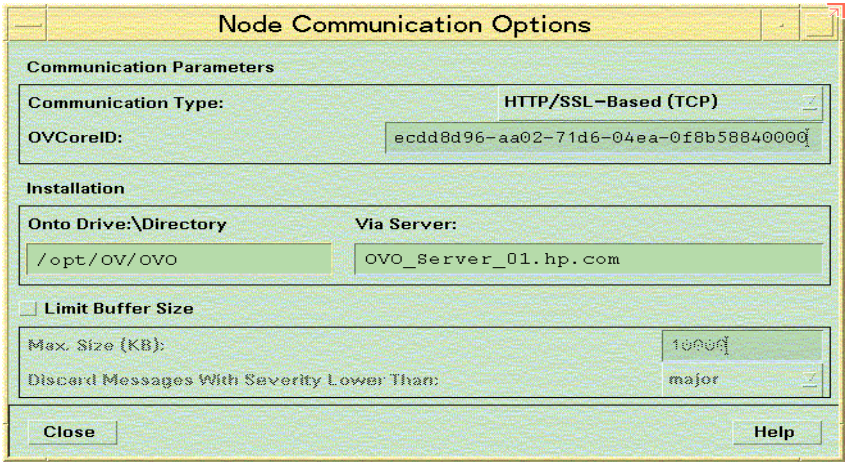
Setting `Message Allowed` as the node type prevents the distribution of software and instrumentation to that node.

Changing the `Type of Managed Node` for HTTPS nodes does not distribute a `nodeinfo` file to the managed node.

7. Enter the desired heartbeat polling settings (optional).
8. Select the `Automatic (De-)Installation` option when adding a managed node to the OVO environment (optional).

   The communication type and settings to be used by the node are displayed in the `Node Communication Options` window. To access this window, click the `Communication Options...` button on the `Add`, `Modify`, or `Copy` Node window for a node.

Figure 5-2 Node Communication Options Window



An HTTPS managed node displays `HTTPS` as its `Communication Type`. The unique identifier, `OVCoreID`, is displayed for reference.

Switching between communication type HTTPS and another communication type automatically changes the platform for the node and removes all values for this node that are only relevant for the newly selected communication type.

HTTPS is the default for new nodes. SNMP-based automatic agent platform detection for newly-added nodes always selects, if available, the HTTPS-based platform.

When you change a node's platform, all node, communication and advanced options are retained, where necessary. This way, switching a node to HTTPS-based management is simplified by retaining the existing settings and generally maintaining the original monitoring view.

The root directory for installation of the agent software is configurable for the HTTPS agent on Microsoft Windows nodes.

NOTE It is not possible to specify a customized log directory and maximum log size for HTTPS nodes, since the new OpenView file system layout and OpenView logging mechanism are used.

9. Information about HTTPS-based High Availability clustered systems that make up a virtual node can be specified under the Cluster Virtual Node section of the Node Advanced Options window if required. To access this window, click the Advanced Options... button on the Add, Modify, or Copy Node window for a node.

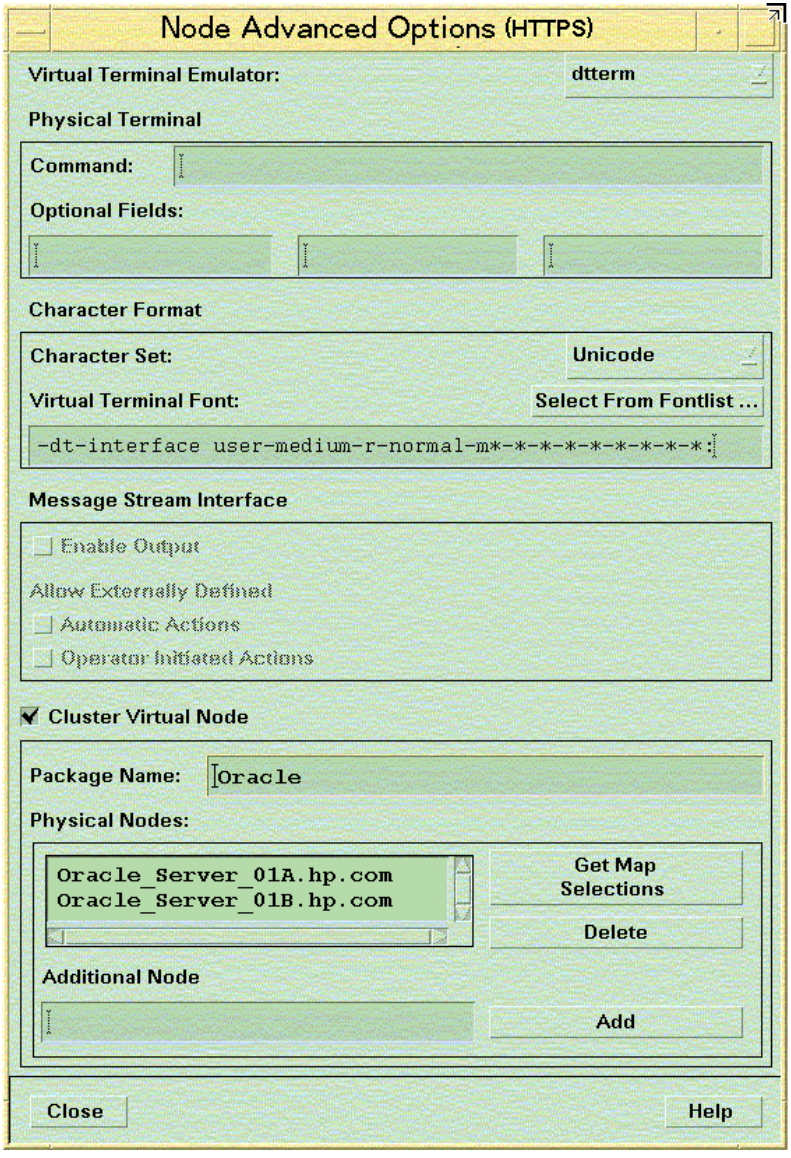
   If you have a virtual machine comprised of two or more systems being managed as HTTPS nodes, check the Cluster Virtual Node checkbox and enter the required information for the cluster and its systems.

   Enter the cluster HA Resource Group name that identifies the cluster in the mandatory field.

   Click the Add button to add the physical systems that make up the cluster package to the Cluster Virtual Node information.

NOTE The character set for HTTPS nodes is always set to Unicode.

Figure 5-3 Node Advanced Options Window



NOTE

Only OVO management server features are available for virtual nodes and one agent feature: distribution of policies and instrumentation to the virtual node. Automatically distributes policies and instrumentation to all physical nodes of the virtual node.

The following options cannot be used for virtual nodes:

- `Nodeinfo` and `mgrconf` cannot be distributed.
- `Agent Sends Alive Packets`.
- All software installation and related options.
- Node Type `Message Allowed`.
- `Limit Buffer Size`.

After installing the OVO software on a managed node, you must make sure that the certificates required by HTTPS communication are created and distributed. The default is for these to be generated automatically. These steps are explained in Chapter 6, “Working with Certificates,” on page 139.

Define Common Settings for Managed Nodes

You can define settings on the management server, which are deployed to the managed nodes at installation time. Basic parameters, such as communication ports or http proxy settings, that are used by many nodes can be define this way. Common scenarios include:

- Need to install many OVO agents on a subnet or domain. Due to firewall restrictions, the default port of the Communication Broker (383) cannot be used and you want to avoid having to manually set the Communication Broker port on every node during agent installation.
- Configure default settings for installation of managed nodes at a central point as the nodes of a subnet or domain share many settings.
- OVO agents are manually installed on a subnet behind a firewall. Common parts of the installation can be automated.

You can maintain these common settings on the OVO management server using the file:

`/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults`

A sample configuration file with examples of how to set up parameters is available at:

`/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults.sampl`

Take a copy of the `bbc_inst_defaults.sampl`, rename it `bbc_inst_defaults`, and modify in accordance with the syntax specified in the sample file.

Allocate a Specific OvCoreId to a Managed Node

If you want to allocate a specific OvCoreId for a new node, manually add it as follows before starting the agent software installation:

On the OVO management server, enter one of the following commands:

`opcnode -chg_id ... id=<id>`

or

`opcnode -add-node ... id=<id>`

During agent installation, the OvCoreId from the OVO database is used for the specified managed node.

This is recommended when re-installing a node managed by many management servers. Re-using the original OvCoreId avoids having to update all the OVO management servers.

When installing certificates manually, everything is prepared on the OVO management server before an agent is installed, including creating an OvCoreId, generate a certificate, add the node with the new OvCoreId to the database. Only after these steps can the agent software be installed on the managed node. Finally the certificate must be copied to the managed node.

Installing on Window Managed Nodes

Set Startup Type on Windows Managed Nodes

Windows does not have a boot startup system comparable to UNIX. To start `ovcd` on Windows independent of user login, `ovcd` is registered as a service. Based on the default `START_ON_BOOT` value, the installation sets the service startup to `Automatic` or `Manual`. However, subsequent changes to the `START_ON_BOOT` flag have no effect on the `ovcd` service registration.

On Windows, you must change the service startup manually as follows:

1. Go to `Start` -> `Settings` -> `Control Panel` -> `Administrative Tools` -> `Services`
2. Double-click the `HP OpenView Ctrl Service` and from the `General` tab of the `Properties` window, set the required `Startup Type`.

This behavior can be noticed in the following use cases:

Agent Installation from the GUI
When add the managed node to the Node Bank, you can also select the option `Automatically update system resource files` in the `Add Node` window. If you select this option for a Windows node, the `ovcd` control service is registered with start-up type Automatic, and the agent starts automatically after a reboot. If you do not select this option, the ovcd service is registered with start-up type `Manual`. In this case, you must manually start the agent after each reboot.

Manual Agent Installation
Using `opcactivate`, you can specify the `-nb` option (or an equivalent option) which has the same effect as selecting `Automatically update system resource files` from the OVO GUI.

Settings selected during the agent installation cannot be changed using OVO. To change these settings, use the Windows Control Panel.

Installation Log File on Windows Managed Nodes

The Windows agent install script `opc_inst.vbs` creates the `opc_inst.log` log file. Installation steps and results are automatically records in this file. While the script is running it resides in `%TMP%` of the user under which the installation is run. The default is `Administrator`.

It is copied, after a successful installation, to *<OVInstDir>*`\data\log`.

Configure a Windows Installation Server

OVO HTTPS Agents can be fully automatically installed onto Windows systems using an installation server system. An installation server is a regular Windows managed node with an OVO HTTPS agent installed. Once the OVO HTTPS agent is installed, you can install any further Windows HTTPS nodes from the OVO Admin GUI or using `inst.sh` on the OVO management server without the need to manually execute the `opc_inst.vbs` utility on the target nodes.

NOTE It it is necessary to set the installation server in the `Communication Options` window of the target nodes.

The following guidelines describe the specific configurations required for the OVO HTTPS agent acting as installation server:

- The Windows system hosting the OVO agent which acts as installation server must be in the OVO Node Bank and must be of the same communication type (HTTPS) as the target nodes.
- It is recommended to use a dedicated system as an installation server system because it is necessary that the OVO agent acting as the installation server runs with extensive capabilities (see below). This means, that this OVO agent should not receive any policies or instrumentation to avoid accidental or malicious start of functionality with these capabilities.
- The OVO agent must run as a user who is able to access the target systems using standard Windows access mechanisms. In particularly it must be able to copy files to the target system as the software is transferred to the Windows nodes using a windows share.

To configure an OVO HTTPS managed node to act as a Windows Installation Server, complete the following steps:

1. Install and start a Windows service on the target system. This can be accomplished by making this OVO Agent run as either:
   - A domain administrator
   - Any other user who has:
     - Networking capabilities.
     - Windows pass-through authentication is in place (identical user/password on both nodes).
     - Administrative capabilities on the target nodes.

TIP

For information about Windows user rights and privileges, refer to the Miscrosoft documentation at the following location:

`http://www.microsoft.com/technet/security/prodtech/`

To install Windows agent software using an installation server, the OVO agent acting as installation server cannot run as SYSTEM (which is the default) because it is not able to access remote systems. Instead, this agent must run under an identity, which is able to access the target managed node using regular Windows access mechanisms to the admin drive.

To change the user under which the OVO agent acting as an installation server runs, perform the following steps:

2. Stop the OVO agent with the command:

   ```
   ovc -kill
   ```

3. Create the Windows user account to be used.
4. Make the following user and permission changes to the selected Windows user account to make sure that the agent is running with the appropriate priveleges as well as the agent directory structure has the appropriate priveleges set:
   - Changes the start-up user of the Windows Service.
   - Change the permissions of OVO data files.

Entering the following command:

```
cscript <InstallDir>\bin\ovswitchuser.vbs -existinguser
<user> -existinggroup <group> -passwd <user_pwd>
```

This command requires a few minutes to execute.

5. Due to a limitation in `ovswitchuser.vbs` , complete the following steps:

   a. Open the `Control Panel` -> `Administrative Tools` -> `Services`

   b. Change the Windows user to one which is configured to run the service `HP OpenView Ctrl Service` and re-enter the user password.

NOTE

The `SYSTEM` account is not sufficient to do the install-server tasks as it does not have the appropriate network rights. Because of this, you must change the agent user on the installation server to an existing administrative account with sufficient network rights. This user is not created automatically.

   c. Confirm that the user has been given the `Start as service` capability.

6. Start the agent with the command:

   `ovc -start`

7. Verify that the processes are running and note the user under which they are running as follows:

   a. **`ovc -status`**

   b. Open the `Task Manager` and display the user.

Migrate a DCE Agent to an HTTPS Agent

WARNING

The major version of your OVO agent software must not be higher than the version of your OVO management server software. For example, an OVO version A.08.x HTTPS agent cannot communicate with a OVO version A.07.1x management server.

If you are operating in a flexible management environment with A.07.1x and A.08.x management servers, make sure that all OVO agents remain on version A.07.1x until all management servers have been upgraded to OVO version A.08.x.

NOTE

The `opcinfo` file is converted when you upgrade an OVO 7.1 agent to an HTTPS agent. A copy is saved to the local `/tmp/opcinfo.save` file.

To migrate a DCE agent to an HTTPS agent:

1. On the OVO management server:

   Prepare the agent profile generation by checking whether the contents of the `bbc_inst_defaults.ini` file is appropriate for the node. This step should only be necessary once for complete subnets or domains.

2. Select the node from the `Node Bank`.

   From the menu bar of the `OVO Node Bank` window Administrator's UI (see Figure 5-1), open the `Modify Node` window by selecting:

   ```
   Actions: Node -> Modify
   ```

   Select the agent type from the `Modify Node` window. For example, `MS Windows (HTTPS)`.

3. Install the new agent software by selecting:

   `Actions: Agents -> Install / Update OVO Software and Configuration`

   Templates, actions, commands and monitors are only re-installed on the managed node system with the `Update OVO Software and Configuration` selection.

   When you are asked whether the DCE agent should be de-installed, confirm to continue with the HTTPS agent installation.

   Local DCE-specific agent configurations are automatically converted to HTTPS agent formats. These include opcinfo settings, ECS data stores and fact stores, Embedded Performance Agent database files.

4. After the agent software installation has completed on the remote node, you can check the status of the installation by entering one of the following commands.

   - On the managed node system:

     `ovc -status`

   - On the OVO management server system:

     `opcragt -status <nodename>`

   A message confirming successful distribution should be displayed in the Message Browser.

Migrate an HTTPS Agent to a DCE Agent

It is not possible to directly migrate an HTTPS agent to a DCE agent.

NOTE

The `opcinfo` file is converted when you upgrade an OVO 7.1 agent to an HTTPS agent. A copy is saved to the local `/tmp/opcinfo.save` file.

When migrating an HTTPS agent to a DCE agent, it is not possible to convert the configuration settings into the opcinfo file. You must make a copy of the configuration information from the `eaagt` namespace. This data can be displayed before removing the HTTPS agent with the command:

`ovconfget`

After installing the DCE agent, manually enter the configuration information into the `opcinfo` file and delete the = signs between each key and value pair.

To migrate an HTTPS agent to a DCE agent:

1. When migrating an HTTPS agent to a DCE agent, it is not possible to convert the configuration settings into the `opcinfo` file.

   Display this data before removing the HTTPS agent with the command:

   `ovconfget`

   Make a copy of the configuration information from the `eaagt` namespace.

2. De-install the HTTPS agent from the managed node.

3. In the Motif GUI, change the communication type for this managed node from HTTPS to DCE:

   Select the node from the `Node Bank`.

   From the menu bar of the `OVO Node Bank` window Administrator's UI (see Figure 5-1), open the `Modify Node` window by selecting:

   `Actions: Node -> Modify`

Select the agent type from the `Modify Node` window. For example, `MS Windows`.

4. Manually install the DCE agent on this managed node:

   `Actions: Agents -> Install / Update OVO Software and Configuration`

   Templates, actions, commands and monitors are only re-installed on the managed node system with the `Update OVO Software and Configuration` selection.

5. After installing the DCE agent, manually enter the configuration information from the HTTPS installation into the `opcinfo` file and delete the = signs between each key and value pair.

6. After the agent software installation has completed on the remote node, you can check the status of the installation by entering one of the following commands.

   - On the managed node system:

     `ovc -status`

   - On the OVO management server system:

     `opcragt -status <nodename>`

   A message confirming successful distribution should be displayed in the Message Browser.

Installing HTTPS Managed Nodes Manually

In some situations, you may want to install the OVO HTTPS agent software without using the management server. This manual installation enables you to prepare the system to become an OVO managed node when it is later connected to the network. Manual installation is useful if you are preparing many systems in a central location, or if you want to avoid the network connection necessary for standard installation. Manual installation may be necessary for systems behind a firewall or behind an HTTP proxy.

Certificate Installation Tips

If an agent is installed before it is added to the OVO management server node bank, a certificate request is issued from the node, but it remains in the list of pending certificate requests in the `Node Certificate Requests` window, because it cannot be automatically mapped to any node from the node bank.

It is possible to add a node to the `Holding Area` from the `OVO Node Certificate Requests` window by selecting `Certificate Request` and clicking the `Add Node to Node Bank` button. The `Add Node` window opens and you can edit the fields and then add nodes to the `Holding Area`. Certificate requests are then automatically mapped to that node, but they are not granted. An administrator must manually grant the certificate requests as required.

When a certificate request is granted, the certificate server signs the certificate and sends it to the certificate client. The certificate client now installs the certificate on the node.

NOTE Remote certificate deployment type can be used during manual agent installation.

After the certificate is installed on the node, either by using remote certificate deployment or by manually importing the certificate to the node, the certificate client notifies the certificate server that the

certificate has been successfully installed. The certificate server notifies the certificate server adapter and certificate server adapter then sets the `Node Certificate State` in the database to `Installed`.

For more detailed information about handling certificates, refer to Chapter 6, “Working with Certificates,” on page 139.

For troubleshooting certificates handling, refer to “Certificate Deployment Problems” on page 277.

Install an Agent Manually from Package Files

For an Agent installation you need super-user rights, for example, `root` on UNIX and `Administrator` on Windows. This is required because native installers, such as `swinstall` on HP-UX and MSI on Windows, which are used for the OVO agent installation, need super-user rights to work.

To install an agent manually from package files, complete the following steps:

1. **Check Node Status and Select Configuration**

   - Check if the system is already added to the `Node Bank`. Add the system to the `Node Bank` if desired.
   - Decide whether the managed node installation should have:
     - — No configuration (only if system is not yet in the Node Bank)
     - — Customized configuration (system must already be in the Node Bank)
     - — Default configuration

     The type of managed node installation that you select, determines which of the following steps you are required to complete.

2. **Create a Default Profile**

NOTE This step is required only if the managed node is already in the Node Bank and the configuration has been customized.

On the OVO management server system, create a default profile with the command:

/opt/OV/bin/OpC/opcsw -create_inst_info *<nodenames>*

For each managed node from *<nodenames>*, the following file is created:

`/var/opt/OV/share/tmp/OpC/distrib/<hex_IP_addr>.i`

The file contains the installation defaults for the managed node with IP address *<hex_IP_addr>*. The file is automatically copied to the target managed node via remote agent installation (`inst.sh`) or you can use it for manual agent installation.

To check the mapping between managed node name and its hex_IP_addr use:

/opt/OV/bin/OpC/install/opc_ip_addr *<nodename>*

This will print you the resulting hex_IP_addr for the specified managed node.

After the system is added to the OVO Node Bank, copy the `/var/opt/OV/share/tmp/OpC/distrib/<hex_IP_addr>.i` profile file to the managed node system.

3. **Copy the OVO Agent Components to the Managed Node**

   Copy the OVO managed node packages, installation script and package description to a temporary directory on the managed node.

   The files on the OVO management server that you require are:

   - `HPOvBbc.<platform>`

     `HPOvBbc.xml`
   - `HPOvConf.<platform>`

     `HPOvConf.xml`
   - `HPOvCtrl.<platform>`

     `HPOvCtrl.xml`
   - `HPOvDepl.<platform>`

     `HPOvDepl.xml`
   - `HPOvEaAgt.<platform>`

     `HPOvEaAgt.xml`

- `HPOvPCO.<platform>`

  `HPOvPCO.xml`
- `HPOvPacc.<platform>`

  `HPOvPacc.xml`
- `HPOvPerlA.<platform>`

  `HPOvPerlA.xml`
- `HPOvSecCC.<platform>`

  `HPOvSecCC.xml`
- `HPOvSecCo.<platform>`

  `HPOvSecCo.xml`
- `HPOvXpl.<platform>`

  `HPOvXpl.xml`
- `opc_inst` (UNIX) or `[cscript] opc_inst.vbs` (Windows)

The following are the optional language packages:

- `HPOvLcja.<platform>`

  `HPOvLcja.xml`
- `HPOvEaAja.<platform>`

  `HPOvEaAja.xml`
- `HPOvEaAes.<platform>`

  `HPOvEaAes.xml`
- `HPOvEaAko.<platform>`

  `HPOvEaAko.xml`
- `HPOvEaAzS.<platform>`

  `HPOvEaAzS.xml`

The `.xml` files are common to all architectures.

The depot files for the supported platforms are identified with a platform-specific extension *<platform>*. The value of *<platform>* is as follows:

`depot.Z`. Files for HP-UX nodes

`sparc.Z.`	Files for Solaris nodes
`rpm.gz.`	Files for Linux nodes
`msi.`	Files for Windows nodes

The files are located in the following directory on the management server:

`/<OvDataDir>/share/databases/OpC/mgd_node/vendor/ \`
`<vendor>/<newarch>/<ostype>/A.08.10.xx/RPC_BBC/`

where, for example, *<vendor>/<newarch>/<ostype>* is:

`hp/pa-risc/hpux1100`

`hp/ia64-32/hpux1122`

`ms/x86/winnt`

`ms/ipf64/winxp`

`linux/x86/linux24`

`linux/ipf64/linux24`

`sun/sparc/solaris7`

4. **Install the Agent Software**

   On UNIX systems, you may need to change the permissions of the agent installation script to ensure that it can be executed. If you need to change the permissions, enter the command:

   `chmod +x ./opc_inst`

   There are three methods of installing and configuring an agent manually:

   - Default configuration
   - No configuration (to be configured later)
   - Customized configuration (configuration file must be specified)

   Select the type of configuration and complete the steps from the appropriate section below.

- **Managed Nodes with Default Configuration**

  For managed nodes to be installed with the default configuration, go to the temporary directory to which you have copied the packages and start the agent installation script `opc_inst` by entering the command appropriate for your operating system:

  For UNIX systems:

  ```
  ./opc_inst -srv <management_server_name>
  -cert_srv <certificate_server_name>
  ```

  For Windows systems:

  ```
  [cscript] opc_inst.vbs -srv <management_server_name>
  -cert_srv <certificate_server_name>
  ```

  Wait until installation and configuration on the remote managed node are finished.

- **Install, Configure, and Activate Customized Managed Nodes**

  To configure a customized configuration and activate the profile created in step 2 for systems already in the Node Bank, use one of the following commands:

  — **`opc_inst -configure <hex_IP_addr>.i`**

  — **`opcactivate -configure <hex_IP_addr>.i`**

  Wait until installation and configuration on the remote managed node are finished.

  The settings are placed under `local_settings` and have highest priority in the same way as `opcinfo` settings for DCE nodes.

  You can maintain these common settings on the OVO management server using the file:

  `/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults`

  A sample configuration file with examples of how to set up parameters is available at:

  `/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults.sampl`

  Take a copy of the `bbc_inst_defaults.sampl`, rename it `bbc_inst_defaults`, and modify in accordance with the syntax specified in the sample file.

- **Pre-install Managed Node Software without Configuration**

  If you want to pre-install the managed node software on a system with no immediate configuration and prepare the system for later use, for example, by another department, enter the following command and do not specify an OVO management server:

  ```
  ./opc_inst -no_start
  ```

  The software is installed but the processes are not started.

  When the node needs to be activated and the processes started, enter one of the following commands, depending on the type of configuration you want to apply.

  To apply the default configuration, enter the command:

  ```
  ./opcactivate -srv <management_server_name> \
  -cert_srv <certificate_server_name>
  ```

  To apply a customized configuration, enter the command:

  ```
  opcactivate -configure <hex_IP_addr>.i
  ```

  Wait until installation on the remote managed node is finished.

TIP

If you want to reset the `MANAGER_ID` parameter after a failed `opcactivate` call, manually set the `MANAGER_ID` or establish communication between the OVO management server and the managed node and run `opcactivate` again. These methods are described below.

— On the management server system, enter the command:

```
/opt/OV/bin/ovcoreid -ovrg
<management_server_name>
```

On the managed node, enter the following command and specify the value of the `OvCoreId` of the OVO management server:

```
ovconfchg -ns sec.core.auth -set MANAGER_ID
<management_server_ovcoreid>
```

— Make sure that the following command from the managed node is successful:

`bbcutil -ping http://<management_server_name>`

Call `opcactivate` again.

This might not be possible in all types of environments, for example where HTTP without SSL is not possible from the managed node to the management server.

5. **Examine the Managed Node Logfile**

   If any errors occurred during installation, correct the problems and reinstall. Errors are written to the native installer logfile for the managed node. For example on HP-UX, the logfile is at the following location:

   `/var/adm/sw/swagent.log`

   Alternatively, `opc_inst` creates a logfile on all platforms in:

   `/<OvDataDir>/log/opc_inst.log`

6. **Map the Certification Request**

   On the OVO management server, if necessary, map the certification request to the newly installed managed node.

   a. From the `Node Bank` window, select the following menu sequence:

      `Actions-> Node-> OVO Certification Request`

      The `OVO Node Certificate Requests` window is displayed.

   b. If the certificate request from the newly installed managed node is not mapped, select this request.

      The `Add Node to Node Bank...` button is enabled.

      Click the `Add Node to Node Bank..` button. The `Node Modify` window for this node is displayed. Click `OK` in the `Node Modify` window.

      The node is entered into the `Holding Area`.

7. **Grant the Certification Request**

   On the OVO management server, grant the certification request for the newly installed node.

   a. In the `OVO Node Certificate Request` window select the mapped request for the node that you have newly installed.

      Click the now enabled `Grant` button to grant the certificate request.

   b. Close the OVO Certification Request window

8. **Add Pre-installed Nodes to the OVO Node Bank**

   For pre-installed nodes only, from the OVO management server, add them to the OVO Node Bank.

   a. Open the `Holding Area` window.

   b. Move the node to the Node Bank.

   c. Drag and drop the node onto a node group in the `OVO Node Group Bank` window.

      or

      use the `opcnode` tool:

      For example for an HP-UX 11 node, enter the command:

      ```
      /opt/OV/bin/OpC/opcnode -add_node
      mach_type=MACH_BBC_HPUX_PARISC \
      net_type=NETWORK_IP group_name=<node_group> \
      node_name=<node_name> node_label=<node_label>
      ```

      Refer to the `opcnode` man page for further details.

   d. If the message browser is already open, request a `Browser Reload`.

      You all messages from the node should be displayed.

9. **Update the Database and Start Heartbeat Polling for the Node**

   After the node is connected to the network:

   From the command line, enter the following command on the OVO management server:

   ```
   /opt/OV/bin/OpC/opcsw -installed <node>
   ```

10. **Verify that the OVO Agent is Running on the Managed Node**

    Enter the following:

    ```
    /opt/OV/bin/OpC/opcragt -status <node>
    ```

NOTE Valid certificates must be installed on the managed node, otherwise the agent will not run and the verification will fail.

Comparing `opc_inst` and `opcactivate`

- Manually installing the agent software using `opc_inst` also activates the node. The `opc_inst` tool installs the software packages and calls `opcactivate`. `opcactivate` sets some initial configuration parameters. A separate activation step is not necessary.
- The purpose of `opcactivate` is to configure the agent by establishing the three fundamental configuration settings:

  sec.core.auth:MANAGER

  Corresponds to the `-srv` option of `opc_inst` and `opcactivate`.

  sec.cm.client:CERTIFICATE_SERVER

  Corresponds to `-cert_srv` option of `opc_inst` and `opcactivate`.

  sec.core.auth:MANAGER_ID

  The `MANAGER_ID` setting defines who is allowed to access the agent from outside. By default, this is the OVO management server and therefore you need its `core_ID`.

  The is no equivalent `opc_inst` or `opcactivate` option for this parameter. Instead `opcactivate` tries to contact the OVO management server (`MANAGER_ID` setting) using `bbcutil -ping` (no SSL). If it cannot reach the management server, the `MANAGER_ID` parameter cannot be set and management server - agent communication is not possible even not if you have a valid certificate on the agent.

- When working with agent profiles:
  - All 3 settings from above are automatically included.
  - Any settings available for the managed node in the following defaults file are included:

    `/etc/opt_OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults`
  - The `core_ID` of the managed node is included if it is available from the management server database.

Install Managed Nodes Using Clone Images

When installing a large number of similar managed nodes, it may be advantageous to create a clone image of a typical system configuration and use this as the basis for installing the other systems. This section provides basic information on using cloned images. If you require further details, refer to the white paper titled: *HP OpenView Operations Installing Agents Using Clone Images*. This is available from the following web site:

`http://ovweb.external.hp.com/lpe/doc_serv/`

Select `operations for unix` and version `8.x`.

From an OVO point of view, there are two levels of clones that could be created:

- Agent software installed on OVO managed node system.
- Agent software installed with policies deployed to OVO managed node system.

The clone image should not contain the unique identifier of the original managed node, the `OvCoreId`. If all cloned systems contain the same identifier, there will be a significant amount of manual reconfiguration required before these systems are recognized as individual managed node with no confusion.

To install the OVO managed node software using a cloned image, complete the following steps:

1. Install the OVO managed node software and configure a system that will be cloned.
2. Stop all managed node processes with the command:

   `ovc -kill`

3. Display all installed certificates from the managed node to be cloned by executing the following command:

   `/opt/OV/bin/ovcert -list`

The output of the following form is displayed:

```
+---------------------------------------------+
| Keystore Content                            |
+---------------------------------------------+
| Certificates:                               |
|     edb87a09-1511-75ff-13c1-f6aef454aa2b (*) |
|     edb...                                  |
+---------------------------------------------+
| Trusted Certificates:                       |
|     CA_edb66a23-1422-04ff-77c1-f1aef555aa1b  |
|     CA_edb...                               |
+---------------------------------------------+
```

4. Remove all installed certificates from the managed node to be cloned by executing the following command:

   /opt/OV/bin/ovcert -remove *<certificate name>*

   For example:

   ```
   /opt/OV/bin/ovcert -remove \
   edb87a09-1511-75ff-13c1-f6aef454aa2b \
   CA_edb66a23-1422-04ff-77c1-f1aef555aa1b
   ```

5. Check that the `CERT_INSTALLED` parameter is set to `FALSE` with the following command:

   /opt/OV/bin/ovconfget

   If the parameter is not correctly set, set it with the command:

   **/opt/OV/bin/ovconfchg -ns sec.cm.certificates **
   -set CERT_INSTALLED FALSE

6. Remove the `OvCoreID` value of the managed node to be cloned by executing the following command:

   /opt/OV/bin/ovconfchg -ns sec.core -clear CORE_ID

7. Make a clone image of the system without certificates and the `OvCoreID` value.

8. Copy the image to the new managed node system.

9. Create a new the `OvCoreID` value on the new managed node:

   ovcoreid -create

NOTE Use the **-force** option if the `OvCoreID` value was not deleted and it needs to be overwritten.

10. Run the `opcactivate` command to send a certificate request to the OVO management server:

    ./opcactivate -srv <*srv_name*>

NOTE Care must be taken if policies were already deployed to the managed node that was cloned. If new managed nodes created from the clone are configured to report to a different OVO management server than that managing the original managed node, the policies will no longer be trusted and they are signed by the Certificate Authority of the original OVO management server. To trust these policies, add the hostname of the original OVO management server as a secondary manager in the mgrconf file on the new managed node.

De-installing Agents

You can automatically or manually de-install HTTPS agents.

De-install Agents Automatically

To find out how to de-install agents automatically, see the *OVO Administrator's Reference*.

De-install Agents Manually

To de-install an OVO agent from an HTTPS managed node manually, execute the following steps.

For UNIX managed nodes:

1. Go to the installation directory:

   cd /opt/OV/bin/OpC/install

2. Enter the following command:

   ./opc_inst -r

For Windows managed nodes:

1. Stop all OVO agents running on the managed node.
2. Enter the following command:

   $INSTALLDIR\bin\OpC\install\opc_inst.vbs -r

De-installation Errors

If errors occur during the de-installation, check the local de-installation log files. Errors are written to the native installer logfile for the node. For example on HP-UX, the logfile is at the following location:

`/var/adm/sw/swagent.log` and `/var/adm/sw/swremove.log`

For Windows managed nodes, the logfile is:

`%SYSTEMROOT%\temp\inst.log`

Alternatively, `opc_inst` creates a logfile on all platforms in:

`/<OvDataDir>/log/opc_inst.log`

6 Working with Certificates

Creating and Distributing Certificates

Certificates are needed for network communication using the Secure Socket Layer (SSL) protocol with encryption. Server and client authentication are enabled. Managed nodes of the managed environment are identified using certificates. The “SSL handshake” between two managed nodes only succeeds if the issuing authority of the certificate presented by the incoming managed node is a trusted authority of the receiving managed node.

