HP Operations Manager

for the Windows operating system

Software Version: 8.16

PDF version of the online help

This document is a PDF version of the online help that is available in HP Operations for Windows.
It is provided to allow you to print the help, should you want to do so. Note that some interactive topics are not included because they will not print properly, and that this document does not contain hyperlinks.

Document Release Date: November 2009

Software Release Date: November 2009

TABLE OF CONTENTS

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notices

Trademark Notices

HP Operations Manager for Windows Overview

HP Operations Manager for Windows (HPOM) is a distributed, client-server software solution designed to provide service-driven event and performance management of business-critical enterprise systems, applications, and services.

HPOM enables management of distributed, heterogeneous, e-business infrastructures and includes support for a broad range of Windows and UNIX systems and applications, including e-commerce, web and application servers, conferencing and emails, databases, ERP software, and more.

Using HPOM, administrators can maximize IT system performance, reduce downtime, delegate tasks to operators, and reduce costs. HPOM constantly monitors thousands of events occurring on all your managed nodes and presents just the information you want to know just when you need it.

To manage your enterprise environment, HPOM performs the following functions:

- Auto-discovers the managed environment and auto-deploys management policies.
- Monitors and detects events or potential performance problems arising from managed nodes and services.
- Extends your management viewpoint beyond event and performance management to include a business service perspective.
- Notifies you in one view when a problem occurs.
- Displays data graphically for in-depth problem diagnosis.
- Displays map views of selected services or nodes, showing relationships and dependencies, multi-level views, and root cause and impact analysis.
- Solves problems automatically or manually to prevent downtime in your service environment.
- Integrates with other HP BTO Software components to provide breadth and perspective, as well as specialty management solutions for specific disciplines.
- Manages key systems and applications with out-of-box intelligence using Smart Plug-in management modules.
- Offers a web console view in addition to the MMC console for remote and mobile operational control.
- Collects network, managed node, and performance metrics to help you optimize performance and prevent problems.
- Creates user roles to configure an operator's view of the environment to focus on specific assigned tasks and responsibilities.
- Provides an industry-standard database, including clustered database support.

Related Topics:

- Getting started with HP Operations Manager for Windows
- HP Operations Manager for Windows and the MMC

HP Operations Manager for Windows and the MMC

When you install HP Operations Manager for Windows, you are installing both the MMC and HPOM for Windows, which is a snap-in to the MMC. MMC is a framework that hosts snap-ins such as administrative tools used to administer networks, nodes, services, and other system components. MMC does not perform administrative functions. It hosts programs, called snap-ins, that do.

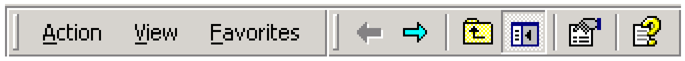
Together, the MMC and its snap-ins combine to form a console. You can save a console by adding snap-ins to the MMC framework and saving the configuration. Console views are saved as files with an `.msc` extension. Any snap-in settings for administering specific components or nodes can be saved as a customized console.

All of the configuration settings for the tools and controls are saved with the console view and restored when the customized console file is opened. As an administrator, you can create a console file, save it as an `.msc` file, then distribute it to different computers across your environment for operators to use in performing their tasks.

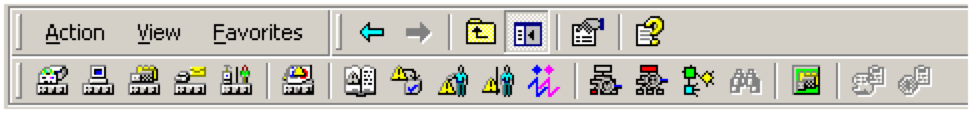
Typically, the `.msc` file is specific to one management server, which is named in the `.msc` file. If you distribute the `.msc` file to a different management server, when you open the `.msc` file, it attempts to find the server originally specified. If unable to locate that server, the program prompts you for a new server name.

The components of an MMC console are contained in the MMC window, which has several menus and a toolbar that provide the commands to open, create, and save MMC console views. The toolbar on the MMC window is called the main toolbar.

MMC menu and toolbar



HP Operations Manager for Windows menu and toolbar



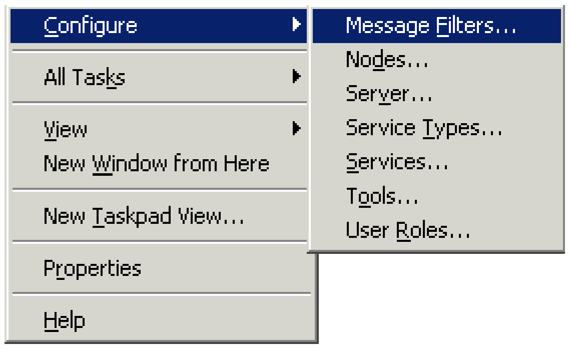
Each console view has its own menus and toolbar, separate from those of the main MMC window, that help the user perform tasks.

MMC and HPOM for Windows menu behavior

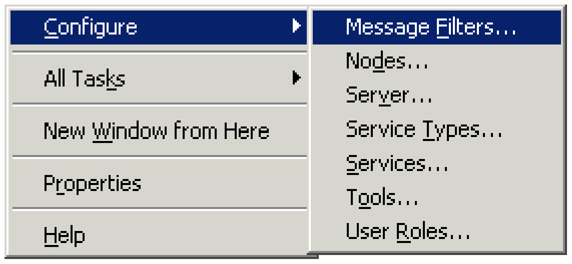
The MMC and HPOM for Windows make extensive use of context, or shortcut menus. These menus may contain commands provided by:

- The MMC console
- HP Operations Manager for Windows

Menus may display different commands depending on the context in which they are opened and the way they are selected. For example, click services, nodes, or tools in the console tree to select the item, then right-click to display a shortcut menu for that item that looks like this:



If you right-click an item in the console tree without first selecting it with a left-click, a slightly different menu displays.



Behavior of New Windows

In the console, when you can open a new window in the following ways:

1. Click one of the View toolbar buttons to view a message browser or map.
2. Choose a different view from the context menu in the map or message browser.

You can also request a new window from the console tree by selecting an item and then right-clicking to open the console menu. From this menu, choose the option New Window from Here.

Related Topics:

- HP Operations Manager for Windows Overview
- First steps

Software Support Online

The HP Software Support Online web site offers in-depth information on a variety of topics:

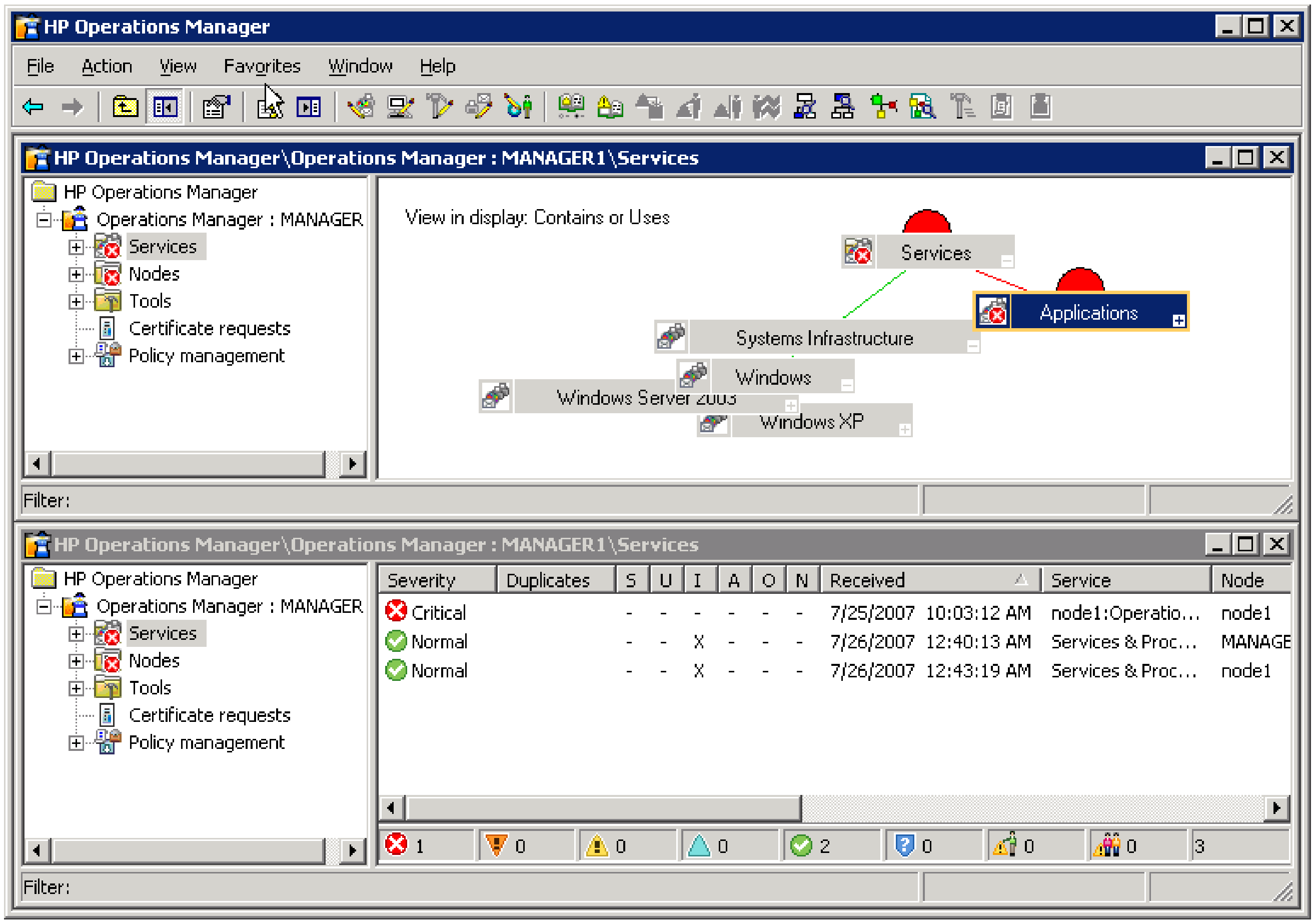
- Troubleshooting, knowledge base search, known problems
- Problem reporting and support information
- User manuals, software updates and patches, demos
- Training and education
- Discussion forum

Click Software Support Online to open the Support web site in a separate browser window.

Getting started with HP Operations Manager for Windows

HP Operations Manager for Windows helps you manage the increasing complexity of your IT service environment from a single console view to increase productivity and prevent or correct downtime of critical business services.

The illustration shows a typical HPOM for Windows view, as configured by the administrator, with the console tree on the left, a map view in the details pane, and the active messages browser window open above the map.



Severity	Duplicates	S	U	I	A	O	N	Received	Service	Node
Critical		-	-	-	-	-	-	7/25/2007 10:03:12 AM	node1:Operatio...	node1
Normal		-	-	X	-	-	-	7/26/2007 12:40:13 AM	Services & Proc...	MANAGE
Normal		-	-	X	-	-	-	7/26/2007 12:43:19 AM	Services & Proc...	node1

HPOM for Windows provides a view of your environment configured by the administrator to:

- Display the status of managed nodes and services through color-coded icons and messages, as configured by the administrator through deployed policies.
- Provide tools for further diagnosis and administration.

- Launch corrective actions required to resolve problems before critical services are impacted, as configured by the administrator through deployed policies .

Related Topics:

- HP Operations Manager for Windows Overview
- HP Operations Manager for Windows and the MMC
- About the map view
- About the message browser

First steps

At installation, HP Operations Manager for Windows installs several default policies that can be used as-is to manage your enterprise environment. Deploying the default policies to your managed nodes provides the fastest, simplest way for you to begin receiving data and messages from your environment.

For more detailed, specific information from your managed services and nodes, you must configure your environment by editing the existing policies or creating new policies to meet your special needs. After configuring the policies, you deploy them to managed nodes and services.

In either case, you need to perform several steps to configure your console to report and display the performance information and messages important to your enterprise operations.

Configuration overview

Configuring your enterprise environment requires several steps:

- Select and configure managed nodes
- Configure tools
- Configure services
- Create, manage, and deploy policies
- Configuring event policies
- Apply tools to nodes and services
- Create a new policy
- Create user roles
- Create service types

License HP Operations Manager for Windows

You must have a license key password to use HPOM for Windows. At installation, you are given a 60-day trial license. Within this 60 day period, you must obtain a permanent license key password to continue to use HPOM. To obtain your license key password at installation, follow the instructions in the message box that appears on screen at product startup. It takes a minute or so for the licensing program to launch.