You can install certificates automatically, and manually. Please refer to the following sections for further information.

- “Deploying Certificates Automatically” on page 143.
- “Certificate Generation for Manual Certificate Deployment” on page 149.
- “Manual Certificate Deployment with Installation Key” on page 154.

Certificate installation is monitored with OVO messages. After a certificate request has been granted automatically, a notification message confirming the successful deployment of a certificate is sent to the message browser. If a certificate request is not automatically granted, a message in the message browser indicates the reasons for request denial and the steps that an administrator must take to solve the problem.

Certificates are managed from the `Node Certificate Requests` window. To open this window, select:

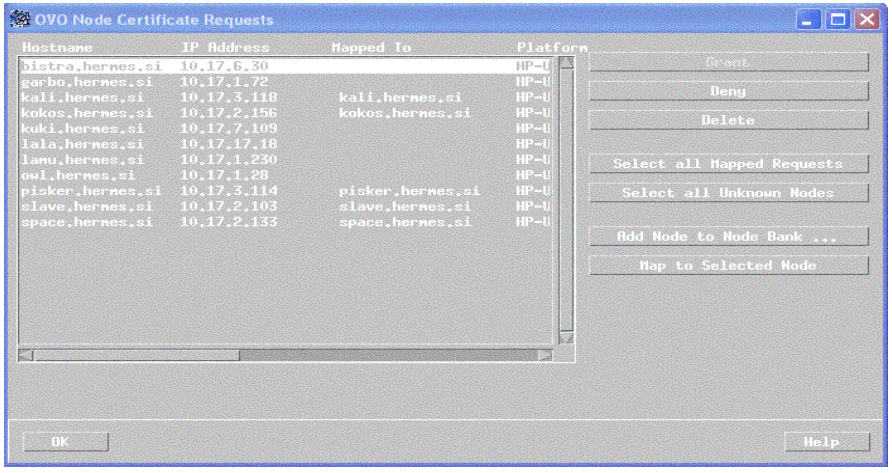
`Actions` → `Node` → `OVO Certificate Requests`

From the `Node Certificate Requests` window you can:

- Grant, deny, or delete certificate requests.
- Map certificate requests with the corresponding node from the Node Bank.
- Track certificate request flow.
- Add nodes to the Holding Area.

Initiating an action on selected nodes displayed in the node listbox, such as [Grant], [Deny], and [Delete], executes the action and removes the nodes from the listbox. The contents of the list may also be refreshed by the Certificate Server and the window automatically reloads the list every 10 minutes.

Figure 6-1 Node Certificate Requests Window



Node Information in the Node Certificate Requests Window

Hostname Hostname of the node that initiated the certificate request (not a unique identifier).

IP Address IP address of the node that initiated the certificate request (not a unique identifier).

OvCoreID The only unique identifier of an OVO HTTPS node. When you grant a request, you also grant all communication originating from the node with this OvCoreID. The hostnames can be changed, but the OVCoreID remains the unique identifier of the node.

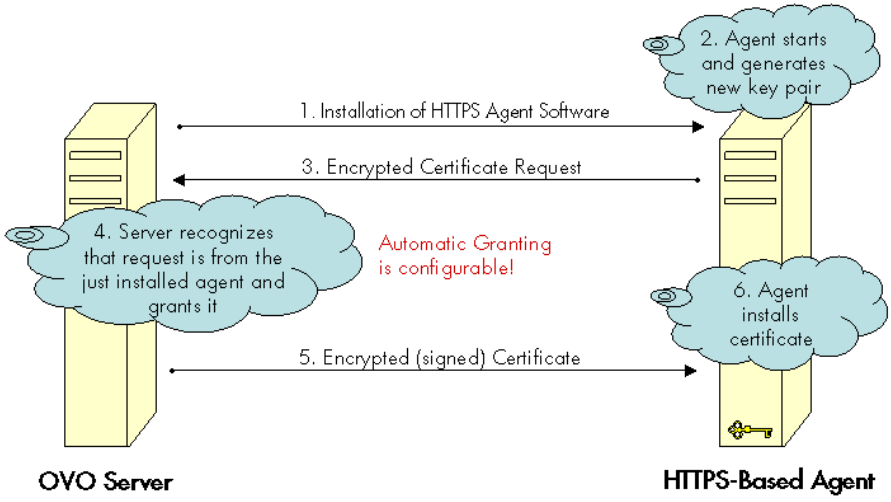
Mapped to Hostname of the node to which listed certificate requests are mapped. For requests that are not yet mapped, the `Mapped to` column is empty. Clicking `[Select all Mapped Requests]` selects all certificate requests having a hostname listed in the `Mapped to` column. See “Map to Selected Node” on page 148.

Platform Operating system of OVO managed node.

Deploying Certificates Automatically

The most common certificate deployment method is to let OVO create, grant and distribute certificates automatically. Figure 6-2 illustrates how OVO issues certificates to HTTPS managed nodes.

Figure 6-2 Certificate Deployment Process



After the HTTPS agent software is installed on a managed node system, the certificate management client on the node system creates a private key and a certificate request. A secret key is used to encrypt the certificate request which is sent over the network to the server system. Automatic granting is the default configuration and the autogrant interval is set to 30 minutes. If a request arrives after the allowed time interval, it must be handled manually using the Node Certificate Requests Window. If you wish to change this interval, use the following command:

```
ovconfchg -ovrg server -ns opc -set \
OPC_CSA_AUTOGRANT_INTERVAL <time interval in minutes>
```

If the message is encrypted with the correct key, the receiving management server trusts the sender. This does not provide full security, and is not recommended for highly secure environments but is more

secure than transmitting the requests as plain text. This mode is only used for transmitting the certificate request and the signed certificate, which should be a short period of time.

In secure environments, it is recommended that automatic granting of certificate requests is disabled and that an administrator assesses each request before granting those that are valid. You can do this with the command:

```
ovconfchg -ovrg server -ns opc -set \
OPC_CSA_USE_AUTOGRANT <TRUE|FALSE>
```

However, manual installation of certificates is the only fully secure method.

NOTE

An OpenView secret key is part of the HP Openview HTTPS security software and is used by default for all HP OpenView HTTPS-based applications. Every installation uses the same secret key.

A configurable secret key is a user configured key that replaces the OpenView secret key. This can be done before the management environment is setup. Ensure that every system that may request a certificate is using the same secret key as the certificate server.

Using a configured secret key ensures that a client system is not able to request a certificate from a foreign certificate server system, for example another HP OpenView installation.

NOTE

The Certificate Server system must be setup and active before certificates can be generated and distributed.

To automatically deploy certificates, install the HTTPS agent software on a managed node system. After the installation, the following steps are executed by OVO:

1. A new public/private key pair is generated on the managed node system by the certificate management client.
2. The managed node system initiates a certificate request on the node system.
3. The generated private key is stored in an encrypted file.

4. The certificate request is encrypted with the secret key and sent to the Certificate Server system (using a non-SSL connection as the node system does not yet have a valid certificate).
5. After the certificate request has been decrypted successfully on the Certificate Server it is added to the pool of pending certificate requests and a notification is sent to all registered components, and corresponding entry in the OVO Event Browser is also displayed.
6. The certificate request is either granted or denied by matching certain preconfigured criteria. For example, the request was made within 2 minutes of the HTTPS agent software being installed on the node system.

NOTE

Granting of a certificate request is the most security sensitive step in this process. The instance that grants the request should have a good reason to do this. An example would be an administrator who is waiting for a request after deploying a package to the node that now requests a certificate from the certificate server.

7. If the request is granted, the certificate request is signed by the Certificate Server. The signed certificate is then encrypted with the secret key and sent to the node system.

   If the certificate request is denied, the server system sends a message to the node system indicating that the request has been rejected and corresponding entry in the OVO Event Browser is also displayed.
8. The Certificate Client on the node system receives the response. If the request has been granted, it installs the new certificate and is now ready to use SSL for authenticated connections.

   If the certificate request has been denied, the Certificate Client stores this information to prevent an automatic retry.

Managing Certificates for HTTPS Managed Nodes

Certificate management is handled from the `OVO Certificate Requests` window, illustrated in Figure 6-1. To open the `OVO Certificate Requests` window, use one of the following options:

- Click the `[Actions]` menu in the `Node Bank` window and select `Node: Add...`, and then `OVO Certificate Requests` menu item from any node-related submap.
- Right-click a certificate-related message and select the `OVO Certificate Requests` menu item.

Actions Available from the Node Certificate Requests Window

Grant

Grant selected certificate request(s). Only mapped requests can be granted. After the operation is completed, the certificate server automatically refreshes the hostname list.

Certificate requests which are not successfully granted remain selected, and an error message is displayed.

If multiple certificate requests are selected, any unmapped requests are ignored and a message is displayed informing you that unmapped certificate requests cannot be granted. If only unmapped certificate requests are selected, the `[Grant]` button is deactivated (gray).

Deny

Deny selected certificate requests. After the operation is completed, the certificate server automatically refreshes the hostname list. You can deny any certificate request, mapped or not. The `[Deny]` button is active whenever a certificate request is selected.

Delete

Delete selected certificate requests. After the operation is completed, the certificate server automatically refreshes the hostname list. You can delete any certificate request. The `[Delete]` button is active whenever a certificate request is selected.

Select all Mapped Requests

Select mapped certificate requests. This button is always active. If no certificate request from the list of queued requests is selected, pressing this button results in selecting all mapped requests in the list. If one or more certificate requests are selected, pressing this button de-selects all unmapped requests from the originally selected requests.

Select all Unknown Nodes

Select requests originated by nodes that do not exist in the `Node Bank`. If one or more certificate requests are selected, pressing this button de-selects all nodes that are not in the `Node Bank` from the originally selected requests.

Add Node to Node Bank

Add nodes from which certificate requests are being originated. This button is active if either of the following conditions are met:

- One or more unmapped certificate requests are selected.
- There are one or more certificate requests where `Hostname` is not identical to `Mapped To`, and `Hostname` cannot be found in the Node Bank.

If only one certificate request is selected, the `Add Node` window opens with the `Hostname` already entered and platform type selected.

If more than one certificate requests are selected when this button is pressed, a pop-up confirmation is displayed, warning that multiple nodes will be added to the Node Bank.

Click `[OK]` and the nodes are added to the Holding Area. After the nodes are added to the Holding Area, a message is sent to the Message Browser. If all nodes are added successfully, a message with severity `Normal` is sent. If any node is not added successfully, a message with severity `Critical` and a list of the nodes which failed to be added to the Holding Area is sent.

Click [Cancel] and no node is added to the Holding Area.

Double-clicking a certificate request item opens an Add Node dialog only if one item is selected and the certificate request is unmapped, or Hostname is not identical to Mapped to or cannot be found in the Node Bank.

Map to Selected Node

Map selected certificate requests. This button is active only if you select one unmapped certificate request or a request where Hostname is not identical to Mapped To. The system must be an HTTPS OVO node. The successful mapping operation causes the Mapped to hostname to change accordingly.

If you try to map a certificate request to a node with a hostname that is different to the hostname from which the certificate request originated, a pop-up window opens with a warning that you must perform a forced operation.

OK

Stop dynamic refresh and close the Node Certificate Request window.

Certificate Generation for Manual Certificate Deployment

Certificates can be deployed totally manually. This avoids sending any certificate-related information over the network before SSL communication is established. The public/private key pair is generated on the certificate server and then transported to the managed node system. This method is often chosen for highly secure environments where it is undesirable to transmit certificate and key data over a network.

NOTE The Certificate Server system must be setup and active before certificates can be generated and distributed.

To manually deploy certificates that have been generated on the Certificate Server:

1. If you are dealing with a particularly large environment, you can create the `bbc_inst_defaults` file to maintain common settings for managed node on the OVO management server. The file should be located as follows:

   `/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults`

   In the namespace `sec.cm.client`, set the deployment type for your managed nodes to manual by adding an entry of the following type for each managed node:

   <IP address> `: CERTIFICATE_DEPLOYMENT_TYPE = MANUAL`

   for example:

   `192.168.10.17 : CERTIFICATE_DEPLOYMENT_TYPE = MANUAL`

   The IP address can accept wildcards to specify ranges of managed node.

   For further information, refer to the file:

   `/etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults.sampl`

See “Changing the Default Port” on page 88 and “Agent Profile” on page 89 for some examples of how to use the `bbc_inst_defaults` file.

2. If installing the OVO HTTPS agent software manually, create a default profile as described in point 2 of “Install an Agent Manually from Package Files” on page 125.
3. Install the OVO HTTPS agent software on the selected managed node system, using the GUI, manually, or remotely.
4. Make a note of the `OvCoreId` value assigned to the selected managed node. `OvCoreId` can be retrieved by calling one of the following commands:

   - **ovcoreid**
   - **ovconfget sec.core**

   When an agent is newly installed using the Administrator’s GUI, a new OvCoreId is created. However, if an OvCoreId is already present in the OVO database for the managed node system, this is used in preference.

   When installing the agent software manually, you must create a profile, copy it and the software packages to the managed node system. The profile includes the original OvCoreId from the OVO database. Install the profile with the command:

   opc_inst -configure *<profile>*

NOTE

The `OvCoreId` stored on a remote system can be determined by using the command:

bbcutil -ping http://*<remote system>*

provided that the Communication Broker is running on the remote system.

The `OvCoreId` can also be locally displayed with the command:

ovcoreid

The `OvCoreId` value stored for the managed node in the OVO database can be displayed with the command:

opcnode -list_id node_list=*<nodename>*

5. On the OVO management server system, ensure that the selected managed node is added to the OVO Node Bank.
6. As an OpenView administrator, create a signed certificate and the corresponding private key for a specific managed node manually on the Certificate Server system using the `opccsacm` command line tool. You must provide a password to encrypt the created data.

NOTE

If certificates must be created before the OVO HTTPS Agent software is installed on the selected managed node, it is possible to specify the `OvCoreId` (`coreid` parameter) in the following command. A `OvCoreId` is still created and it is stored in the database. The `OvCoreId`, which is part of the certificate file name, can be retrieved with the command if the managed node is already stored in the OVO database:

opcnode -list_id node_list=*<node name>*

This value must then be set on the corresponding node system after the OVO HTTPS Agent software is installed with the command:

ovcoreid -set *<id>* -force

If no `OvCoreId` is already stored, use the value from the managed node:

The `OvCoreId` stored on a remote system can be determined by using the command:

bbcutil -ping http://*<remote system>*

provided that the Communication Broker is running on the remote system.

Alternatively, the `OvCoreId` can be locally displayed with the command:

ovcoreid

To create a certificate for the selected managed node, on the OVO management server system, enter the command:

**opccsacm -issue -file *<filename>* [-pass *<password>*] **
-name *<full_qual_hostname>* -coreid *<OvCoreId>*

The tool asks you to specify a password to encrypt the created certificate. This is later required to decrypt the certificate when importing the certificate to the managed node system.

7. Set the installation type to `MANUAL`, either in the `bbc_inst_defaults` file or with the command:

   ```
   ovconfchg -nssec.cm.client -set \
   CERTIFICATE_DEPLOYMENT_TYPE MANUAL
   ```

   Copy the file containing the signed certificate, its corresponding private key and the root certificate onto a floppy disk or other portable media.

   The default file location directory if the `-file` option was omitted is:

   `/<OvDataDir>/temp/OpC/certificates`

   The file name takes the following form:

   `<hostname>-OvCoreId.p12`

8. Go to the managed node system and stop the agent locally with the command:

   ```
   ovc -stop
   ```

9. Install the certificate, the trusted root certificates and the private key from the portable media using the `ovcert` command line tool. Specify the password used in step 5 when requested during installation of the certificate.

   To import the certificate, enter the following command:

   ```
   ovcert -importcert -file <file created in step 5>
   ```

   The tool will ask for the password that was provided in step 5.

NOTE Access to the medium that contains private keys should be tightly controlled to ensure that only authorized people can use them.

10. After installation, delete the certificate installation file from the managed node, and delete the data on the portable medium or store it in a secured place.
11. Start the agent locally with the command:

    ```
    ovc -start
    ```

12. Delete the file created for the certificate import from the certificate server system.

Manual Certificate Deployment with Installation Key

Manual certificate deployment with installation key offers the advantage that the private key never leaves the system to which it belongs. However, it requires that some security-related data is transmitted over the network before the certificate can be installed on the managed node system.

NOTE The Certificate Server system must be setup and active before certificates can be generated and distributed.

To manually deploy certificates using an installation key:

1. Manually install the OVO agent software on the managed node system. For further information, refer to “Installing HTTPS Managed Nodes Manually” on page 124.
2. As an OpenView administrator, initiate the creation of a new installation key on the Certificate Server system. Provide a password to encrypt the created key.

   `opccsacm -geninstkey -file <filename> [-pass <password>]`

   The Certificate Server adds the key to its installation key repository and writes it, together with some management information to a file.
3. Copy the file with the installation key information onto a floppy disk or other portable media.
4. Go to the managed node system and, using the `ovcert` command line tool, initiate a new certificate request. A new public/private key pair is generated. Use the following command:

   `ovcert -certreq -instkey <filename>`

   The encrypted request is sent to the Certificate Server.

The Certificate Server decrypts the request with the key from its repository. If the correct installation key was used, the Certificate Server automatically grants the request and sends the signed certificate back to

the managed node. Then it removes the installation key from the repository. If an invalid installation key was used, the request is automatically denied.

7 Virtual Nodes in OVO

Virtual Nodes in OVO

Clusters are multiple systems, or nodes, that operate as a unit to provide applications, system resources, and data to users. In modern cluster environments such as Veritas Cluster, Sun Cluster or TruCluster, applications are represented as compounds of resources. Those resources construct a resource group, which represents the application running in cluster environment. Each resource has a special function in this compound.

With OVO 8, managing node clusters model has been enhanced. There is now a common mechanism to model applications running in cluster environments.

Terminology

The following High Availability terms and abbreviations are used in OVO.

General High Availability Terms

HA (High Availability)

High availability is a general term used to characterize environments that are business critical and which are therefore protected against downtime through redundant resources. Very often, cluster systems are used to reach high availability.

HA Cluster (High Availability Cluster)

High availability clusters are hardware resources grouped together by a cluster management application such as MC/ServiceGuard (MC/SG), Veritas Cluster, and Sun Cluster. Redundant resources are used to guarantee high availability through, for example, multiple computers, redundant network connections, and mirrored storage devices.

HA package | HA resource group | Cluster package | HARG

These terms are all used to denote a resource defined in the 'cluster world' which can be linked to an application instance. It runs on a cluster and can be switched from one cluster node to another. A cluster package is usually also linked to an element from the 'networking world' known as a virtual node.

Virtual Node

A virtual node is the network representation of an application package running on an HA cluster. A virtual node typically has a hostname and an IP address, is known to the name resolution and can be addressed like an ordinary system.

Physical Node | Cluster Node

This is one single system belonging to the cluster hardware and acting as a potential host for the HARG. A set of physical nodes makes up the cluster.

Switch-over

Controlled switch of a cluster package from one cluster node to another, for example, due to load balancing.

Fail-over

Unplanned switch of a cluster package from one cluster node to another, for example, due to an application error.

Cluster Terms used in OVO

OVO Virtual Node

An OVO virtual node is a concept to represent HA packages in the OVO database and GUI. A virtual node is assigned the hostname and IP address belonging to the HA package. An OVO virtual node has an `HARG Name` attribute. Typically, the value of this attribute is the HA resource group name. An OVO virtual node is comprised of the physical nodes where the HA resource group can run on the cluster.

ClAw (Cluster Awareness)

Cluster awareness is Openview functionality which is used to monitor start and stop events of cluster packages. The ClAw module must be installed on each physical node of a cluster that is to be monitored, as the cluster awareness software only monitors start and stop events on the LOCAL node. The ClAw module is part of the OVO HTTPS agent and the functionality is located in the `ovconfd` process.

APM (Application Package Monitor)

Application Package Monitor is Openview functionality which is used to monitor start and stop events of cluster packages. The APM module is part of the OVO 7.x DCE Agent.

The functionality is mainly located in the `opcapm` process, with additional components located in `opcctla` and `opctemplate`.

APM serves the same purpose as ClAw and is the predecessor of ClAw. It was introduced through the OVO Windows product and is also available for OVO UNIX from version 7.10. However, APM was never exposed for OVO UNIX. For example, the DB SPI and the Windows Exchange SPI use the APM.

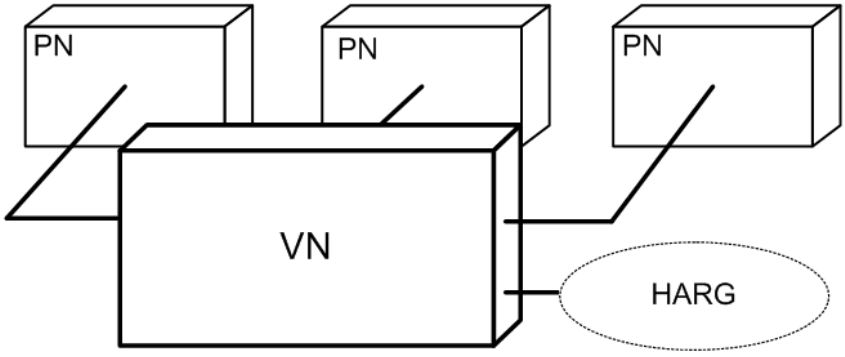
HARG Name (High Availability Resource Group Name)

HARG Name is a string attribute which can be assigned to an OVO virtual node in the OVO 8 database. A HARG name in OVO must be identical to the HA resource group's name in a cluster. This name is the link between the OVO world (OVO database) and the cluster world.

Virtual Node Concepts

An OVO virtual node can be regarded a group of physical nodes linked by a common `HA Resource Group` name. The Cluster Awareness (ClAw) extension of the agents on these physical nodes can switch the policies on a physical node as the package itself switches within the virtual node.

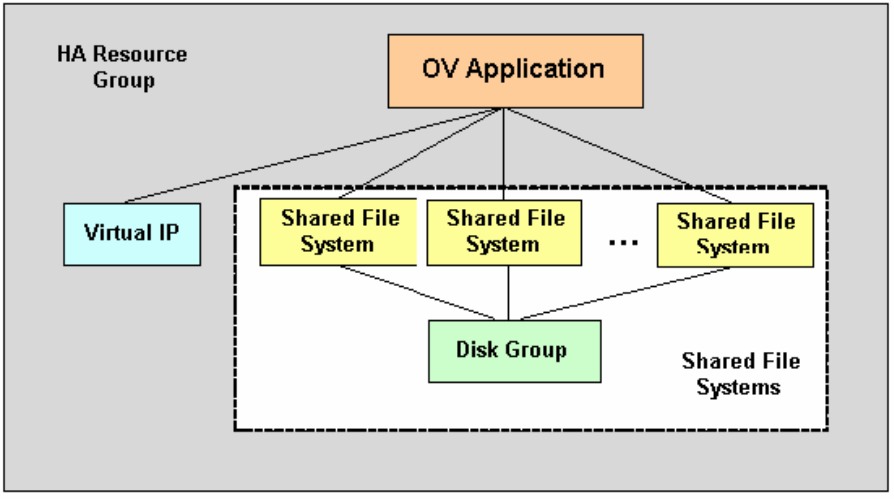
Figure 7-1 Virtual Nodes



The `HA Resource Group` name linking the managed node provides the following advantages:

- Events detected in the scope of the `HA Resource Group`, for example, by policies assigned to the virtual node, may receive that name as the originating node.
- Correct filtering and highlighting on the management station GUI.
- Provide appropriate service names and message key correlations for true management of the cluster.

Figure 7-2 HA Resource Group



NOTE This functionality is only available for HTTPS nodes.

A virtual node can be associated with just one HA resource group name.

An HA resource group name can be assigned to more than one virtual node, but these virtual nodes should not share any common physical nodes. This is because any policy assigned to both virtual nodes would receive the same HARG a second time and the cluster awareness of the agent would not be able to distinguish the virtual nodes.

Working with Virtual Nodes

The following sections describe how to work with virtual nodes in OVO:

Adding Virtual Nodes to OVO

To add a virtual node, from the OVO Node Bank window:

NOTE

You can enter a node into the node bank as a physical node and later change it into a virtual node by selecting `Cluster Virtual Node` in the `Node Modify` window.

A virtual node cannot be directly switched back to a physical node in OVO. To do so, the node must be deleted from the node bank and then added again.

1. Open the `Add Node` window:

   `Actions: Node -> Add`

2. Enter the necessary node-related information:
   - Node name
   - IP address
   - Node communication type: HTTPS or DCE
   - Check `the Cluster Virtual Node` check box
   - Enter a list of physical nodes - no virtual nodes and all nodes must be of the same communication type.
   - HA Resource Group name that the cluster will be hosting

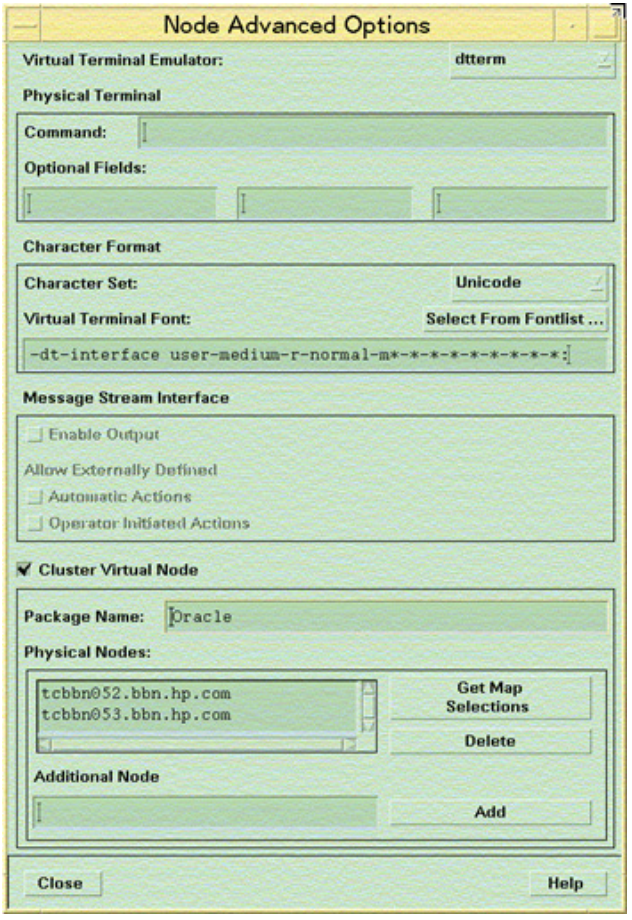
NOTE

All nodes that are to be a part of a cluster must also be members of the OVO node bank. They must all share the same node type characteristics (platform, operating system, communication type).

The virtual node must not be a DHCP node.

The physical nodes of a cluster must not be virtual nodes themselves.

Figure 7-3 Adding Virtual Nodes to OVO



3. Click [OK] to confirm.

Configuring Virtual Nodes using opcnode(1m)

Virtual nodes can also be configured in an OVO node bank by uploading them with the `opccfgupld(1m)` utility or the `opcnode(1m)` utility.

The new call parameters added to opcnode(1m):

```
-set_virtual

node_list = "node1 node2 …"

cluster_package = HARG_name
```

Example:

```
./opcnode -set_virtual node_name=ovguest3 node_list="talence
ovguest3" cluster_package=HARG_name
```

Modifying Virtual Nodes in OVO

To modify a virtual node, from the OVO Node Bank window:

1. In the `Node Bank` window, select the virtual node to be modified.
2. Open the `Modify Node` window:

   `Actions: Node -> Modify`
3. Modify the virtual node-related information:
   - Change the HA Resource Group name
   - Change the list of physical nodes

NOTE

All nodes that are to be a part of a cluster must also be members of the OVO node bank. They must share the same node type characteristics (platform, operating system, communication type).

The physical nodes of a cluster must not be virtual nodes themselves.

4. Click `[OK]` to confirm.

Assigning Policies to Virtual Nodes in OVO

To assign policies to virtual nodes, from the OVO Node Bank window:

1. In the `Node Bank` window, select the virtual nodes to which policies are to be assigned from `De-assigned/Removed`.
2. Open the `Assign Templates` window:

   `Actions: Agents -> Assign Templates`
3. Open the `Add ...` window:
4. Insert the virtual node name and the desired policies.
5. Click `[OK]` to confirm.

Deploying Policies to Virtual Nodes in OVO

Policies assigned to virtual nodes get deployed to the associated physical nodes during an `Install & Update Software and Configuration` request for the virtual node.

NOTE The OVO agent software cannot be deployed to a virtual node. It must be installed on physical nodes. Policies can be deployed to virtual nodes.

To deploy policies to virtual nodes, from the OVO Node Bank window:

1. In the `Node Bank` window, select the virtual nodes to which policies are to be deployed.
2. Open the `Install & Update Software and Configuration` window:

   `Actions: Agents -> Install & Update Software and Configuration`
3. Select the templates to be deployed.
4. Click `[OK]` to distribute the templates to all physical nodes belonging to the selected virtual node.

Distribution to virtual node automatically includes all associated physical nodes. The related HA Resource Group name is added to all policies that are sent to the managed nodes assigned and belonging to the specified virtual node.

If a physical node is being updated which belongs to other virtual nodes, the HA Resource Group name collection is extended to those nodes. As a result, each policy sent to a physical node has all HA Resource Group names attached for the virtual nodes to which it belongs.

Modifying Policy Configuration on Virtual Nodes in OVO

To modify a policy:

1. Open the policy in the `Message Source Templates` window.
2. Make the required changes to the policy.
3. Click `[OK]` to confirm the changes and close the window.

   The changed policy is updated on all physical nodes when a new policy deployment is initiated.

De-assigning Policies from Virtual Nodes in OVO

To de-assign policies from virtual nodes, from the OVO Node Bank window:

1. In the `Node Bank` window, select the virtual nodes to which policies are to be assigned.
2. Open the `Assign Templates` window:

   ```
   Actions: Agents -> Assign Templates
   ```

3. Select the row with the policy/node combination to be removed.
4. Click `[Remove Selected]`.
5. Click `[OK]` to confirm.

Deleting Virtual Nodes from OVO

To delete a virtual node from the OVO Node Bank, from the OVO Node Bank window:

1. In the `Node Bank` window, select the virtual node to be deleted.
2. Delete the selected node:

   ```
   Actions: Node -> Delete
   ```

Uses of ClAw and APM

Cluster awareness and application package monitoring can be very helpful to:

- Monitor applications which are running as HA packages.
- React on HA package switch-over or fail-over.
- Represent HA-related information for operators.

These scenarios are discussed in more detail in the following sections.

Monitoring Applications Running as HA packages

ClAw and APM serve the same purpose; they monitor package existence and package switch-over and fail-over, enabled and disabled through OVO configuration.

Derived from HA Package events, templates or policies which monitor application instances running on cluster nodes are enabled or disabled. A template or policy is enabled on a node, as soon as an HA package is running on the cluster node. It is disabled when the last package switches to another node or if none was present when the OVO agent was started.

React to HA package Switch Over or Fail Over

ClAw and APM can be configured to run customizable start- and stop-actions at package switch-over or fail-over time.

Representing HA-related Information for Operators

ClAw and APM can be used to represent HA-related information for operators. For example, messages for a clustered application should go to the virtual node in the browser and they should color the service graph representing this application.

HA-related information for operators is handled using OVO-message enrichment to represent clustered applications.

ClAw can link the application world and cluster world. The OVO interceptors with their policies and related instrumentation, such as monitor scripts, logfile pre-processors, and automatic actions, work on application level. For example, `opcle` monitors the logfile of an Oracle instance. Typically, such policies and instrumentation are not aware of any underlying cluster on which the application instance runs. ClAw can link an application instance to a virtual node. Messages that are generated for a certain application or application instance, are associated with the virtual node instead of the physical node. This helps to more clearly model clustered applications in service graphs and message browsers.

APM contains a subset of the ClAw features and most of the logic is performed in instrumentation files (actions and monitor scripts). ClAw and the HTTPS agent are policy-based.

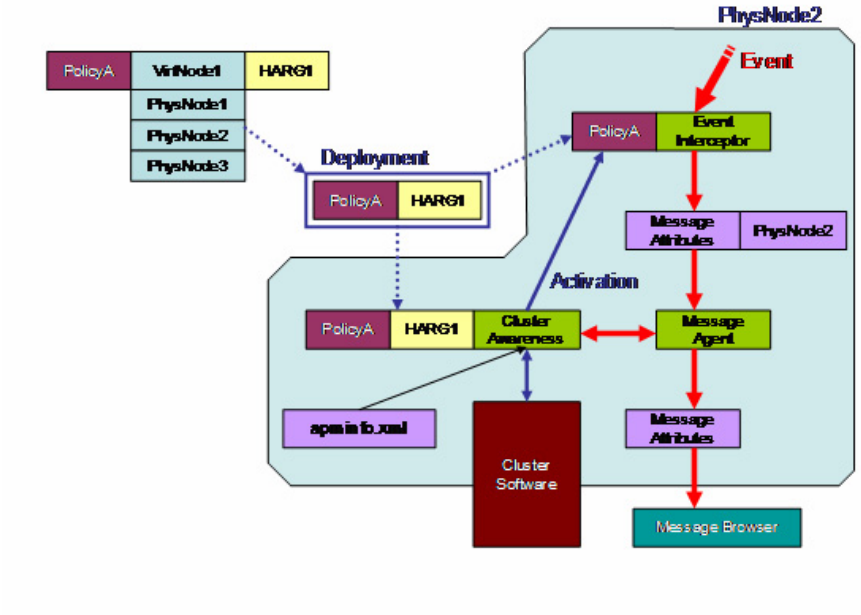
The Virtual Node Concept, ClAw, APM and Message Enrichment

There are three important concepts in managing clustered applications:

- ClAw (HTTPS agent) and APM (DCE agent)
- Virtual Node Concept in OVO 8
- Message Enrichment Using ClAw

The relationships between these concepts are used to represent clustered applications. Let us take a closer look at these three concepts.