NOTE:
You cannot request a license key password from a remote console. You must log on to the management server to obtain a license key password.

If you do not obtain your permanent license key password on startup, you can obtain it from within the program at a later time by launching the licensing program from the console tree.

HP Operations Manager for Windows Manager

The full management server license allows you to operate one HPOM for Windows management server, including the management server node .

Additional licenses are required for the following items:

- Managed nodes: Any additional node managed by the HPOM for Windows management server requires an additional HP Operations agent license.
- Monitored nodes: Target connector licenses are required for each remote system that is being monitored by HPOM. Monitored nodes are systems that do not have an HP Operations agent installed, for example, agentless nodes, systems that are monitored through proxy connections, or systems that send messages to HPOM by way of external nodes. For nodes monitored through external nodes, target connector licenses are only required if a message has been received for that node. Target connector licenses are not required if the event-originating node is an L2/L3 networked device and the event is from HP Network Node Manager, or if the event is from other HP monitoring software (for example, HP SiteScope, HP Internet Services, HP Storage Essentials, HP Insight Manager, NonStop management, Webjetadmin, Procurve monitoring).
- Smart Plug-ins: Additional licenses are required for the installation and use of HP Operations Smart Plug-ins.

The HP Operations Manager for Windows Manager LTU entitles you to a license each for HP Reporter and HP Performance Manager. The HP Reporter license allows you to install HP Reporter locally on the management server or remotely on another system. A remote installation is recommended to balance the available resources. HP Performance Manager can only be installed locally on the HPOM management server.

HP Operations Manager for Windows Limited Edition

The HP Operations Manager for Windows Limited Edition LTU allows you to operate the HPOM for Windows management server with 20 nodes. (You can extend the number of nodes to 30 by purchasing an extension to the HP Operations Manager for Windows Limited Edition LTU.)

The number of licensed nodes includes:

- Management server node: The HP Operations agent running on the management server.
- Managed nodes: Nodes that have an HP Operations agent installed.
- Monitored nodes: Nodes that are monitored by HPOM but do not have an HP Operations agent installed.

The HP Operations Manager for Windows Limited Edition LTU includes the right to use one copy each of selected SPIs, of HP Reporter, and of HP Performance Manager.

NOTE:
With an HP Operations Manager for Windows Limited Edition, you will *not* be able to manage more than the license-specified number of nodes, even if there are additional standard HP Operations agent licenses being installed.

An HP Operations Manager for Windows Limited Edition license can be upgraded with an HP Operations Manager for Windows Manager license at any time. However, in this case, you also need to buy the appropriate amount of licenses for the HP Operations agents running on all the nodes managed by the HPOM for Windows management server, for all remote nodes that are monitored by HPOM, and for the HP Operations Smart Plug-ins to be used.

For details about the HP Operations Manager for Windows Limited Edition, its extension, and migration possibilities, contact your sales representative or see the customer letter included with your media bundle.

HPOM counts the number of available and used licenses at the following times:

- Each time the console starts
- Each time a policy or package is deployed
- Each time a node is added
- At least every 24 hours

For complete details about licensing HP Operations Manager for Windows, see the help files provided with the HP AutoPass licensing program.

Related Topics:

- Obtain a license
- Create and view a license report

Obtain a license

Use these instructions to obtain a permanent license number for HP Operations agents, HP Performance Agent Software agents, target connectors, Smart Plug-ins, and other related HPOM for Windows products.

To obtain a license, you must provide the number of the HP Purchase Order that you received from your HP Software authorized reseller when you bought the product that you want to license. If you have not yet purchased the product, call 1-877-686-9637 (in the United States and Canada) or visit hp.com to locate an HP Software authorized reseller.

To obtain a license

1. In the console tree, select Tools → HP Operations Manager Tools → Licensing .
2. Right-click Obtain License and select All Tasks →Launch Tool... .
3. Select the product for which you want to purchase licenses from the window that appears.
4. When the HP AutoPass window appears, use the wizard to enter the required information and to receive your license number.

Related Topics:

- License HP Operations Manager for Windows
- Create and view a license report

Create and view a license report

To verify that you have purchased enough licenses for the HP Operations agents, HP Performance Agent Software agents, target connectors, Smart Plug-ins, and other HP Operations Manager for Windows products that you have installed, use the License Report tool. This report shows what products you have installed, how many copies are installed, and how many licenses you have purchased for these products.

To create and view a license report

1. In the console tree, select Tools → HP Operations Manager Tools → Licensing .
2. Right-click License Report and select All Tasks →Launch Tool... .
3. Wait for the report to be generated. This takes about 20 seconds, or longer, depending on the speed of your computer and the size of your managed environment.
4. When the report appears in your browser, select a product from the pull-down menu. For each product, the following information appears:

   Running with Instant On:
   : This field indicates whether the product is running under an Instant On license. The first time you install a product, you get an Instant On license that allows you to use the product for 60 days.

   Remaining days:
   : The number displayed here is the number of days for which the Instant On license will remain valid.

   Expiration Date:
   : The date the license expires.

   License Capacity:
   : The number displayed here is the number of licenses that you have purchased for this product.

   Number of installed nodes:
   : The number displayed here shows the number of nodes where this product is installed.

   Available licenses:
   : The number displayed here shows the number of unused licenses that are available for this product.

   Number of unlicensed installations:
   : The number of product installations that are running without a valid license. If this number is not zero (0), purchase the necessary licenses.

NOTE:
For target connector licenses, the report calculates an average value for all used licenses, based on the license usage of the last 30 days. For HP Operations agent licenses, the report gives the number of licenses used at the time the report is generated.

Related Topics:

- Obtain a license
- License HP Operations Manager for Windows

High contrast option

If you want to use the high contrast mode for better readability, set the high contrast mode using the following key sequence before you start the HPOM for Windows console. Press all keys simultaneously.

Left ALT, Left Shift key, and the Print Screen key

This opens a dialog box in which you can set your preferred high contrast options. Alternatively, if you set the high contrast options by following the path Control Panel → Display → Appearances, you should still use the key stroke sequence after setting your options in the dialog box to enable all the features of the high contrast option.

Keyboard navigation

Keyboard navigation in HP Operations Manager for Windows conforms to standard Microsoft behavior with the following exceptions:

Tab key: The Tab key is not functional in the Configure Nodes and Configure Tools dialog boxes. To move through the enabled controls in either of these dialog boxes, use these key combinations instead:

Ctrl + T (tabs through enabled controls)
Ctrl + Shift + T (tabs through enabled controls in reverse order). Press the Spacebar to choose what you have selected in the dialog box. (For example, to select the OK or Cancel button.)

To move in and out of the dialog boxes using the keyboard, use the Application key (between the Windows logo key and the Ctrl on the right side of the keyboard), or Shift + F10.

NOTE:
Pressing the Tab key shifts the focus from the dialog box to one of the main windows in HP Operations Manager for Windows. To tab back into the dialog box, press the Tab key (if necessary) to set focus on either the console tree or details pane, then press Shift + F10 to open the context menu.

Press the letter o to return to the Configure Tools dialog box. Press the letter d to return to the Configure Nodes dialog box.

How to use help

HP Operations Manager for Windows provides help files to guide you through the tasks you want to perform. Help is available from several locations:

- Help drop-down menu
- Any shortcut menu
- Help buttons in dialog boxes

Help files consist of overview, conceptual, and procedural topics and interactive demonstrations.

- Overview topics are associated with the book level icon in the table of contents. Procedural topics may also contain some overview information, to explain concepts related to a particular task.
- Conceptual topics provide a high-level view of a task or a series of related tasks and are identified in the table of contents with an icon that resembles a document page.
- Procedures are step-by-step instructions that describe a particular user task. These are identified in the table of contents with an icon that illustrates numbered steps.
- Interactive demos illustrate complex concepts such as the flow of a process or the interrelationship of rules and parameters. In some cases you can change information in a help demonstration to test the results of an action before you implement the task. See the Basic Training for HP Operations interactive tutorial in the help system table of contents for an example of an interactive demonstration.

Links to more information

Blue underlined text is linked to related topics that appear in various ways, depending on the content of the linked information:

- Popup windows display acronym and glossary information.
- Links to other topics replace the existing text in the help window. Use the Back button to return to your starting place.
- Links to hidden text expand to provide additional information without requiring you to leave your current location. A small blue arrow indicates expandable information. When printing from help files, expand the hidden text to print it.

Cautions and notes

- Cautions show important information that may significantly affect your work.
- Notes present helpful information.

Printing help files

Print topics of interest by printing one topic page at a time or groups of topics by right-clicking the topic in the contents pane and selecting `Print` . If you want to print the entire help, start from the top-level topic (*HP Operations Manager for Windows*). Note that, depending on the number of SPIs installed, the help may contain more than 2500 printed pages.

You can also download the core help (which excludes help for SPIs) in PDF format from the HP Software Product Manuals web site.

Copying help files

You may want to copy the help files from your management server to your local computer. This enables to you to read HPOM documentation when you cannot access the management server. The help files are .chm files, and are located on your management server at `%OvInstallDir%\help\en\` . Copy all the .chm files into one directory on your local computer. Double-click `console.chm` to start the online help.

For more information...

Additional information about HP Operations Manager for Windows is available in manual format as .pdf files. Unless you specified a different directory, manuals are installed in the default installation directory:

`C:\Program Files\HP\HP BTO Software\paperdocs\en` .

Manuals are also available on the Documentation directory on the installation media in this location:

`Documentation\HPOM Guides`

Additional technical information is provided in the form of white papers, available on the installation media in this location:

`Documentation\White Papers`

Basic Training for HP Operations Manager

Basic Training is an interactive tutorial that opens automatically after the first installation of HP Operations Manager for Windows. To open the tutorial at any time after installation, go to the help system table of contents and click the Basic Training for HP Operations contents entry. Look for the light bulb icon, the symbol for an interactive demo.

HPOM for Windows Manuals

- *HP Operations Manager for Windows Upgrade Guide*

  The interactive upgrade guide lets you choose options to indicate the type of upgrade that you want to perform. The procedure changes according to your selection, creating an upgrade procedure that is tailored to your individual situation. You can view the file from the Documentation directory on the installation media.

- *HP Operations Manager for Windows Installation Guide*

  The installation guide contains detailed installation information and an overview of how to use other HP BTO Software products with HP Operations Manager. A printed guide comes with each set of installation media. You can view the file from the Documentation directory on the installation media.

- *HP Operations Manager for Windows Release Notes*

  The release notes file (HPOM_Release_Notes.htm) is also available in the Documentation directory on the installation media.

To open and view or print a file, navigate to the directory where you have installed the manuals and double-click the name of the file you want to view or print.

Terminal Services support

HP Operations Manager for Windows supports remote operation using Microsoft Windows Terminal Services and Citrix Metaframe XP on Windows 2000 and 2003 server. Both the Terminal Services Administration Mode and the Terminal Services Application Mode are supported.

For detailed information on Microsoft Terminal Services and Citrix Metaframe, refer to the HPOM Support Matrix .

Administering your environment

The administrator of an enterprise IT environment has many responsibilities. Using HPOM, you can perform all necessary configuration tasks for your operators on a single management console from one central location.

As administrator, you will:

- Install HPOM and deploy to managed nodes .
- Define the environment for operators by managing installation and the configuration of nodes, tools , and services.
- Configure policies for receiving messages or the collection of metrics and deploy them to managed nodes.
- Customize the automatic commands (scripts that are configured to run automatically when a specific event occurs) and operator-initiated commands (the set of commands/executable scripts that are available to an operator when a specific event occurs) required to complete a task.
- Provide specific message instructions to help with problem resolution.
- Extend the scope of HPOM by integrating additional tools and managed services and nodes.

Set up an operator's view

As administrator, you determine the view of your enterprise environment that will be displayed in HP Operations Manager for Windows by configuring the environment using policies . The view you create is used by operators to perform these tasks:

- Monitor performance and events.
- Perform routine maintenance tasks.
- Correct problems as they arise.
- Prevent problems before they can impact your network.

HPOM provides predefined policies that you can use as-is or modify to meet your requirements. You can also create your own new policies. After installing HPOM, you perform a number of configuration and deployment tasks to create a view of the managed nodes and services in your environment, the status messages that apply to them, and the corrective actions available to resolve problems.

By creating user roles and assigning users to these various roles, you configure an operator's view of the environment to focus on specific assigned tasks and responsibilities. By defining roles for specified operators and administrators, you control the users view of your enterprise and the range of activities which that user has permission to perform. By assigning users to well-defined, specific roles, you can distribute monitoring and maintenance tasks across a group of individuals with their own particular areas of expertise and experience and customize each users console view.