Figure 7-4 Virtual Node Concept with ClAw



ClAw (HTTPS agent) and APM (DCE agent)

ClAw and APM read the `apminfo.xml` file. This is the link between the application layer, for example, Oracle instances and Exchange instances, and the cluster layer (HA resource groups).

The cluster layer should be transparent for application instances.

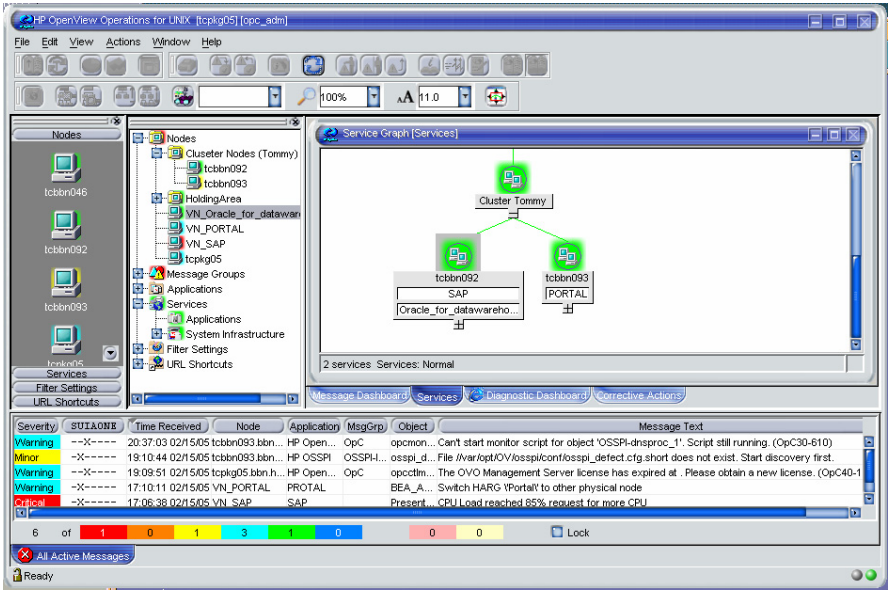
There are applications such as OVO, and NNM which cannot run as multiple instances. Applications such as Oracle can support multiple instances. When there is more than one instance, the instance name from `apminfo.xml` file is important.

Virtual Node Concept in OVO 8

For every monitored HA resource group, you need one virtual node.

The most important attribute of a virtual node is the `HARG name`; the name of the monitored HA resource group.

Figure 7-5 HA Concept with ClAw in OVO



Policies inherit the HARG name attribute from the virtual nodes at policy deployment time. While stored in the database, a template or policy has no HARG name attribute. If the policy is assigned to several virtual nodes and the virtual nodes share physical nodes (several HA packages running on the same cluster) then the policies deployed to such physical nodes inherit the HARG name attributes from all concerned virtual nodes.

The HARG name attributes are then stored in the policy header and ClAw accesses them to perform enable/disable operations accordingly.

Example:

Two Oracle instances db_app1 and db_app2 run on the same cluster. They are linked to the HA resource groups HA_pkg_db_app1 and HA_pkg_db_app2. A single logfile template HA_pkg_db monitors both logfiles of both instances.

You need two virtual nodes, one with HARG name HA_pkg_db_app1 the other with HA_pkg_db_app2. The policy must be assigned to both virtual nodes.

The policy exists once per cluster node after deployment. It is enabled on the nodes where HA_pkg_db_app1 or HA_pkg_db_app2 is currently running and it is disabled on any nodes where neither HA resource group is running.

NOTE

Templates/policies are NOT installed on the shared disk of an HA resource group. Instead they are installed on all physical nodes of a cluster, which belong to a HA resource group.

If OVO could install on the shared disk of a HA resource group, the enabling and disabling of policies would not be necessary. However, OVO does not have the rights to install on the shared disk of any HA capable application.

You can also setup virtual nodes for DCE nodes. The benefit of virtual nodes compared to the cluster representation method as used in OVO 7 is that for OVO 7 you need a node group with the physical nodes for template distribution and a separate node entry with the virtual IP address for action execution. With a virtual node these are combined. For DCE nodes, the HARG name attribute is left blank, and the Communication Type for the virtual node must be set to DCE.

Message Enrichment Using ClAw

There are two main types of ClAw message enrichment:

- Message enrichment using Custom Message Attributes (CMA).
- Message enrichment to obtain the virtual node of an application instance.

Message Enrichment Using Custom Message Attributes

There are two pre-defined custom message attributes (CMAs) which can be set in policies. These are CMA names:

- `namespace`
- `instance`

Both must map to entries in the `apminfo.xml` file. `namespace` maps to `application namespace`, and `instance` maps to `instance`. The two CMAs can be populated as follows.

- For policies:

```
  ...
CONDITION ...
  ...
  SET
    ...
    CUSTOM "namespace" "<$OPTION(my_ns)>"
    CUSTOM "instance" "<$OPTION(my_instance)>"
```

NOTE

Policies may be defined to include the user variable *<$MSG_GEN_NODE_NAME>*. For policies assigned to an HTTPS virtual node, *<$MSG_NODE_NAME>* represents the virtual node name and *<$MSG_GEN_NODE_NAME>* the physical node name of the event, if the Custom Message Attributes values for `namespace` and `instance` are set.

- For monitor scripts or the `opcmsg` command line interface:

  ```
  opcmsg ... -option my_ns=<my_appl_ns> -option \
  my_instance=<my_instance>
  ```

  or respectively

  ```
  opcmon ... -option my_ns=<my_appl_ns> -option \
  my_instance=<my_instance>
  ```

The interceptors feed the namespace, `instance CMA` into the matched messages. Next `opcmsga` reads the special CMAs and requests the HA resource group (IP address and nodename) from ClAw for *<my_appl_ns>* and *<my_instance>*. The physical name, as added by the interceptors and stored in the following message attributes, is replaced by the virtual name and corresponding IP address as received from ClAw: `node_name`, `msg_key`, `msg_key relation`, `service_name`, `auto-perator action node_name` and `operator action node_name`. This is useful, for example, if you want to show a service graph in Service Navigator, which represents a clustered application.

If an action should be executed on the physical node where the message was created, use `<$MSG_GEN_NODE_NAME>` in the action node field of the corresponding policy.

The CMAs instance and namespace are visible in the Java message browser. In addition, a further CMA is automatically added to such messages, denoting the HA resource group, called `harg`.

Configuring Custom Message Attributes Using the Administrator's GUI

To configure custom message attributes, complete the following steps in the Administrator's GUI:

1. Open the `Message Source Templates` window.
2. Select the template to which you want to add CMAs.
3. Double click the selected template to open the `Message and Suppress Conditions` window.
4. Select one of the conditions where a CMA should be added and click the **Modify** button.
5. In the `Modify Condition` window, click **Custom Attributes**.

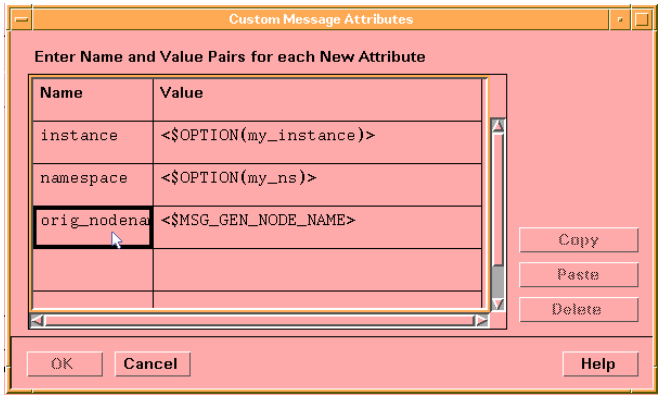
6. In the Custom Message Attributes window, enter the name and value pairs.

   For example:

   Enter **instance** in Name field, **<$OPTION(my_instance)>** in Value field.

   Enter **namespace** in the Name field, **<$OPTION(my_ns)>** in the Value field.

Figure 7-6 Custom Message Attributes Window



7. Click **OK** in the Custom Message Attributes window.
8. Click **OK** in the Condition window.
9. Click **Close** to close the Message and Suppress Conditions window.

Message Enrichment to obtain the Virtual Node of an Application Instance

ClAw incorporates the ovappinstance tool, located in the $OvBinDir directory, which can be used on command line level to get information about application instances and their related HARGs. For example, the following command prints the virtual IP address of the instance *<instance>*:

```
ovappinstance -i <instance> -host
```

Configuring ClAw and APM

ClAw and APM can be configured with exactly the same configuration files.

NOTE Some of the configuration elements are supported by ClAw only for backward compatibility. ClAw can work in `APM-mode` and in `ClAw-mode`, where the `ClAw-mode` needs no configuration on the agent in the case where you want to enable and disable policies.

There are two configuration file types:

- `$OvDataDir/conf/conf/apminfo.xml`
- `$OvDataDir/bin/instrumentation/conf/<appl_name>.apm.xml`

The following sections introduce how to use these configuration files and present some examples.

NOTE The directories `$OvDataDir/conf/conf/` and `$OvDataDir/bin/instrumentation/conf/` do not exist by default. When you are configuring apminfo.xml for the first time, you must first manually create these directories.

$OvDataDir/conf/conf/apminfo.xml

The `apminfo.xml` file is used to:

- Define which resource groups are to be monitored. (OVO 7 APM-only. OVO 8 ClAw monitors ALL resource groups).
- Define the mappings between HA resource groups and application instances.

There must only be one `apminfo.xml` file per node. There is no special distribution mechanism to transport the `apminfo.xml` file from the OVO management server to the managed nodes. Typically, the `apminfo.xml` file is installed manually on the agents. There is no merge mechanism to

add further entries to the `apminfo.xml` file. You must update it manually, for example, if you want to add another application instance -> HA resource group link.

apminfo.xml Syntax

```
<APMClusterConfiguration>
       <Application>
               <Name> ... </Name>
                  <Instance>
                        <Name> ... </Name>
                        <Package> ... </Package>
                  </Instance>
       </Application>
</APMClusterConfiguration>
```

apminfo.xml Examples

Example 1: In the following example, one application, `OpenView_Application`, is defined. It defines one instance with the name `openview` and an HA Resource Group with the name `ov-server`.

```
<?xml version="1.0"?>
<APMClusterConfiguration>
       <Application>
               <Name>OpenView_Application</Name>
               <Instance>
                        <Name>openview</Name>
                        <Package>ov-server</Package>
               </Instance>
       </Application>
</APMClusterConfiguration>
```

Example 2: In the following example, two applications, `SQL_Server` and `Exchange`, are defined. Each application defines two instances with a name and corresponding HA Resource Group name.

```
<?xml version="1.0"?>
<APMClusterConfiguration>
       <Application>
               <Name>SQL_Server</Name>
               <Instance>
                       <Name>Instance1</Name>
                       <Package>sqlsrvpkq1</Package>
               </Instance>
               <Instance>
                       <Name>Instance2</Name>
                       <Package>sqlsrvpkq2</Package>
              </Instance>
       </Application>
       <Application>
               <Name>Exchange</Name>
               <Instance>
                       <Name>Instance1</Name>
                       <Package>msexpkq1</Package>
               </Instance>
               <Instance>
                       <Name>Instance2</Name>
                       <Package>msexpkq2</Package>
               </Instance>
       </Application>
</APMClusterConfiguration>
```

$OvDataDir/bin/instrumentation/conf/ <*appl_name*>.apm.xml

The <*appl_name*>.apm.xml file is used to:

- Specify template to resource group mappings. Required for OVO 7 only. Not required for OVO 8 ClAw, but incorporated for backward compatibility.
- Specify start and stop hooks used by APM and ClAw. These are used to execute additional tasks at HA package switch or fail over. Template and policy enable and disable operations are not handled from this file.

NOTE

There is an important distinction between OVO 7 APM and OVO 8 ClAw:

With ClAw you do not need to define template to resource group mappings in <*appl_name*>.apm.xml. Instead, you can perform these mappings by assigning HARG names to virtual nodes on the OVO 8 management server.

Nevertheless, the template or policy name to HA resource group mappings are understood by ClAw.

Usage of <appl_name>.apm.xml:

As for the apminfo.xml file, there are no special deployment mechanisms to distribute the configuration files to the agents.

<*appl_name*> must be defined in the apminfo.xml file so that the link between apminfo.xml entries and <appl_name>.apm.xml files can be made by APM and ClAw.

NOTE

SPIs may retain OVO 7-style <*appl_name*>.apm.xml files with the mappings on HTTPS agents. This is to avoid the need for different SPIs for OVO 7 and OVO 8.

<appl_name>.apm.xml template name to resource group mappings do not interfere with virtual node HARG names, if both are used together. It is effectively a type of redundancy; a policy mentioned in both methods is enabled or disabled twice.

NOTE

<appl_name>.apm.xml is dependent on the application namespace. It is not dependent on the instance level. Therefore, the start and stop actions are provided with the associated instance name as their first parameter when they are executed at package switch time (see the $instanceName in example 2 below). The environment variable $instanceName is set by ClAw when start or stop tasks are performed.

<appl_name>.apm.xml Syntax

```
<APMApplicationConfiguration>
        <Application>
                <Name> ... </Name>
                <Template> ... </Template>
                <Template> ... </Template>
                <StartCommand> ... </StartCommand>
                <StopCommand> ... </StopCommand>
        </Application>
</APMApplicationConfiguration> ?
```

Application (or application namespace)	Application or Name
Policies (or templates)	Template
Start Actions (or start commands)	StartCommand
Stop Actions (or start commands)	StopCommand

<appl_name>.apm.xml Examples

<appl_name>.apm.xml must be located at:

/var/opt/OV/bin/instrumentation/conf

Example 1:

The following example application configuration, `OpenView_Application`, defines the start action `/tmp/test_clawstart.sh clawstart` and the stop action `/tmp/test_clawstop.sh clawstop`.

The application configuration file should be located as follows:

```
/var/opt/OV/bin/instrumentation/conf/Openview_Application.apm.xml
```

```
<?xml version="1.0"?>
<APMApplicationConfiguration>
   <Application>
     <Name>OpenView_Application</Name>
     <StartCommand>/tmp/test_clawstart.sh clawstart</StartCommand>
     <StopCommand>/tmp/test_clawstop.sh clawstop</StopCommand>
   </Application>
</APMApplicationConfiguration>
```

Example 2:

The following example application configuration, `SQL_Server`, defines two policies `SQLTemplA` and `SQLTemplB` with start action `C:\startSQLSrv.bat $instanceName` and stop action `C:\stopSQLSrv.bat $instanceName`.

The application configuration file should be located as follows:

```
/var/opt/OV/bin/instrumentation/conf/SQL_Server.apm.xml
```

```
<?xml version="1.0"?>
<APMApplicationConfiguration>
   <Application>
     <Name>SQL_Server</Name>
     <Template>SQLTemplA</Template>
     <Template>SQLTemplB</Template>
     <StartCommand>C:\startSQLSrv.bat $instanceName</StartCommand>
     <StopCommand>C:\stopSQLSrv.bat $instanceName</StopCommand>
   </Application>
</APMApplicationConfiguration>
```

Example 3:

The following example application, `Exchange`, defines one policy `ExchangeTempl` and one custom agent or subagent `ExchangeSubAgent`.

The application configuration file should be located as follows:

`/var/opt/OV/bin/instrumentation/conf/Exchange.apm.xml`

```
<?xml version="1.0"?>
<APMApplicationConfiguration>
   <Application>
        <Name>Exchange</Name>
        <Template>ExchangeTempl</Template>
        <Subagent>ExchangeSubAgent</Subagent>
   </Application>
</APMApplicationConfiguration>
```

Syntax check tool for `apminfo.xml` and `<appl_name>.apm.xml` is located at:

`/opt/OV/bin/ovappinstance -vc`

where `-vc` = verify Configuration

This tool can be called on the managed node where the configuration files are used.

Command Line Utilities of ClAw

1. `$ovBinDir/ovclusterinfo` prints cluster related information.

2. **`$OvBinDir/ovappinstance`** provides information about application instances and their related HA resource groups (based on the data available in the `apminfo.xml` configuration file).

For further information, refer to the man pages for these commands.

Command Line Utilities of APM

`<OVO_bin_dir>/opcclustns` provides information about application instances and related resource groups.

Customizing ClAw to Monitor Cluster States

ClAw checks the state of a cluster to decide whether policies have to be enabled or disabled. If a state maps to `online` then a policy is enabled, if it maps to `offline` or `unknown`, then it is disabled.

For certain use cases it can make sense to modify the mapping. For example, with Veritas Cluster Server, an administrator decides that the cluster state `|PARTIAL|` should also be regarded as `online`. This means that the administrator wants to monitor HA resource groups even if they are only partially running. A partially running HARG could mean that a minor subservice did not start, but the main service is operational.

For Veritas Cluster Server, this can be performed with the command:

```
ovconfchg -ns conf.cluster.RGState.VCS -set _PARTIAL_ online
```

NOTE

OVO configuration setting names can only contain alpha-numeric characters and the underscore character (A ... Z, a ... z, 0 ... 9 and _).

For example, the Veritas Cluster state `|PARTIAL|` is translated by OVO as `_PARTIAL_`.

The `ovconfchg` call must be executed on all cluster nodes.

Cluster Application Default States

The following is a list of the default states and their meaning for different cluster applications.

The term [`conf.cluster.RGState.`*<HA_Application>*], for example `[conf.cluster.RGState.MCSG]`, defines the namespace where a configuration setting is made.

Any state which is not defined in this namespace is treated as being offline. However, you can specify entries for additional states to the configuration settings with command of the following form:

```
ovconfchg -ns conf.cluster.RGState.<HA_Application> \
-set <New_State_Name> <State>
```

For MC Service Guard, Red Hat Advanced Server, Sun Cluster and Veritas Cluster Server, you can directly add the states with their value (offline or online) using the `ovconfchg` command under the appropriate name space. ClAw uses cluster commands to get the current state and then refers to the configuration settings to find whether this state is online or offline. So, while adding a new state in the configuration settings, the state string should be the same as the string returned by the cluster command used to retrieve its state.

However, this is not the case with Microsoft Cluster Server. ClAw uses Microsoft Cluster Server APIs instead of CLI tools and Microsoft Cluster Server APIs return enumerated values for its state instead of a state string.

At the time of writing, all the possible states for Microsoft Cluster Server are supported. If Microsoft Cluster Server introduces new states, it will be necessary to update ClAw to incorporate these modifications.

NOTE

The settings are not HA resource group specific. They impact the monitoring of all configured HARGs. As a result, you cannot configure resource group A so that state S maps to `online`, while for resource group B, state S maps to `offline`. The will both be `online` or `offline`.

MC Service Guard

```
[conf.cluster.RGState.MCSG]
down=offline
halting=unknown
starting=unknown
unknown=unknown
up=online
```

Microsoft Cluster Server:

```
[conf.cluster.RGState.MSCS]
ClusterGroupFailed=offline
ClusterGroupOffline=offline
ClusterGroupOnline=online
ClusterGroupPartialOnline=offline
ClusterGroupStateUnknown=unknown
```

Red Hat Advanced Server

[conf.cluster.RGState.RHAS]
started=online

Sun Cluster

```
[conf.cluster.RGState.SC]
ERROR_STOP_FAILED=unknown
OFFLINE=offline
ONLINE=online
PENDING_OFFLINE=unknown
PENDING_ONLINE=unknown
UNMANAGED=unknown
```

Veritas Cluster Server

NOTE OVO configuration setting names can only contain alpha-numeric characters and the underscore character (A ... Z, a ... z, 0 ... 9 and _).

For example, the Veritas Cluster state |PARTIAL| is translated by OVO as _PARTIAL_.

```
[conf.cluster.RGState.VCS]
OFFLINE=offline
ONLINE=online
_OFFLINE_=offline
_ONLINE_=online
_PARTIAL_=unknown
_UNKNOWN_=unknown
```

Getting the First Message for a Virtual Node

This is an example to generate a message for a virtual node. Prerequisite is an HA cluster on which one or more HA resource groups are running. For simplicity, just select one of the existing resource groups and model it in OVO as a virtual node. You need to know the resource group name, either the IP address or nodename to do this.

1. Make sure that the OVO agent software is installed on each physical node of the cluster.
2. Add the virtual node into the OVO Node Bank.
3. Add the physical nodes belonging to the virtual node.
4. Specify the HA resource group name associated with the virtual node.

   For steps b, c, and d, you can refer to the details which are described in “Adding Virtual Nodes to OVO” on page 163.

   In the following steps, the HA resource group name is referred to as *<my_resource_group>*.
5. Configure CMAs in the template.

   Take `opcmsg(1|3)` as an example. In this example, we will add a simple test condition to this template and specify CMAs in this test condition.

   a. Open the `Message Source Templates` window.

   b. Select the template `opcmsg(1|3)`.

   c. Double click the selected template to open the `Message and Suppress Conditions` window.

   d. Click the **Add** button to add a test condition.

e. Edit the following fields in the Condition window:

Description: **test_CMA**

Condition:

Application: **a**

Object: **testcma**

Message Text: **I want to test CMA**

Set Attributes:

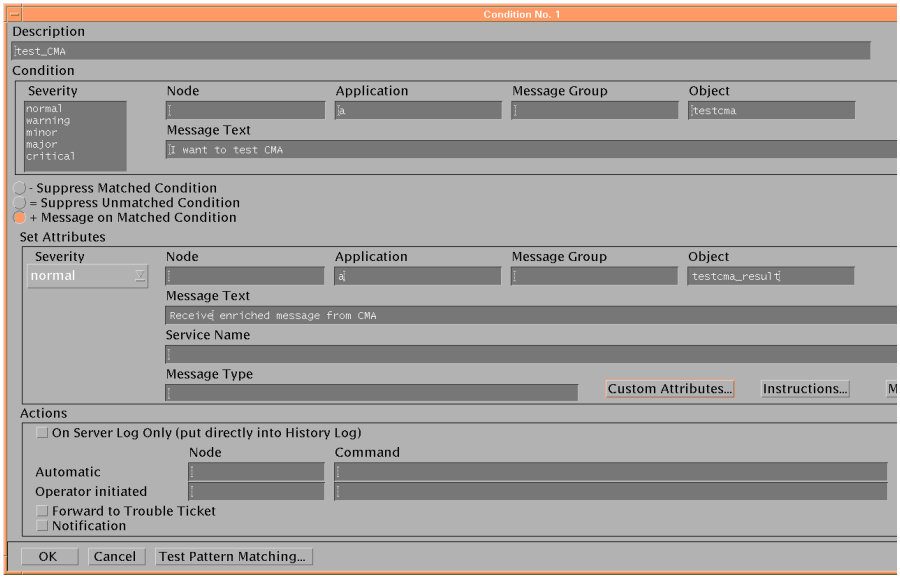
Application: **a**

Object: **testcma_result**

Message Text: **Receive enriched message from CMA**

Severity: **normal**

Figure 7-7 opcmsg(1|3) Conditions Window



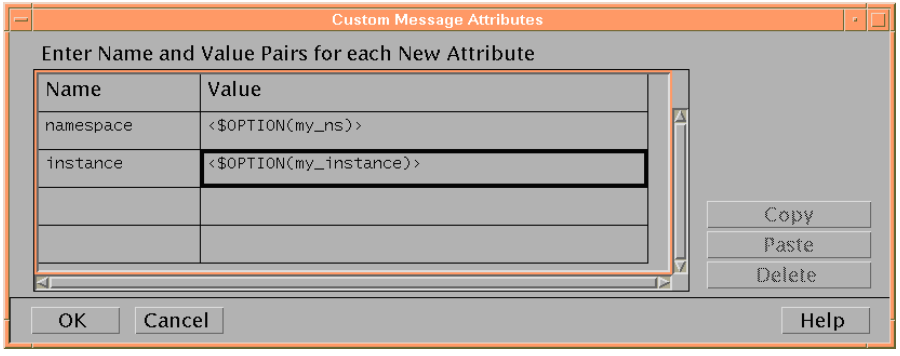
f. Click on Custom Attributes to open the Custom Message Attributes window.

g. In the Custom Message Attributes window, enter the name and value pairs.

Enter **namespace** in the Name field, **<$OPTION(my_ns)>** in the Value field.

Enter **instance** in Name field, **<$OPTION(my_instance)>** in Value field.

Figure 7-8 Custom Message Attributes Window



NOTE

When following this example, it is helpful to de-assign the opcmsg(1|3) policy from the physical nodes of the virtual node if the opcmsg(1|3) policy was already assigned to these nodes. This step is not essential, but could help to avoid confusion, as if it is not de-assigned, the template is assigned to a physical and virtual node at the same time and would be always enabled.

h. Click **OK** in the Custom Message Attributes window.

i. Click **OK** in the Condition window.

j. Click **Close** to close the Message and Suppress Conditions window.

The generic steps are specified in section “Message Enrichment Using Custom Message Attributes” on page 173.

6. Assign the opcmsg(1|3) policy to the virtual node.

7. Distribute the `opcmsg(1|3)` policy to the virtual node.
8. Check if the policy is installed on the agent using the `ovpolicy` command.

   On each physical node, enter the command:

   `ovpolicy -l -level 4`

   The following information is displayed:

```
msgi      "opcmsg(1|3)"   <enabled or disabled> 1
policy id : "15012f6e-ab2a-71d9-1d2e-0a110b850000"
owner     : "OVO:<full_qualified_virtual_node_name>"
category  : <no categories defined>
attribute : "HARG:<my_resource_grp_name>" "no_value"
```

NOTE

If the policy is assigned to the virtual node only, on the node where the HA package is running, this policy is enabled. On the node where the HA package is not running, this policy is disabled.

You can obtain policy status information (enabled or disabled) using the command `ovpolicy -l`.

For example, to list installed policies for the local agent, enter the command:

`ovpolicy -l`

The information is displayed in the following form:

```
Type              Name              Status     Version
-------------------------------------------------------
configsettings    "OVO settings"    enabled    1
msgi              "opcmsg(1|3)"     enabled    1
monitor           "mondbfile"       disabled   1
```

9. Check whether the `apminfo.xml` file is already installed on each physical node.

   On the management server, execute the following command for each of your physical nodes:

```
"
for node in <all your physical nodes>
do
opcdeploy -cmd "ls"  -par "\$OvConfDir/conf/apminfo.xml"
-node $node
done
"
```

10. If the `apminfo.xml` file is NOT installed, edit the `apminfo.xml` file on the management server and install it on each physical node as follows:

    a. **`cd /tmp`**

    b. **`vi apminfo.xml`**

    c. Put the following contents into the `apminfo.xml` file and save the file:

```
"
<?xml version="1.0"?>
<APMClusterConfiguration>
      <Application>
            <Name>OpenView_Application</Name>
            <Instance>
                  <Name>openview</Name>
                  <Package>ov-server</Package>
            </Instance>
      </Application>
</APMClusterConfiguration>
"
```

NOTE

The extract for the `apminfo.xml` file mentioned above is an example, and the application `OpenView_Application` is defined, which is mapped to `my_ns` defined in a CMA. It also defines the mapping between the application instance `openview` and the HA Resource Group `ov-server`. Instance `openview` is mapped to `my_instance` defined in the CMA.

d. Install the `apminfo.xml` file on each physical node as follows:

```
"
for node in <all of your physical node names>
do
opcdeploy -deploy -file /tmp/apminfo.xml -node $node
-targetdir "conf/conf" -trd data
done
"
```

11. If the `apminfo.xml` file is already installed on the agent, you must edit the existing `apminfo.xml` file manually as follows:

    a. Log on to the system where the `apminfo.xml` file is installed.

    b. **`cd \$OvConfDir/conf/`**

    c. **`vi apminfo.xml`**

    d. Keep the existing application definitions and define your application:

```
"
<?xml version="1.0"?>
<APMClusterConfiguration>
    <Application>
        <Name>Existing_Application</Name>
        <Instance>
            <Name>Existing_instance</Name>
            <Package>Existing_resource_group_name
                                        </Package>
        </Instance>
    </Application>
    <Application>
        <Name>OpenView_Application</Name>
        <Instance>
            <Name>openview</Name>
            <Package>ov-server</Package>
        </Instance>
    </Application>
</APMClusterConfiguration>
"
```

12. Configure the file:

    `$OvDataDir/bin/instrumentation/conf/<appl_name>.apm.xml`

This file should be created in the directory:

```
$OvDataDir/bin/instrumentation/conf
```

on each physical node and it should take the form of the following example:

```
<?xml version="1.0"?>
<APMApplicationConfiguration>
    <Application>
        <Name>OpenView_Application</Name>
        <Template>opcmsg(1|3)</Template>
    </Application>
</APMApplicationConfiguration>
```

For more detailed information about configuring the `<appl_name>.apm.xml` file, see "$OvDataDir/bin/instrumentation/conf/ <appl_name>.apm.xml" on page 179.

13. If the `opcmsg(1|3)` policy is installed on the agent and `enabled`, and if the `apminfo.xml` file is installed, execute the following command from this agent:

    `opcmsg a=a o=testcma msg_t="I want to test CMA" \`
    `-option my_ns=OpenView_Application \`
    `-option my_instance=openview`

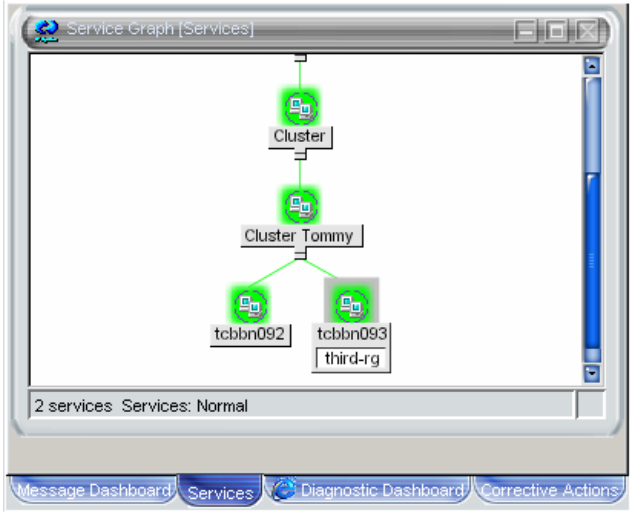
    You should receive a normal message for the virtual node with the following details in the browser:

    ```
    Node:<virtual_nodename>
    Application: "a"
    Object: "testcma_result"
    Message Text: "Receive enriched message from CMA"
    ```

Monitoring HARGs in the Java UI

Clusters and their nodes can be monitored in the `Services Graph` window. You can configure the cluster so that the active node is labelled with, for example, the application that it is hosting. When this node is no longer active, the label is switched to the new active node.

Figure 7-9 Cluster Displayed in the Services Graph



To monitor HA resource groups in the Java UI, the following configurations need to be made:

- Create an APM definition file to define the mappings between HA resource groups and application instances.
- Create or configure a command, script or executable, which is run when an HA resource group is started or stopped.
- Specify start and stop hooks used by APM and CLAw to execute additional tasks at HA package switch or fail over.
- Configure the custom message attributes.
- Create policies to label and unlabel the system in the Java GUI on which the HA resource group is active or inactive when an HA resource group is started or stopped.

Our example is based on a cluster `tommy2`, consisting of two physical nodes, `tcbbn092` and `tcbbn093`. Three HARGs are installed on this cluster; `OpenView_Application`, `second-rg` and `third-rg`. This example concentrates on the `third-rg` application. To be able to monitor HARGs in the Java GUI, the following steps need to be provided.

1. Create the APM definition file to define the mappings between HA resource groups and application instances. In the following example, for simplicity, we configure the application name and instance name to be the same as the HARG name for HA resource group "`second_rg`" and "`third_rg`". For further information, see “$OvDataDir/conf/conf/apminfo.xml” on page 176.

```
# more /var/opt/OV/conf/conf/apminfo.xml

<?xml version="1.0"?>
<APMClusterConfiguration>
      <Application>
            <Name>OpenView_Application</Name>
            <Instance>
                  <Name>openview1</Name>
                  <Package>ov-server</Package>
            </Instance>
      </Application>
      <Application>
            <Name>second-rg</Name>
            <Instance>
                  <Name>second-rg</Name>
                  <Package>second-rg</Package>
            </Instance>
      </Application>
      <Application>
            <Name>third-rg</Name>
            <Instance>
                  <Name>third-rg</Name>
                  <Package>third-rg</Package>
            </Instance>
      </Application>
</APMClusterConfiguration>
```

2. Create a shell script which will be executed when a HARG is started or stopped. It will log the start and stop information to the logfile `/tmp/clawapplication_log` and send a status message to the browser. The shell script should look like the following example:

```
# more /tmp/test_clawst.sh

application=$1
label=$2
start_stop=$3
echo "app=$application st=$start_stop label=$label"
>>/tmp/clawapplication_log
echo "$application $start_stop at:" >>/tmp/clawapplication_log
date >>/tmp/clawapplication_log
echo "OVO_instance is $application" >>/tmp/clawapplication_log
echo "Sending $start_stop message..."
>>/tmp/clawapplication_log
/opt/OV/bin/OpC/opcmsg a=a o=o msg_t="$application $start_stop"
-option label=$label -option my_instance=$application -option
my_ns=OpenView
echo "$application ends at:" >>/tmp/clawapplication_log
date >>/tmp/clawapplication_log
echo "=======================" >>/tmp/clawapplication_log
```

3. Specify start and stop hooks used by APM and CLAw to execute additional tasks at HA package switch or fail over. For further information, see the section titled “$OvDataDir/bin/instrumentation/conf/ <appl_name>.apm.xml” on page 179.