Configure the management server

- Configure managed nodes and deploy necessary policies to these nodes.
- Configure tools that operators use to resolve problem situations.
- Configure services by defining the service hierarchy and specifying the way status is calculated and propagated for display in the message browser and map views.
- Configure service types that are used when an instance of a service is created.

Develop policies

- Create policies for messages or the collection of metrics and deploy them to managed nodes. You can use the predefined default policies supplied with HPOM if you prefer.
  - Specify the messages that result from events that occur in the environment.
  - Customize the automatic actions (scripts that are configured to run automatically when a specific event occurs) and operator-initiated actions (the set of commands/executable scripts that are available to an operator when a specific event occurs) required to complete a task.

- Configure measurement collections. Specify the data to be collected, the nodes from which it is collected, and the thresholds, which when exceeded, trigger events that produce messages in the console for operators to act upon.
- Provide specific message instructions to help with problem resolution.

Deploy policies

- Deploy appropriate policies to specified managed nodes
- Extend the scope of HPOM by integrating additional applications and managed services and nodes.

Assign operators and administrators to user roles

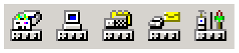
- Create user roles that focus the attention of operators or administrators on their primary tasks without the distraction of information that is not relevant to their assigned responsibilities.
- Specify which users can view, create, modify, delete, deploy, and undeploy policies and packages.
- Specify which services, nodes, and tools will be available to specific user roles.
- Specify which Message Groups will be available to operators.

Related Topics:

- Configuring Managed Nodes
- Configuring Tools
- Configuring Services
- Configure Service Types
- Policy Management and Deployment
- Configure User Roles
- Configuring event policies

Using the configuration toolbar

Configuration tasks are grouped on the Configuration toolbar to provide quick access to all configuration editors.



Click the appropriate icon in the console view to open one of the editors to configure services, nodes, or tools.

HPOM security

Security is involved in many of the HPOM tasks performed by administrators and operators. As an administrator, you need to consider your environment and plan your security needs before installing HPOM.

Users establishing security for HPOM should be:

- Experienced Windows administrators who understand Windows security concepts and terminology.
- Experienced with the Windows domain environment.

Related Topics:

- HPOM group accounts
- HPOM user accounts
- Update HPOM user accounts

HPOM group accounts

HPOM uses Windows group accounts to identify valid users of the product. During the installation, HPOM creates the following two groups, either locally on the management server (if you are performing a workgroup installation), or in the domain for domain installation:

- HP-OVE-ADMINS (HPOM administrators)
- HP-OVE-OPERATORS (HPOM operators)

You can also specify different names during the installation, or, instead of having HPOM create them for you, you can also create these groups yourself before or during the installation.

The purpose of these groups is to allow users to access the HPOM console and HPOM resources so that they can perform HPOM administrator or operator tasks. For example, you can add Rosa Galvez, who is not a Windows administrator, to the HP-OVE-ADMINS group so that she can perform HPOM administrator tasks, or add Sean Payne, another HPOM user who is not a Windows administrator, to the HP-OVE-OPERATORS group to perform operator tasks. Local or domain administrators by default cannot access HPOM.

NOTE:
No user should be part of both groups. An HPOM administrator should not be added to the HP-OVE-OPERATORS group.

HP-OVE-ADMINS

The HP-OVE-ADMINS group by default contains the following members:

- HP-OVE-User
- HP-OVE-Deleg-User
- Installing user

  After installing the management server, the installing user is always an HPOM administrator, even if you remove the installing user from the HP-OVE-ADMINS group. You can remove the installing user's HPOM administrator permissions as follows:

  a. Configure an alternative HPOM administrator by adding at least one other user to the HP-OVE-ADMINS group.

  b. Log in to the management server with an account that is a member of the HP-OVE-ADMINS group.

  c. In the console, launch the tool Tools → HP Operations Manager Tools → Cleanup rights of the installing user .

  d. Remove the installing user from the HP-OVE-ADMINS group.

NOTE:
The installing user still has HPOM administrator permissions until they log out.

HPOM Administrator Tasks

Users classified as HP-OVE-ADMINS can perform the following HPOM administrator functions:

- Add managed nodes
- Configure managed nodes, tools, service types, and user roles
- Deploy policies and packages Required administrator privileges for package deployment
- Create and modify policies, tools, services, and user roles
- Create automatic and operator-initiated commands
- Expand and navigate Policy Management in the console tree
- Run utilities (register packages, upload and download from user roles policies)
- Perform all operator tasks

HP-OVE-OPERATORS

The HP-OVE-OPERATORS group is initially empty.

HPOM Operator Tasks

Users classified as HP-OVE-OPERATORS can perform all HPOM tasks except those listed under HP-OVE-ADMINS tasks:

- All actions on messages
- Use message filters
- Change severity
- Modify message text
- With the proper authority, execute tools, view service maps, and access policy and package management.

As an administrator, after adding a user to the HP-OVE-OPERATORS group, you can further define the rights of that user with the User Roles Editor.

Related Topics:

- HPOM user accounts
- Configuring user roles

HPOM user accounts

When the HPOM management server is installed, you are asked to supply the name of two Windows user accounts that are used by several HPOM services and processes to control their security rights:

- HP-OVE-User

  This is the user account under which the HPOM management server processes run, with the exception of the policy management and deployment process.

- HP-OVE-Deleg-User

  The policy management and deployment process (ovpmad) runs under the HP-OVE-Deleg-User account. In domain environments, the HP-OVE-Deleg-User account must have delegation rights, if you plan to deploy Windows HTTPS agents from a remote console using the *Impersonate* user deployment option

Both user accounts are by default members of the HP-OVE-ADMINS, Administrators, and Users groups. They are domain accounts if HPOM is installed in a Windows domain, or local accounts if HPOM is installed in a Windows workgroup environment.

NOTE:
The HP-OVE-User and HP-OVE-Deleg-User accounts are intended solely for the use of the HPOM product. As such, the HPOM product assumes that it owns these accounts and can safely manipulate them as needed to meet the needs of the product. Specifying accounts that are used for other purposes may cause problems in your environment. Choose account names that will not be used by anyone else in your organization.

The accounts that you supply during the installation for the HP-OVE-User and HP-OVE-Deleg-User users can be changed after the installation using the `ovchgpass.exe` command. Use the command to change the user name, password, or both. This command changes the password everywhere that the HPOM system uses this account, to ensure that HPOM services do not fail.

Related Topics:

- Update HPOM user accounts
- HPOM group accounts

Update HPOM user accounts

You can change account information for the HP-OVE-User and HP-OVE-Deleg-User users with the `OVChgPass.exe` utility. As administrator, when you install HPOM, you choose a user name and password for these accounts. The user accounts are used by the HPOM management server services. User name and password are registered in the services.

Using `OVChgPass.exe` , you can create a new account or change the password for the existing HP-OVE-User and HP-OVE-Deleg-User accounts. When you change the password, it is changed in every instance where it is used. If you change the user name, it is also changed in every instance where it is used. The names of the current HP-OVE-User and HP-OVE-Deleg-User accounts are found and updated in the registry at this location:

```
HKEY_LOCAL_MACHINE\SOFTWARE\HewlettPackard\OVEnterprise\Security\Aliases|HP-OVE-User
HKEY_LOCAL_MACHINE\SOFTWARE\HewlettPackard\OVEnterprise\Security\Aliases|HP-OVE-Deleg-User
```

The utility runs on the management server and performs these updates:

- Creates a new user account on the system where the tool runs, if the given account does not exist. Alternatively, only the password is changed. In both cases, the user is added to the HP-OVE-ADMINS, Administrators, and Users groups.
- Updates all HPOM services which run as the old HP-OVE-User or HP-OVE-Deleg-User by associating them to the new user/password. You must manually restart the services after making this change.
- Updates all DCOM servers currently configured to run under the HP-OVE-User or HP-OVE-Deleg-User identity properties.
- Updates all HPOM tools that are configured to run under the HP-OVE-User or HP-OVE-Deleg-User account.
- Updates all scheduled task policies that are configured to use the HP-OVE-User or HP-OVE-Deleg-User to start commands. Only the latest version of the policy (with the highest version number) is updated; a new version of this policy will be created.

If any action fails, the update procedure continues. To avoid inconsistency, you must make a manual change of the failed account data. In this case, note the failed action and write down the name of the service shown in the error message. If the first step should fail (update or creation of the Windows account), the tool stops execution.

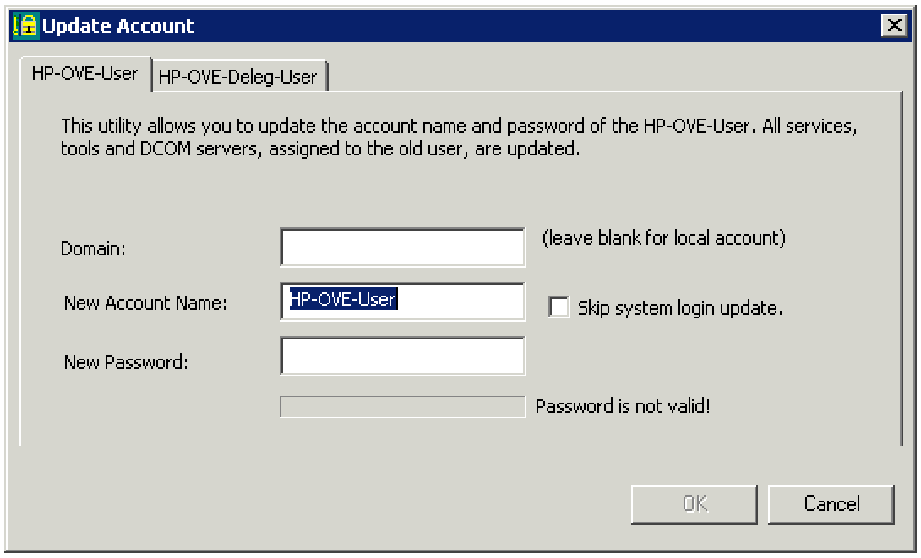
If the given user accounts do not yet exist, the user who runs the `OVChgPass.exe` utility must have sufficient privileges to create user accounts locally on the management server system or, for domain installations, in the domain. Alternatively, the user must have sufficient rights to change passwords.

To update HPOM user accounts

1. From the command prompt, to open the utility, type:

   OVChgPass.exe

2. The Update Account dialog box opens and displays the HP-OVE-User tab.



Update Account

This utility allows you to update the account name and password of the HP-OVE-User. All services, tools and DCOM servers, assigned to the old user, are updated.

Domain: (leave blank for local account)

New Account Name: HP-OVE-User

Skip system login update.

New Password:

Password is not valid!

3. Select the HP-OVE-Deleg-User tab, if you want to change the HP-OVE-Deleg-User account.
4. Enter the domain where the user is located. Leave this box blank for a local account.
5. If desired, enter a new user name.

NOTE:
If you change the user name for the HP-OVE-Deleg-User, you must execute the Windows Node Security Setup tool.

6. Select Skip system login update , if you do not want HPOM to create the new user account or to change the user password. When you select this option, make sure the administrator has already made the required account changes locally or in the domain.
7. Enter your new password.
8. Click OK .

To change account information manually

1. To update HPOM management server services, select Services from Administrative Tools in the Control Panel . The user login is shown in the Log On As column. Change the log on properties of each HPOM service.
2. To update the DCOM server's properties, start `DCOMCNFG.EXE` from the `WINNT\system32` folder.

3. To update the HPOM tools, go to the console tree.

   Select Tools → HP Operations Manager Tools → Modify Tools login / password .

Related Topics:

- HPOM user accounts
- HPOM group accounts
- Start Windows node security setup

Access requirements for NTFS partitions

NTFS partitions requirements include:

- The local groups named Administrators, HP-OVE-ADMINS, and HP-OVE-OPERATORS must have full access to the executable files and subdirectories in the following directories on the management server:
  - `%OvInstallDir%`
  - `%OvDataDir%`
  - `%OvShareDir%`
  - `%SYSTEMROOT%\system32`

  In addition, these groups must either be given the Bypass Traverse Checking privilege or they must also have Execute access to all of the parent directories of these directories. By default, Windows 2000 installations give this type of access to "Everyone," so you should only need to make changes like this if you modified your file system security from the default settings.
- Accounts used for running tools must have the correct access on any NTFS partitions that will be accessed by the tool.

On FAT file systems or partitions, there is no security to set.