   In the following example, we specify start and stop hooks for `third-rg`. When `third-rg` is started, the shell script `/tmp/test_clawst.sh` which we defined in the previous step is executed with input parameters `$instanceName ov_label3 starts`. A message with text `third-rg starts` is then sent to the browser and the value of `label` is set to `ov_label3`. When `third-rg` is stopped, the same shell script is executed with input parameters `$instanceName ov_label3 stops` and a message with text `third-rg stops` is then sent to the browser and the value of `label` is set to `ov_label3`.

   The start and stop definitions should be specified as in the following example:

```
# more /var/opt/OV/bin/instrumentation/conf/third-rg.apm.xml

<?xml version="1.0"?>
<APMApplicationConfiguration>
   <Application>
      <Name>third-rg</Name>
      <StartCommand>
         /tmp/test_clawst.sh $instanceName ov_label3 starts
      </StartCommand>
      <StopCommand>
         /tmp/test_clawst.sh $instanceName  ov_label3 stops
      </StopCommand>
   </Application>
</APMApplicationConfiguration>
```

4. Configure the custom message attributes. For more information, refer to “Configuring Custom Message Attributes Using the Administrator’s GUI” on page 174.

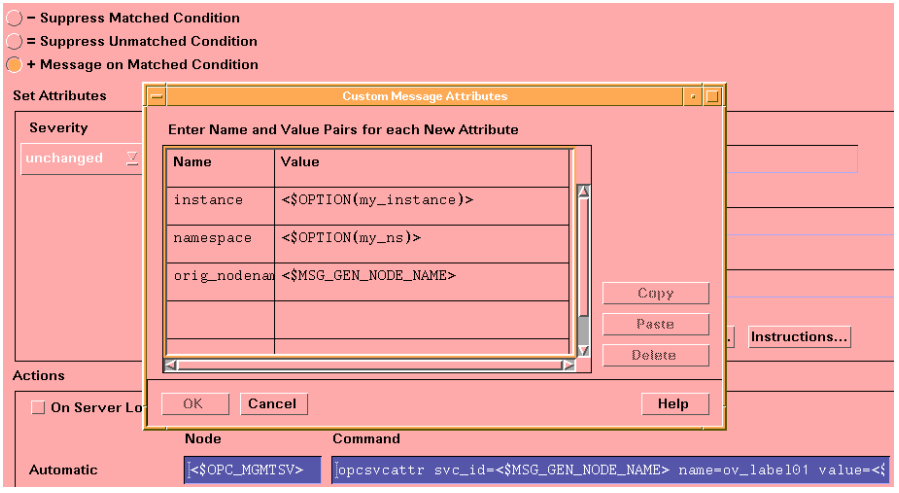
   In our example, specify the following information:

   namespace in the Name field, **<$OPTION(my_ns)>** in the Value field.

   instance in Name field, **<$OPTION(my_instance)>** in Value field.

   orig_nodename in Name field, **<$MSG_GEN_NODE_NAME>** in Value field.

Figure 7-10 Specifying Custom Message Attributes



5. Create a policy to check if an HARG is started and to label the system in the Java UI on which the HARG is active. Deploy this policy to the virtual node.

   The following policy example checks the message text for a running HARG. On finding one, it runs an automatic action to label the active cluster node with the package name, `third-rg` on node `tcbbn093` in our example.

```
OPCMSG "opcmsg(1|3)

DESCRIPTION "starts HARG"
      CONDITION_ID
"96a679b2-b59c-71d9-1ed2-c0a801020000"
      CONDITION
            TEXT "<*> starts<*>"
      SET
            SERVICE_NAME "<$MSG_GEN_NODE_NAME>"
            MSGKEY "<$OPTION(my_instance)>"
            MSGKEYRELATION ACK  "<$OPTION(my_instance)>"
            CUSTOM  "instance"  "<$OPTION(my_instance)>"
            CUSTOM  "namespace"  "<$OPTION(my_ns)>"
            CUSTOM  "orig_nodename"
"<$MSG_GEN_NODE_NAME>"
            AUTOACTION "/opt/OV/bin/OpC/opcsvcattr
svc_id=<$MSG_GEN_NODE_NAME> name=<$OPTION(label)>
value=<$OPTION(my_instance)>" ACTIONNODE IP 0.0.0.0
"<$OPC_MGMTSV>"
ANNOTATE
            SIGNATURE "EAJHjRr9vq48…
```

   Enter the following command to run the `third-rg` HARG on the node `tcbbn093`:

```
/usr/sbin/cmrunpkg -n tcbbno93 third-rg
```

   The `third-rg` HARG is started, the message `third-rg starts` is received and the icon of the node `tcbbn093` in the Java UI is labeled with the active package name, `third-rg`.

Figure 7-11 Cluster Status Showing third-rg Running on tcbbn093

Cluster status confirms that `third-rg` has been started on node `tcbbn093`

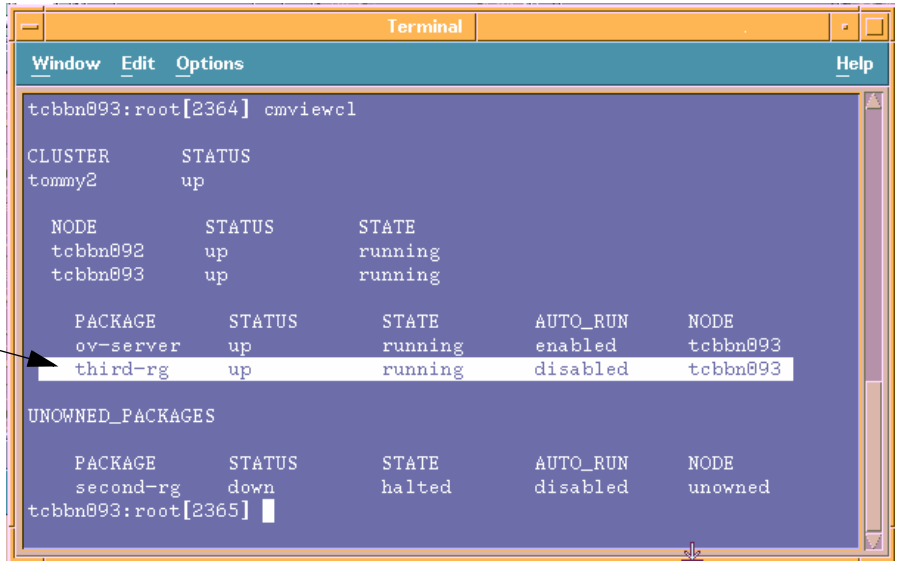
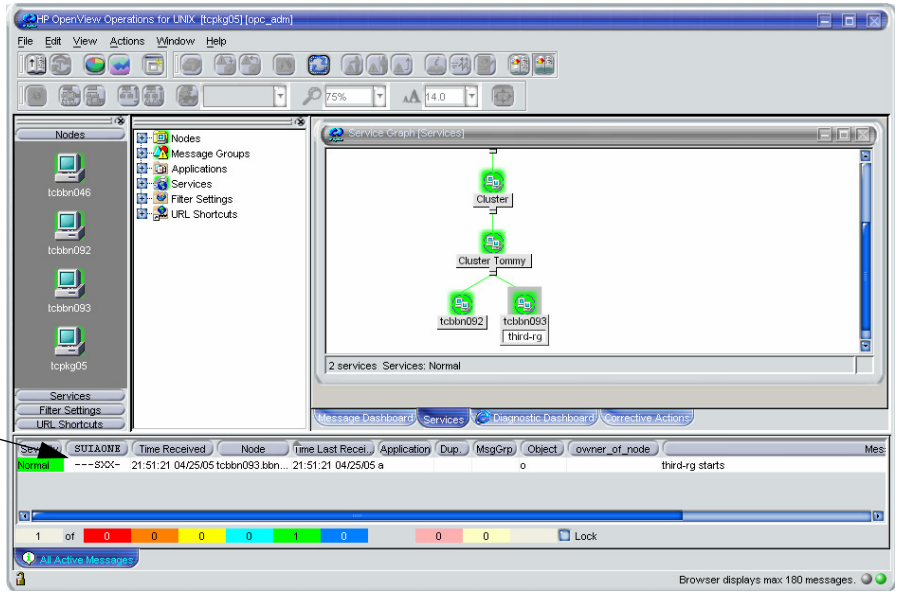


Figure 7-12 Cluster Service View Showing third-rg Running on Node tcbbn093

Message confirms that `third-rg` has been started on node `tcbbn093`



6. Create a policy to check if an HARG is stopped and to remove the label from the system in the Java UI on which the HARG was active. Deploy this policy to the virtual node.

   The following policy example checks the message text for a stopped HARG. On finding one, it runs an automatic action to remove the label from the now no longer active cluster node, `tcbbn093` in our example.

```
OPCMSG "opcmsg(1|3)

    DESCRIPTION "default interception of messages
            submitted by opcmsg(1) and opcmsg(3)"
    FORWARDUNMATCHED
    MSGCONDITIONS
    DESCRIPTION "stops HARG"
        CONDITION_ID "8070b36c-b5b3-71d9-1ed2-c0a801020000"
        CONDITION
            TEXT "<*> stop<*>"
        SET
            SEVERITY Warning
            SERVICE_NAME "<$MSG_GEN_NODE_NAME>"
            MSGKEY "<$OPTION(my_instance)>"
            MSGKEYRELATION ACK  "<$OPTION(my_instance)>"
            CUSTOM  "instance"  "<$OPTION(my_instance)>"
            CUSTOM  "namespace"  "<$OPTION(my_ns)>"
            CUSTOM  "orig_nodename"  "<$MSG_GEN_NODE_NAME>"
            AUTOACTION "/opt/OV/bin/OpC/opcsvcattr -remove
svc_id=<$MSG_GEN_NODE_NAME> name=<$OPTION(label)>" ACTIONNODE
IP 0.0.0.0  "<$OPC_MGMTSV>" ANNOTATE
            SIGNATURE "RgUMFg…
```

   Enter the following command to stop the `third-rg` HARG on the node `tcbbn093`:

```
/usr/sbin/cmhaltpkg -n tcbbno93 third-rg
```

   The `third-rg` HARG on the node `tcbbn093` is stopped. The message `third-rg stops` is received and the label of the package name, `third-rg`, is removed.

Figure 7-13 HARG third-rg is Stopped on Node tcbbn093

Cluster status confirms that `third-rg` has been stopped on node `tcbbn093`

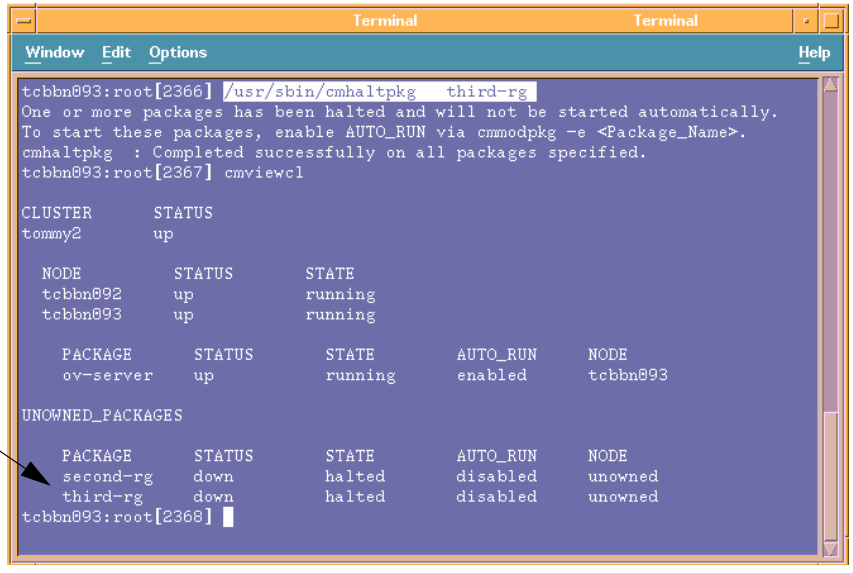
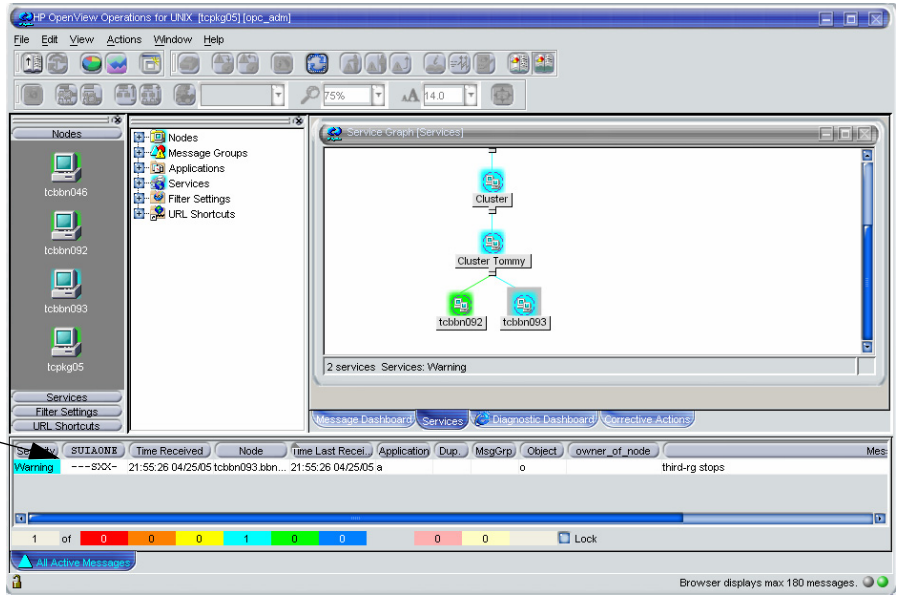


Figure 7-14 Cluster Service View with third-rg No Longer Running on Node tcbbn093

Message confirms that `third-rg` has been stopped on node `tcbbn093`



On switching the `third-rg` HARG to the node `tcbbn092`, the node icon in the `Service Graph` is labeled with the application name `third-rg`.

Figure 7-15 HARG third-rg is Started on Node tcbbn092

Cluster status confirms that `third-rg` has been started on node `tcbbn092`

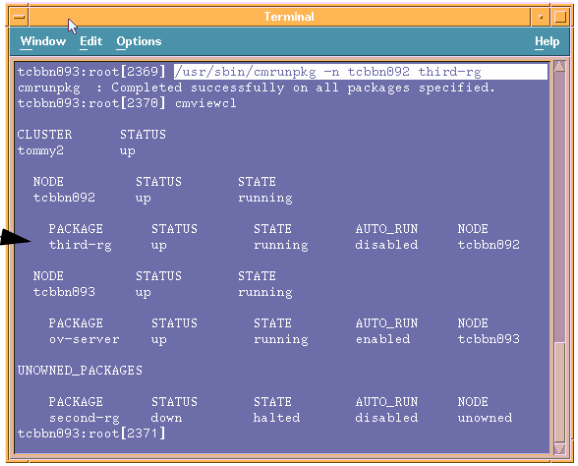
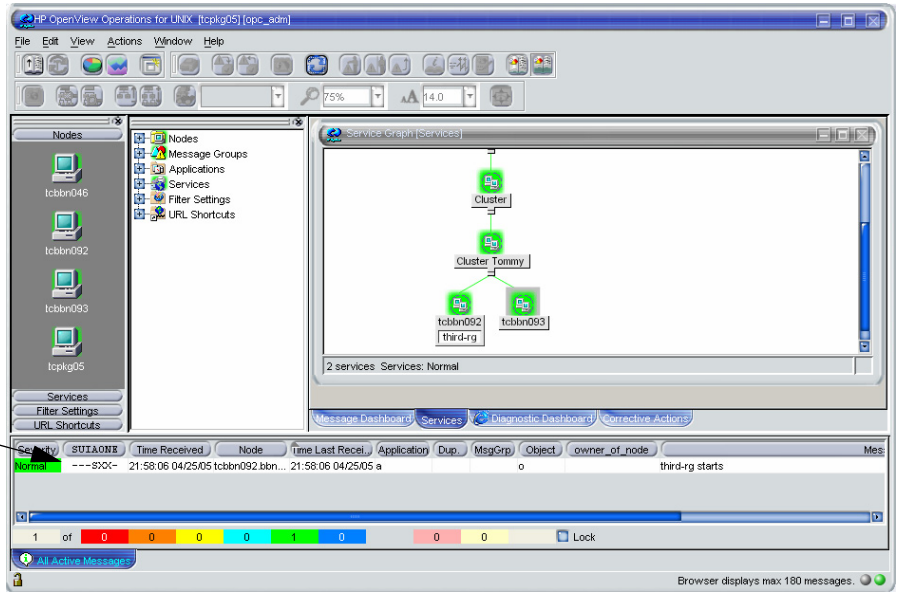


Figure 7-16 Cluster Service View Showing third-rg Running on Node tcbbn092

Message confirms that `third-rg` has been stopped on node `tcbbn092`



Virtual Node FAQs

1. Do I need to configure anything on the agent when using ClAw only for policy enabling and disabling?

   Answer:

   No. The `apminfo.xml` configuration file is not needed for policy enabling and disabling. The ClAw module is designed to monitor ALL resource groups.

2. How can I check which policies on the agent are *cluster-aware*?

   Answer:

   Enter the command:

   `/opt/OV/bin/ovpolicy -list -level 4`

   Check the output for lines with `attribute` and `HARG:`.

3. What happens if a policy is assigned to a virtual node and also assigned to a physical node belonging to that virtual node?

   Answer:

   The policy remains enabled even when the HA package is switched away from the physical node, because there is still the explicit assignment of the policy to the physical node.

4. Can I permanently disable a policy on a cluster? For example, when problems such as message storms arise?

   Answer:

   There is no completely effect method. After a package switch, the policy is normally enabled automatically. Remember that several cluster nodes are concerned and that the policy is duplicated on all cluster nodes.

   A short-term solution can be achieved with the following commands:

   On the managed node:

   `ovpolicy -disable -polname <name>`

   `ovpolicy -remove -polname <name>`

From the OVO management server, you can wrap the same calls into `opcdeploy`:

`opcdeploy -cmd "ovpolicy -..." -node <virtual_nodename>`

5. How should a model for an HTTPS-agent-monitored cluster look like in the OVO database?

   Answer:

   Define one virtual node per monitored HA resource group.

   In addition you can have a normal node group containing the physical nodes of a cluster. This node group can be used to:

   - Assign policies only for the physical nodes.
   - Execute broadcast commands or application calls on ALL cluster nodes instead of only on the active node of an HA package.

6. What happens when I execute an action on a virtual node?

   Answer:

   The task is only executed on the node to which the virtual address refers.

7. Does APM-style (`apminfo.xml` and `<appl_name>.apm.xml` based) policy enabling and disabling collide with virtual node based enabling and disabling when both are defined for the same policy?

   Answer:

   No.

8. Is it of benefit to define virtual nodes for DCE?

   Answer:

   There are limited benefits, but they are not significant enough to justify changes of existing OVO 7 style cluster representations in customer environments.

   For more details see above.

9. How can I switch off ClAw? I do not want to monitor any HA applications on a specific cluster.

   Answer:

   By default, CLAW monitors all resource groups on a system.

   Enter the following command on each cluster node:

```
/opt/OV/bin/ovconfchg -ns conf.cluster -set MONITOR_MODE false
```

This will reduce some CPU load on each system.

10. Can I install or patch the OVO agent software on a cluster by deployment to the virtual node?

    Answer:

    No. You must install or patch each physical node individually.

11. Is the OVO management server running on an HA cluster also modelled as a virtual node?

    Answer:

    Yes. The HA resource group running OVO and NNM is added as a virtual node to the OVO node bank.

Limitations

Status from OVO Patch Level 8.12:

- CMAs are not supported for `trapi`.
- `-option` approach only possible for `opcmon`, and `opcmsg`. Therefore, it is currently not possible to dynamically set CMAs with `opcle`. You can only set the CMAs in logfile policies to hard-coded values or variables from OVO pattern matching. But it is difficult, for example, to subtract an instance name from a directory path of a logfile.

Status before OVO Patch Level 8.12:

- Message key relations are not updated with virtual nodenames. This creates a problem with message correlation on the OVO management server if the nodename is part of the message key. This is the case, for example, when using the `state-based browser` option in the monitor template GUI.
- Operator initiated and automatic actions are executed on the physical nodes specified in a message. Currently it is not possible to let them be executed through virtual nodes, unless virtual nodes are hard-coded into the policy.
- HARG CMA does not exist.

Supported Platforms

See latest OVO 8.x Release Notes.

8 Proxies

Proxies in OVO

Firewall programs and their associated policies, located at a network gateway server, are gateways that are used to protect the resources of a private network from external users. Users of an intranet are usually able to access the approved parts of the Internet while the firewall controls external access to the organization's internal resources.

There are two basic categories of firewalls:

- IP packet filters that work on the network level.
- Proxy servers that work on the application level, for example, a web proxy.

A proxy is a software application that examines the header and contents of Internet data packets and takes necessary action required to protect the systems to which the data is directed. In conjunction with security policies, proxies can remove unacceptable information or completely discard requests.

There are significant security-related advantages of using Application Proxies. These include:

- A fine granularity of security and access control can be achieved as proxies examine packets at the application level. For example, it is possible to restrict specific types of file transfer such as `.exe` files.
- Proxies can provide protection against “Denial of Service” attacks against the firewall.

There are two commonly cited disadvantages of using proxies:

- Proxies require large amounts of computing resources in the hose system but this is no longer a practical issue as powerful computers are now relatively inexpensive.
- Proxies must be written for specific application programs and there may be programs for which proxies are not easily available.

A proxy server stops and inspects all information before letting it access the internal network. Therefore, by using a proxy, there is no direct connection between an internal network and the “outside” world. Users must authenticate to the proxy to be able to send out information. When a client within the intranet attempts to make a request to the Internet,

the proxy actually receives that request. Using Network Address Translation (NAT), the proxy changes the source IP address of the packet to that of the proxy server, which hides the identity of the users on the internal network from the outside. If the request meets the requirements of any established policies, the proxy server forwards this request to the desired address. When a response is received, the process is reversed. As long as the incoming request is deemed to be safe, the request is forwarded to the target client on the network. The source address of the response remains unchanged but the destination address is changed back to that of the requesting machine within the firewall. This confers a dramatic increase in security for the network because there is no direct, uncontrolled route to any network systems.

There are two basic types of proxy servers:

- **Single-Homed Host**

  The proxy server has only one network card and address, and it is the responsibility of the Internet router to forward requests to the proxy server and block all other information to the network.

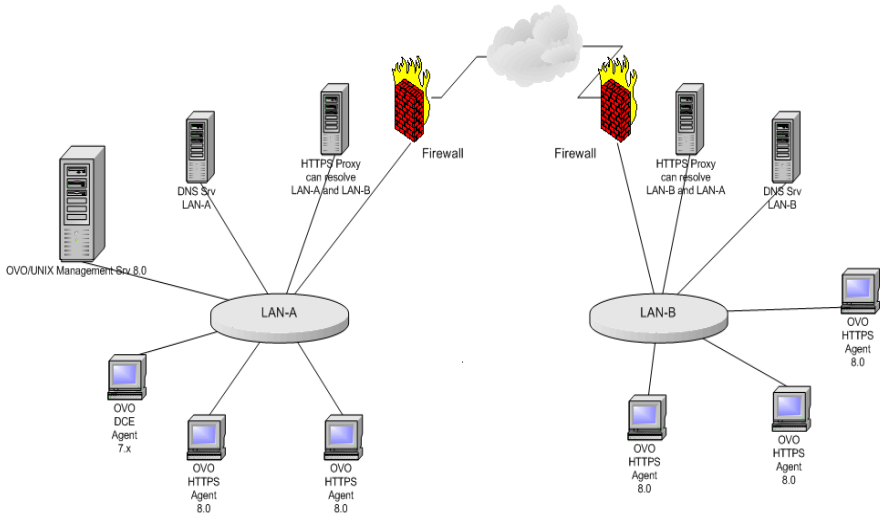
- **Dual-Homed or Multi-Homed Host**

  The proxy server is associated with more than one network card. Requests from the internal network are directed to one of the network cards. Information that comes from the Internet is received by the other network card. There is no routing setup between the network cards, so there is no direct connection between the incoming and outgoing information. The proxy server is responsible for deciding what is sent and to where it is sent.

Configuring Proxies

Most LAN-Internet-LAN architectures can be represented by the following diagram or a subset of the illustration.

Figure 8-1 HTTP Proxy Schematic



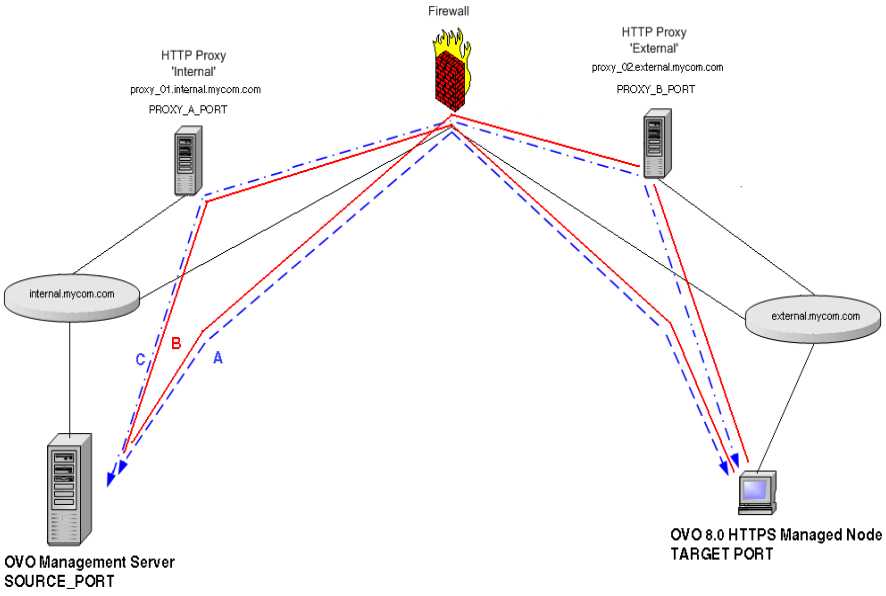
Internal LAN-A includes the OVO management server and an HTTP proxy.

A firewall separates the internal LAN from the Internet and the outside world.

A external LAN-B includes HTTPS managed nodes and an HTTP proxy.

The proxy communication can be represented by the following diagram or a subset of the illustration.

Figure 8-2 HTTP Proxy Infrastructure



A: Direct communication; no Proxy. Firewall must accept all connections from `*.internal.mycom.com:*` to `*.external.mycom.com:TARGET_PORT` and all connections from `*.external.mycom.com.*` to `*.internal.mycom.com:SOURCE_PORT`.

B: `proxy_01` is the proxy in domain `internal.mycom.com` and can access domain `external.mycom.com`. Firewall must accept all connections from `proxy_01.internal.mycom.com:*` to `*.external.mycom.com:TARGET_PORT`.

`proxy_02` is the proxy in domain `external.mycom.com` and can access domain `internal.mycom.com`. Firewall must accept all connections from proxy_01.internal.mycom.com to `*.internal.mycom.com:SOURCE_PORT`.

C: `proxy_01` is the proxy in domain `internal.mycom.com`. `proxy_02` is the proxy in domain `external.mycom.com`. `proxy_01` can access `proxy_02` and `proxy_02` can access `proxy_01`. Firewall must accept all connections from proxy_01.internal.mycom.com:* to

proxy_02.external.mycom.com:PROXY_B_PORT and proxy_02.external.mycom.com:* to proxy_01.internal.mycom.com:PROXY_A_PORT.

The proxies through which an OVO managed node is to communicate must be specified for each system. This is set in the namespace `bbc.http` and stored in the `bbc.ini` file using the `ovconfchg` command. `bbc.ini` must not be edited manually.

Syntax

`ovconfchg -ns <namespace> -set <attr> <value>`

where:

`-ns <namespace>` Sets a namespace for following options.

`-set <attr> <value>` Sets an attribute (proxy) and values (port and addresses) in current namespace.

For example:

```
ovconfchg -ns bbc.http -set PROXY
"web-proxy:8088-(*.mycom.com)+(*.a.mycom.com;*)"
```

Defines which proxy and port to use for a specified hostname.

Format:
`proxy:port +(a)-(b);proxy2:port2+(a)-(b); ...;`

`a`: list of hostnames separated by a comma or a semicolon, for which this proxy shall be used.

`b`: list of hostnames separated by a comma or a semicolon, for which the proxy shall *not* be used.

The first matching proxy is chosen.

It is also possible to use IP addresses instead of hostnames so `15.*.*.*` or `15:*:*:*:*:*:*:*` would be valid as well, but the correct number of dots or colons MUST be specified. IP version 6 support is not currently available but will be available in the future.

```
PROXY=web-proxy:8088-(*.hp.com)+(*.a.hp.com;*)
```

The proxy `web-proxy` is used with port `8088` for every server `(*)` except hosts that match `*.hp.com`, for example `www.hp.com`. If the hostname matches `*.a.hp.com`, for example, `merlin.a.hp.com` the proxy server will be used.

Manual Agent Installation Behind a HTTP Proxy

Manual agent installation where the system is behind a proxy must follow the dedicated sequence of steps:

1. Take all necessary files to the system where you want to install the HTTPS agent software. See “Installing HTTPS Managed Nodes Manually” on page 124 for instructions on manual installation of HTTPS agent software.
2. Start the agent installation script by entering:

   `./opc_inst`

   You can also add server and certificate server options to this command.
3. Set the proxy parameters. For example:

   ```
   ovconfchg -ns bbc.http -set PROXY
   "web-proxy:8088-(*.mycom.com)+(*.a.mycom.com;*)"
   ```
4. When the node needs to be activated and the agent started, enter the command:

   `./opcactivate -srv <srv_name>`

Set Proxies on a Managed Node

To set proxies on an OVO managed node:

1. Manually install the agent software on the managed node system. It will probably not be possible to do a remote installation as the target system cannot yet be reached. See “Installing HTTPS Managed Nodes Manually” on page 124 for instructions on manual installation of HTTPS agent software.
2. Set the proxies over which the OVO agent will communicate with the OVO management server. For example:

   ```
   ovconfchg -ns bbc.http -set PROXY
   "web-proxy:8088-(*.mycom.com)+(*.a.mycom.com;*)"
   ```

3. Stop all agent processes with the command:

   ovc -kill

4. Restart the agent with the command to register the proxy changes:

   ovc -start

Set Proxies on the OVO Management Server

To change the proxy settings on the OVO management server:

1. Set the proxies over which the OVO management server will communicate with its HTTPS managed nodes. For example:

   ```
   ovconfchg -ns bbc.http -set PROXY
   "web-proxy:8088-(*.mycom.com)+(*.a.mycom.com;*)"
   ```

2. Stop all OVO processes with the following commands:

   ovstop ovoacomm

   /opt/OV/bin/OpC/ovc -kill

3. Restart the processes with the following commands to register the proxy changes:

   ovstart ovoacomm

   /opt/OV/bin/OpC/opcsv -start

   /opt/OV/bin/OpC/opcagt -start

Manual Agent Installation Behind a HTTP Proxy With No Name Resolution

In some cases agent and server nodes are not able to resolve eachothers node names. This can be caused by a system configuration where DNS is disabled and `/etc/hosts` file does not introduce other node names. This can be the case with systems behind a proxy which can only resolve the both node names: agent's and management server's.

To install the agent manually on a system behind a proxy, complete the following steps:

1. On the management server, provide a dummy IP address and the real hostname for this managed node in `/etc/hosts` and then add the managed node to the Node Bank (MotifGUI).

   Without this name service entry you cannot add this managed node to Node Bank nor grant a certificate for it.

2. In the Node Bank for this agent, change the Heartbeat Polling (HBP) type from `Normal` to **RPC only**.

   The HBP low-level phase uses ICMP ping, but the pings are not forwarded via HTTP proxy. This is why the need to select `RPC only`, which does not use ICMP ping.

3. Configure the proxy settings on the OVO management server to enable OVO to contact the managed node system. See “Set Proxies on the OVO Management Server” on page 215" for instructions.

4. Copy all necessary files to the system where you want to install the HTTPS agent software. See “Installing HTTPS Managed Nodes Manually” on page 124" for instructions on manual installation of HTTPS agent software.

5. Start the agent installation script with the command:

   `./opc_inst`

   You can also add server and certificate server options to this command and skip step 6.

6. Set the proxy parameter on the agent. For example:

   ```
   ovconfchg -ns bbc.http -set PROXY
   "web-proxy:8088-(*.mycom.com)+(*.a.mycom.com;*)"
   ```

If the agent is already running, restart it:

```
ovc -kill
```

```
ovc -start
```

7. When the node needs to be activated and the agent started (in the case it is not yet started in step 4), enter the command:

   ```
   ./opcactivate -srv <srv_name> -cert_srv \
   <certificate_server_name>
   ```

8. If the certificate has not been manually installed on the agent, trigger certificate request from this managed node system with the command:

   ```
   ovcert -certreq
   ```

9. Grant the certificate for this agent in server's Motif GUI:

   Action →Node →OVO Certificate Requests

10. Optionally remove *<agent_node>* from `/etc/hosts` on the management server.

9 Managing HTTPS Agents on DHCP Client Systems

OVO Agents and DHCP

Dynamic Host Configuration Protocol, or DHCP, enables a DHCP server to dynamically allocate network configurations to computers on an IP network. The primary purpose of this is to reduce the work necessary to administer a large IP network and distributed IP addresses to computers as they are required.

DHCP is a client-server application. When a computer connects to a DHCP server, the server temporarily allocates the computer an IP address. The computer uses this address until the lease expires, at which point it can be replaced with a new IP address.

The main advantage of DHCP is that its addressing scheme is fully dynamic. With a DHCP server running on your network, you can add or move computers around on your network and not have to worry about re-configuring your IP settings.

You can manage OVO HTTPS agents running on DHCP-Client systems. The OVO solution is not dependent on any specific DHCP or DNS product and is based on the following assumptions:

- System names must not change. The system name can be used as an identifier of a system, even in a manager-of-manager (MoM) environment.
- DHCP and DNS are synchronized.
- There are a relatively small number of IP address changes per day so no IP Address Change Event (IPCE) Storm strategy is necessary. An OVO agent sends this event, when it detects an IP address change on one of its network interfaces.
- The Java GUI, and the Administrator and Operator UI processes do not automatically update the IP address changes. Administrators and Operators need to restart their UI processes on receipt of the corresponding warning, to load the latest IP address information.
- DHCP support of agents is configurable for each agent and server.
- Dynamic IP address changes at runtime, not only at startup.

  The time between two IP address change checks can be configured by setting the `IPADDR_CHECK_INTERVAL` variable on the system.

DHCP Settings in OVO

Variables for DHCP

The following variables are used to configure the DHCP-specific behavior of the management server processes.

`OPC_DUMMY_IP_RANGE 1.1.1.*`

If the OVO/UNIX management server detects an IP address conflict while processing an IP change request, the next free IP address out of the `OPC_IP_DUMMY_IP_RANGE` is used. The format of this string is `[1-9*].[1-9*].[1-9*].[1-9*]`. At least one number must be specified. The default is `1.1.1.*`.

`OPC_IPCE_RETRY_NUM 10`

If none of the IP addresses reported by the system matches those of DNS, the IP address change event is buffered. Each event is processed with a maximum number of retries as specified by the `OPC_IPCE_RETRY_NUM` variable. The default is `10`.

`OPC_IPCE_RETRY_INTERVAL 180`

After the `OPC_IPCE_RETRY_INTERVAL` time period has elapsed, all buffered IP change events are processed again. The default is `180` seconds.

opcnode Variables for DHCP

The command `opcnode` has the following DHCP options:

`opcnode -add dynamic_ip=yes|no node_name=<`*`fully qualified domain name`*`>`

The option `-add` includes a parameter `dynamic_ip`. Setting `dynamic_ip` to `yes` configures the OVO management server to accept IP address change events from this new system, in the same way as selecting DHCP in the `Node Modify` window of the Administrator UI.

`opcnode -chg_iptype dynamic_ip=yes|no -node_list=<`*`List of nodes`*`>`

Setting `dynamic_ip` to `yes` configures the OVO management server to accept IP address change events from this modified system, in the same way as selecting DHCP in the `Node Modify` dialog of the Administrator UI.

NNM Synchronization Using dhcp_postproc.sh

The `dhcp_postproc.sh` tool is used by the management server process `ovoareqsdr` after successful processing of an IP address change event. This tool synchronizes NNM after the IP address of a system has been changed. The tool obtains the hostname of the system and its new IP address.

Enabling Management of Agents on DHCP Clients

Complete the following steps to enable management of HTTPS agents on DHCP Clients:

1. Ensure that DHCP and DNS are synchronized, for example by updating from the DHCP Server. If synchronization is not achieved, the OVO management server cannot process any IP address change events and it will decrease the overall performance of the system.
2. Configure NNM to process DHCP. This is described in the OVO online help; section entitled *Deleting Inaccessible DHCP IP Addresses*.
3. Customize `/opt/OV/contrib/OpC/dhcp_postproc.sh`

   Customize the script to suit your environment. The following entries are of particular interest:

```
NETMASK="255.255.248.0" # netmask

MAXRETRY=5     # number of retries for opctranm

SLEEP_TIME=10  # sleep this amount of seconds
               # before the next retry

TRACE="off"    # on=do (or off=do not) create
               # lots of tracefiles in /tmp

NETMON_TOPO_FIX="OFF" #off is highly recommended

FORCE_NODEINFO_DIST #off
```

   You may add `opcmsg` or `opcwall` calls.

10 MOM Environments

Environments with Multiple OVO Management Servers (MoM)

The MoM concepts for the HTTPS agent and DCE agents is very similar. Refer to the chapter titled *Scalable Architecture for Multiple Management Servers* in the *HP OpenView Operations Concepts Guide* for further information. Configuration information is available from the *HP OpenView Operations Administrator's Reference*. Message target rules that specify where messages go to and remote access rules that specify which OVO server is allowed to do which tasks are configured on the agent. Both message target rules and remote access rules are defined in the responsible manager policy (formally known as the `mgrconf` file). On the OVO management server, you must set up the responsible manager policy in the same way as for OVO 7 (manager conf syntax has not changed).

With OVO 8 and the HTTPS agent, new security concepts have been introduced. Some additional factors need to be taken into consideration. For details, refer to the steps in "Environments Hosting Several Certificate Servers" on page 59 to first establish a trust between the multiple configuration servers.

For more information about multiple configuration servers, refer to "Configuring Multiple Configuration Servers" on page 233.

Responsible Manager Terminology for the HTTPS Agent

The OpenView responsible manager concept for HTTPS agents is based on the following terms:

- **OV Access Rights**

  OV components can define access rights. These are the rights to, for example to execute actions, deploy files, and configure settings. The rights are mapped to pre-configured OpenView roles. It is possible to alter the mappings by changing configuration settings under the namespace `sec.core.auth.mapping.*`, for example, to stop remote access to a managed node.

- **OV Defined Roles**

  An OVO management server can assume an OpenView defined role. The mapping between management servers and roles is defined in the responsible manager policy and in specific configuration settings.

- **Local User Role**

  The local user has all rights, assuming appropriate system rights are given, for example `root`.

- **Initial or Authorized Manager Role**

  This manager has all rights and is setup at install time to allow remote access if required. This node is defined by the `MANAGER` and `MANAGER_ID` settings in the security namespace `sec.core.auth`. There can be only one initial manager.

- **Secondary Manager Role**

  A secondary manager has all rights including action execution and configuration deployment. There can be multiple secondary managers defined in the responsible manager policy. The initial manager and the secondary managers make up the group of possible configuration servers.

- **Action-allowed Manager Role**

  An action-allowed manager has no other rights than the action execution right. There can be multiple action-allowed managers defined in the responsible manager policy.

- **Certificate Authority Defined**

  The setting `CERTIFICATE_SERVER` from the security namespace `sec.cm.client` is setup during installation. It defines the system with the certificate authority, which is contacted to get a valid subscribed certificate for the managed node. The certificate server cannot be defined in the responsible manager policy.

Backward Compatibility and the Differences between OVO 7 and OVO 8

Here is a list of changes in the MoM concept and information about backward compatibility:

- Secondary managers have action execution rights on HTTPS agents only.
- OVO 7.x responsible manager files can be used for OVO 8 agents without changes.
- Secondary managers can deploy configuration data without the primary manager switch on HTTPS agents. On DCE agents you must first call the primary manager switch:

  `opcragt -primmgr`

- For OVO 7 and OVO 8, message assignment to managers, `opcragt -primmgr` modifies the primary message target manager.
- For configuration deployment, no existing configuration information is removed from an HTTPS agent by a new configuration distribution (`opcragt -primmgr` call) from a secondary server. For DCE agents, this is possible if the primary and secondary servers are not identically configured.
- If there are no templates assigned for a node, the server clears all configuration on a DCE agent, but it changes nothing on an HTTPS agent unless it uses the same owner string as the other server.

  In environments with both HTTPS and DCE agents, only one configuration server should be used. This server must execute an `opcragt -primmgr` call before deploying data. In pure HTTPS environments you have more flexibility due to the separation of configuration servers and message target servers.
- All OVO 7 MoM templates can also be used on HTTPS agents. However, HTTPS agents cannot communicate with OVO 7 management servers. Therefore, all OVO management servers that are referenced in the responsible manager policy of an HTTPS agent must be upgraded to OVO 8. The MoM configuration file `allnodes.bbc` on the management server is available to aid migration from OVO 7 to OVO 8. This file has higher priority than

the `allnodes` file for data deployment to HTTPS nodes. `allnodes.bbc` should contain only OVO 8 management servers. It can be moved to `allnodes`, when all servers are updated.

- All OVO management servers which are referenced in a responsible managers policy must be added to the node bank, and their `OvCoreId` must be added to the database. `OvCoreIds` are automatically added to the responsible managers policy during deployment. The authorization of servers on HTTPS managed nodes is bases on authorized OvCoreIds.
- If you setup a MoM environment with HTTPS nodes, some certificate related configurations must be made. For details see “Environments Hosting Several Certificate Servers” on page 59.

Upgrading in a MoM Environment

When upgrading in a MoM environment, there are two main steps to consider:

- Upgrading the OVO management servers to OVO 8.
- Upgrading the managed nodes to OVO 8 HTTPS agents.

Execute the following steps to upgrade your OVO 7.x MoM environment to an OVO 8 MoM environment:

1. Upgrade at least one OVO 7.x management server to OVO 8 management server.
2. Migrate the DCE agents to HTTPS agents as described in “Migrate a DCE Agent to an HTTPS Agent” on page 120.

NOTE

The OVO 7.x management server will no longer be able to manage these migrated systems as HTTPS agents are not supported by OVO 7.x management servers.

3. Download the configuration data from the first OVO 8 management server using the `opccfgdwn` utility. For more information, refer to *To Download the Current OVO A.07.1x Configuration* in the *HP OpenView Operations Installation Guide for the Management Server*.
4. Upload the downloaded configuration data to any additional OVO 8 management server using the `opccfgupld` utility. For more information, refer to *To Upload the Saved OVO A.07.1x Configuration* in the *HP OpenView Operations Installation Guide for the Management Server*.
5. Set the install flag in the database for your HTTPS agents. Without this, the uploaded nodes cannot be added automatically into the heartbeat polling list, causing problems for heartbeat polling and configuration distribution.

   Enter the command:

   `opcsw -i <https_node_name>`
6. Repeat steps 4 and 5 for all other OVO 8 management servers.

7. Establish a trust between two or more managers so that their environments are able to communicate with each other. Complete the steps described in “Certificate Handling for a Second OVO Management Server” on page 64.

8. Create a responsible manager file for HTTPS nodes and deploy it to the agents:

   `/etc/opt/OV/share/conf/OpC/mgmt_sv/respmgrs/allnodes.bbc`

   `allnodes.bbc` has a higher priority than the `allnodes` file but lower than the *<hex_IP_addr>* files for an HTTPS node. The file is distributed automatically together with policies, in the same way as the `allnodes` file with templates.

   `allnodes.bbc` can be empty or contain only a subset of the settings from the `allnodes` file. An empty `allnodes.bbc` file means that no MoM configuration is distributed to an HTTPS node and previously deployed MoM configurations are removed if the owner is the same management server that originally distributed the configuration. All OVO management server systems specified in a responsible manager file going to an HTTPS node must use HTTPS as the communication type and have an `OvCoreId`.

Message Target Rules (OPC_PRIMARY_MGR Setting)

There is a setting called `OPC_PRIMARY_MGR` in the OVO agent namespace `eaagt`. It specifies the hostname of the OVO management server to which OVO messages are sent by default. This agent setting is modified by OVO management servers using the command:

```
opcragt -primmgr
```

If `OPC_PRIMARY_MGR` is not set or is invalid, the OVO management server is denoted by the `MANAGER` setting. Invalid means that the `OPC_PRIMARY_MGR` is not specified as a secondary manager nor as an action-allowed manager, nor is it the initial manager. The `OPC_PRIMARY_MGR` is only a message related setting and it maps to the `$OPC_PRIMARY_MGR` variable which may be used in message target rules of the responsible manager policy so that messages are sent to that OVO management server.

Multiple Parallel Configuration Servers

Multiple parallel configuration servers are supported for HTTPS nodes. The OpenView policy concept allows multiple OpenView products to independently work with policies on an agent by providing an owner concept for policies. The policy header includes an attribute `owner`, which can be set by the OVO management server. This is a logical association using a concept of agreements between management servers to decide which management server is responsible for which configuration (policies) on an agent. Normally all policies (templates) associated with an OVO management server can be modified by this management server. This means that two different management servers will not interfere when distributing policies to the same agent, because they have a different name.

Let us now understand when multiple parallel configuration servers could used. For example, a service provider manages the hardware and operating systems of a set of customer systems. The customer himself manages an application on the same set of nodes. Both the service provider and the customer each use their own OVO management server to manage these systems. The implementation of a solution could be as follows:

- Service provider and customer create their own certificates but agree on a trust, so that the agent accepts action and configuration requests from both OVO management servers.
- Service provider and customer agree on a responsible manager policy (`mgrconf` policy). In this case, the service provider acts as the primary manager, and the customer as a competence center. Both OVO management servers must be listed in the `mgrconf` policy.
- The competence center is also allowed to deploy configurations. It provides all policies with a specific attribute which can be matched by the message target rules of the responsible manager file. Related messages are then sent to the competence center, all others to the primary manager.

Configuring Multiple Configuration Servers

Multiple configuration servers can be used in two different scenarios:

- **Backup Server**

  Typically in a backup scenario, two OVO management servers are configured identically. The main installation is referred to as the primary management server and the other as a backup server.

- **Competence Center**

  In a competence center scenario, the management responsibilities are split between different OVO management servers. Typically, the competence center management server is responsible for dedicated applications, such as SAP, and the primary management server is responsible for the rest of the duties.

When policies are deployed to HTTPS agents, they are provided with an owner string. This owner string is fetched from a configuration setting on the management server (`OPC_POLICY_OWNER in the` namespace `opc`). Default value is `OVO:<server_fully_qualified_name>`).

Primary- and backup management servers should share the same owner string.

In the competence center scenario, normally all parties retain their default owner strings.

When a backup management server is desired, you can overwrite the default owner string by using the following command on the backup management server:

```
ovconfchg -ovrg server -ns opc -set \
OPC_POLICY_OWNER <OVO:primary_server_fully_qualified_name>
```

NOTE You can also change the `OPC_POLICY_OWNER` string to any desired value but they must be identical on both management servers.

NOTE Be aware that only one owner string can be set per management server. If a manager acts as backup for a certain OVO domain, but as competence center for another one, this will not work.

NOTE

If the backup management server is not setup in exactly the same way as the primary management server, the agent may be configured differently when templates and instrumentations are deployed. Instrumentation files from the primary management server remain, if not overwritten by the backup management server. Additional instrumentation files from the backup management server will be deployed and cumulated on the agent. Policies are only replaced, if the primary and backup management server use the same owner string or if policies are identical. All other policies on the agent will remain unchanged, because they belong to different owners.

mgrconf and nodeinfo Policies in Multiple Configuration Server Environments

`mgrconf` and `nodeinfo` policies are treated as special cases. The rules described in the section “Dealing with Identical Policies Deployed by Different Management Servers” on page 235 are not applicable to these two policies.

For OVO UNIX, there can only be one instance of either of these policies per manage node. The management server which deploys either of these policies first will permanently remain the owner. The second management server will not overwrite the existing policy. Therefore, it is recommended that in competence center scenarios, only one server is used to deploy `mgrconf` policies.

If it is really necessary to change the owner attribute for `mgrconf` and `nodeinfo` on a managed node, execute the appropriate following commands:

If you are at a management server system, execute the following command:

For nodeinfo:

```
opcdeploy -cmd "ovpolicy -setowner \
OVO:<your_full_qualified_mgmt_server_name> -poltype \
configsettings" -node <your_managed_node_name>
```

For mgrconf:

```
opcdeploy -cmd "ovpolicy -setowner \
OVO:<your_full_qualified_mgmt_server_name> -poltype \
mgrconf" -node <your_managed_node_name>
```

If you are at a managed node system, execute the following command:

For `nodeinfo`:

```
ovpolicy -setowner \
OVO:<your_full_qualified_mgmt_server_name> -poltype \
configsettings
```

For `mgrconf`:

```
ovpolicy -setowner \
OVO:<your_full_qualified_mgmt_server_name> -poltype \
mgrconf
```

Dealing with Identical Policies Deployed by Different Management Servers

Policies are identified using their Ids and policy name, type and version. If an ID is present, it has a higher priority than a name plus policy type and version.

Identical policies are determined in the following way:

- The same policy ID
- The same policy name, type and version, but different policy ID.

Identical policies can be modified by multiple management servers, independent of the policy owner. This avoids many instances of the same policy being installed on an agent and avoids multiple messages being created for the same issue.

If multiple servers are used to deploy the same configuration data, they are acting as backup management servers, and their data should be synchronized.

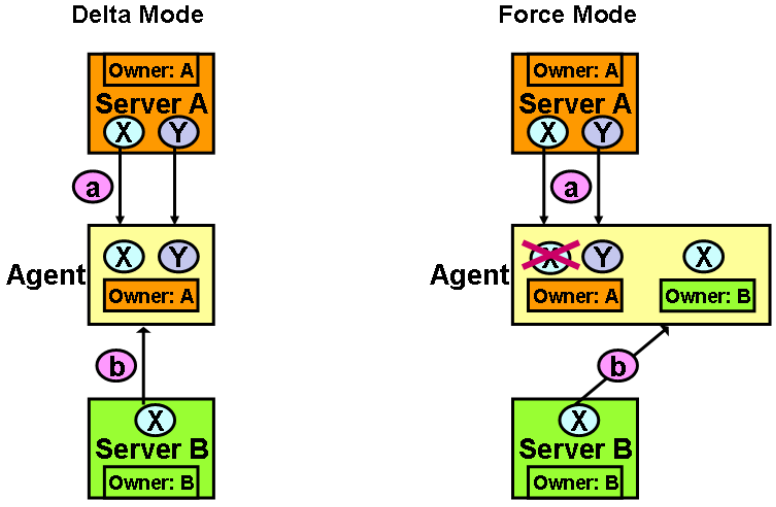
With regards to the owner concept, the following examples show you how the policies are handled between multiple configuration servers.

Let us assume that we have management server A and management server B, policy X and policy Y. Policy X is assigned to the agent from both management server A and management server B. Policy Y is assigned to the same agent only from management server A.

Server A and server B use different owner string. Server A use owner string “A”. Server B use owner string “B”.

1. Trigger configuration distribution.

Figure 10-1 Policy Handling between Multiple Configuration Servers, Server A has Different Owner to Server B.



a. From server A, in delta mode and force mode:

   Policy X and Policy Y are deployed and have owner “A”.

b. From server B, in delta mode:

   Nothing is changed for policy X. Since policy X is already installed, it will remain the same and has owner “A”. Nothing is changed for policy Y. It still has owner “A”.

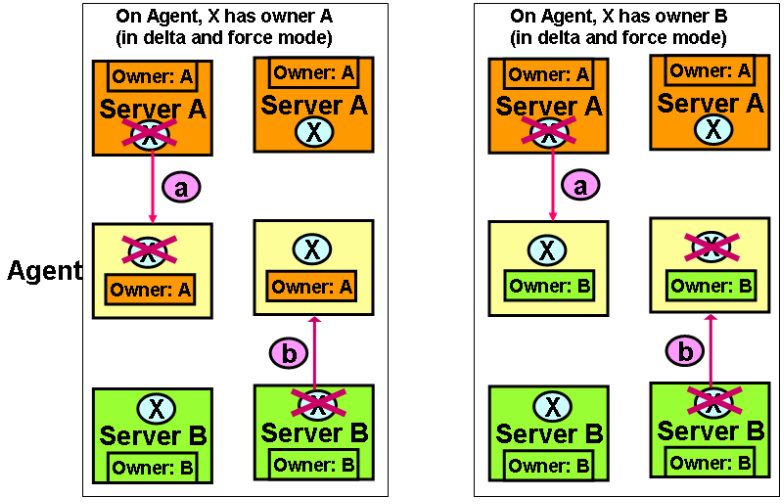
   From server B, in force mode:

   Policy X is overwritten and has owner “B”.

   Nothing is changed for policy Y. It still has owner “A”.

2. De-assign policy X and trigger distribution.

Figure 10-2 Policies Handling between Multiple Configuration Servers, Server A has Different Owner to Server B.



a. If from server A, in delta mode and force mode:

   If policy X has owner “A”, it is removed from the agent.

   If policy X has owner “B”, it remains the same, because of the different owner string.

b. If from server B, in delta mode and force mode:

   If policy X has owner “A”, it remains the same, because of the different owner string.

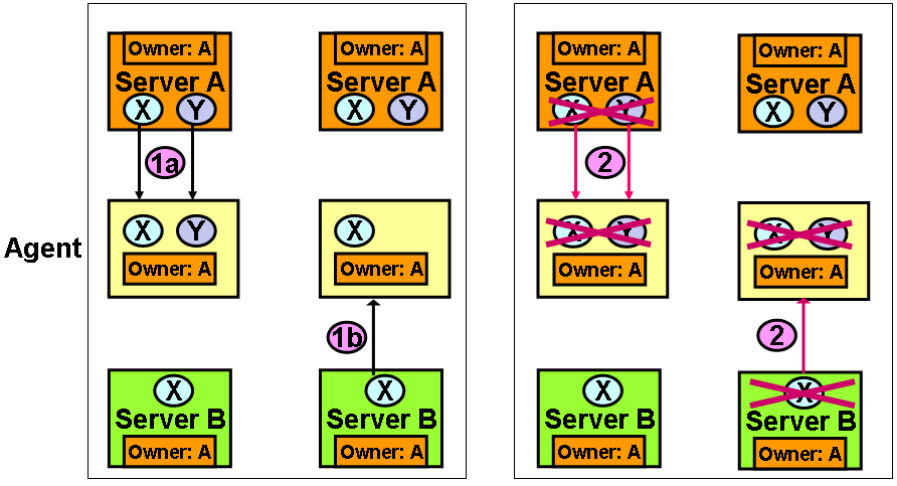
   If policy X has owner “B”, it is removed from the agent.

3. De-assign policy Y from server A, in delta mode and force mode:

   Policy Y is removed.

Server A and server B use same owner string “A”.

Figure 10-3 **Policies Handling between Multiple Configuration Servers, Same Owner for Servers A and B, and Delta or Force Modes**



1. Trigger configuration distribution.
   a. From server A, in delta mode and force mode:

      Policy X and Policy Y are deployed and has owner “A”.

   b. From server B, in delta mode:

      Nothing is changed for policy X. It still has owner “A”. Policy Y is removed.

      In force mode:

      Policy X is overwritten and still has owner “A”. Policy Y is removed.

2. De-assign policy X and trigger distribution.
   a. If from server A, in delta mode and force mode:

      Policy X is removed.

   b. If from server B, in delta mode and force mode:

      Policy X is removed.

3. De-assign policy Y from server A, in delta mode and force mode:

   Policy Y is removed.

A policy can only be removed by its owner. With regards to policy removal, the following important scenario must be considered:

Let us assume that we have a backup management server scenario. Initially, the primary management server (A) deploys policy (PA) to agent (G). Then policy (PA) has owner (A).

Next, the backup management server (B) deploys the same policy (PA) to the same agent (G). Because the policy is identical, the already installed policy (PA) with owner (A) is removed and re-installed from the backup management server B. Now, the reinstalled policy (PA) has owner (B).

Finally, on the primary management server (A), de-assign policy (PA) and issue template distribution to the same agent (G).

The result is that policy (PA) is NOT removed from agent (G), because policy (PA) has owner (B). Thus only the backup management server (B) can remove it.

The `delta` and `force` distribution modes are also available for multiple server environments. `force` replaces all policies of the calling owner and all identical policies, even though they are owned by different management servers.

For non-identical policies, in delta and force mode, a de-assigned policy is only removed by the same owner.

How to List and Modify Policy Owners on the Agent

The local policy utility `ovpolicy` can modify all policies on a system by default. Specify the `-owner` option to select policies belonging to a specific owner.

To list all policies of any owner, use the command:

```
ovpolicy -l
```

For example, to only list the policies for `my_srv`, use the command:

```
ovpolicy -l -owner OVO:<my_srv_full_qualified_name>
```

`ovpolicy` can also be used to modify owner strings of policies, for example, if the owner string of an OVO management server must be changed without the need to redeploy the configuration for a managed node. For details, refer to the `ovpolicy` man page.

Cleaning-up the Agent

To remove and re-deploy all policies from all OpenView applications from HTTPS nodes, use the command:

```
opcragt -distrib -purge -templates <nodename>
```

Instrumentation deployment is cumulative. Neither the `delta` nor the `force` installation removes any file on the agent, but only updates existing configurations and adds new ones.

If you want to cleanup and re-install all configuration data in the instrumentation directory on the agent, use the command:

```
opcragt -distrib -purge -actions -monitors -commands \
<nodenames>
```

11 Variables in OVO

Setting Variables in OVO

NOTE

The `opcsvinfo` file is no longer used by OVO 8. It is saved during the upgrade procedure to the directory:

`/tmp/save710/`

The `opcsvinfo` file from an OVO 7.1 installation is NOT automatically converted when upgrading to OVO 8. If you want to convert the contents of the `opcsvinfo` files, save the file to a temporary location and use the tool:

`/opt/OV/contrib/OpC/opcinfoconv`

The `opcinfo` file is converted when you upgrade an OVO 7.1 agent to an HTTPS agent. A copy is saved to the local `/tmp/opcinfo.save` file.

To set variables on the OVO management server:

1. Enter the command:

   `/opt/OV/bin/ovconfchg -ovrg server -ns opc -set \`
   `<var_name> <value>`

2. Restart server processes.

All relevant variables that were available in the `opcsvinfo` files are also used by OVO 8. The OVO 8 schema uses namespaces (the parameter `-ns` from the example above). All former `opcsvinfo` variables now have the namespace `opc`, all former `opcinfo/nodeinfo` variables on HTTPS nodes have the namespace `eaagt`. The DCE agents still use the `opcinfo` files.

You can suffix the namespace by the process name if required. For example, to set the port for the DCE message receiver `opcmsgrd`, enter the command:

```
ovconfchg -ovrg server -ns opc.opcmsgrd -set \
OPC_COMM_PORT_RANGE 12345
```

To read the variables on the OVO management server, enter the command:

```
/opt/OV/bin/ovconfget -ovrg server \
[ <namespace> [ <var_name> ] ]
```

which either prints all settings, all settings of a namespace, or one variable.

To read variables on a managed node, use the `ovconfget` command, but without `-ovrg server` option.

To set a variable on an agent use `ovconfchg` without the `-ovrg server` option.

```
/opt/OV/bin/ovconfget [ <namespace> [ <var_name> ] ]
```

You can delete variables with `ovconfchg -clear` option.

```
/opt/OV/bin/ovconfget -clear [ <namespace> [ <var_name> ] ]
```

You can find more documentation and examples about configuration settings under:

```
/opt/OV/misc/xpl/conf/defaults/*.ini
```

A Troubleshooting HTTPS-based Communication

Troubleshooting

If communication between an OVO Management Server and an HTTPS agent appears to be interrupted, for example, messages do not arrive at the Message Browser, or software or instrumentation is not distributed, execute the appropriate troubleshooting steps as described in the following sections.

Before you continue with the described actions, you should be familiar with the new HTTPS agent and the underlying communication concepts such as certificates.

This guideline describes possible actions to identify and solve HTTPS communication problems between OVO management servers, Certificate Authority Servers and OVO managed node agents.

It is assumed, that the OVO HTTPS agent software is installed, but there is a problem in the communication between OVO managed nodes and OVO management servers in one or both directions.

In most installations, the OVO management server and Certificate Authority servers are installed on the same system.

Troubleshooting problems encountered with the communication between an OVO management server and an OVO HTTPS agent is split into the following areas:

- Troubleshooting Tools
- Logging
- Troubleshooting Processes

Troubleshooting Tools

Ping an HTTPS-Based Application

HTTPS-based applications can be pinged to test if the application is active and responding. A ping may be executed against an application whether or not it has SSL enabled.

The `bbcutil` utility supports a `-ping` command line parameter that can be used to ping an HP OpenView HTTPS-based application.

Use the following command to ping a specified HTTPS-based application:

From an OVO managed node:

```
<OvInstallDir>/bin/bbcutil -ping [<hostname_or_ip_addr>] [count]
```

From an OVO management server:

```
<OvInstallDir>/bin/bbcutil -ovrg server -ping \
[<hostname_or_ip_addr>] [count]
```

For example:

HTTP `bbcutil -ovrg server -ping http://...`

HTTPS `bbcutil -ovrg server -ping https://...`

Checks whether the communication service on the managed node specified by *<hostname_or_ip_addr>* is alive. If the hostname or IP address is omitted, localhost is assumed. An optional loop count can be specified after the hostname or IP address which causes the ping command to be repeated by the number of times specified.

See the `bbcutil` man page for details of the command line parameters.

In general, all `bbcutil` calls from an OVO management server to a managed node should include the `-ovrg server` parameter. for example:

```
bbcutil -ovrg server -ping https://...
```

If the OVO management server is a stand-alone system, the `-ovrg server` parameter maybe omitted. However, if the OVO management server is installed on an HA cluster, the `-ovrg server` parameter is required because a managed node certificate and a server certificate including two `OvCoreIds` are installed on each OVO

management server. While on stand-alone systems, the managed node certificate and server certificate, including the `OvCoreIds`, are identical, they differ on cluster installations. The agent is only aware of the management server `OvCoreId`. It is not aware of the `OvCoreId` value of the management server.

Display the Current Status of an HTTPS-Based Application

An HTTPS-based application at a specified location can be requested to display its current status.

Use the following command to query a specified application:

`bbcutil -status <hostname_or_ip_addr:port>`

Queries the communication server located at the hostname and port specified by *<hostname_or_ip_addr:port>* for details about the current state of the server.

See the `bbcutil` man page for details of the command line parameters. If a port is not specified, the port number of the Communication Broker is used.

Display All Applications Registered to a Communication Broker

The Communication Broker at a specified location can be requested to display all applications that are registered to it.

Use the following command to list all applications that are registered to the specified Communication Broker:

`bbcutil -registrations|-reg <hostname_or_ip_addr>`

Queries a Communication Broker on the managed node specified by *<hostname_or_ip_addr>* and displays a list of all registered applications. If the hostname or IP is omitted, localhost is assumed.

See the `bbcutil` man page for details of the Communication Broker command line parameters.

Use What String

All executables contain a detailed UNIX-style `what` string that can be used to determine the precise version of the HTTPS-based communication software installed. Microsoft Windows executables also contain standard property strings.

List All Installed OV Filesets on an HTTPS Managed Node

The `ovdeploy` tool can be used to list the installed OpenView products and components. The following three levels of information can be displayed:

- Basic inventory
- Detailed inventory
- Native inventory

The following sections illustrate how to list the inventory and show examples of the output.