Trust relationships between Active Directory domains

If your managed environment includes more than one Active Directory domain, you must ensure that the correct trust relationships exist between these domains. Management servers, consoles, and nodes can run in different domains, and if two-way trusts exist between all your domains, no issues should arise. However, if some trust relationships do not exist, certain HPOM features may not function properly.

Specifically, to make full use of all HPOM features, the following trust relationships must exist:

- Trust is required between the management server's domain and the domain where the HPOM service accounts exist. The management server consists of a number of services, which run under service accounts (called HP-OVE-User and HP-OVE-Deleg-User by default).
- Trust is required between the management server's domain and the console user's domain.

  The console uses DCOM to communicate with the management server. The management server must be able to verify the console user's credentials.

  NOTE:
  You can open remote consoles in the same domain as the management server, and in its trusted domains. You cannot open remote consoles in domains that the management server does not trust. You also cannot open remote consoles if the management server is part of a workgroup. Instead, you can use Microsoft Terminal Services or Citrix Metaframe to remotely access the management server, and then open the console locally on the management server.

- Trust is required between the domain of the computer on which the console runs and the domain where the HPOM service accounts exist.

  The management server uses a DCOM interface to notify the console of updates that it must display (for example, the status change of a deployment job). The computer on which the console runs must be able to verify the management server's credentials. (The management server runs under the HPOM service accounts.)

Some remote agent installation options do not require trust relationships to exist. However, to enable all remote agent installation options, the following trust relationships are required:

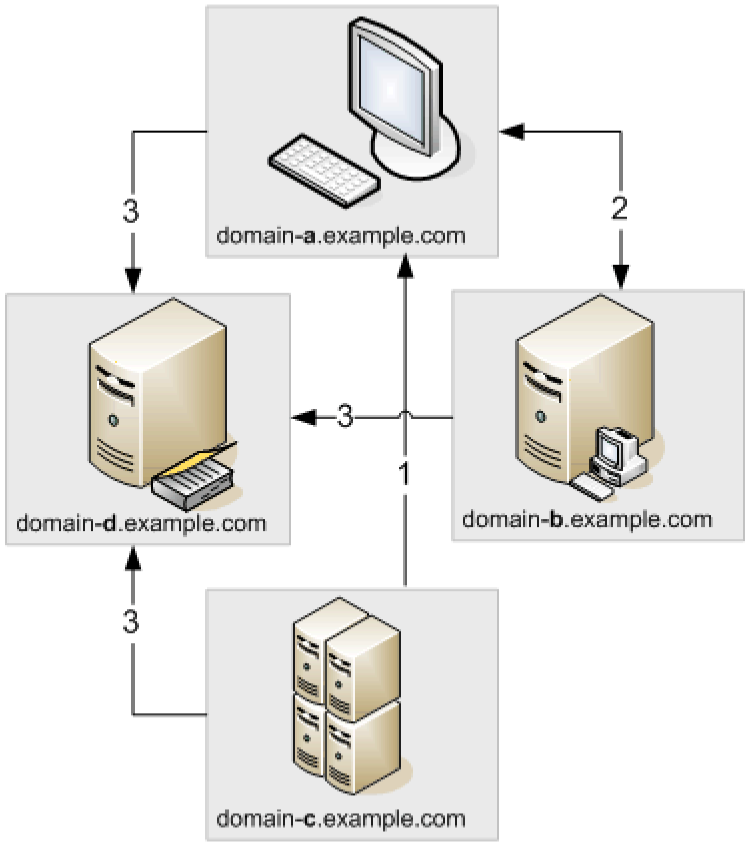
- Trust between the managed node's domain and the domain where the HPOM service accounts exist.
- Trust between the managed node's domain and the console user's domain.

The figure below shows a management server, console, and managed nodes, all in separate domains. In addition, the HPOM service accounts exist in a fourth domain.

- domain-a.example.com : remote console domain
- domain-b.example.com : management server domain

In addition another management console or managed nodes can exist in the domain as well.

- domain-c.example.com : managed nodes domain
- domain-d.example.com : HPOM service users domain



The figure above shows the following trust relationships:

1. The trust relationship from the managed nodes domain to the remote console domain.
2. The two-way trust relationship between the management server domain and the remote console domain.
3. The trust relationships from the remote console domain, management server domain, and managed nodes domain, to the HPOM service users domains.

Related Topics:

- Install agents remotely
- Install agents in trusted domains

Installing DCE agents on backup domain controllers

If you need to install a DCE agent to run under the HP ITO account on a Windows backup domain controller, you must complete several prerequisite steps. You do not have to complete these steps if you install the HTTPS agent on a domain controller, or if you install the DCE agent to run under the Local System account. HP recommends that you use the Local System account if you install the DCE agent on a domain controller.

Before you install a DCE agent to run under the HP ITO account on a Windows backup domain controller, you must complete the following steps:

1. Add the primary domain controller as a managed node .
2. Install the DCE agent on the primary domain controller.
3. Synchronize your backup domain controllers.

NOTE:
You must ensure that all or none of the domain controllers in the domain are managed nodes.

CAUTION:
If you add a Windows domain controller as a managed node, you allow tools and scheduled commands to run without a password. This means that any administrator who configures tools in HPOM can configure a tool to run as any user (including domain administrator) in that domain without a password.

You can address this security concern using `SetMgmtServer /auth /on` to configure the DCE agent installation defaults before you remotely install DCE agents.

Related Topics:

- Configure DCE agent installation defaults
- User accounts for tools

Security audits

For security reasons, if is essential to be able to monitor activity, which occurs on the HPOM management server. If you can track use of or changes to the configuration of the HPOM management server and record attempts to gain unauthorized access to data, you have a good chance of being able to determine exactly who was responsible for any security breach and when.

The audit feature allows you to monitor how HPOM for Windows is used and what, if any, configuration changes are made. HPOM writes auditable events to its own, internal, custom event log, `%OvDataDir%\log\OvConfigChangeEvents.Evt` , which you can browse using the standard Windows Event Viewer.

In order to run an audit, you have to configure HPOM to monitor the underlying Windows services, which are used to control the various HPOM management areas. You can do this individually for each management area that you want to audit, or collectively if you want to audit all management areas concurrently. Once enabled, the security-audit feature allows you to track events in the following areas:

- Policy Management and Deployment

  To audit the management and deployment of policies, you need to monitor the Policy Management and Deployment Server (PMAD). Enabling auditing of the PMAD server logs activity in the following areas:

  - The deployment of policies to (and removal from) managed nodes
  - Any alteration to (and editing of) policies

    NOTE:
    The detail which the security audit feature collects is limited. For example, you can determine that version 1.2 of a policy was replaced with version 1.3, but you cannot use the security-audit feature to find out what the changes between the two policies are.

- HP Operations agent: Management and Deployment

  To audit the management and deployment of the HP Operations agent, you need to monitor the Policy Management and Deployment server (PMAD). Enabling auditing of the PMAD server logs activity in the following areas:

  - The deployment of the HP Operations agent to (and removal from) managed nodes

- HPOM Tools

  To audit the use of HPOM tools, you need to monitor the Message-action Server. Enabling auditing of the Message-action Server records activity in the following areas:

  - Any access to (and execution of) HPOM tools on the HPOM management server

- Automatic and Operator-initiated Actions

  To audit the execution of either automatic or operator-initiated actions, you need to monitor the Message-action Server. Enabling auditing of the Message-action Server logs activity in the following areas:

  - Each time an automatic or operator-initiated action is started or stopped on the HPOM management server

- HPOM Users and User Roles

  To audit the activity of HPOM users, such as the administrator and the operator, you need to monitor the Security Server. Enabling auditing of the Security Server logs activity in the following areas:

  - Any changes to HPOM user roles, rights, and permissions

- Messages

  To audit changes to messages, you need to enable monitoring of message changes done locally or on forwarded messages. Enabling message auditing logs activity in the following areas:

  - Each time the state of a message, the severity, the text, an annotation, or a custom message attribute changes, an audit log is created in the eventlog.
  - Audit logs are created each time messages are downloaded or deleted using DB Maintenance.

For each entry in the audit records, HPOM for Windows records the following information:

- The source of the event, for example: the Policy-Management server or the Message-action server
- An Event ID: a unique identifier for each event type, which makes searching the log much easier
- The date and time, at which the event occurred
- A User ID: a unique identifier to link an HPOM operator or administrator to the reported event

Related Topics:

- Log files and event sources
- Audit policy management and deployment
- Audit the message and action server
- Audit the security server
- Audit message changes
- Enabling and disabling security audits

Log files and event sources

If you make use of the security-audit feature, HPOM for Windows writes changes to the configuration of the HPOM for Windows management server to a custom application log named, `OvConfigChangeEvents.Evt` , which is stored in the following location: `%OvDataDir%\log` on the HPOM for Windows management server.

Three HPOM for Windows components are currently able to make use of the Security Audit feature, namely;

- Policy management and deployment server (OvPmad)

  The event source indicated in the Event Viewer for the policy management and deployment server is: `OvPolicyMgmt`

- Security Server (OvSecurityServer)

  The event source indicated in the event viewer for the security server is: `OvSecSvr`

- Message and action Server (OvEpMessageActionServer)

  The event source indicated in the event viewer for the message-action server is: `OvEpMsgActSvr`

- Local message changes

  The event source indicated in the event viewer for message changes done locally on the HPOM server is: `MessageChanges_local`

- Changes to forwarded messages

  The event source indicated in the event viewer for message changes done on another management server and forwarded to this management server is : `MessageChanges_forwarded`

In order for the Windows event service to be able to distinguish between the custom or the standard application log, the names used to identify event sources in the custom log used by the security-audit feature must be different to the names used to identify sources in the standard application log. For example, the name OvPolicyMgmt is used to identify the source of events associated with the PMAD sever in the security-audit log: the name HPOV-PMAD is used when PMAD writes entries to the standard application log.

NOTE:
The *standard* application log contains errors, warnings, and other useful information about Windows applications or services, for example; when they are shut down. The *security-audit* log contains entries relating strictly to events, which have passed or failed the HPOM or Windows audit.

Related Topics:

- Security audits
- Enabling and disabling security audits

Audit policy management and deployment

You enable auditing of policy management and deployment by monitoring the underlying Windows service, which controls policy management and deployment, namely: `OvPmad` .

When you enable auditing for policy management and deployment, the PMAD server writes entries to the custom application log `OvConfigChanges` for each change that is made to the policy-management configuration and, in addition, for each and every deployment job that is started, restarted, suspended, or canceled. The PMAD server also writes an entry to the custom application log for any operation that fails due to insufficient user rights, for example; when an operator tries to edit a policy without the necessary permissions.

Each entry in the custom application log concerning an audit event for policy management and deployment contains the following information in the event header:

- Date :
  The date when the event occurred, for example: "1/23/2004"
- Time :
  The time at which the event occurred, for example: "10:18:52 AM"
- Type :
  The type of event, for example: "Success Audit" for events concerning policy management and deployment jobs which succeed and "Failure Audit" for events relating to jobs that fail, for example; due to insufficient permissions.
- User :
  The name of the user who called the PMAD interface, for example: the user of the MMC console
- Hostname :
  the hostname of the HPOM for Windows management server on which the event was logged
- Source :
  This value is always set to "OvPolicyMgmt" for the HPOM for Windows PMAD Server.
- Category :
  This value is always set to "None"
- Event ID :
  A unique identifier for the logged event
- Description :
  A short description of the event, which has been logged, for example:

  ```
  (PMD393) A deployment job (ID '1012') to install version 9.0 of policy 'WINOSSPI-Ex60
  Exchange Application Warnings' (type 'Windows Event Log') on node 'YODA (management
  server)' has been added to the job queue.
  ```

Related Topics:

- Security audits
- Log files and event sources
- Enable and disable security audits

Audit the security server

You enable auditing of the Security Server by monitoring the underlying Windows service which controls the security server, namely: `OvSecurityServer` .