Basic Inventory

To display basic inventory information, enter the following command:

From a managed node:

ovdeploy -inv -host *<hostname>*

From an OVO management server:

ovdeploy -ovrg server -inv -host *<hostname>*

For example:

```
ovdeploy -ovrg server -inv -host hp_System_002

NAME                                          VERSION    TYPE
ARCHITECTURE
HP OpenView HTTP Communication                05.00.070  package
Windows 4.0 5.0 5.1 5.2
HP OpenView Deployment                        02.00.070  package
Windows 4.0 5.0 5.1 5.2
HP OpenView Security Certificate Management   01.00.070  package
Windows 4.0 5.0 5.1 5.2
```

```
HP OpenView Security Core                    02.00.070  package
Windows 4.0 5.0 5.1 5.2
...
```

Detailed Inventory

To display detailed inventory information, enter the following command:

From a managed node:

ovdeploy -inv -all -host *<hostname>*

From an OVO management server:

ovdeploy -ovrg server -inv -all -host *<hostname>*

For example:

```
ovdeploy -ovrg server -inv -all -host hp_System_002
```

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?> <inventory
    xmlns=">http://openview.hp.com/xmlns/depl/2003/inventory">
  <host>hpspi002.bbn.hp.com</host>
  <date>Thursday, October 30, 2003 12:24:48 PM</date>
  <package>
    <name>HP OpenView HTTP Communication</name>
    <version>05.00.070</version>
    <systemtype>IA32</systemtype>
    <ostype>Windows</ostype>
    <osvendor>MS</osvendor>
    <osversion>4.0 5.0 5.1 5.2</osversion>
    <osbits>32</osbits>
    <nativeinstallertype>msi</nativeinstallertype>
  </package>
  <package>
    <name>HP OpenView Deployment</name>
    <version>02.00.070</version>
    <systemtype>IA32</systemtype>
...
```

Native Inventory

To display native inventory information, enter the following command:

From a managed node:

ovdeploy -inv -it native -host *<hostname>*

From an OVO management server:

ovdeploy -ovrg server -inv -it native -host *<hostname>*

For example:

```
ovdeploy -ovrg server -inv -it native -host hp_System_002
```

```
NAME                                          VERSION
WebFldrs XP                                   9.50.5318
HP OpenView Core Library                      2.50.70
HP OpenView Certificate Management Client     1.0.70
HP OpenView HTTP Communication                5.0.70
ActivePerl 5.6.1 Build 633                    5.6.633
HP OpenView Deployment                        2.0.70
Microsoft FrontPage Client - English          7.00.9209
```

Standard TCP/IP Tools

If SSL is not enabled, standard TCP/IP tools such as telnet can be used to contact HP OpenView HTTPS-based application. To use telnet to ping an HTTPS-based application execute the following commands:

Two carriage returns are required after the `PING` input line to telnet.

To end the telnet session, enter **`control-D`** and **Return**:

```
telnet <host> <port>
PING /Hewlett-Packard/OpenView/BBC/ping HTTP/1.1
```

The output takes the following form:

```
HTTP/1.1 200 OK
content-length: 0
content-type: text/html
date: Thu, 08 Aug 2002 08:20:24 GMT
senderid: fd7dc9c4-4626-74ff-9e5a09bffbae
server: BBC X.05.00.01.00; ovbbccb 05.00.100
```

`HTTP status 200 OK` indicates the HTTPS-based application has recognized the request and successfully responded. Other status may indicate a failure in the request or other error.

For a list of error codes, refer to :

`http://www.w3.org/Protocols/rfc2616/rfc2616.html`

RPC Calls Take Too Long

If an RPC call takes longer than the default timeout of 5 minutes, the following error messages may be displayed, for example, for a policy installation:

```
ERROR:   General I/O exception while connecting to host '<hostname>'.
           (xpl-117) Timeout occurred while waiting for data.
```

or

```
ERROR:  The Configuration server is not running on host '<hostname>'.
Check
        if the Configuration server is in state running.
        (bbc-71) There is no server process active for address:
        https://<hostname>/com.hp.ov.conf.core/bbcrpcserver
```

This may happen if 1000 policies are installed using the `PolicyPackage` interface from `OvConf` or if the connection or target-machine is slow.

To prevent this the communication timeout (response timeout) can be changed using the following commands with the required time out value:

On the target system:

ovconfchg -ns bbc.cb -set RESPONSE_TIMEOUT *<seconds>*

On the OVO management server:

**ovconfchg -ovrg server -ns bbc.http.ext.conf -set \
RESPONSE_TIMEOUT *<seconds>***

NOTE The `RESPONSE_TIMEOUT` parameter must be set on both managed nodes.

A similar situation can arise when running any command that takes over 5 minutes to complete. The timeouts should be extended as follows.

On the OVO managed node enter the commands:

NOTE The unit is milliseconds in the second case.

ovconfchg -ns bbc.cb -set RESPONSE_TIMEOUT *<seconds>*

ovconfchg -ns depl -set CMD_TIMEOUT *<milliseconds>*

On the OVO management server, enter the command:

```
ovconfchg -ovrg server -ns bbc.http.ext.depl -set \
RESPONSE_TIMEOUT <seconds>
```

Logging

Errors in violation of security rules are recorded in a logfile. For HTTPS-based servers, all client access can be additionally logged, if enabled.

To enable logging of all client access, set the following parameter value using the command:

```
ovconfchg -ns bbc.cb -set LOG_SERVER_ACCESS true
```

This will log all access to the Communication Broker. To view the logs, open the text file:

***<OvDataDir>*/log/System.txt**

You can additionally log access to all OV Communication Broker servers using the command:

```
ovconfchg -ns bbc.http -set LOG_SERVER_ACCESS true
```

You can additionally log all client access to the configuration and deployment application using the command:

```
ovconfchg -ns bbc.http.ext.conf -set LOG_SERVER_ACCESS true
```

Communication Problems between Management Server and HTTPS Agents

The most likely areas where communication problems may be experienced are divided into the following sections:

Network Troubleshooting Basics

Basic network troubleshooting uses the following commands:

`ping`	**`<SYSTEMPATH>/ping`**
`nslookup`	**`<SYSTEMPATH>/nslookup`**
`telnet`	**`<SYSTEMPATH>/telnet`**
`ovgethostbyname`	**`<INSTALLDIR>/bin/ovgethostbyname`** (for use on Solaris systems only in place of `nslookup`)

NOTE

The actions described below may not work if communication between an OVO management server or Certificate Authority server and OVO managed node has to pass:

- Firewalls
- NATs
- HTTP Proxies

Contact your Network Administrator for more information.

To check for basic network problems, complete the following steps:

1. Check if the name resolution for the OVO management server, Certificate Authority server and OVO managed node is consistent on all affected systems.

   Use `ping`, and `nslookup` (on Solaris: `ovgethostbyname`) with the Fully Qualified Domain Name (FQDN) on all systems with all systems as targets.

   `bbcutil -gettarget <nodename>`

2. Check if all systems (OVO management server, Certificate Authority server and OVO managed node) are accessible.

   Use one of the following commands:

   - **`<OvInstallDir>/bin/bbcutil -ping <FQDN>`**
   - **`telnet <FQDN>`**

3. Check if HTTP communication is working by using a Web browser to connect to the Communication Broker. The Communication Broker, `ovbbccb`, must be running for this check.

   To retrieve the assigned *<AGENT-BBC-PORT>* value, enter the command:

   `bbcutil -getcbport <agenthostname>`

   For example, if you enter the command:

   ```
   bbcutil -getcbport mysystem.mycom.com
   ```

   Output of the following form is displayed:

   ```
   mysystem.mycom.com:8008
   ```

   On the OVO management server system, open a Web browser and enter the following URL:

   `http://<OVO managed node>:<AGENT-BBC-PORT>/ \`
   `Hewlett-Packard/OpenView/BBC/`

   The default port number for *<AGENT-BBC-PORT>* is 383.

   Repeat this step from the managed node to the OVO management server:

   `http://<OVO management server>:<AGENT-BBC-PORT>/ \`
   `Hewlett-Packard/OpenView/BBC/`

The `HP OpenView BBC Information Modules` page should appear and allow you to check `ping` and `status` or list registered services and OV resource groups (ovrg).

HTTP Communication Troubleshooting Basics

Basic HTTP communication troubleshooting uses the following commands:

`ovc`	**<*INSTALLDIR*>/bin/ovc**
`ovconfget`	**<*INSTALLDIR*>/bin/ovconfget**
`ovbbccb`	**<*INSTALLDIR*>/bin/ovbbcutil**
`ps`	**<*SYSTEMPATH*>/ps**

NOTE Even if the communication between OVO management server or Certificate Authority server and OVO managed node has to pass:

- Firewalls
- NATs
- HTTP Proxies

the following actions must work! If they do not, contact your Network Administrator for more information.

NOTE If the communication between OVO management server or Certificate Authority server and OVO managed node is not allowed to pass through the firewalls, one or more HTTP Proxies must be used (see the corresponding sections).

To check for HTTP communication problems, complete the following steps:

1. On all systems, the OVO management server, Certificate Authority server and OVO managed node, check if:

   The OV Communication Broker ovbbccb is running with the following commands:

   ovc -status

   The ovbbccb process must be listed as running. The output takes the following form:

```
ovcd      OV Control                CORE       (2785)  Running
ovbbccb   OV Communication Broker    CORE       (2786)  Running
ovconfd   OV Config and Deploy       CORE       (2787)  Running
ovcs      OV Certificate Server      SERVER     (3024)  Running
coda      OV Performance Core        AGENT      (2798)  Running
opcmsga   OVO Message Agent          AGENT,EA   (2799)  Running
opcacta   OVO Action Agent           AGENT,EA   (2800)  Running
opcmsgi   OVO Message Interceptor    AGENT,EA   (2801)  Running
opcle     OVO Logfile Encapsulator   AGENT,EA   (2805)  Running
opcmona   OVO Monitor Agent          AGENT,EA   (2806)  Running
opctrapi  OVO SNMP Trap Interceptor  AGENT,EA   (2810)  Running
```

   ps *<OPT>* | grep ovbbccb

   ovbbccb must be listed.

   ***<OvInstallDir>*/bin/bbcutil -status**

   Status of ovbbccb must be ok.

NOTE

Make a note of the ports listed using the command:

bbcutil -getcbport *<hostname>*

- on OVO managed node as *<AGENT-PORT>*
- on OVO management server as *<MGMT-SRV-PORT>*
- on Certificate Authority server as *<CA-SRV-PORT>*

Alternatively, you can use the command:

ovconfget -ns bbc.cb.ports PORT

You can start the Communication Broker with the command:

ovc -start

No error messages should be displayed.

If the ovbbccb process is not running:

a. Check the logfile for error messages in the file:

<OvDataDir>/log/System.txt

b. Start the Communication Broker with the command:

<OvInstallDir>/bin/bbcutil -nodaemon -verbose

If there is any problem, errors are displayed in detail at startup. The port number it uses is also displayed on startup.

c. For more detailed output use the command:

```
OVBBC_TRACE=true <OvInstallDir>/bin/ \
bbcutil -nodaemon -verbose
```

This displays a very significant amount of detailed information. This detail can also be obtained using OV tracing.

2. Check the configuration of the Communication Broker port settings with the following commands:

a. Lists all Communication Broker ports:

bbcutil -getcbport *<hostname>*

b. Check if the default `DOMAIN` parameter is correctly set for the managed nodes using the command:

ovconfget bbc.http DOMAIN

This should be set to the default domain, for example, `myco.com`. This parameter may be used to find a match for the parameters configured in step 2.a above.

c. Check if a process has the Communication Broker port open and is listening for connections using the command:

netstat -an | grep \.383

You should see something similar to (varies on each platform):

```
tcp        0      0  *.383          *.*          LISTEN
```

`LISTEN` verifies that a process is listening on the specified port. If this is displayed and the Communication Broker is not running, another process is using the port and the Communication Broker will not startup. This can be verified with steps 1.a and 1.b.

3. Check the HTTP Communication capabilities by entering the following commands.

On the OVO management server and the Certificate Authority server:

**<*OvInstallDir*>/bin/bbcutil -ovrg server -ping **
http://<OVO managed node>[:<*AGENT-PORT*>]/

On the OVO managed node:

**<*OvInstallDir*>/bin/bbcutil -ping **
http://OVO management server[:<*MGMT-SRV-PORT*>]/

**<*OvInstallDir*>/bin/bbcutil -ping **
http://Certificate Authority server[:<*CA-SRV-PORT*>]/

NOTE If no port is specified in these command, the default port 383 is used.

Each call should report:

```
status=eServiceOK
```

4. Check if the managed nodes have the correct Communication Broker port configuration. Do *not* specify a port number in the URI. OV communication *must* be able to resolve the Communication Broker port number on its own. If the ping works with the port number, but does not work without the port number, the local managed node is not correctly configured. Go back to step 2.
5. Check if the HTTP Proxy is correctly configured using the command:

   bbcutil -gettarget <*nodename*>

   For example, if you enter the command:

   ```
   bbcutil -gettarget mysystem.mycom.com
   ```

   Output of the following form is displayed:

   ```
   Node: mysystem.mycom.com:8008 (14.133.123.10)
   ```

   If a proxy is configured, it will be displayed.

   For example, if you enter the command:

   ```
   bbcutil -gettarget www.mycom.com
   ```

   Output of the following form is displayed:

   ```
   HTTP Proxy: web-proxy:8008 (14.193.1.10)
   ```

   ovconfget bbc.http PROXY

   Although not recommended, applications may set their own private PROXY setting. The above setting is valid for the whole managed node. An individual application may override this value in its own private namespace:

   ovconfget bbc.http.ext.*<comp id>*.*<appname>*

   If the *<comp id>* or *<appname>* is not known, check using `ovconfget` the entire configuration for all proxy settings in the namespaces starting with:

   bbc.http.ext
6. Check on the OVO management server and the Certificate Authority server systems that the proxy is working and supports the `CONNECT` command.

NOTE

The blank lines are important.

On some platforms, it may not be possible to echo commands typed into telnet.

Enter the command:

telnet *<proxy>* *<proxy port>*
CONNECT *<AGENT>*:*<AGENT PORT>* HTTP/1.0

PING /Hewlett-Packard/OpenView/BBC/ HTTP/1.0

To exit telnet, enter **Control-D**

The output should be similar to the following. If the Communication Broker is up and running on the target managed node, the HTTP status should be `200 OK`.

```
HTTP/1.1 200 OK
cache-control: no-cache
content-type: text/html
date: Fri, 06 Feb 2004 15:15:02 GMT
senderid: fd7dc9e4-4626-74ff-084a-9e5a09bffbae
server: BBC 05.00.101; ovbbccb 05.00.101HP OpenView BBC
Information Modules:

    Node:         ping.bbn.hp.com
    Application:  ovbbccb
    Version:      05.00.101
    Modules:      ping
                  status
                  services
                  ovrg

Connection closed by foreign host.
```

7. Check on the OVO managed node that the proxy is working and supports the CONNECT command.

NOTE

The blank lines are required.

On some platforms, it may not be possible to echo commands typed into telnet.

Enter the command:

```
telnet <proxy> <proxy port>
CONNECT <MGMT-SRV>:<MGMT-SRV PORT> HTTP/1.0

PING /Hewlett-Packard/OpenView/BBC/ HTTP/1.0
```

or

```
telnet <proxy> <proxy port>
CONNECT <CA-SRV>:<CA-SRV PORT> HTTP/1.0

PING /Hewlett-Packard/OpenView/BBC/ HTTP/1.0
```

To exit telnet, enter **Control-D**

See the previous point for a sample output.

8. Enable logging for HTTP access to the Communication Broker.

   ```
   ovconfchg -ns bbc.cb -set LOG_SERVER_ACCESS true
   ```

   This will log all access to the Communication Broker. To see the logs use:

   ```
   ovlogdump <OvDataDir>/log/System.txt
   ```

   You can additionally log access to all OV servers using:

   ```
   ovconfchg -ns bbc.http -set LOG_SERVER_ACCESS true
   ```

Authentication and Certificates Troubleshooting for HTTP Communication

Troubleshooting Basic HTTP communication uses the following commands:

ovc	**<*INSTALLDIR*>/bin/ovc**
ovconfget	**<*INSTALLDIR*>/bin/ovconfget**
ovconfchg	**<*INSTALLDIR*>/bin/ovconfchg**
ovcoreid	**<*INSTALLDIR*>/bin/ovcoreid**
ovcert	**<*INSTALLDIR*>/bin/ovcert**
bbcutil	**<*INSTALLDIR*>/bin/bbcutil**

To check for authorization and certificate related HTTP communication problems, complete the following steps:

1. Check the OvCoreID of each system.

   On the OVO management server or the Certificate Authority server, enter the command:

   ovcoreid -ovreg server

   On OVO managed node, enter the command

   ovcoreid

   Make a note of each of the displayed OvCoreID values:

   - *<MGMT-SRV-COREID>*
   - *<CA-SRV-COREID>*
   - *<AGENT-COREID>*

2. Check the certificates on the OVO management server or Certificate Authority server and on OVO managed node using the following command:

   ovcert -list

NOTE

There are 3 certificates on the OVO management server system or Certificate Authority system:

- OVO management server certificate
- Certificate authority certificate
- Managed node certificate

When an OVO management server is installed on a cluster (high availability environment), the certificates of the OVO management server and the agent on the management server are not the same. On non-cluster installations, the certificates must be identical.

On each system there must be at least following Certificates.

On OVO managed node:

```
| Certificates:                                    |
|     <AGENT-COREID>                  (*)          |
```

On the OVO management server or the Certificate Authority server:

```
| Certificates:                                    |
|     <MGMT-SRV-COREID>|<CA-SRV-COREID>    (*)     |
```

On all systems:

```
| Trusted Certificates:                            |
|     <CA-SRV-COREID>                              |
```

NOTE

The `(*)` signifies that the private key for the certificate is available.

If one of the certificates is missing, refer to Chapter 6, “Working with Certificates,” on page 139 and generate the required certificates.

To get more detailed info about the installed certificates, use the following commands:

On OVO managed node:

ovcert -check

On the OVO management server:

ovcert -check -ovrg server

An example of the output is shown below:

```
OvCoreId set                        : OK
Private key installed               : OK
Certificate installed               : OK
Certificate valid                   : OK
Trusted certificates installed      : OK

Check succeeded.
```

To check that the installed certificates are valid, use the following command and make sure that the current date is between the `valid from` and `valid to` dates of the installed certificates:

ovcert -certinfo *<CertificateID>*

NOTE

The CertificateID of a trusted certificates is the OvCoreID of the certificate server prefixed with a `CA_`.

An example of the output is shown below:

```
# ovcert -certinfo 071ba862-3e0d-74ff-0be4-b6e57d0058f2

Type        :  X509Certificate
Subject CN  :  071ba862-3e0d-74ff-0be4-b6e57d0058f2
Subject DN  :  L:  alien2.ext.bbn.com
               O:  Hewlett-Packard
               OU: OpenView
               CN: 071ba862-3e0d-74ff-0be4-b6e57d0058f2

Issuer CN   :  CA_99300c4e-f399-74fd-0b3d-8938de9900e4
Issuer DN   :  L:  tcbbn054.bbn.hp.com
               O:  Hewlett-Packard
               OU: OpenView
               CN: CA_99300c4e-f399-74fd-0b3d-8938de9900e4
Serial no.  :  04
Valid from  :  01/27/04 12:32:48 GMT
Valid to    :  01/22/24 14:32:48 GMT
Hash (SHA1):   60:72:29:E6:B8:11:7B:6B:9C:82:20:5E:AF:DB:D0: ...
```

NOTE

An HTTPS agent is also installed on an OVO management server system.

If calling `ovcert -list` on an OVO management server system, you are given the certificate details of the agent on the OVO management server system as well as the details of the certificate for the management server and the CA.

3. Check the HTTPS communication capabilities using the following commands.

NOTE

The following actions must work even if communication between an OVO management server or a Certificate Authority server and an OVO managed node has to pass:

- Firewalls
- NATs
- HTTP Proxies

If they do not, contact your Network Administrator for more information.

NOTE

If the communication between OVO management server or Certificate Authority server and OVO managed node is not allowed to pass through the firewalls, one or more HTTP Proxies must be used (see the corresponding sections).

On an OVO management server or Certificate Authority server:

```
<OvInstallDir>/bin/bbcutil -ovrg server -ping \
https://<OVO managed node name>[:<AGENT-PORT>]/
```

On an OVO managed node:

```
<OvInstallDir>/bin/bbcutil -ping \
https://<OVO management server name>[:<MGMT-SRV-PORT>]/
```

```
<OvInstallDir>/bin/bbcutil -ping \
https://Certificate Authority server[:<CA-SRV-PORT>]/
```

Each call should report:

```
status=eServiceOK
```

The reported OvCoreID must match with the OvCoreIDs that you noted in the first step:

```
coreID=<COREID>
```

OVO Communication Troubleshooting

Troubleshooting OVO communication uses the following commands:

ovc	**<*INSTALLDIR*>/bin/ovc**
ovconfget	**<*INSTALLDIR*>/bin/ovconfget**
ovconfchg	**<*INSTALLDIR*>/bin/ovconfchg**
ovcoreid	**<*INSTALLDIR*>/bin/ovcoreid**
ovpolicy	**<*INSTALLDIR*>/bin/ovpolicy**
ovcs	**<*INSTALLDIR*>/bin/ovcs**
opcagt	**<*INSTALLDIR*>/bin/OpC/opcagt**
opcragt	**<*INSTALLDIR*>/bin/OpC/opcragt**
opccsa	**<*INSTALLDIR*>/bin/OpC/opccsa**
opcssam	**<*INSTALLDIR*>/bin/OpC/opccsam**
opcsv	**<*INSTALLDIR*>/bin/OpC/opcsv**
opcnode	**<*INSTALLDIR*>/bin/OpC/opcnode**
opc	***/usr*/bin/OpC/opc**

To check for OVO communication problems, complete the following steps:

1. OVO managed nodes must be in the `OVO Node Bank`.
2. The Fully Qualified Domain Name (FQDN) of the OVO managed node must match.
3. The communication type of the OVO managed node must be `HTTPS`.
4. The `OvCoreID` of the OVO managed node must match.

   Check the value of the OVO managed node `OvCoreID` stored in the OVO database using the command:

   opcnode -list_id node_list=*<OVO managed node>*

   It must match the *<AGENT-COREID>*.

   To check, on the managed node call the command:

   ***<OvInstallDir>*/bin/ovcoreid**

You can change the OVO managed node OvCoreID from the OVO management server using the command:

```
opcnode -chg_id node_name=<OVO managed node> \
id=<AGENT-COREID>
```

You can change the OvCoreID on the OVO managed node using the command:

```
ovcoreid -set <NEW-AGENT-COREID>
```

NOTE

Changing the OvCoreId of a system is an operation that must be done with great care because it changes the identity of a managed node. All managed node-related data, such as messages, are linked by the OvCoreId of a managed node. Changing the value of the OvCoreID should only be executed by experienced users who know exactly what they want to do and what is being affected by attempting this change, especially on the OVO management server.

5. Check, that all OVO Management Server processes are running using the commands:

   ```
   opcsv -status
   ```

   All registered processes must be in the state running.

   ```
   ovc -status
   ```

   All registered core processes must be in state running.

6. Make sure that the operator is responsible for the:

   - OVO managed node and its node group
   - Message group

   Reload the Message Browser.

7. Check for pending certificate requests.

   On the Certificate Authority server enter the command:

   ```
   opccsa -list_pending_cr
   ```

   Check if the OVO managed node is listed by nodename, IP address or OvCoreID and whether all parameters are consistent.

Manually grant pending certificate requests with the command:

opccsa -grant *<NODE>*|*<Certificate_Request_ID>*

If the parameter are not consistent, change the values on the OVO management server and OVO managed node, as required.

On the OVO managed node, stop and restart all processes with the commands:

ovc -kill

Verify, that all processes are stopped with the command:

ps *<OPT>* | grep /opt/OV

ovc -start

NOTE

To manually trigger a Certificate Request, first check that there is no certificate already installed with the command:

ovcert -status

If no certificate is installed, enter the command:

ovcert -certreq

The `ovcd` process of the OVO agent must be running for the `ovcert -certreq` call to work. Certificate requests are automatically sent during agent startup, so just the agent startup is sufficient, unless the `CERTIFICATE_DEPLOYMENT_TYPE` is set to `Manual`. This is done with the command:

```
ovconfchg -ns sec.cm.client -set \
CERTIFICATE_DEPLOYMENT_TYPE Manual
```

Therefore, the `ovcert -certreq` command is only of interest if `Manual` certificate deployment type is chosen, or if the certificate was removed while the agent was running. For example, no `ovc -kill` command run before removing the certificate.

If a certificate is already installed, the following error message is displayed:

```
ERROR:   (sec.cm.client-125) There is already a valid
certificate for this node installed.
```

8. If there are no OVO managed node messages in the Message Browser on OVO managed node, execute the following checks:

   - Check if all processes are running:

     ovc -status

     All registered processes must be running and no process should run twice.

   - Check if the expected policies are deployed:

     ovpolicy -list

   - Check the MANAGER, MANAGER_ID, and CERTIFICATE_SERVER settings:

     ovconfget sec.cm.client CERTIFICATE_SERVER

     This must match the Certificate Authority server.

     ovconfget sec.core.auth MANAGER

     This must match the OVO management server.

     ovconfget sec.core.auth MANAGER_ID

     This must match the OvCoreID of the OVO management server.

     To check the OvCoreId of the management server, on the management server enter the command:

     ovcoreid -ovrg server

     ovconfget eaagt OPC_PRIMARY_MGR

     This setting is optional, but when set, it must match the OVO management server.

NOTE

If the OVO management server is not the primary manager, additional checks have to be performed.

The OVO management server must appear with consistent values in the file:

<OvDataDir>/datafiles/policies/mgrconf/*<ID>*_data

- Check the settings of message suppression.
- Check the settings of message buffering.
- Check if the message buffer file is growing:

  ls -l *<OvDataDir>*/tmp/OpC/msgagtdf

  or on OVO management server:

  opcragt -status *<nodename>*

- Send a message to be forwarded to the server:

  opcmsg a=appl o=object msg_t=*<my_text>*

- Check if messages appear in the message manager queue file:

  **strings /var/opt/OV/share/tmp/OpC/mgmt_sv/ **
  msgmgrq | grep *<my_text>*

9. If DEPLOYMENT, ACTIONS or HBP to an OVO managed node fails, on the OVO managed node, check the status of the agent with the command:

   opcragt -status

   If this reports no problems, the problem is not HTTPS communication dependent.

HTTPS Communication and Time Zones

ovbbccb provides increased security on UNIX operating systems by using a feature known as chroot(). A chroot on UNIX operating systems is an operation which changes the root directory. Whenever the ovbbccb process starts up, it is rooted to *<OvDataDir>* in UNIX. This ensures that it can access files only under *<OvDataDir>*. It cannot access any other files.

For time zone conversions, the system files for time zone are needed, which are located in a now inaccessible directory. ovbbccb cannot access the time zone file and writes the date information in UTC(GMT) format rather than actual time zone set for the system.

To establish the correct time zone, create a similar directory structure under *<OvDataDir>* as is available under . . . /zoneinfo/*<TZ>* and copy the actual time zone file:

1. Stop all the OV processes:

   /opt/OV/bin/ovc -kill

2. Check the /etc/TIMEZONE file for the current timezone (TZ value), for example:

   TZ=US/Eastern

3. Create the following directory based on the TZ value.

   mkdir -p *<OvDataDir>*/usr/share/lib/zoneinfo/*<TZ>*

   If the TZ value contains entries separated by a /, as in our example with TZ=US/Eastern, create the directory structure up to the last slash:

   mkdir -p *<OvDataDir>*/usr/share/lib/zoneinfo/US

NOTE

Substitute *<OvDataDir>* with path used by the managed node platform. For details refer to “Generic Directory Structure on OVO Managed Nodes” on page 33.

For example: *<OvDataDir>* on Solaris is: /var/opt/OV

Make sure that the directory structure under *<OvDataDir>* is exactly same as that of /usr/share/lib/zoneinfo/<TZ>.

4. Copy the timezone resource file to the newly created directory:

   ```
   cp /usr/share/lib/zoneinfo/<TZ> \
   <OvDataDir>/usr/share/lib/zoneinfo/<TZ>
   ```

   On HP-UX systems, also copy the following file:

   usr/lib/tztab

5. Start all the OV processes:

   ```
   /opt/OV/bin/ovc -start
   ```

All the messages subsequently logged by `ovbbccb` should have the correct timestamp.

Certificate Deployment Problems

During certificate deployment, the situation may arise that there are two pending certificate requests for the same managed node in the Certificate Server Adapter's list of pending certificate requests.

For example, this can occur if the certificate request is triggered from the managed node. This certificate request is not granted and remains pending in the Certificate Server Adapter's internal list. If you now de-install the agent software and re-install it, another certificate request is triggered. The new request also contains a new `OvCoreID`, because re-installing the managed node generates a new `OvCOreID`. This certificate also remains in the list of pending certificate requests.

The listing of the pending certificate requests also contain a time stamp of when the certificate request was received by the OVO management server. It is clear which certificate request is newer and valid. Grant the newest one and remove any older requests.

Alternatively, there are two further ways of removing unwanted certificate requests:

- Log in as an OVO administrator and remove all certificate requests for a “problematic” managed node and then issue a new certificate request from the managed node with the command:

  `ovcert -certreq`

NOTE

The `ovcd` process of the OVO agent must be running for the `ovcert -certreq` call to work. Certificate requests are automatically sent during agent startup, so just the agent startup is sufficient, unless the `CERTIFICATE_DEPLOYMENT_TYPE` is set to `Manual`.

Therefore, the `ovcert -certreq` command is only of interest if `Manual` certificate deployment type is chosen, or if the certificate was removed while the agent was running. For example, no `ovc -kill` command run before removing the certificate.

This results in a single certificate request for the managed node which can then be mapped and granted in the usual way. See Chapter 5, “Working with HTTPS Managed Nodes,” on page 107.

- If as administrator, you cannot execute the `ovcert -certreq` command on the managed node and so cannot issue a new certificate request, then retrieve the valid `OvCoreID` from the managed node by executing the command:

  `<OvInstallDir>/bin/bbcutil -ovrg server -ping <nodename>`

  List all certificate requests and grant the certificate request that contains valid `OvCoreID` and remove any others.

Change the Management Server Responsible for a Managed Node

It is sometimes necessary to change the management server which manages a managed node. In the following steps, we concentrate on the changes required on the managed node. With DCE agents, you basically just had to change the `OPC_MGMT_SERVER` entry in the `opcinfo` file. With HTTPS agents, it is more complicated and the following topics must be taken into consideration:

1. **Policy Cleanup on the Managed Node**

   If the new server has a different certificate authority than the old one and the new and old server do not have a trust setup, the agent needs a new certificate. This also means that the policies on the agent become unreadable as soon as the agent gets a certificate from the new CA.

   Remove all policies, because they cannot be read anymore, using the command:

   `ovpolicy -remove all`

   If the CAs are the same or a trust exists, then the policies are basically readable, but the `OvCoreId` of the old server, which is contained in the certificates as part of the policy header files, must still be authorized. This is achieved by entering the name of the old management server in the `mgrconf` policy. The following file must exist and the old manager must be mentioned in it:

   <OvDataDir>`/datafiles/policies/mgrconf/*data`

   If this is not the case, enter the command:

   `ovpolicy -remove all`

   An alternative to running the command `ovpolicy -remove all`, you can also remove all files from the following directory:

   <OvDataDir>`/datafiles/policies`

2. **Stop the Agent**

   The agent should be stopped before doing further modifications:

   `ovc -kill`

3. **Certificate Cleanup on the Agent**

   If the new target server shares the same certificate authority as the old one or there is a trust setup between new and old servers, then the certificates can remain as they are. If not, then you must create new certificates.

   Remove the existing ones using the command:

   ovcert -remove *<all_certs_listed_in_ovcert_-list_output>*

4. **Configuration Settings Cleanup**

   Change some basic settings on the agent. The `OvCoreId` can remain unchanged:

   - If the certificate authority has changed, enter the following command to specify the new certificate authority:

     ovconfchg -ns sec.cm.client -set CERTIFICATE_SERVER \ *<new_CA>* (typically the fully qualified hostname)

   - Set the new management server using the following command:

     ovconfchg -ns sec.core.auth -set MANAGER *<new_mgmtsv>* (typically the fully qualified hostname)

   - Obtain the management server `OvCoreId` value with the command:

     ovcoreid -ovrg server

     Set the `OvCoreId` of the new management server on the agent:

     ovconfchg -ns sec.core.auth -set MANAGER_ID \ *<new_manager_core_id>*

   - For MoM environments only, check if the `OPC_PRIMARY_MGR` setting is already set. If it is set, you can clear it or set it to the new management server (both actions have the same effect).

     ovconfchg -ns eaagt -clear OPC_PRIMARY_MGR

5. **Create New Certificates**

   If the old certificates were removed, request new certificates. Restart the agent, and it will make a request for a new certificate (except when the `CERTIFICATE_DEPLOYMENT_TYPE` setting under namespace `sec.cm.client` is set to `Manual`. In this case, perform a manual certificate installation. For further details see “Manual Certificate Deployment with Installation Key” on page 154.

6. **Prepare the Management Server**

   On the new management server proceed in the same way as for adding a new managed node, including granting the certificate request, assigning policies, and deploying configuration. For further details, see “Installing HTTPS Managed Nodes Manually” on page 124.

Certificate Backup and Recovery in OVO

It is extremely important to be aware of the impacts of loosing a private key or when keys and certificate errors arise. The normal configuration upload and download does not include certificate and key data.

There is a utility on the OVO management server to backup and recover certificates plus the associated private keys and OvCoreIds:

`/opt/OV/bin/OpC/opcsvcertbackup/`

This utility has the following options:

- **`-remove`**

  Removes all certificates from an OVO management server, including:

  — Certificate Authority root certificate and its private key.

  — Server certificate and its private key.

  — Managed node certificate on the OVO management server.

  However, a backup is also created automatically before the removal takes place.

- **`-backup`**

  A tar archive is created at the following default address:

  `/tmp/opcsvcertbackup.<date_time>.tar`

  The *<date_time>* format is `YYMMDD_hhmmss`.

  The default storage location can be changed by using the **`-file`** option.

  The information recorded includes:

  — Certificate Authority root certificate, private key and ID

  — OVO management server certificate with key and OvCoreId

  — Managed node certificate with key and `OvCoreId`

  You must secure the data by using the **`-pass`** option with a password.

  The tar archive contains a text file named:

  `opcsvcertbackup.<date_time>.txt`

This information can be useful for archiving and includes OvCoreIds of the backed up certificates, hostname, and time stamp of the backup. This information is not used during a restore.

- **-restore**

  A tar archive as created using the `-backup` option can be restored using this command.

  The filename must be provided with the `-file` option. The password used at backup time must be entered with the `-pass` option.

  The restore cannot work, if any of the certificates or private keys for the Certificate Authority, OVO management server, or managed node already exists on the OVO management server system but are not the same as the corresponding values stored in the backup archive.

  To avoid this, enforce the restore by using the **-force** option. `opcsvcertbackup` also returns with an error when the OvCoreIds of the certificates to be restored do not fit with those stored in the OVO database. When the `-force` option is used, the OvCoreIds are replaced and confirmation is displayed.

When to Backup Certificates

The following are the times when a backup using `opcsvcertbackup` is recommended:

- **Initial OVO Installation**

  After a successful OVO management server installation, it is highly recommended to make a backup of the certificate data with the command:

  opcsvcertbackup -backup

  The resulting tar archive should be stored in a secure place.

- **OVO Management Server Re-installation on Alternative System**

  Perform a standard OVO management server installation on the alternative system. Install the backup from the original OVO management server installation onto the newly installed system with the command:

  opcsvcertbackup -restore -file *<filename>* -pass *<password>* -force

NOTE

The `-force` option must be used because the server installation has automatically created a Certificate Authority, OVO management server, and managed node certificates. These certificates are unsuitable because the managed nodes are configured to use the existing ones from the first installation.

- **Recovery**

  If something is deleted accidentally, use the command:

  opcsvcertbackup -restore -file *<filename>* -pass *<password>*

  Carefully check any error output.

- **Recovery from Configuration Errors**

  If a normal recovery without force option id not successful, check the error messages from the `opcsvcertbackup` call. If this does not help, clean the certificate information stuff with the command:

  opcsvcertbackup -remove

  or directly overwrite the existing certificate configuration with the command:

  opcsvcertbackup -restore -file *<filename>* -pass *<password>* -force

- **Configuring a Certificate Trust for MoM Environments**

  After creating a certificate trust it is recommended that you make a new backup. This ensures that the additional root certificate(s) can be restored in case a recovery is needed.

- **Configuring a Shared Certificate Authority**

  When configuring a shared Certificate Authority, the following command can be useful for removing the unwanted certificates from a second OVO management server installation.

  opcsvcertbackup -remove

  For further details “Environments Hosting Several Certificate Servers” on page 59.

B Tracing OVO

Quick Start to Tracing OVO

To help you investigate the cause of problems, OVO provides problem tracing. Trace logfiles can help you pinpoint when and where problems occur, for example, if processes or programs abort, performance is greatly reduced, or unexpected results appear.

The following tracing mechanisms can be used with OVO:

- OpenView tracing is the mechanism for tracing the latest OpenView products and will be incorporated into all future OpenView products. OpenView tracing can be used to help solve problems with HTTPS agents and the OVO management server.

  OpenView tracing allows remote access using a proprietary format. SSL encryption is not used. By default, the communication port is 5053.

- OVO-style tracing, using configuration settings, can also be used to problem solve HTTPS agents as well as the OVO management server from patch level 8.11. The configuration settings set with the `ovconfchg` command.

- OVO-style tracing must be used to problem solve DCE agents. The `opcinfo` and `opcsvinfo` files are used to specify the trace settings.

  Exceptions:

  OVO-style tracing cannot be used on any process from OpenView shared components. Normally, if a process name begins with `opc`, OVO-style tracing works, if it begins with `ov`, only OpenView tracing can be used.

  The `$OvDataDir/datafiles/xpl/OVTraceCfg.dat` file contains a list of all known OpenView trace areas (second column). Areas with a component prefix of `opc.` or `eaagt.` belongs to OVO and can be traced using OVO-style tracing. All other trace areas are not visible in OVO-style tracing.

  Trace messages from OpenView shared component libraries cannot be traced using OVO-style tracing, even if these shared libraries are used by “`opc`” processes.

OVO-Style Tracing Overview

All opcinfo and opcsvinfo trace settings used in OVO 7 and earlier can also be applied to the OVO 8 and later management servers and to HTTPS agents. However, these are now configuration settings, which are set with the ovconfchg command. DCE agents must be traced using the OVO-style tracing.

Activate OVO-Style Tracing on the Management Server

You can activate the OVO trace facility for the management server processes by entering the following ovconfchg command:

To enable tracing on an OVO Management server, enter the command:

```
ovconfchg -ovrg server -ns opc -set OPC_TRACE TRUE
```

This entry is always required and enables tracing for the areas MSG and ACTN.

Patch level < 8.13: To inform the processes about new configuration settings on the management server, enter the command:

```
/opt/OV/bin/OpC/opcsv -trace
```

It is not necessary to restart any processes. Doing so may also remove the cause of the problem you are investigating.

Activate OVO-Style Tracing on Managed Nodes

You can activate the OVO trace facility for the HTTPS agent processes by entering the following ovconfchg command and, for DCE agents, by modifying the opcinfo file:

HTTPS Agents Enter the command:

```
ovconfchg -ns eaagt -set OPC_TRACE TRUE
```

This entry is always required and enables tracing for the areas MSG and ACTN.

On HTTPS agents, the tracing settings are automatically read by the processes at runtime, as soon a trace configuration setting has changed.

DCE Agents

Open the `opcinfo` file. For the locations of the `opcinfo` files on other supported platforms, see Table B-1:

Add the following entry to the `opcinfo` file and save it:

```
OPC_TRACE TRUE
```

This entry is always required and enables tracing for the areas `MSG` and `ACTN`.

For DCE agents, the tracing settings must be activated with the following command:

```
/opt/OV/bin/OpC/opcagt -trace
```

.

Table B-1 Location of the opcinfo File on OVO DCE Managed Nodes

Platform	opcinfo File
HP-UX 11.x Linux Solaris IBM/ptx Siemens Nixdorf SINIX SGI IRIX	`/opt/OV/bin/OpC/install/opcinfo`
AIX	`/usr/lpp/OV/OpC/install/opcinfo`
MPE/iX	`OPCINFO.BIN.OVOPC`
Novell NetWare	`sys:/opt/OV/bin/OpC/install/opcinfo`
Tru64 UNIX	`/usr/opt/OV/bin/OpC/install/opcinfo`
Windows	`\usr\OV\bin\OpC\install\opcinfo`

NOTE

When changing the attributes of a managed node, the `nodeinfo` file is overwritten by the distribution process. For this reason, you should include the trace statement in the `opcinfo` file and not in the `nodeinfo` file.

De-activate OVO-Style Tracing

To de-activate OVO problem tracing, complete the following steps:

Mgmt Server To disable tracing, enter one of the following commands:

```
ovconfchg -ovrg server -ns opc -clear
OPC_TRACE
```

or

```
ovconfchg -ovrg server -ns opc -set OPC_TRACE
FALSE
```

To inform the processes about new configuration settings on the management server, enter the command:

```
/opt/OV/bin/OpC/opcsv -trace
```

HTTPS Agents Enter the command:

```
ovconfchg -ns eaagt -clear OPC_TRACE
```

or

```
ovconfchg -ns eaagt -set OPC_TRACE FALSE
```

DCE Agents Add the following entry to `opcinfo` file:

```
OPC_TRACE FALSE
```

or remove the `OPC_TRACE TRUE` entry.

```
ovconfchg -ns eaagt -set OPC_TRACE TRUE
```

For DCE agents, the tracing settings must be activated with the following command:

```
/opt/OV/bin/OpC/opcagt -trace
```

Trace Output File Locations

Trace information is written to the `trace.bin` logfile:

- Management Server

  `<OvDataDir>/share/tmp/OpC/mgmt_sv/trace.bin`

  Default: `/var/opt/OV/share/tmp/OpC/mgmt_sv/trace.bin`

- Managed Nodes

  `<OvDataDir>/tmp/OpC/trace.bin`

  HP-UX default: `/var/opt/OV/tmp/OpC/trace.bin`

  For locations of trace files on other supported platforms, refer to Table B-2.

Table B-2 Location of the Trace Output File on OVO DCE Managed Nodes

Platform	opcinfo File
HP-UX 11.x Linux Solaris IBM/ptx Siemens Nixdorf SINIX SGI IRIX Tru64 UNIX	`/var/opt/OV/tmp/OpC/`
AIX	`/var/lpp/OV/tmp/OpC/`
Novell NetWare	`sys:/var/opt/OV/tmp/OpC/`
Windows	`\usr\OV\tmp\OpC\`

Configuring OVO-Style Tracing of the Management Server and Managed Nodes

This reduces the amount of data that is entered into the trace output file and simplifies the interpretation of the trace logfile. You can activate tracing for specific functional areas by specifying one or more functional areas in the trace statement.

Functional Areas

You can select the most suitable functional areas from the following list to more precisely target the area of investigation. Functional areas are set using the `OPC_TRACE_AREA` statement.

NOTE Not all functional areas are available for all processes.

`ACTN`	Actions.
`ALIVE`	Agent-alive check.
`ALL`	All tracing areas (except `DEBUG` and `PERF`).
`API`	Configuration API.
`AUDIG`	Auditing.
`DB`	Database.
`DEBUG`	Debugging information. Use this option carefully, as it provides extensive and detailed information, but the trace logfile will also be correspondingly large.
`DIST`	Distribution.
`GUI`	User interface.
`INIT`	Initialization.
`INST`	Installation.
`INT`	Internal.
`LIC`	Licensing.

MISC	Miscellaneous.
MSG	Message flow.
NAME	Name resolution.
NLS	Native language support.
NTPRF	NTPerfMon.
OCOMM	Open agent communication.
PERF	Performance.
SEC	Security.
SRVC	Service.

Customize Tracing

To configure tracing:

1. Specify **OPC_TRACE TRUE**

   This is always required and enables tracing for the areas MSG and ACTN.

2. To trace a specific functional areas, select the appropriate functional area or management server/agent process by entering statements of the following formats:

   OPC_TRACE_AREA *<area>*[,*<area>*]

   OPC_TRC_PROCS *<process>*[,*<process>*]

   OPC_DBG_PROCS *<process>[,<process>]*

<area>	OVO area to be traced or debugged. By default, MSG and ACTN are enabled. For a list of all available areas, see “Functional Areas” on page 291.
<process>	OVO process to be traced or debugged.

NOTE Spaces are not allowed between entries in the lists for each process or area.

The following examples illustrate how to enable tracing for the message/action flow and initialization and debug. Generate trace output only for `opcmsga` and `opcacta`. Enable debug output only for `opcmsga`.

Example B-1 Management Server Configuration Commands

```
ovconfchg -ovrg server -ns opc -set OPC_TRACE TRUE \
-set OP_TRACE_AREA MSG,ACTN,INIT,DEBUG \
-set OPC_TRC_PROCS opcacta,opcmsga \
-set OPCDBG_PROCS opcmsga
```

Example B-2 HTTPS Managed Node Configuration Commands

```
ovconfchg -ns eaagt -set OPC_TRACE TRUE \
-set OP_TRACE_AREA MSG,ACTN,INIT,DEBUG \
-set OPC_TRC_PROCS opcacta,opcmsga \
-set OPCDBG_PROCS opcmsga
```

Example B-3 Trace Configuration in opcinfo for DCE Managed Nodes

```
OPC_TRACE TRUE
OPC_TRACE_AREA MSG,ACTN,INIT,DEBUG
OPC_TRC_PROCS opcmsga,opcacta
OPC_DBG_PROCS opcmsga
```

If the granularity of the above tracing options is not sufficient, use the variable `OPC_RESTRICT_TO_PROCS` to enable tracing for a particular area of a OVO process.

3. To receive verbose trace information output, enter the following command or add the entry in the `opcinfo` file:

   Mgmt Server `ovconfchg -ovrg server -ns opc -set OPC_TRACE_TRUNC FALSE`

   HTTPS Agents `ovconfchg -ns eaagt -set OPC_TRACE_TRUNC FALSE`

   DCE Agents `OPC_TRACE_TRUNC FALSE`

   By default, `OPC_TRACE_TRUNC TRUE` is enabled.

   For more information on tracing configuration, see “Examples of Tracing” on page 294.

Examples of Tracing

This section contains some examples to show how tracing can be activated for different areas and processes.

Enter the appropriate command or add the entry in the `opcinfo` file:

- **Default**

  Collect trace information for the trace areas `MSG` (message flow) and `ACTN` (actions).

  Mgmt Server
  ```
  ovconfchg -ovrg server -ns opc -set
  OPC_TRACE TRUE
  ```

  HTTPS Agents
  ```
  ovconfchg -ns eaagt -set OPC_TRACE TRUE
  ```

  DCE Agents
  ```
  OPC_TRACE TRUE
  ```

- **Tracing for Heartbeat Polling and Message Flow**

  Collect trace information for the trace area `ALIVE` (agent-alive check).

  Mgmt Server
  ```
  ovconfchg -ovrg server -ns opc -set
  OPC_TRACE TRU -set OPC_TRACE_AREA ALIVE
  ```

  HTTPS Agents
  ```
  ovconfchg -ns eaagt -set OPC_TRACE TRUE
  -set OPC_TRACE_AREA ALIVE
  ```

  DCE Agents
  ```
  OPC_TRACE TRUE
  OPC_TRACE_AREA ALIVE
  ```

- **Tracing for Specific Areas of Specific Processes**

  Collect trace information for the trace area `GUI` (graphical user interface) of the operator GUI processes `opcuiop` and `opcuiopadm`.

  Mgmt Server
  ```
  ovconfchg -ovrg server -ns opc -set
  OPC_TRACE TRU -set OPC_TRACE_AREA GUI -set
  OPC_TRC_PROCS opcuiop,opcuiopadm
  ```

- **Tracing and Debugging**

  - Collect trace information for *all* trace areas (except `PERF`), as well as debug information for all debug areas. Debug areas are to be used by HP Support Personnel only.

Mgmt Server
```
ovconfchg -ovrg server -ns opc -set
OPC_TRACE TRU -set OPC_TRACE_AREA
ALL,DEBUG
```

HTTPS Agents
```
ovconfchg -ns eaagt -set OPC_TRACE TRUE
-set OPC_TRACE_AREA ALL,DEBUG
```

DCE Agents
```
OPC_TRACE TRUE
OPC_TRACE_AREA ALL,DEBUG
```

- Collect trace information for *all* trace areas (except `PERF`) for the process `ovoareqsdr` (request sender), as well as debug information for all debug areas of the process `ovoareqsdr` (request sender).

  Mgmt Server
  ```
  ovconfchg -ovrg server -ns opc -set
  OPC_TRACE TRU -set OPC_TRACE_AREA
  ALL,DEBUG -set OPC_TRC_PROCS ovoareqsdr
  -set OPC_DBG_PROCS ovoareqsdr
  ```

  HTTPS Agents
  ```
  ovconfchg -ns eaagt -set OPC_TRACE TRUE
  -set OPC_TRACE_AREA ALL,DEBUG -set
  OPC_TRC_PROCS ovoareqsdr -set
  OPC_DBG_PROCS ovoareqsdr
  ```

  DCE Agents
  ```
  OPC_TRACE TRUE
  OPC_TRACE_AREA ALL,DEBUG
  OPC_TRC_PROCS ovoareqsdr
  OPC_DBG_PROCS ovoareqsdr
  ```

❑ **Different Trace Areas for Different Processes**

Restricting tracing to a specified process must specify the process in the tracing command or, in the case of the `opcinfo` file entries, must start with the keyword `OPC_RESTRICT_TO_PROCS`, followed by the process (or processes) for which tracing is to be started.

The areas are to be traced are specified as usual.

For `opcinfo` file entries, always add the `OPC_RESTRICT_TO_PROCS` sections at the end of the or `opcinfo` file.

The first configuration entry enables tracing for the trace areas INIT (initialization) and INT (internal) of the control agent process (opcctla). The second configuration entry enables tracing for the trace areas MSG (message flow) and ACTN (actions) of the message agent process (opcmsga).

Mgmt Server

```
ovconfchg -ovrg server -ns opc.opcctla
-set OPC_TRACE TRUE -set OPC_TRACE_AREA
INIT,INT

ovconfchg -ovrg server -ns opc.opcmsga
-set OPC_TRACE TRUE
```

HTTPS Agents

```
ovconfchg -ns eaagt.opcctla -set OPC_TRACE
TRUE -set OPC_TRACE_AREA INIT,INT

ovconfchg -ns eaagt.opcmsga -set OPC_TRACE
TRUE
```

DCE Agents

```
OPC_RESTRICT_TO_PROCS opcctla
OPC_TRACE TRUE
OPC_TRACE_AREA INIT,INT

OPC_RESTRICT_TO_PROCS opcmsga
OPC_TRACE TRUE
```

The example above can also be written as follows:

```
OPC_TRACE TRUE
OPC_TRC_PROCS opcctla,opcmsga
OPC_RESTRICT_TO_PROCS opcctla
OPC_TRACE_AREA INIT,INT
```

Syntax for Trace Files

The general format of the trace information is as follows:

<mm/dd/yy> <hh:mm:ss> <process_name>(pid)[<area>...]: <detailed_ information>

mm/dd/yy Date.

hh:mm:ss Time.

process_name	Process name.
pid	Process ID.
area	Functional area(s) as specified in the trace statement.
detailed_information	Detailed information about the process.

NOTE New trace information is appended to existing trace logfiles. For this reason, you should delete the file to prevent it from becoming too large.

OpenView-Style Tracing Overview

OpenView Tracing implements a hierarchy of elements: `Applications`, `Components`, `Categories` and `Attributes`. By specifying a combination of these in the trace GUI or in a trace configuration file, the area of interest can be traced.

Table B-3 illustrates how these elements relate to OVO components, processes, and areas.

Table B-3 OpenView Tracing Terminology

OpenView Name	OVO-Style Name	Examples
Application	Process, `OPC_TRC_PROCS` and `OPC_DBG_PROCS`	`opcmsga`, `opcmsgrd`, `ovpolicy`
Component	n.a.	`opc`, `eaagt`
Subcomponent	Trace Areas, `OPC_TRACE_AREA`	`actn`, `msg`, `init`, `debug`
Category	`OPC_TRACE TRUE`	`Trace`
Attribute	n.a.	`Info`, `Warn`, `Error`, `Developer`, `Verbose`

There are two ways to trace OVO using OpenView tracing:

- Configure Remote Tracing using the Windows Tracing GUI.
- Configure Manual Tracing Using Trace Configuration Files.

These methods are described in the next sections.

Configure Remote Tracing using the Windows Tracing GUI

The tracing GUI, available after installing the OVO agent on a Windows system, helps to simplify tracing configuration. It can be used to connect to the remote trace server to identify the application, component, and category names and to view the attributes. It requires that port 5053 is opened in firewalls between the system where the GUI is running and the system where the trace output is generated. Using the features provided within the Tracing GUI, the required configuration setting can be selected and a configuration file saved.

To configure OpenView tracing on OVO processes with the Tracing GUI:

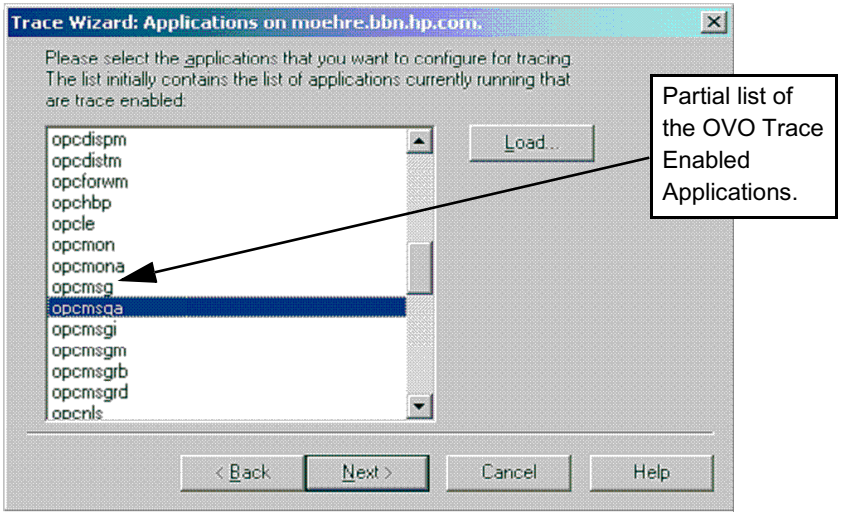
1. Identify the OVO processes that you want to trace. The following example uses the `opcmsga` and `opcmsgm` processes.
2. Start the Tracing GUI on a Windows system. In a Windows Explorer window, go to the directory:

   `<OvInstallDir>\support\`

   Default location: `c:\Program Files\HP OpenView\support\`
3. Start the GUI by double-clicking the file:

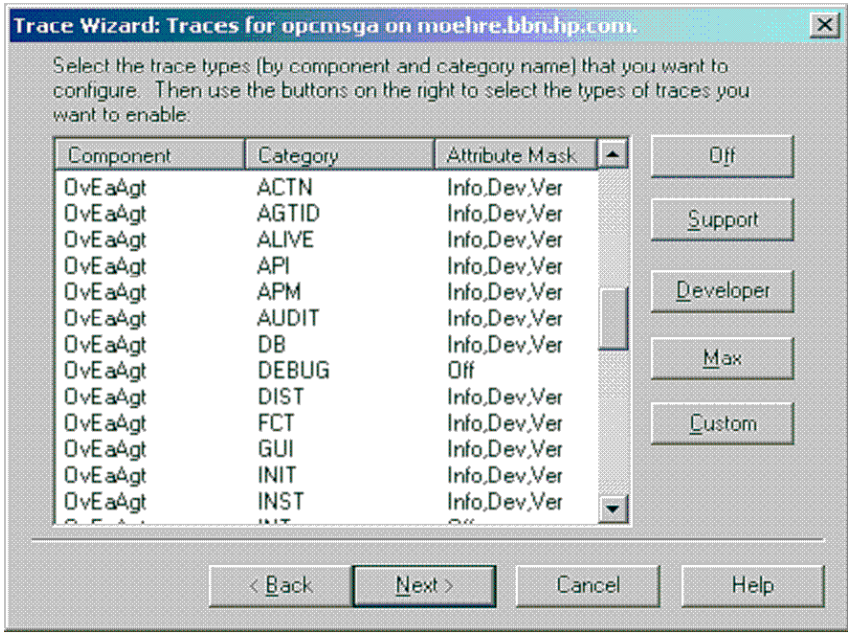
   `ovtrcgui`

Figure B-1 TraceMon Applications Dialog for OVO Applications



4. Select the `opcmsga` and `opcmsgm` applications.
5. Set all `opc` and `OvEaAgt` sub-components, except for the `DEBUG` to `Support`. This sets tracing attributes to the `Support` defaults of `Info`, `Warn`, and `Error` for all sub-components, with the `Verbose` attribute added to each component/sub-component combination entry.

Figure B-2 TraceMon Trace Dialog for OVO Applications



After you have selected the required configuration settings, save the configuration file.

Configure Manual Tracing Using Trace Configuration Files

In many cases, in particular on UNIX systems, the simplest way is to manually create the trace configuration files specifying the components to be traced and log the trace output into a file. Three management server and three agent example trace configuration files are provided at the following location on the management server system:

`/opt/OV/contrib/OpC/TraceConfig`

You must copy the appropriate file to the managed node system, if you want to use it to trace an agent.

NOTE

You can also use the Tracing GUI to create a trace configuration file on a Windows system and then copy this to the system where you want to investigate a problem.

These files include trace configuration statements for all OVO processes. refer to the lines beginning with `APP:`. If you want to trace specific processes, create a new trace configuration file and copy and paste the appropriate pieces from the example files and add the header line - the first line, beginning with `TCF`.

OpenView Tracing implements a hierarchy of elements starting with `Applications`, `Components`, `Categories` and `Attributes`. In OpenView Tracing terminology, the processes defined by `OPC_TRC_PROCS` and `OPC_DBG_PROCS` are referred to as `Applications`. The `TRACE AREAS` defined by the `OPC_TRACE_AREA` parameter are referred to as `subcomponents`. `Component` and `Attribute` elements were not part of the tracing configuration for OVO versions before OVO 8.

Component = *<component name>*

Trace area = *<sub-component>*

Category = `Trace`

To configure the same type of trace configuration using OpenView Tracing, you create a Trace Configuration File (See Example B-4), enable tracing using the `ovtrccfg` tool, and monitor the trace messages using the `ovtrcmon` tool.

Example B-4 OpenView Trace Configuration File

```
TCF Version 3.2
APP: "opcmsga"
SINK: Socket "prodnode" "node=10.1.221.22;"
TRACE: "eaagt.actn" "Trace" Info Warn Error Developer
Verbose
TRACE: "eaagt.debug" "Trace" Info Warn Error Developer
Verbose
TRACE: "eaagt.init" "Trace" Info Warn Error Developer
Verbose
TRACE: "eaagt.msg" "Trace" Info Warn Error Developer Verbose
APP: "opcacta"
SINK: Socket "prodnode" "node=10.1.221.22;"
TRACE: "eaagt.actn" "Trace" Info Warn Error Developer
Verbose
TRACE: "eaagt.init" "Trace" Info Warn Error Developer
Verbose
TRACE: "eaagt.msg" "Trace" Info Warn Error Developer Verbose
```

The following table lists the example OpenView trace configuration files and how these files map to the OVO 7 tracing configurations.

OpenView Trace Configuration Files	OVO 7 Tracing Configurations
*Default.tcf	OPC_TRACE TRUE
*All.tcf	OPC_TRACE TRUE, OPC_TRACE_AREA ALL
*AllDebug.tcf	OPC_TRACE TRUE, OPC_TRACE_AREA ALL, DEBUG

Activating Tracing

To activate tracing into a local file, complete the following steps:

/opt/OV/support/ovtrcadm -a localhost

**/opt/OV/support/ovtrccfg -server localhost **
<my_trace_config_file>

For example:

```
ovtrccfg -server localhost \
/opt/OV/contrib/OpC/TraceConfig/ServerAll.tcf
```

Viewing Trace Results

To view the trace output you need to use the formatting tool `ovtrcmon`:

/opt/OV/support/ovtrcmon -fromfile <binary_output> [-tofile <ascii-output>]

You can specify output formats. Details are available from the `ovtrcmon` usage text:

```
/opt/OV/support/ovtrcmon -help
```

An alternative way to capture trace output, assuming you want to use one of the pre-configured trace configuration files from the directory on the management server:

`/opt/OV/contrib/OpC/TraceConfig/*.tcf`:

is as follows:

1. In your trace configuration file (file extension `.tcf`), replace the lines beginning with `SINK: File` with the string:

   SINK: Socket "localhost" "node=localhost;"

2. Load the trace configuration file using the command:

   /opt/OV/support/ovtrccfg *<my_trace_config_file>*

3. Start `ovtrcmon` to dump the output into a file:

   /opt/OV/ovtrcmon -server localhost >
   <my_ascii_trace_output_file>

   See `ovtrcmon` usage message for output formatting options.

Disable Remote Tracing (No Ports Opened)

The ovtrcd process, by default, opens port 5053 for external access. You can switch off the opening of this externally visible port using one of the following methods:

- **On a Managed Node**

  1. Disable remote tracing using the following command:

     ovtrcadm -disableremotetracing

  2. Restart the OpenView trace daemon (ovtrcd) process using the following command:

     /opt/OV/support/ovtrcadm -srvshutdown

     /opt/OV/lbin/xpl/trc/ovtrcd

     Or use the OVTrcSrv boot script located in the platform-dependent boot directory, for example, on Solaris:

     /etc/init.d/OVTrcSrv

  3. After ovtrcd restarts, localhost:5053 is still used, but on local loopback only. Restart the applications that you want to trace. For example the OVO agent.

- **On a Management Server**

  The bbc_inst_defaults file on the management server contains the configuration setting:

  eaagt:DISABLE_REMOTE_TRACE_AT_INSTALL

  If this setting is set to TRUE, all appropriate newly installed agents automatically execute the following steps, as required by the above method, before they are started.

  ovtrcadm -disableremotetracing

  /opt/OV/support/ovtrcadm -srvshutdown

  /opt/OV/lbin/xpl/trc/ovtrcd

  For more information on how to configure the bbc_inst_defaults file, refer to the section “Changing the Default Port” on page 88 or the example file at the following location:

  /etc/opt/OV/share/conf/OpC/mgmt_sv/bbc_inst_defaults.sampl

Switch Off Tracing

To switch of tracing, enter the command:

```
/opt/OV/support/ovtrccfg off
```

NOTE

For systems with OVO agent patch level prior to 8.14, it is possible that you experience problems when you stop the OpenView trace daemon (`ovtrcd`) process.

To recover from this type of problem, do the following:

1. If `ovtrcd` was not correctly shutdown but stopped using the `kill` command, remove some temporary files using the following commands:

   ```
   rm /var/opt/OV/tmp/ovtrc.server.lock
   ```

   ```
   rm /var/opt/OV/tmp/hp.trc.sem.TraceCfg*
   ```

2. If `ovtrcd` was correctly shutdown and restarted, or you have completed the clean up from the previous step, restart the applications that you want to trace. For example the OVO agent.

An Example of Tracing OVO Processes

The following sample procedure provides an example of how to setup OpenView tracing on OVO processes. The example makes the following configuration assumptions:

- The `opcmsga` and `opcmsgm` processes running on a UNIX system must be traced.
- The `ovtrccfg` trace configuration client will be used to make configuration changes.
- The trace configuration file must be named: `$OV_CONF/OVOTrace.tcf`
- The `ovtrcmon` trace monitor client will be used to monitor the traces.
- The trace output must be written to a file named: `$OV_LOG/OVOTrace.trc`

To setup OpenView tracing on OVO processes:

1. Identify the OVO processes that you want to trace. (The following example uses the `opcmsga` and `opcmsgm` processes).
2. Create a trace configuration file named `OvoTrace.tcf`. Locate the file in the `$OV_CONF` directory.

   This sample trace configuration file (See Example B-5) enables tracing on the two OVO applications, `opcmsga` and `opcmsgm`. The Sink is configured as a socket with the machine `supnode1` as the target server. The components selected are the `opc` and `eaagt`. All the associated sub-components are selected except for the `DEBUG` sub-components. This would correspond to selecting `All Areas except DEBUG`. The tracing attributes are set to the `Support` defaults of `Info`, `Warn`, and `Error` for all, with the `Verbose` attribute added to each component/sub-component combination entry.

Example B-5 Trace Configuration File $OV_CONF/OVOTrace.tcf