When you enable auditing for the Security Server, the security server writes entries to the custom application log `OvConfigChanges` for every change that is made to the configuration of a user role in the user-roles editor. Each entry in the custom application log concerning an audit event for the security server contains the following information in the event header:

- Date :
  The date when the event occurred, for example: "1/23/2004"
- Time :
  The time at which the event occurred, for example: "10:18:52 AM"
- Type :
  The type of event, for example success or failure: This value is always set to the value "Success Audit" for the HPOM for Windows Security Server.
- User :
  The name of the user who called the security-server interface, for example: the user of the MMC console
- Hostname :
  The hostname of the HPOM for Windows management server, on which the event was logged
- Source :
  This value is always set to "OvSecSvr" for the HPOM for Windows Security Server.
- Category :
  This value is always set to "None"
- Event ID :
  A unique identifier for the logged event
- Description :
  A short description of the event, which has been logged, for example:

  ```
  (SS74) Message group 'Default' of the role 'PSoft Admin' has been updated, flags enabled:
  'Own', 'Disown', 'Acknowledge', 'Unacknowledge', 'Change Severity', 'Change Text',
  'Launch Operator Initiated Command', 'Relaunch Automatic Command'.
  ```

Related Topics:

- Security audits
- Log files an event sources
- Enable and disable security audits

Audit the message and action server

You enable auditing of the message and action server by monitoring the underlying Windows service which controls the message and action server, namely: `OvEpMessageActionServer` .

When you enable auditing for the message and action server, the message and action server writes entries to the custom application log `OvConfigChanges` each time an automatic or operator-initiated action is started or stopped and, in addition, each time a tool is used by an HPOM administrator or operator.

Each entry in the custom application log concerning an audit event for the message-action server contains the following information in the event header:

- Date :
  The date when the event occurred, for example: "1/23/2004"
- Time :
  The time at which the event occurred, for example: "10:18:52 AM"
- Type :
  The type of event that is being logged. For the message-action server, this is always set to the value: "Success Audit"
- User :
  The name of the HPOM user who initiated the event, for example:
  - "<*domain* >\Administrator" for operator-initiated actions, where the HPOM user is known
  - "NT AUTHORITY\SYSTEM" for automatic actions, where the HPOM user is not known and the action is performed under the System account
- Hostname :
  the hostname of the HPOM for Windows management server on which the event was logged
- Source :
  This value is always set to "OvEpMsgActSvr" for the HPOM message and action server.
- Category :
  This value is always set to "None"
- Event ID :
  A unique identifier for the logged event
- Description :
  A short description of the event, which has been logged, for example:

  ```
  (AS103) Tool execution task sent to agent on node "yoda.test.dom (<$OPC_MGMTSV>)". The
  tool execution will be started as user $AGENT_USER. The output will be sent to display
  "yoda.test.dom:0.0". The call ID of this tool execution task is "bb97111f-fd5b-4641-9678-
  ```

`57b1923ece56". The command call is "SetMgmtServer.exe /system /forced".`

Exceptions and Restrictions

When you run an audit of the HPOM message and action server, note that the restrictions described in the following list apply to the type of events that can be logged:

- Automatic actions

  The message and action server does not either start or control automatic actions, which are configured to run on the managed node immediately after message generation, for example, when the target-node is defined as "<$*MSG_NODE_NAME* >" in the automatic action. This means that there is no way to log these automatic-action events as part of the audit of the Message-action Server.

  NOTE:
  Automatic actions that are set up to be run either on the HPOM management server (by using the setting <$*OPC_MGMTSV* > in the automatic action) or on a managed node other than the node which generates the message, *are* handled by the message and action server and, as a result, *are* logged in the custom event log used by the auditing feature.

- Operator-Initiated Actions

  Operator-initiated actions that are set up to be run on <$*OPC_GUI_CLIENT* > or on <$*OPC_GUI_CLIENT_WEB* > are executed directly from the GUI: they are not started or controlled by the message and action server, which means they do not appear in the audit log.

- Tools

  The message and action server does not start or control any tools, which run in the context of the HPOM console (as opposed to on the HPOM management server), even if the console is running on the HPOM management server. Tools which you configure to run in the context of the HPOM console are started and stopped directly from the GUI, which means that the start and stop events do not appear in the audit log.

Related Topics:

- Security audits
- Log files and event sources
- Enable and disable security audits

Audit message changes

Message change auditing makes it possible to audit changes made to messages. As a user in a regulated environment, it is important to you to be able to audit which changes to messages were made, when messages were changed, and by whom.

Message change auditing provides auditing capability for the following message changes:

- Message state change (such as own, disown, acknowledge, unacknowledge)
- Message severity change
- Message text change
- Message annotation change
- Message download using DB Maintenance (only a single audit message is written, not a message change event for every downloaded/deleted message)
- Custom message attribute change

Message change auditing does not audit the following message changes:

- New message creation or forwarded message arriving
- Message action state change (such as running, successful, failed)
- Message counter change

NOTE:
Action Execution ("Action started by user .." and "Action execution cancelled by user ..") itself is audited as part of the message and action server auditing. It is not necessary to audit the action state changes to messages that were caused by action execution.

Audit message changes for forwarded messages

Server-based flexible management allows you to forward messages between two management servers (HPOM for Windows and HPOM for UNIX), and also to forward message operations (such as acknowledge, own, and severity change) for forwarded messages. This means that messages are kept in sync between the management servers even when a message is changed on one of the servers.

It is possible to configure auditing for forwarded message changes independently of the auditing for local message changes. As with auditing for local message changes, auditing for forwarded messages is turned off by default. To get auditing for forwarded messages, you must enable it manually, in addition to enabling auditing for local message changes.

Audit DB maintenance

The DB Maintenance functionality of HPOM downloads acknowledged messages at specified times and deletes them from the database. This is a change to messages that need to be audited. However, because this is a mass update that may concern lots of messages, not every change to a message should be audited.

So for DB Maintenance, just one message change audit event is generated that explains how many messages have been touched by DB Maintenance.

For more information on DB Maintenance, see the help topics DB Maintenance Component details and Change StartTime DBMaint and DBMaintTimeSpec Settings .

Related Topics:

- Security audits
- Enable and disable security audits
- Server-based flexible management

Message change audit event log

When message change auditing is turned on, audit events related to message changes are written to the custom event log &quotOvConfigChanges."

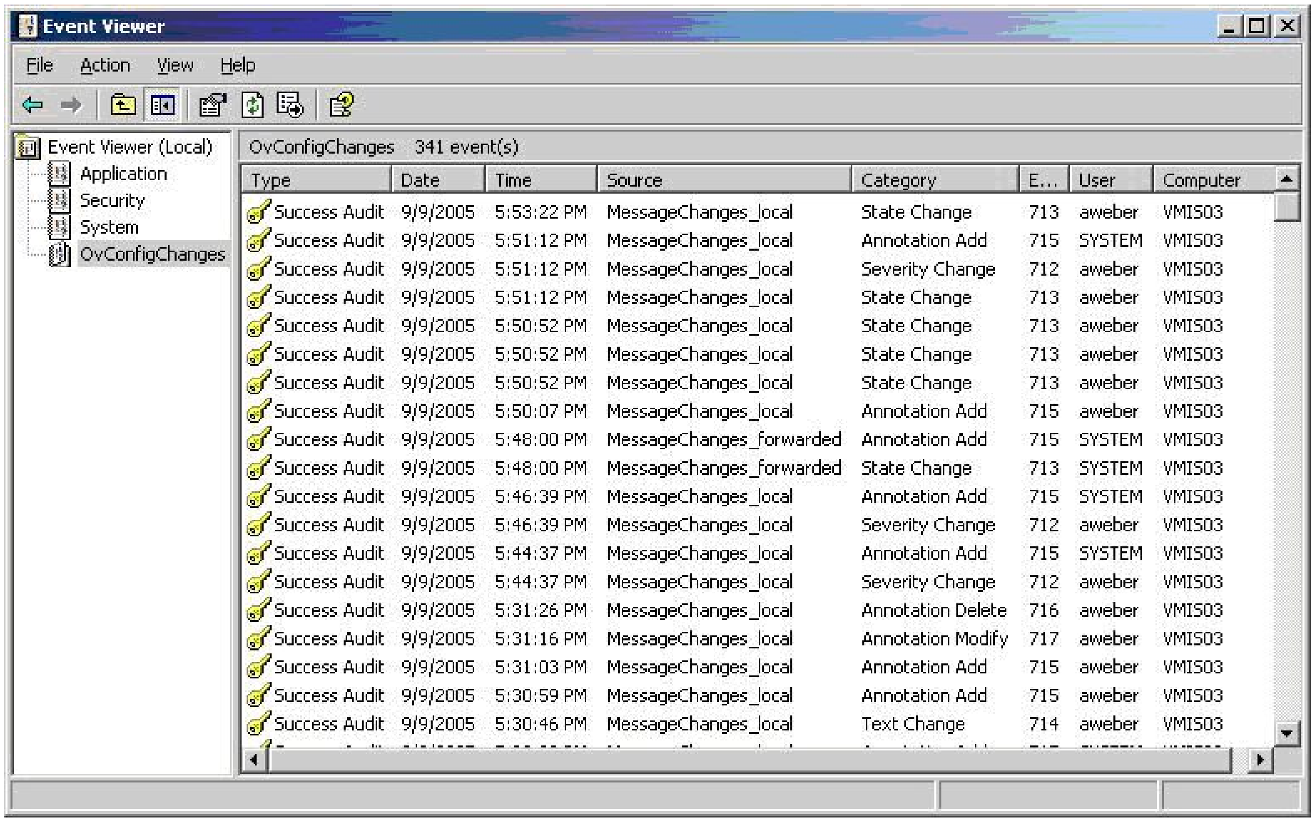
As administrator, you should be aware of the amount of auditing data that can be generated with message change auditing and should consider increasing the maximum log size for the OvConfigChanges event log. It is important to choose the right strategy for the automatic overwrite of the event log, because the default "Overwrite events as needed" may not be suitable for every customer environment. "Do not overwrite" might be better from an auditing point of view.

Certain specific registry settings can help in this area. See the following URL for details:

`http://support.microsoft.com/kb/312571/en-us`

Using this information, you can make sure that you do not lose events and that the event log size is still limited. When you use these settings, it is helpful to set up an HPOM event log policy that waits for event 524, described in http://support.microsoft.com/kb/312571/en-us.

Event 524 indicates that the event log has reached its maximum size. When this event occurs, the policy would move the event log file just backed up to another location, so that there is a complete history of all previous event logs (and auditlog entries) available. When message change auditing is turned on, message changes are visible in the event viewer, as shown below:



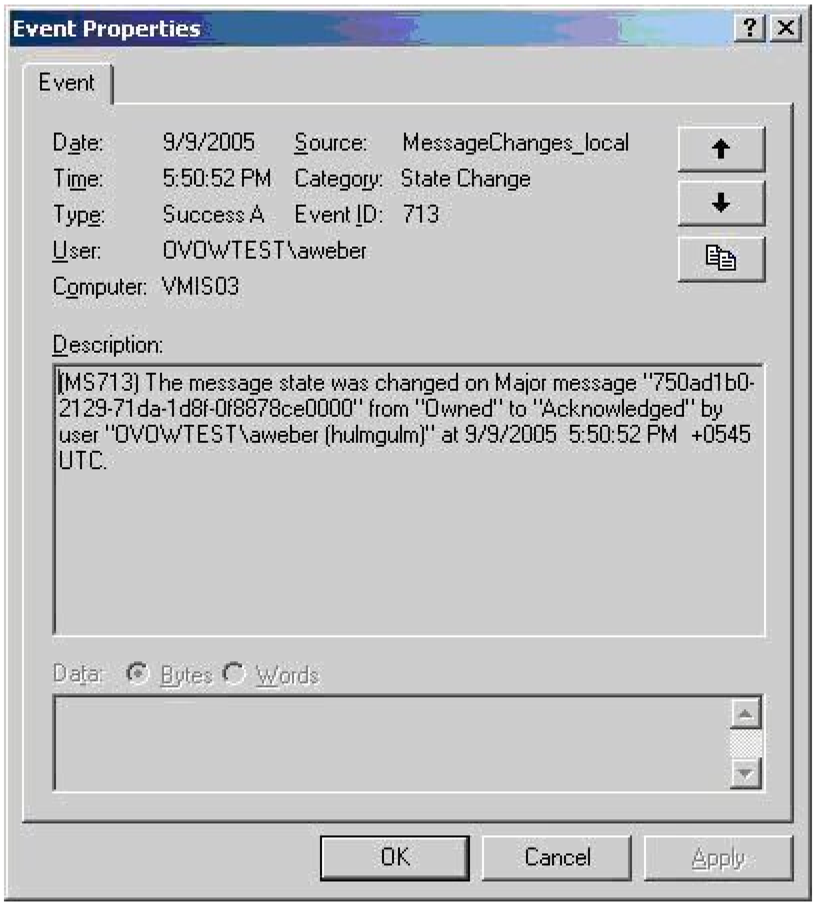
OvConfigChanges 341 event(s)

Type	Date	Time	Source	Category	E...	User	Computer
Success Audit	9/9/2005	5:53:22 PM	MessageChanges_local	State Change	713	aweber	VMIS03
Success Audit	9/9/2005	5:51:12 PM	MessageChanges_local	Annotation Add	715	SYSTEM	VMIS03
Success Audit	9/9/2005	5:51:12 PM	MessageChanges_local	Severity Change	712	aweber	VMIS03
Success Audit	9/9/2005	5:51:12 PM	MessageChanges_local	State Change	713	aweber	VMIS03
Success Audit	9/9/2005	5:50:52 PM	MessageChanges_local	State Change	713	aweber	VMIS03
Success Audit	9/9/2005	5:50:52 PM	MessageChanges_local	State Change	713	aweber	VMIS03
Success Audit	9/9/2005	5:50:52 PM	MessageChanges_local	State Change	713	aweber	VMIS03
Success Audit	9/9/2005	5:50:07 PM	MessageChanges_local	Annotation Add	715	aweber	VMIS03
Success Audit	9/9/2005	5:48:00 PM	MessageChanges_forwarded	Annotation Add	715	SYSTEM	VMIS03
Success Audit	9/9/2005	5:48:00 PM	MessageChanges_forwarded	State Change	713	SYSTEM	VMIS03
Success Audit	9/9/2005	5:46:39 PM	MessageChanges_local	Annotation Add	715	SYSTEM	VMIS03
Success Audit	9/9/2005	5:46:39 PM	MessageChanges_local	Severity Change	712	aweber	VMIS03
Success Audit	9/9/2005	5:44:37 PM	MessageChanges_local	Annotation Add	715	SYSTEM	VMIS03
Success Audit	9/9/2005	5:44:37 PM	MessageChanges_local	Severity Change	712	aweber	VMIS03
Success Audit	9/9/2005	5:31:26 PM	MessageChanges_local	Annotation Delete	716	aweber	VMIS03
Success Audit	9/9/2005	5:31:16 PM	MessageChanges_local	Annotation Modify	717	aweber	VMIS03
Success Audit	9/9/2005	5:31:03 PM	MessageChanges_local	Annotation Add	715	aweber	VMIS03
Success Audit	9/9/2005	5:30:59 PM	MessageChanges_local	Annotation Add	715	aweber	VMIS03
Success Audit	9/9/2005	5:30:46 PM	MessageChanges_local	Text Change	714	aweber	VMIS03

If you double-click an event from the list, the Event Properties dialog displays attributes of the specified event. See the help topic Message change audit event attributes for details.

Message change audit event attributes

The Event Properties dialog displays message change audit attributes, as shown below:



Date:	9/9/2005	Source:	MessageChanges_local
Time:	5:50:52 PM	Category:	State Change
Type:	Success A	Event ID:	713
User:	OVOWTEST\aweber		
Computer:	VMIS03		

Description:

(MS713) The message state was changed on Major message "750ad1b0-2129-71da-1d8f-0f8878ce0000" from "Owned" to "Acknowledged" by user "OVOWTEST\aweber (hulmgulm)" at 9/9/2005 5:50:52 PM +0545 UTC.

The table explains the message change audit event attributes:

Attribute	Description
Type	Is always "Success Audit."
Date, Time	Contains the date/time when the auditing event was logged. Due to race conditions, this may not be exactly the time when the message change was done in the local HPOM for Windows database; for forwarded message changes this may not be the time when the message change was originally made to the message on the sending server.
Source	Is "MessageChanges_local" or "MessageChanges_forwarded," depending on where the original message change was done. If changed locally, the source is "MessageChanges_local." If the message change was received from another management server, the source is "MessageChanges_forwarded."

Category	Can be one of the following, depending on the message change that was audited: ■ Custom Message Attributes Change ■ Severity Change ■ State Change ■ Text Change ■ Annotation Add ■ Annotation Delete ■ Annotation Modify ■ DBMaint execution (for English locale) The Category name is locale dependent and localized to Japanese, Korean, and Simplified Chinese.
Event	The identifier for the type of message change that was audited: ■ custom message attribute change, event = 711 ■ message severity change, event = 712 ■ message state change, event = 713 ■ message text change, event = 714 ■ message annotation add, event = 715 ■ message annotation delete, event = 716 ■ message annotation modified, event = 717 ■ DB Maintenance message download and deletion, event = 718 ■ DB Maintenance message deletion, event = 719
User	For local message changes, this is the local user who made the change. For forwarded message changes, this is the user account of the service that logged the auditing event; in most cases this will be the LOCAL SYSTEM account.
Computer	Always the host name of the local HPOM for Windows server.
Description	Depends on the type of message change that was audited, as described below: ■ For a custom message attribute change, the description is: (MS711) The custom message attributes were changed on $(MsgSeverity) message "$(MsgId)" to $(NewListOfCMAs) by user "$(UserName)" at $(DateTime). ■ For a message severity change, the description is: (MS712) The message severity was changed on message "$(MsgId)" from "$(OldSeverity)" to "$(NewSeverity)" by user "$(UserName)" at $(u). ■ For a message state change, the description is: (MS713) The message state was changed on $(MsgSeverity) message "$(MsgId)" from "$(OldState)" to "$(NewState)" by user "$(UserName)" at $(DateTime). ■ For a message text change, the description is: (MS714) The message text was changed on message "$(MsgId)" from "$(OldText)" to "$(NewText)" by user "$(UserName)" at $(DateTime). ■ For a message annotation add, the description is: (MS715) A message annotation was added to message "$(MsgId)" with the annotation text "$(NewAnnoText)" by user "$(UserName)" at $(DateTime).

- For a message annotation delete, the description is:

  (MS716) A message annotation was deleted from message "$(MsgId)" with the annotation text "$(OldAnnoText)" by user "$(UserName)" at $(DateTime)

- For a message annotation modified, the description is:

  (MS717) A message annotation was modified on message "$(MsgId)" from annotation text "$(OldAnnoText)" to "$(NewAnnoText)" by user "$(UserName)" at $(DateTime).

- For DB Maintenance message download and deletion, the description is:

  (MS718) DBMaint execution has downloaded $(NumberOfMessages) messages that have been acknowledged earlier than $(DateTime) to file "$(Path/FileName)." After the download, these messages have been deleted from the database.

- For DB Maintenance message deletion, the description is:

  (MS719) DBMaint execution has deleted $(NumberOfMessages) messages from the database that have been acknowledged earlier than $(DateTime).

Localization Information

The Description string is locale dependent and localized to Japanese, Korean, and Simplified Chinese, as follows:

$(DateTime) will be replaced by the date/time when the message change was done in the database on the local HPOM for Windows server. For forwarded message changes, this may not be the time when the message change was originally done to the message on the sending server.

$(DateTime) will be localized to the regional and language options of the HPOM for Windows server system (system settings, not the user settings, as done in the console).

For example, a system locale of English (United States) will look like this: `8/30/2005 7:43:49 PM +0200 UTC` . This also indicates that the time is regionalized to central European daylight savings time (UTC+2).

$(UserName) will be replaced by the user logon name, followed by the display name, as configured in the user configuration on the domain server, for example, "HPOMTest\Administrator (John Doe, HP USA)." The display name is only provided in $(UserName) if it can be resolved by asking the Domain server. To be able to resolve this name, the HPOM for Windows server must be a domain member of the user's domain. In the example above, the HPOM for Windows server must be member of the HPOMTEST domain to be able to resolve the user's display name.

NOTE:
The lookup of the display name is very expensive. For performance reasons this lookup is only done once and the lookup result is stored in a memory cache. A change of a user's display name on the domain server will not become visible in the HPOM for Windows message change auditing until the WMI service is restarted with the following command:

```
net stop winmgmt; net start winmgmt
```

Restrict message change auditing to higher severities

Some users need message change auditing, but only need to audit changes to messages with a higher severity. For example, you might need to audit all changes to messages with a severity of "Major" or "Critical," but not messages with a severity of "Minor" or below.

For severity changes events, all message changes are audited when the old severity is at least the configured severity, or when the new severity is at least the configured severity. In the example given above, a severity change of "Major" to a severity of "Normal" and a severity change of "Normal" to "Critical" would be audited. A severity change of "Minor" to "Normal" would not.

You can set the severity threshold when to audit message changes by setting the minimal message severity that should be audited.

To restrict message change auditing to higher severities

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog appears.
2. Click Namespaces , and then click Auditing . A list of values appears.
3. Set the value of Restrict message severities for message change auditing to the lowest severity of message that you want to audit. The default value is Normal , which means that all message changes are audited.
4. Click Apply .

Handle large annotations and message texts

Annotations can be very large (for example, file listings that have been created by automatic actions and are added to the message can be up to several MB). Automatically generated message texts can also be large.

In general, it is not helpful to fill up the auditing event log with these huge amounts of data. Additionally, the event log API enforces a limit of 32KB for each parameter that is passed to an event log text. A maximum of 32KB of an annotation can be written to the event log; anything over the maximum is truncated.

You can restrict the size threshold of how much of annotations and message texts should be audited.

To handle large annotations and message texts

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Auditing . A list of values appears.
3. Set the value of Restrict event size for message change auditing to the maximum number of bytes for message texts and annotations in the audit log.

   The maximum value is 32 KB; when annotations or message texts are audited and exceed this value, the text that exceeds this limit is truncated. The message change auditing feature indicates that an annotation or a message text has been cut by adding "..." to the end of the text.

   The default value is 32767, which means that the maximum of 32 KB text (the Windows Event Log API limit) is audited. Even if you configure a value above 32 KB, only 32 KB of text will be audited.
4. Click Apply , and then OK to save your changes and close the Server Configuration dialog box.

Enable and disable security audits

You can enable and disable security audits in HPOM *globally* by setting the value Turn on in general in the auditing namespace in the Server Configuration dialog box. Security audits are disabled by default and can only be enabled by setting Turn on in general to True .

NOTE:
Changing the value of Turn on in general is not sufficient by itself to enable or disable the security-auditing feature. You also have to restart the HPOM services that you want to audit, for example: OvPmad (policy management and deployment).

After you set Turn on in general to True and restart the services, you can change the value of Turn on at runtime to enable and disable auditing without having to restart the resources and services again. Auditable events are written to the HPOM custom event log. Normal events which cannot (or do not need to) be audited, such as application errors and warnings, are written to the standard Windows event log.

The auditing namespace also contains values for enabling and disabling each auditable event source, for example:

- Turn on action execution auditing
- Turn on agent certificate request handling auditing
- Turn on config change auditing
- Turn on forwarded message change auditing
- Turn on local message change auditing
- Turn on outage auditing
- Turn on policy management and deployment auditing
- Turn on user roles configuration auditing

You can enable or disable auditing of each event source at any time either individually or collectively.

To enable or disable auditing globally for the first time

The steps described in this procedure allow you to enable or disable auditing globally, that is, for all the HPOM components, which are able to write to the HPOM custom log for auditing.

CAUTION:
This procedure is not recommended for enabling or disabling auditing on an HPOM management server, which is running in a high-availability cluster. For more information, see Security audits in a high-

availability environment .

1. In the console tree, right-click **Operations Manager** , and then click **Configure→Server...** . The Server Configuration dialog opens.
2. Click **Namespaces** , and then click **Auditing** . A list of values appears.
3. Set the value of **Turn on in general** as follows:
   - Enable auditing: **True**
   - Disable auditing: **False**
4. Click **Apply** .
5. Restart the services associated with the event sources you want to audit. You can do this globally with the following two commands:

   ```
   c:>net stop winmgmt
   
   c:\>vpstat -3 -r
   ```

   Alternatively, you can restart services individually (or in a batch file) as follows:

   ```
   c:\>net stop WinMgmt
   c:\>net stop OvSecurityServer
   c:\>net start OvSecurityServer
   c:\>net start OvPmad
   c:\>net start OvEpStatusEngine
   c:\>net start OvOWReqCheckSrv
   c:\>net start OvAutoDiscovery
   ```

   NOTE:
   After you restart the services, you can change the value of **Turn on at runtime** to enable and disable auditing without having to restart the services again.
6. Start the Windows Event Viewer and, in the console tree, click the **OvConfigChanges** item.
7. You can now control auditing more quickly and accurately using the procedure below.

The sample VB script "SetAuditing.vbs" in the directory "examples\OvOW\Policy Management\scripts" can be used to globally enable or disable auditing. Call "cscript.exe SetAuditing.vbs /enable" and auditing will be enabled.

To manage auditing for individual event sources

This procedure allows you to enable or disable auditing at runtime for individual or multiple event sources, without having to restart any associated Windows services for the change to take effect.