```
TCF Version 3.2
APP: "opcmsgm"
SINK: Socket "supnode1" "node=10.111.1.21;"
TRACE: "opc.actn" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.agtid" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.alive" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.api" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.audit" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.db" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.dist" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.fct" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.gui" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.init" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.inst" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.int" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.lic" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.mem" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.memerr" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.misc" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.mon" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.msg" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.name" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.nls" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.ntprf" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.ocomm" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.pdh" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.perf" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.pstate" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.sec" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.srvc" "Trace" Info Warn Error Developer Verbose
TRACE: "opc.wmi" "Trace" Info Warn Error Developer Verbose
APP: "opcmsga"
SINK: Socket "supnode1" "node=10.111.1.21;"
TRACE: "eaagt.actn" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.agtid" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.alive" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.api" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.audit" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.db" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.dist" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.fct" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.gui" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.init" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.inst" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.int" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.lic" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.mem" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.memerr" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.misc" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.mon" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.msg" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.name" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.nls" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.ntprf" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.ocomm" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.pdh" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.perf" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.pstate" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.sec" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.srvc" "Trace" Info Warn Error Developer Verbose
TRACE: "eaagt.wmi" "Trace" Info Warn Error Developer Verbose
```

If you have access to a Windows system with the `TraceMon` tool installed, it can be used to connect to the remote trace server to identify the application, component, and category names and to view the attributes. Refer to Figure B-3 and Figure B-4 for screen shots of associated dialogs from TraceMon GUI. Using the features provided within the TraceMon GUI tool, the required configuration setting can be selected and the configuration file saved.

Figure B-3 TraceMon Applications Dialog for OVO Applications

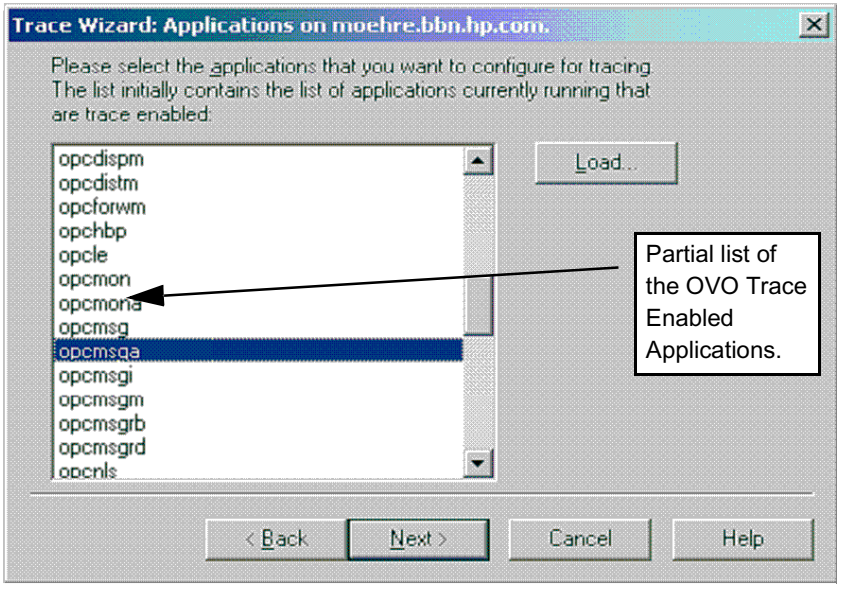
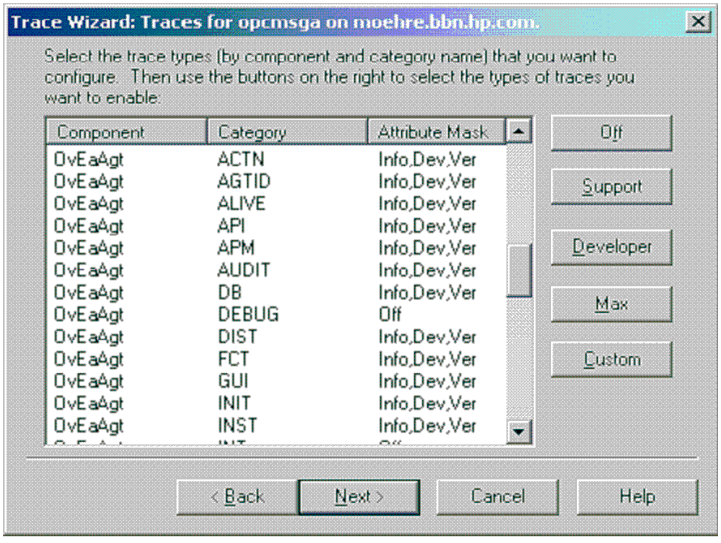


Figure B-4 TraceMon Trace Dialog for OVO Applications



3. Verify that the trace server is running on the system by executing the command:

   `ps -ef | grep ovtrcd`

   If the process is running, the information returned should be of the following form:

   ```
   root@ supnode1: ps -ef | grep ovtrcd
   root 18750  1  0  Mar  5  ?0:00 /opt/OV/bin/ovtrcd
   ```

4. Verify that the applications being traced, `opcmsga` and `opcmsgm`, are running on the system.

   To verify a process is running, execute commands of the following form:

   `ovstatus -c opcmsga opcmsgm`

   The information returned should be of the following form:

   ```
   root@ supnode1: ovstatus -c opcmsga opcmsgm
   Name PID State Last Message(s)
   opcmsga 15422  RUNNING  Initialization complete.
   opcmsgm 26605  RUNNING  OVO Server Initialization
   Complete.
   ```

5. Use the `ovtrccfg` configuration client to set the tracing configuration, using the command:

   ```
   $OV_BIN/ovtrccfg -server supnode1 $OV_CONF/OvoTrace.tcf
   ```

6. Use the `ovtrcmon` monitor client to monitor the trace messages generated from the `opcmsga` and `opcmsgm` applications. To monitor the trace server running on the `supnode1` system and output the trace messages in binary format to the `$OV_LOG/OvoTrace.trc file`, enter the command:

   ```
   $OV_BIN/ovtrcmon -server supnode1 -tofile
   $OV_LOG/OvoTrace.trc
   ```

7. Provided that the processes to be traced are running (`opcmsga` and `opcmsgm` in our example), they should now be generating trace messages. Once enough trace information has been captured, stop the tracing. To Stop tracing, enter the command:

   ```
   $OV_BIN/ovtrccfg off
   ```

8. View the trace output using the `ovtrcmon` monitor client. The trace output can be read from the binary trace file created using the `ovtrcmon -fromfile` option. This option reads in a binary trace file and converts it to text. The converted trace messages can be sent directly to standard out or can be redirected to trace text file.

   To convert the binary trace file to text and send the output to standard out, enter the following command:

   ```
   $OV_BIN/ovtrcmon -fromfile $OV_LOG/OvoTrace.trc
   ```

   To redirect the converted trace messages to a text file, enter the following command:

   ```
   $OV_BIN/ovtrcmon -fromfile $OV_LOG/OvoTrace.trc \
   > /tmp/trc.text
   ```

   The binary `$OV_LOG/OvoTrace.trc` can be viewed from within the TraceMon Windows tool, where additional filtering can be done.

9. If analysis of the trace output is inconclusive, additional tracing can be done to capture more trace information. If needed, the trace configuration file can be modified to include or remove applications, components, categories or attributes.

OVO Trace-Enabled Applications

All OVO 8.x processes use OpenView Tracing. The OVO Trace Enabled processes can be divided into three groups:

- The Server processes
- The Agent processes
- The processes that link with a lower level component which implemented XPL Tracing.

There are no pre-configuration steps required to enable tracing within OVO 8.x. This is accomplished by either adding XPL Tracing into the OVO code base or by incorporating core functionality from a foundation component and linking with the corresponding library. In the case where XPL Tracing was added to the OVO code base, the existing tracing was converted to XPL Tracing. In cases where functionality from a foundation component was added, the XPL Tracing incorporated into these foundation components is pulled into OVO.

Table B-4 OVO Trace-enabled Applications on Management Server and Managed Nodes

Platform	Application Name		
UNIX	coda	ovas	ovconfget
	codautil	ovbbccb	ovcoreid
	ctrlconfupd	ovc	ovcreg
	logdump	ovcd	ovcs
	opc_getmsg	ovcert	ovdeploy
	opc_ip_addr	ovcm	ovpolicy
	opccrpt	ovconfchg	
	opcnls	ovconfd	

Table B-5 OVO Trace-enabled Applications on Management Server

Platform	Application Name		
UNIX	opc	opcdbck	opcragt
	opc_dbinit	opcdbinst	opcservice
	opc_dflt_lang	opcdbmsgmv	opcsvcm
	opc_rexec	opcdbpwd	opcsvreg
	opcactm	opcdispm	opcsw
	opcagtdbcfg	opcdistm	opcttnsm
	opcagtutil	opcforwm	opcuiadm
	opcauddwn	opchbp	opcuiopadm
	opcbbcdist	opchistdwn	opcuiwww
	opccfgupld	opcmsgm	ovoareqhdlr
	opccsacm	opcmsgrb	ovoareqsdr
	opccsad	opcmsgrd	
	opcctlm	opcnode	

Table B-6 OVO Trace-enabled Applications on Managed Nodes

Platform	Application Name		
UNIX	opcacta	opcmon	opcmsgi
	opceca	opcmona	opctrapi
	opcecaas	opcmsg	
	opcle	opcmsga	

Server and Agent Applications

OVO Specific and OpenView Components

There are many components and sub-components defined for each application. The most important are `eaagt` and `opc`. Table B-7 lists the OpenView Tracing Components which are defined for the Server and Agent processes.

Table B-7 OVO Server and Agent Components

OVO Component Name	Component Description
`eaagt`	Event Action Agent
`opc`	Management Server Control

Table B-8 lists the components defined for the shared components which have been incorporated into the product.

Table B-8 OV Shared Components

Application with Component and Subcomponent Names	
Black Box Communication	
bbc.cb	bbc.http.output
bbc.fx	bbc.http.server
bbc.fx.client	bbc.messenger
bbc.fx.server	bbc.rpc
bbc.http	bbc.rpc.server
bbc.http.client	bbc.soap
bbc.http.dispatcher	
Control Component	

Table B-8 OV Shared Components (Continued)

Application with Component and Subcomponent Names	
ctrl.action	ctrl.ovc
ctrl.autoshutdown	ctrl.process
ctrl.component	ctrl.rpcclient
ctrl.controller	ctrl.rpcserver
ctrl.main	ctrl.soap
ctrl.monitor	ctrl.xml
ctrl.monitorproxy	
Configuration Management Component	
conf.cluster	conf.ovconfd
conf.cluster.clioutputs	conf.ovpolicy
conf.config	conf.policy
conf.message	
Certificate Server Adapter	
CSA-CertRequestImpl	Csa-Main
CSA-CertReqContainer	csa.ovcmwrap
CSA-Database	Csa-RpcServer
Csa-Log	CSA-UpdateHandler
Security Core Component	
sec.cm.client	sec.core.base
sec.cm.server	sec.core.ssl
sec.core.auth	
Cross Platform Library	

Table B-8 OV Shared Components (Continued)

Application with Component and Subcomponent Names	
xpl.cfgfile	xpl.net
xpl.config	xpl.runtime
xpl.io	xpl.thread
xpl.log	xpl.thread.mutex
xpl.msg	
Embedded Performance Agent	
coda	coda.mesa
coda.dataaccess	coda.mesainstances
coda.kmdatamatrix	coda_mesametricrdr
coda.localmesa	coda.mesarea
coda.logger	coda.prospector
Deployment Component	depl

OVO Specific and XPL Standard Categories

OVO trace areas are designated by OpenView categories. In addition, a number of the OpenView standard categories are used by both OVO processes and the lower level OpenView components used by OVO.

Table B-9 lists the OpenView tracing categories which are defined for the `eaagt` and `opc` components.

NOTE These categories are referred to as `areas` in version of OVO before OVO 8.

Table B-9 OVO opc and eaagt Sub-components

Sub-Component Name	Sub-Component Description
OVO Specific Tracing Categories	
actn	Actions
agtid	IP independence using AgentID
alive	Agent alive checking
api	Configuration API
apm	Cluster APM
audit	Auditing
db	Database (dblib)
debug	Debug
dist	Distribution
fct	Function (control flow)
gui	Motif Userinterface
init	Initialization (e.g. err init, conf init)
inst	Installation
int	Internal
lic	Licensing

Table B-9 OVO opc and eaagt Sub-components (Continued)

Sub-Component Name	Sub-Component Description
memerr	Problems with Memory allocation
memory	Rest of memory allocation
misc	Miscellaneous
mon	Monitor
msg	Message flow
name	Name resolution
nls	National Language Support (character set conversion,...)
ntprf	NT Performance trace
ocomm	Openagent communication
pdh	Performance data helper
perf	Performance
pstate	Policy and Source state changes
sec	Security
srvc	Service
wmi	Conversion of LE-Templates to WMI-Templates
Generic XPL Tracing Categories	
Trace	Generic traces
Proc	Procedure traces
Operation	Operational traces
Init	Initialization
Cleanup	Cleanup operation
Event	Event
Parms	Parameters
ResMgmt	Resource Management

NNM Pre-Configuration Requirements

There are no pre-configuration steps required to enable OpenView Tracing in OVO 8.x.

If NNM/ET is installed, some of the NNM processes require a pre-configuration step. The required steps are summarized below:

- The NNM/ET applications, these have names starting with: `ovet_`, require their associated `lrf` file be modified to include the hidden `-debug 4` option to enable tracing.
- The ECS Correlation Composer applications require the ECS and PMD tracing be configured to enable OpenView tracing.

C Configuring HTTPS-based Communication

Communication Configuration Parameters

HP OpenView applications may be customized for an installation using configuration parameters. The communication broker configuration parameters are contained in the `bbc.ini` file located at the following address:

`<OVDataDir>/conf/confpar/bbc.ini`

The parameters used for communication are described in "HTTPS Communication Configuration File" on page 322.

The Communication Broker uses the namespace `bbc.cb`. An additional namespace, `bbc.cb.ports`, has been defined to specify the Communication Broker port number for all managed nodes. This enables different Communication Brokers to have different port numbers. This configuration takes precedence over the `SERVER_PORT` parameter defined in the namespace `bbc.cb`.

NOTE

A namespace is a unique URL (Uniform Resource Locator).

For example:

`www.anyco.com` or `abc.xyz`

Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URL references.

The name/value pairs in the `bbc.cb.ports` namespace define the port numbers for the Communication Brokers within the network. The syntax of the name/value pairs is:

`NAME=<host>:<port>` or `NAME=<domain>:<port>`

Multiple host/port or domain/port combinations may be defined per line. Each is separated by a comma or semicolon.

A domain takes the form `*.domainname`. All entries for this domain will use the specified port. More specific entries take precedence. The name of the name/value pair is ignored, although the names must be unique within this namespace. The following are entry examples:

- `HP=jago.sales.hp.com:1383, *.sales.hp.com:1384; *.hp.com:1385`
- `SUN= *.sun.com:1500`

In this example the Communication Broker running on the host `jago.sales.hp.com` will have the port number `1383`.

All other hosts within the domain `sales.hp.com` use the port number 1384. All other hosts within the domain `hp.com` use the port number 1385. Hosts in the domain `sun.com` use the port number 1500. All other hosts use the default port number 383.

HTTPS Communication Configuration File

bbc.ini(4)

NAME

bbc.ini – Configuration file for HTTPS communication.

DESCRIPTION

`bbc.ini` is the configuration file of an OVO managed node using HTTPS communication and is located at:

`/<OVDataDir>/conf/confpar`

It consists of sections headed by namespaces which contain the settings for each namespace. The `bbc.ini` file contains the namespaces listed below. Possible and default settings are described for each namespace.

bbc.cb

The Communication-Broker Namespace. You can use the following parameters:

`string CHROOT_PATH = <path>`

On UNIX systems only, the `chroot` path is used by the `ovbbccb` process. If this parameter is set, the `ovbbccb` process uses this path as the effective root thus restricting access to a limited part of the file system. Default is `<OvDataDir>`. This parameter is ignored on MS Windows and Sun Solaris 7 systems. See the `chroot` man page for details on `chroot`.

`bool SSL_REQUIRED = false`

If this parameter is set to `true`, the communication broker requires SSL authentication for all administration connections to the communication broker. If this parameter is set to `false`, non-SSL connections are allowed to the communication broker.

`bool LOCAL_CONTROL_ONLY = false`

If this parameter is set to `true`, the communication broker only allows local connections to execute administrative commands such as `start` and `stop`.

`bool LOG_SERVER_ACCESS = false`

If this parameter is set to `true`, every access to the server is logged providing information about the sender's IP address, requested HTTP address, requested HTTP method, and response status.

`int SERVER_PORT = 383`

By default this port is set to `383`. This is the port used by the communication broker to listen for requests. If a port is set in the namespace `[bbc.cb.ports]`, it takes precedence over this parameter.

`string SERVER_BIND_ADDR = <address>`

Bind address for the server port. Default is `INADDR_ANY`.

bbc.cb.ports

The Communication-Broker-Port Namespace. This parameter defines the list of ports for all Communications Brokers in the network that may be contacted by applications on this host. The default port number for all communication brokers is 383. You can use the following parameters:

`string PORTS`

This configuration parameter must be the same on all managed nodes. To change the port number of a communication broker on a particular host, the hostname must be added to this parameter, e.g. `name.hp.com:8000`. You can use an asterisk "`*`" as a wild card to denote an entire network, e.g.; `*.hp.com:8001`. Note too, that either a comma "," or a semi-colon ";" should be used to separate entries in a list of hostnames, for example;

`name.hp.com:8000, *.hp.com:8001`.

In these examples, all hostnames ending in "hp.com" will configure their BBC Communication Broker to use port 8001 except host "name" which will use port 8000. All other hosts use the default port 383.

You can also use IP addresses and the asterisk wild card (*) to specify hosts. For example;

`15.0.0.1:8002, 15.*.*.*:8003`

bbc.http

The HTTP Namespace for managed node-specific configuration. For application-specific settings, see the section bbc.http.ext.*. Note that application-specific settings in `bbc.http.ext.*` override managed node-specific settings in `bbc.http`. You can use the following parameters:

`int SERVER_PORT = 0`

By default this port is set to `0`. If set to `0`, the operating system assigns the first available port number. This is the port used by the application *<appName>* to listen for requests. Note that it only really makes sense to explicitly set this parameter in the `bbc.http.ext.`*<appName>* namespace, as the parameter is application specific with any other value than the default value.

`string SERVER_BIND_ADDR = `*<address>*

Bind address for the server port. Default is `localhost`.

`string CLIENT_PORT` = 0

Bind port for client requests. This may also be a range of ports, for example 10000-10020. This is the bind port on the originating side of a request. Default is port `0`. The operating system will assign the first available port.

Note that MS Windows systems do not immediately release ports for reuse. Therefore on MS Windows systems, this parameter should be a large range.

`string CLIENT_BIND_ADDR = `*<address>*

Bind address for the client port. Default is `INADDR_ANY`.

`bool LOG_SERVER_ACCESS = false`

If this parameter is set to `true`, every access to the server is logged providing information about the sender's IP address, requested HTTP address, requested HTTP method, and response status.

`string PROXY`

Defines which proxy and port to use for a specified hostname.

Format:
`proxy:port +(a)-(b);proxy2:port2+(a)-(b); ...;`

a: list of hostnames separated by a comma or a semicolon, for which this proxy shall be used.

b: list of hostnames separated by a comma or a semicolon, for which the proxy shall *not* be used.

The first matching proxy is chosen.

It is also possible to use IP addresses instead of hostnames so `15.*.*.*` or `15:*:*:*:*:*:*:*` would be valid as well, but the correct number of dots or colons MUST be specified. IP version 6 support is not currently available but will be available in the future.

bbc.fx

BBC File-Transfer Namespace for managed node-specific configuration. For application-specific settings, see the section bbc.fx.ext.*. Note that application-specific settings in `bbc.fx.ext.*` override managed node-specific settings in `bbc.fx`. You can use the following parameters:

`int FX_MAX_RETRIES = 3`

Maximum number of retries to be attempted for the successful transfer of the object.

`string FX_BASE_DIRECTORY = <directory path>`

Base directory for which files may be uploaded or downloaded. Default directory is <*OvDataDir*>.

`string FX_TEMP_DIRECTORY = <directory path>`

Temporary directory where uploaded files are placed while upload is in progress. At completion of upload, the file will be moved to *<directory path>*. Default directory is *<OvDataDir>*/tmp/bbc/fx.

`string FX_UPLOAD_DIRECTORY = <directory path>`

Target directory for uploaded files. By default this is the base directory. The upload target directory may be overridden with this configuration parameter. Default directory is `FX_BASE_DIRECTORY`.

bbc.snf

BBC Store-and-Forward Namespace for managed node-specific configuration. For application-specific settings, see the section bbc.snf.ext.*. Note that application-specific settings in `bbc.snf.ext.*` override managed node-specific settings in `bbc.snf`. You can use the following parameters:

`string BUFFER_PATH = <path>`

Specifies the SNF path were the buffered requests are stored. Default is:

<OVDataDir>/datafiles/bbc/snf/*<appName>*

`int MAX_FILE_BUFFER_SIZE = 0`

Specifies the maximum amount of disk space that the buffer is allowed to consume on the hard disk.

`0` = No limit

bbc.http.ext.*

HTTP External-Communication Namespaces: `bbc.http.ext.`*<compID>*`.`*<appName>* and `bbc.http.ext.`*<appName>*.

This is the Dynamic External-Communication Namespace for application-specific settings. Note that application-specific settings in `bbc.http.ext.*` override managed node-specific settings in `bbc.http`.

See the section `bbc.http` for a list of the parameters you can use in the `bbc.http.ext.*` namespace.

bbc.fx.ext.*

The Dynamic File-Transfer (fx) Namespace for external-component and application-specific settings. Note that application-specific settings in `bbc.fx.ext.*` override managed node-specific settings in `bbc.fx`.

File Transfer External Namespaces:
`bbc.fx.ext.<compID>.<appName>` and `bbc.fx.ext.<appName>`.

See the section `bbc.fx` for a list of the parameters you can use in the `bbc.fx.ext.*` namespace.

bbc.snf.ext.*

The Dynamic Store-and-Forward (snf) Namespace for external-component and application-specific settings. Note that application-specific settings in `bbc.snf.ext.*` override managed node-specific settings in `bbc.snf`.

Store and Forward External Namespace:
`bbc.snf.ext.<compID>.<appName>` and `bbc.snf.ext.<appName>`.

See the section `bbc.snf` for a list of the parameters you can use in the `bbc.snf.ext.*` namespace.

AUTHOR

`bbc.ini` was developed by Hewlett-Packard Company.

EXAMPLES

```
PROXY=web-proxy:8088-(*.hp.com)+(*.a.hp.com;*)
```

The proxy `web-proxy` is used with port `8088` for every server `(*)` except hosts that match `*.hp.com`, for example `www.hp.com`. If the hostname matches `*.a.hp.com`, for example, `merlin.a.hp.com` the proxy server will be used.

SEE ALSO

ovbbccb (1)

D HTTPS Communication Architecture

Communication (Broker) Architecture

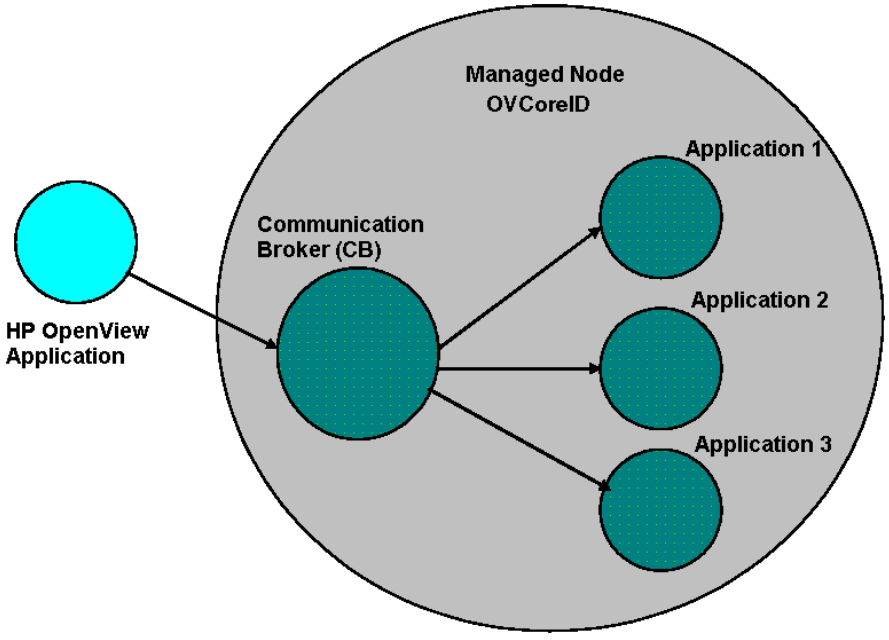
The Communication Broker acts as a proxy on the local managed node and provides a central point of entry to the managed node for all applications on that managed node. Applications that want to receive data register an address with the Communication Broker. The registration defines the port number, protocol, bind address, and base path the application wants to receive data on. Other applications, local or remote, either query the Communication Broker for the location of the application or use the Communication Broker as a proxy to forward the request to registered applications. The Communication Broker loads configuration data from the standard OpenView Configuration File.

The Communication Broker has the following characteristics:

- The Communication Broker provides a single port solution for the managed node. Requests for all registered servers on this managed node can be directed through the Communication Broker. The Communication Broker transparently forwards the request to the registered server in the same way as an HTTP proxy forwards an HTTP request. The default port for the Communication Broker is 383 but can be changed.
- For higher security on UNIX systems, `chroot` can be used at start up of the Communication Broker. `chroot` restricts the part of the file system visible to the Communication Broker process by making the specified path act as the root directory, thus reducing exposure to hackers.
- The Communication Broker can be run as non-root on UNIX systems if its port number is greater than 1024.
- The Communication Broker can be configured to run as root-only on UNIX systems to open its port and then switch to a non-root user for all other operations.
- The Communication Broker can be:
  - — Started as a daemon on UNIX systems.
  - — Installed as a Windows NT Service on Windows systems.
- Control commands for the Communication Broker can be restricted to the local managed node only.

- The Communication Broker applies SSL encryption of data transmission over the network.
- The Communication Broker applies SSL authentication through guaranteed identity of senders and receivers.

Figure D-1 Communication Broker Architecture



A Communication Broker configures a minimum of one port for accepting incoming data to a managed node. The port is associated with an OpenView ID (OVCoreID) to identify the managed node. The Communication Broker can be configured to open multiple ports for high availability managed nodes. Each port can have a different identity associated with it. If SSL is enabled, the port is configured with X509 certificates. These certificates allow connecting applications to verify the identity of both message senders and receivers.

All applications on the current managed node that register with the Communication Broker are automatically registered for all active incoming ports opened by the Communication Broker. The port

associated with the default namespace, `bbc.cb`, is automatically activated on startup of the Communication Broker. Other ports can be activated or deactivated dynamically after startup. See the command line interface parameters for the Communication Broker for details.

E Firewalls and HTTPS Communication

Firewall Scenarios

Firewalls are used to protect a company's networked systems from external attack. They usually separate the Internet from a company's private intranet. It is also quite common to implement multiple levels of firewalls to restrict access to the more trusted environments from those of lower sensitivity. For example, the research and finance departments may be contained in the environment of highest security, while direct sales may need to be easily accessible from the outside. Systems on the intranet are allowed, under certain conditions, to cross the firewall to access systems on the internet, for example located in the DMZ. The firewall can also allow systems on the Internet to cross the firewall and access systems on the private intranet. For either of these situations, the firewall must be configured to allow that operation.

HP OpenView's HTTPS communication provides features that allow firewall administrators to configure HP OpenView applications to communicate through firewalls.

Contacting an Application on the Internet from an Intranet using an HTTP Proxy

An HP OpenView HTTPS-based application on a private intranet wants to contact an application outside of the firewall on the public Internet or Demilitarized Zone (DMZ). The OpenView application initiates the transaction and acts as a client contacting a server application on the Internet. The server application could be another HP OpenView application acting as an HTTP server or any other HTTP server application. A common example of a client is a web browser on the private intranet wanting to contact a web server on the Internet. An HTTP proxy must be configured in the browser which forwards the request across the firewall and contacts the web server in the Internet. The firewall is configured to allow the HTTP proxy to cross the firewall. The firewall does not allow the web browser to directly cross the firewall. In the same way, HP OpenView's HTTPS communication applications can also be configured to use HTTP proxies to cross firewalls.

Contacting an Application on the Internet from an Intranet without an HTTP Proxy

An HP OpenView HTTPS-based application on a private intranet wants to contact an application outside of the firewall on the Internet without using an HTTP proxy. The firewall must be configured to allow the HP OpenView application on the private intranet to cross the firewall. This is very similar to configuring a firewall to allow an HTTP proxy to cross the firewall. The firewall administrator may want to set source and target ports for the transaction to restrict communication across the firewall. The `CLIENT_PORT` configuration parameter specifying the source ports can be set from the HP OpenView application when initiating the transaction. The target or destination port is defined in the URL (Uniform Resource Locator or Identifiers) address used to contact the HTTP server on the Intranet. This is the communication broker port on the target node.

Contacting an Application within a Private Intranet from an OpenView Application on the Internet

An HP OpenView HTTPS-based application on the Internet wants to contact an application on a private intranet. This means that a firewall must be crossed from the outside and is usually only allowed by organizations under very restricted conditions set by the firewall administrator. The initiating or client application may do this using an HTTP proxy or go directly through the firewall. The HTTP proxy is outside the firewall and the firewall must be configured to allow the HTTP proxy to cross it. The HTTP proxy could either directly contact the server on the private intranet or go through another proxy, in a cascading proxies arrangement. In either case, the HP OpenView's HTTPS communication client application is configured in the same way. However, the HTTP proxies must be configured differently.

Contacting an Application within a Private Intranet from an OpenView Application on the Internet without using HTTP Proxies

An HP OpenView HTTPS-based application on the Internet wants to contact an application on the private intranet, but there is no HTTP proxy. The firewall must be configured to allow the HP OpenView client application to cross the firewall. The firewall administrators may want to

set the source and target ports for the transaction to restrict communication across the firewall. The `CLIENT_PORT` configuration parameter specifying the source port can be set from the HP OpenView application when initiating the transaction. The target or destination port used to contact the HTTP server on the Intranet is defined in the URL address and is the Communication Broker port on the target node.

If the target server is registered with the Communication Broker, the target port will always have the port number of the Communication Broker. This makes it easier when configuring firewalls. It can greatly reduce the number of target ports an administrator must configure at the firewall.

For information on configuring OVO with firewalls, refer to the *HP OpenView Operations Firewall Concepts and Configuration Guide*.

F OVO 8 Quick Start Guide

OVO Server Components and Processes

The following server components communicate as RPC clients with the HTTPS agent:

- `ovoareqsdr` to send action request and to do heartbeat polling.
- `opcragt` to perform remote control and to do primary manager switches.
- `opcbbcdist` controls the configuration deployment to HTTPS nodes. The deployer is used during remote agent installation.

HTTPS-based communication RPC servers are:

- `ovbbccb` (communication broker).
- `opcmsgrb` (message receiver for HTTPS agents).
- `ovcs` (the security certificate server).

New Processes on the OVO Management Server

A number of new processes are introduced on the OVO management server. The command `opcsv -status` lists all processes that are relevant to OVO, but excludes Oracle and NNM processes. The call displays the following new processes:

- `opcbbcdist`: configuration deployment to HTTPS nodes. Analogous to `opcdistm` for DCE nodes. Both processes are controlled by `opcctlm`.
- `opcmsgrb`: message receiver for HTTPS nodes. Analogous to `opcmsgrd` for DCE nodes. Both processes controlled by `ovoareqsdr`.
- `ovcd`: control daemon; self-controlled.
- `ovbbccb`: communication broker; controlled by `ovcd`.
- `ovconfd`: configuration and deployment process; controlled by `ovcd`.
- `ovcs`: server extension to handle certificate requests; controlled by `ovcd`.
- `opccsad`: OVO certificate server adapter; controlled by `opcctlm`.
- `ovtrcd`: OVO trace server.

When calling `ovstop ovoacomm`, no core OpenView processes are stopped. This includes also the ovcs server extensions. To stop all core OpenView processes, you must enter the command:

```
ovstop ovctrl
```

To terminate all core OpenView processes, enter the command:

```
ovc -kill
```

This also stops the OVO agent on the management server node.

New Commands in OVO 8

Table F-1 gives you a concise overview of the new commands in OVO 8 and their counterparts in OVO 7.1. For full details about any command, refer to the command man page.

The most important new OVO 8 commands are introduced in “HTTPS Communication Administration Commands in OVO” on page 42.

NOTE The wrapper utilities such as `opcagt`, and `opctemplate`, do NOT provide the same output format as the DCE-based `opcxxx` commands.

Table F-1 Command Mapping Table between OVO 7.x and OVO 8

OVO 7.x Command	OVO 8 Command
opcagt	**ovc**
-help	ovc -help
-start	ovc -start AGENT ovc -restart AGENT
-stop	ovc -stop
-status	ovc -status
-kill	ovc -kill
-trace	ovc -trace
-version	ovc -version
opcragt	**ovdeploy**, **ovconfpar**
-agent_version	ovdeploy -inv -host *<node>*
-get_config_var	ovconfpar -get
-set_config_var	ovconfpar -set

Table F-1 Command Mapping Table between OVO 7.x and OVO 8

OVO 7.x Command	OVO 8 Command
opctemplate	**ovpolicy**
-help	ovpolicy -help
-l	ovpolicy -list
-e	ovpolicy -enable
-d	ovpolicy -disable
opcsv	**ovc**
-help	ovc -help
-start	ovc -start SERVER -restart SERVER
-stop	ovc -stop
-status	ovc -status
-trace	ovc -trace
opctranm	**ovdeploy** (HTTPS Agents) **opctranm** (DCE Agents)

Comparison of HTTPS and DCE Agents

Configuration Deployment

Configuration deployment to HTTPS agents differs slightly from that of DCE-based nodes:

- Policies are used by HTTPS agents. These refine and replace the Templates used by DCE-based agents.

  Policies are pushed out by the OVO management server. Templates for DCE agents are pulled by the OVO distribution agent. When the OVO management server system is located inside the trusted environment, policy deployment to managed nodes across a firewall is outbound only.

- Instrumentation is the single term used by HTTPS agents for Actions, Commands, and Monitors. All scripts and binaries are stored in a common instrumentation directory.
- A configuration parameter schema with a name-value pair policy type for HTTPS agents replaces `nodeinfo` and `opcinfo` files.
- `mgrconf` file is enhanced for HTTPS agents by a role model-based security authorization mechanism allowing the deployment of policies and instrumentation from more than one OVO management server.

For detailed information about HTTPS agent configuration management, refer to “Configuration Deployment to HTTPS Nodes” on page 99.

Distribution Managers

`opcbbcdist` is the configuration management adapter between the OVO management server and the HTTPS agents. Its main functions are:

- Convert existing templates into policies.
- Convert ECS templates and the associated circuits into policies.
- Convert node properties into the format used on HTTPS nodes. This replaces the `nodeinfo` file found on DCE nodes.

`opcbbcdist` only accepts requests from the OVO management server. `opcdistm`, the configuration management adapter between the OVO management server and the DCE agents accepts requests from the distribution agent (`opcdista`) of the DCE managed nodes.

Multiple Parallel Configuration Servers

Multiple parallel configuration servers are supported for HTTPS nodes through an owner concept for policies.

Comparison of Resource Requirements

Table F-2 OVO Agent Footprint

Description	HTTPS Agent	DCE Agent
RAM	☺	☺
CPU	☺	☺
Disk	😐	☺

NOTE The managed node footprint for the HTTPS agent becomes progressively more favorable as the number of new OpenView products installed increases. These products share the underlying OV infrastructure and so significantly less software needs to be installed and run as compared to traditionally designed software.

Comparison of Agent Performance

Table F-3 OVO Agent Performance Comparison

Description	HTTPS Agent	DCE Agent
OVO agent binary installation	Full - 😐 Patch - ☺	Full - ☺ Patch - ☹
Policy and instrumentation deployment	Full - 😐 Delta - ☺	Full - ☺ Delta - ☹
OVO message throughput	😐	☺

Comparison of Agent Commands

Table F-4 OVO Agent Command Comparison

Description	HTTPS Agent	DCE Agent
Start, stop, status, and control of the OVO agent	`ovc` `opcagt` wrapper	`opcagt`
Policy/template management	`ovpolicy` `opctemplate` wrapper	`opctemplate`

Local configuration settings	`ovconfget`
	`ovconfchg`
	Configuration parameter schema with a name-value pair policy type.
		`nodeinfo` file
		`opcinfo` file
		Configuration files

Remote agent control from OVO server	`opcragt` `ovconfget/set`	`opcragt`

Comparison of Agent Processes

Table F-5 OVO Agent Process Comparison

Description	HTTPS Agent	DCE Agent
Start, stop, and control of the OVO agent	`ovcd`	`opcctla`
Policy and instrumentation deployment	`ovconfd`	`opcdista`
Communication	`ovbbccb` HTTPS-RPC server using one configurable port. Default: 383.	`llbserver` dced, rpcd, or llbd on fixed port 135.
Security	`ovcs` - Certificate server `opccsad` - Certificate Adapter `ovcd` - Certificate client	n.a.
HTTPS agent configuration adapter	`opcbbcdist`	n.a.
Message agent	`opcmsga`	`opcmsga`
Monitor agent	`opcmona`	`opcmona`
Embedded Performance Component	`coda`	`coda`
Logfile encapsulator	`opcle`	`opcle`
Message Interceptor	`opcmsgi`	`opcmsgi`
SNMP Trap Interceptor	`opctrapi`	`opctrapi` `opcevti (Windows)`
Event Correlation	`opceca`	`opceca`
ECS Annotate Server	`opcecaas`	`opcecaas`

Comparison of Troubleshooting Methods

Table F-6 OVO Agent Troubleshooting Comparison

Description	HTTPS Agent	DCE Agent

Tracing	`ovtrcadm`[a]
	`ovtrcmon`
	`ovtrcadm`
	`ovtrccfg`
	`ovtrcd`
	Tracing is more powerful but there is some increased complexity associated with the greater functionality.
		`opcagt -trace`

a. Tracing capabilities of the HTTPS agent are described in detail in the dedicated document *HP OpenView Operations - Tracing Concepts and User's Guide*.

Index

Index

Index

Index

Index

Index


invent


Java™
COMPATIBLE