NOTE:
The values that you modify in this procedure only take effect after auditing has been enabled globally for the first time, and the Windows services that you want to audit restarted.

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog opens.
2. Click Namespaces , and then click Auditing . A list of values appears.
3. Set the value of Turn on at runtime as follows:
   - Enable auditing: True
   - Disable auditing: False
4. Click Apply .
5. To enable or disable auditing for individual event sources:
   a. Set the value of Turn on at runtime to True .
   b. Set values for the individual event sources that you want to enable or disable, for example:
      - Turn on action execution auditing
      - Turn on agent certificate request handling auditing
      - Turn on config change auditing
      - Turn on forwarded message change auditing
      - Turn on local message change auditing
      - Turn on outage auditing
      - Turn on policy management and deployment auditing
      - Turn on user roles configuration auditing
   c. Set the values to True (enabled) or False (disabled), as required, and then click Apply .

Related Topics:

- Security audits
- Event sources in security audits
- Security audits in a high-availability environment

Security audits in a high-availability environment

If you want to enable and make use of the auditing feature on an HPOM management server that is installed in a high-availability cluster, you need to bear in mind the following special considerations.

- In a high-availability cluster, the custom-event logs which the audit feature uses to record activity on the HPOM for Windows management server are located on both cluster nodes in `%OvDataDir%\log` in the same way as a stand-alone HPOM management server: they are not located in `%OvShareDir%\log` . The cluster service synchronizes the custom event log at regular intervals between the active and backup nodes in the cluster, too.

  NOTE:
  Synchronization between active and backup cluster nodes can only take place if the custom-event log, `OvConfigChangeEvents.evt` , exists and is writable on the individual backup nodes in the cluster, for example; after a fail over.

- You only have to enable auditing once in the high-availability environment, on the *active* HPOM for Windows management server in the cluster. You change the values in the auditing namespace in Server Configuration dialog to change the registry keys that configure auditing. In the event of a fail over, the registry keys on the active HPOM management server are automatically replicated to the backup nodes in the cluster, along with their sub-keys and settings (on or off).
- The custom logs for audit events are constantly synchronized between the active cluster node where the HPOM for Windows management server is running, and the backup cluster node.
- Any changes that are subsequently made either manually or automatically to these registry keys (or their sub-keys) on the *active* cluster node (where the HPOM management server is running) are automatically replicated by the cluster service to the other nodes in the cluster in the event of a fail over.

  The registry keys are "attached" to a cluster resource called "OvOW Registry Replication". Whenever this resource is brought online on a cluster node, the MS Cluster Service overwrites any existing keys with the keys from the machine, where this resource was previously online.

For example, if you disable auditing for the Policy Management and Deployment component in the cluster by setting the value Turn on policy management and deployment auditing to False , this sets the registry key
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\OvConfigChanges\OvPolicyMgmt` , and the cluster service sets the same value on the other nodes in the cluster.

NOTE:
Registry changes are only replicated from the active to the backup HPOM in the high-availability cluster. If you make manual changes to the key settings management server in the registry of the *backup* HPOM management server in the high-availability cluster, these changes will be *not* be replicated to the *active* cluster node, where the HPOM management server is running. More importantly, the changes made to the registry on the backup cluster node are lost in the event of a fail over, when the cluster service starts

the HPOM management server on the backup cluster node and, in the process, creates a replica of the active cluster node's registry on the backup cluster node.

To enable or disable auditing globally in a high-availability cluster

You enable and disable security audits on an HPOM management server that is installed in a high-availability cluster in the same way you enable and disable security audits on a stand-alone HPOM management server, with one exception: the method you use to stop and restart Windows services.

⚠ CAUTION:
The steps described in this procedure must be carried out on the *active* HPOM management server in the high-availability cluster.

1. In the console tree on the *active* HPOM management server in the high-availability cluster, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog opens.
2. Click Namespaces , and then click Auditing . A list of values appears.
3. Set the value of Turn on in general as follows:
   - Enable auditing: True
   - Disable auditing: False
4. Click Apply .
5. On the *active* HPOM management server in the high-availability cluster, restart the services associated with the event sources you want to audit, as follows:
   a. In the Cluster Administrator, locate the cluster group for your HPOM management server installation, for example: HPOM
   b. Take the HPOM management server off line

      Right-click the cluster group for your HPOM management server installation and select the Take offline option from the menu which pops up. This command stops *all* the resources and Windows services associated with the HPOM management server, without provoking a fail over.
   c. Stop and restart the Windows-Management service (`WinMgmt`).

      At the command prompt, type:
      ```
      c:\>net stop WinMgmt
      c:\>net start WinMgmt
      ```
   d. Bring the HPOM management server back on line

      Next, right-click the cluster group for your HPOM management server installation once again and select the Bring online option from the menu which pops up. This command restarts *all* the resources and Windows services associated with the HPOM management server.

NOTE:
After you restart the services, you can change the value of Turn on at runtime to enable and disable auditing without having to restart the services again.

6. Start the Windows Event Viewer and, in the console tree, click the OvConfigChanges item.

To manage auditing for individual event sources

This procedure explains how to enable or disable auditing at runtime for individual or multiple event sources on the *active* HPOM management server in the high-availability cluster without having to restart any associated Windows services for the change to take effect.

NOTE:
The registry keys that you modify using the Server Configuration dialog do not exist on the backup nodes in the cluster until a fail over occurs and the registries on the cluster nodes are synchronized.

1. In the console tree on the *active* HPOM management server in the high-availability cluster, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog opens.
2. Click Namespaces , and then click Auditing . A list of values appears.
3. Set the value of Turn on at runtime as follows:
   - Enable auditing: True
   - Disable auditing: False
4. Click Apply .
5. To enable or disable auditing for individual event sources on the *active* HPOM management server in the high-availability cluster:
   a. Set the value of Turn on at runtime to True on the *active* HPOM management server
   b. Set values for the individual event sources that you want to enable or disable, for example:
      - Turn on action execution auditing
      - Turn on agent certificate request handling auditing
      - Turn on config change auditing
      - Turn on forwarded message change auditing
      - Turn on local message change auditing
      - Turn on outage auditing
      - Turn on policy management and deployment auditing
      - Turn on user roles configuration auditing

   c. Set the values to True (enabled) or False (disabled), as required, and then click Apply .

6. Start the Windows Event Viewer on the *active* HPOM management server in the high-availability cluster and, in the console tree, click the OvConfigChanges item.

Related Topics:

- Security audits
- Log files and event sources
- Enabling and disabling security audits

Auditing security

When you change values in the auditing namespace in the Server Configuration dialog, this sets the registry keys that configure auditing. Therefore, auditing can only be turned on or off by an administrator who has write access to the registry on the HPOM server.

In regulated environments (21CFR Part 11), special security requirements apply that can be fulfilled using the standard Windows EventLog security mechanisms shown below:

- Users cannot change or delete audit log entries. This is not possible in general for single eventlog entries. (You cannot change or delete a single event from a Windows eventlog).
- Access to the eventlog file can be restricted in Windows, so viewing the eventlog is not possible for non-admin users. Non-admin users also cannot delete eventlog files using the event viewer.
- Non-admin users can be restricted in Windows so they are not allowed to edit the registry; this means they cannot turn HPOM auditing on or off.

DCE RPC Communication without using the Endpoint Mapper

Allowing one or more well-known ports through a firewall is often considered as a security risk. Especially, allowing the well-known port of the DCE RPC endpoint mapper, 135, can be a risk.

Security in firewall environments can be significantly improved by reducing communication to a single, user-defined port. This section describes a solution where the DCE RPC endpoint mapper is not used, allowing you to close port 135 in the firewall, thus significantly increasing the security of your environment. RPC communication of HPOM for Windows will then only require one open destination port in each direction.

HTTP-based communication used for performance data or service discovery data is not affected by this change and requires additional, but user-defined ports, as outlined in the firewall white paper.

This applies only to communication with DCE agents. HTTPS agents use a different communication mechanism and different ports. See Configuring HTTPS communication through firewalls .

NOTE:
It is assumed that you are familiar with HP Operations Manager for Windows fundamentals, and are knowledgeable with agent-server communication.

NOTE:
Information contained within this section assumes that firewalls have been established in accordance with the HP Operations Manager for Windows Firewall Configuration White Paper.

Current HPOM Communication

With HPOM for Windows 7.x, communication between managed nodes and management servers is generally based on DCE RPC or Microsoft’s implementation of it: Microsoft RPC.

HP Operations services and processes acting as RPC servers register at the local DCE endpoint mapper (UNIX: rpcd or dced, Windows: RPC service) to publish their offered services. They specify a certain port or they are assigned a free port, through which they can be contacted.

HP Operations processes acting as RPC clients first contact the endpoint mapper on the destination node to find the registered server. The client is not initially aware of the port that the server is using and must request this information from the DCE endpoint mapper.

There are RPC servers and clients on both management server systems and managed node systems: RPC clients on the management server contact RPC servers on the nodes, and RPC clients on the nodes contact the RPC server on the management server. In addition, there is local DCE RPC communication on the HP Operations management server system and managed nodes, which means RPC clients contact RPC servers on the same system.

Related Topics:

- Concepts of DCE RPC communication without endpoint mapper
- Configuring the HP Operations management server system
- Configuring HP Operations managed node systems
- Server port specification File
- Configuration examples
- Verifying configuration
- Troubleshooting
- Variable reference
- Internal process handling

Concepts of DCE RPC Communication without Endpoint Mapper

The fundamental requirement is to enable HPOM to run in an environment where the port 135 can be closed on the firewall. Additionally, it would be helpful if the DCE RPC endpoint mapper could be disabled on the node itself, if no other applications are using it.

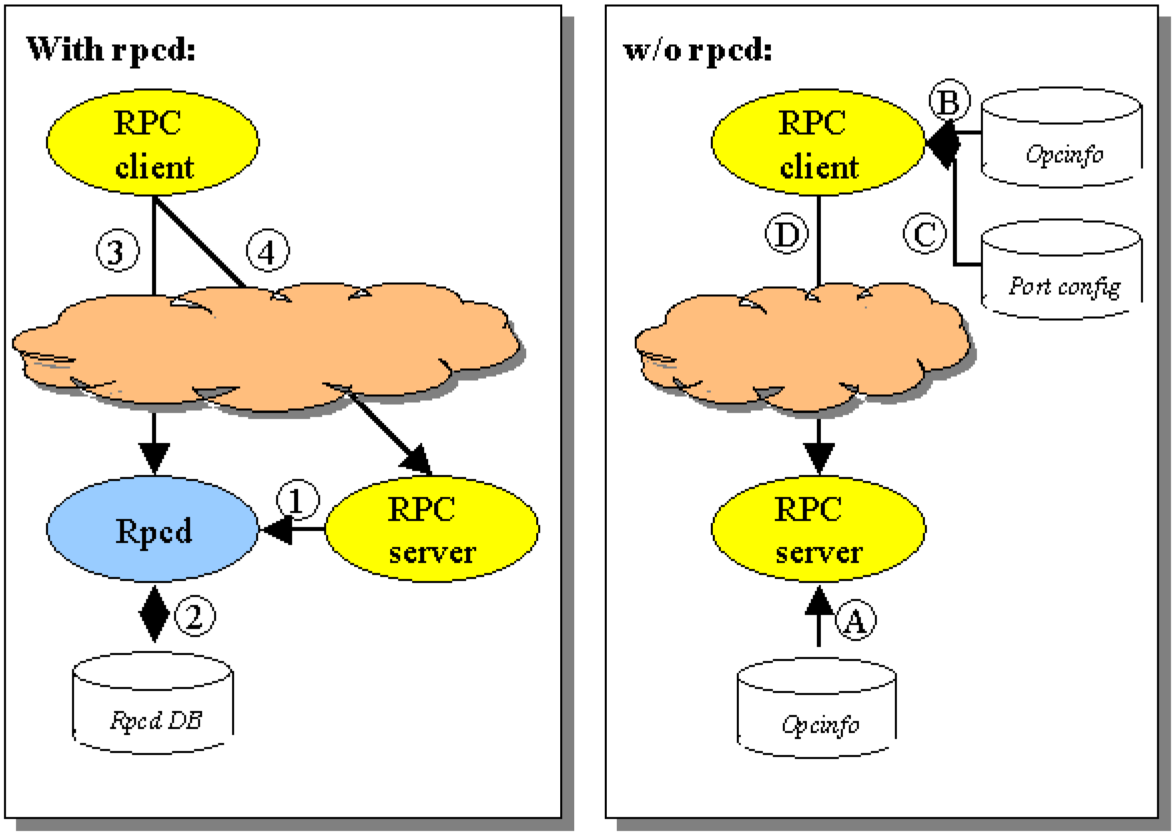
In particular, communication between the HP Operations management server and managed nodes must not use port 135.

This behavior can be achieved by setting certain variables in the opcinfo file on the HP Operations managed nodes and the registry on the HPOM for Windows management server.

To implement this, the HP Operations RPC servers must use specific ports, which can be configured by the customer. The RPC servers and clients will read these ports from configuration files and RPC clients will directly contact the HP Operations RPC servers using the specified ports, without using the DCE endpoint mapper.

NOTE:
Variables used in the registry have the same names as the ones used in the `opcinfo` file, except that the prefix `OPC_` is omitted. There are variable names such as `OPC_COMM_PORT_RANGE` throughout this document - the prefix *`OPC_`* in italics means that the variable used in the `opcinfo` file is called `OPC_COMM_PORT_RANGE` , whereas the variable used in the registry is called `COMM_PORT_RANGE` .

A comparison of both models is shown below:



Behavior with DCE RPC endpoint mapper

1. The RPC server starts up. It either uses the port specified in the variable *OPC_* `COMM_PORT_RANGE` or it will get a free port assigned. The RPC server registers itself with this port at the endpoint mapper.
2. The endpoint mapper stores this information in its database.
3. The RPC clients starts but does not know the port number used by the RPC server. It queries the endpoint mapper with the type of server it wants to contact and some additional interface specification uniquely identifying the destination server. The endpoint mapper returns the port number.
4. The RPC client can now contact the specified RPC server.

Behavior without DCE RPC endpoint mapper

1. The RPC server starts up. It uses the port specified in the variable *OPC_* `COMM_PORT_RANGE` . It does not register itself at the endpoint mapper (switched off using *OPC_* `COMM_REGISTER_RPC_SERVER FALSE`) and listens at this port (A).
2. The RPC client determines from its local configuration that the RPC server must be contacted without an endpoint mapper lookup (*OPC_* `COMM_LOOKUP_RPC_SRV FALSE`). It reads the port of the RPC server

either from a variable in the opcinfo file directly or reads it from a server port specification file specified in the variable *OPC_* `COMM_RPC_PORT_FILE “…”` (B).

3. The RPC client searches for the specified RPC server within the server port specification file, based on the server type and destination node. The file entry contains the port where the RPC server is supposed to be listening (C).
4. The RPC client now can contact the RPC server directly (D).

NOTE:
Mixed environments are also possible, where some RPC clients are using the endpoint mapper, and other clients do not. For example, RPC clients (=nodes) inside the firewall could use the RPC endpoint mapper lookup (because they do not have to cross the firewall), whereas RPC clients (=nodes) outside the firewall do not use the endpoint mapper. This can be controlled using the *OPC_* `COMM_LOOKUP_RPC_SRV` variable.

In such a mixed scenario it would be necessary to register the HP Operations management server RPC server at the endpoint mapper, so that the nodes inside the firewall can find the server.

This can be controlled using the *OPC_* `COMM_REGISTER_RPC_SERVER` variable.

Related Topics:

- Configuring the HP Operations management server System
- Configuring HP Operations managed node systems
- Server port specification File
- Configuration examples
- Verifying configuration
- Troubleshooting
- Variable reference
- Internal process handling

Configuring the HP Operations management server system

The following RPC communication from the HP Operations management server to the managed nodes is affected:

RPC Client	RPC Server	Direction	Explanation
`ovpmad`	`opcctla`	Mgmt Server → Mgd Node	Deployment of policies (started by management server).
`OvEpMsgActSrv`	`opcctla`	Mgmt Server → Mgd Node	Action requests, Control requests, Heartbeat polling, ...
`opcragt`	`opcctla`	Mgmt Server → Mgd Node	Start/ stop/ status and other opcragt requests.

On the management server, you can configure all settings using the Server Configuration dialog box.

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box appears.
2. Click Namespaces , and then click Message Action Server General . A list of values appears.
3. After you have changed a value, restart the OvEpMessageActionServer service for the change to take effect.

RPC Servers

On the HPOM for Windows management server system, the RPC service (endpoint mapper) must be running because it is required by the operating system and other applications. However, it is possible not to register the HP Operations RPC server at the endpoint mapper by setting the value DCE RPC server port registration to false.

Then the RPC clients on managed node systems must have a server port configuration, so that agents can contact the RPC server on the management server system without querying the endpoint mapper.

Configuration values for RPC server on the management server

Value	Type	Default	Explanation
DCE RPC server port registration	Boolean	True	Selects whether to register RPC interfaces with endpoint mapper. If set to false for the message action server, all managed nodes must have a dedicated server port configuration to be able to reach the HP Operations management server. If true, managed nodes can send messages in the usual way, but managed nodes with configured RPC server ports will use those and will not perform an endpoint mapper lookup.
DCE RPC server port	Integer	0	Specifies exactly one port to be used by the RPC server. The value 0 means that the DCE runtime will assign a port.

The Message and Action server implements two interfaces, the RPC interface of the Message Receiver and the interface of the Distribution Manager, which is actually a link to the message receiver so that HPOM for Windows provides the same interfaces as HPOM for UNIX. Therefore the value of DCE RPC server port applies to both of these interfaces.

RPC Clients

The RPC clients ovpmad and `OvEpMsgActSrv` on the management server system share the same settings.

Configuration values for RPC clients on the management server

Value	Type	Default	Explanation
DCE RPC server port specification file	String	none	Complete path pointing to a server port specification file as described below. Do NOT use " " when specifying the path. Example: `c:\Program Files\ports`
DCE RPC server port lookup	Boolean	True	Whether to perform an endpoint mapper lookup if no matching port for a managed node is found in the server port specification file.

Related Topics:

- Concepts of DCE RPC communication without endpoint mapper
- Configuring HP Operations managed node systems
- Server port specification file
- Configuration examples
- Verifying configuration
- Troubleshooting
- Variable reference
- Internal process handling

Configuring HP Operations managed node systems

The following RPC communication from managed nodes to the HP Operations management server is affected:

RPC Client	RPC Server	Direction	Explanation
opcmsga	OvEpMsgActSrv	Mgd Node → Mgmt Server	HP Operations messages and action responses.
opcdista (UNIX) opcctla (Windows)	OvEpMsgActSrv	Mgmt Server → Mgd Node	Distribution agent asks server if new distribution data is available (only used with HPOM for UNIX, not needed for HPOM for Windows). Can be switched off using the `OPC_NO_CFG_RQST_AT_STARTUP TRUE` variable setting.

Setting of Variables by Process

Most parameters for communication without the endpoint mapper on the managed node are configured in the opcinfo file. It is sometimes very important to apply settings to selected processes only, using the following syntax:

An entry `OPC_RESTRICT_TO_PROCS` starts a section that only applies to the specified process. All following entries are only evaluated for this one process. A second `OPC_RESTRICT_TO_PROCS` entry starts a section that only applies to the next specified process.

Specify all entries that should apply to all processes before the first occurrence of a `OPC_RESTRICT_TO_PROCS` section.

Example

```
OPC_COMM_RPC_PORT_FILE         /tmp/port_conf.txt

OPC_RESTRICT_TO_PROCS          opcctla

OPC_COMM_REGISTER_PORT_RANGE  5001
```

In this case, the specified port specification file is valid for all processes, while the port setting is only valid for the control agent, `opcctla`.

RPC Servers - opcinfo File Variables for RPC Servers on Managed Node Systems

Key	Type	Default	Explanation
`OPC_COMM_REGISTER_RPC_SRV`	Boolean	TRUE	Selects whether to register RPC server interfaces with endpoint mapper.
`OPC_COMM_PORT_RANGE`	string	empty	Specifies exactly one port to be used by the RPC server. Must be set for the control agent.

NOTE:
Make sure that the configured `OPC_COMM_PORT_RANGE` for the control agent (`opcctla`) contains only one port and that no other process uses this port.

RPC Clients - opcinfo File Variables for RPC Clients on Managed Nodes

Key	Type	Default	Explanation
`OPC_COMM_PORT_MSGR`	Int	0	Specifies at which port the message interface of the Message Action Server (`OvEpMsgActSrv`) is listening on the management server. For the HP Operations management servers, this must be the same value as used for `OPC_COMM_PORT_DISTM` .
`OPC_COMM_PORT_DISTM`	Int	0	Specifies at which port the deployment interface of the Message Action Server (`OvEpMsgActSrv`)is listening to configuration requests from the agent. For the HP Operations management servers, this must be same value as used for `OPC_COMM_PORT_MSGR` . This setting is needed if the `opcinfo` file of the agent does not contain the variable setting: `OPC_NO_CFG_RQST_AT_STARTUP TRUE` .
`OPC_COMM_RPC_PORT_FILE`	String	empty	Complete path pointing to a server port specification file as described below.
`OPC_COMM_LOOKUP_RPC_SRV`	Boolean	TRUE	Whether to perform an endpoint mapper lookup if no matching port is found in the server port specification file.

The settings `OPC_COMM_PORT_MSGR` and `OPC_COMM_PORT_DISTM` can be used if only one management server will be contacted, or all management servers are configured to use the same RPC server ports.

If multiple management servers with different port settings are used, a server port specification file must be configured using the `OPC_COMM_RPC_PORT_FILE` variable.

The server port specification file, if configured, is evaluated first. If not, the two additional `opcinfo` values (`OPC_COMM_PORT_MSGR` and `OPC_COMM_PORT_DISTM`) are evaluated. If these have a value of 0, they are considered as not set. See also Internal Process Handling .

An example of the server port specification file on a managed node is shown below:

```
#
# SelectionCriteria SrvType    Port Node
# --------------------------------------
NODE_NAME           opcmsgrd   5000 primaryserver.hp.com
NODE_NAME           opcdistm   5000 primaryserver.hp.com
NODE_NAME           opcmsgrd   6000 backupserver.hp.com
NODE_NAME           opcdistm   6001 backupserver.hp.com
```

somename.hp.com matches <*>somename.hp.com<*>
^somename.hp.com matches$ exactly somename.hp.com matches

NOTE:
The keywords used in the `SrvType` column refer to processes of the HPOM for UNIX management server. These keywords are also used with the HPOM for Windows management server, even though the corresponding interfaces or roles are implemented in one single process in HPOM for Windows (in the message action server).

Related Topics:

- Concepts of DCE RPC Communication without Endpoint Mapper
- Configuring the HP Operations management server system
- Server port specification file
- Configuration examples
- Verifying configuration
- Troubleshooting
- Variable reference
- Internal process handling

Server Port Specification File

Standard HPOM patterns can be used. A description of all available patterns is available elsewhere in the HPOM for Windows online help.

NOTE:
Patterns without anchoring match values may be prefixed or suffixed by anything.

The RPC client reads the file before opening a connection to an RPC server and tries to find a matching pattern. The first match completes the operation. If no match is found (or the file does not exist or has not been configured), the variable `OPC_ COMM_LOOKUP_RPC_SRV` decides whether to perform an endpoint mapper lookup. If an endpoint mapper lookup is not performed or it fails, the communication failure is handled in the usual way.

A configured port value of 0 is the same as if no matching entry is found and causes the RPC client to perform a regular endpoint mapper lookup (unless disabled entirely).

This can be used in a similar way to HPOM suppress conditions to specify an entry at the very beginning to filter out all nodes (by pattern) that still have an endpoint mapper running.

For all other nodes that do not match this suppress condition, the RPC client continues to search for a match in the remaining entries of the file.

File Syntax

- Empty lines are ignored.
- Comments start with the character # " but it must be the very first character. A line containing configuration data must not have trailing comments.
- Specify configuration data using 4 standard elements, separated with white spaces:
  - SelectionCriteria
    NODE_NAME - Node name pattern or exact match (case sensitive)
    NODE_ADDRESS - IP Addresses pattern or exact match (case sensitive)
  - SrvType
    opcctla - contacting the agent
    opcmsgrd - contacting the management server (message interface)
    opcdistm - contacting the management server (distribution interface)
  - Port - Port number to contact this RPC server

- Node - Node name or address pattern for this rule

An example of the server port specification file on the management server is shown below:

```
#
# SelectionCriteriaSrvType   Port   Node
# ------------------------- ----------
-                   --        -
NODE_NAME           opcctla   12345 ‹*›.hp.com
NODE_ADDRESS        opcctla   12346 15.136.‹*›
NODE_ADDRESS        opcctla   12347 ^192.‹1 -lt ‹#› -lt 10›.‹*›
NODE_ADDRESS        opcctla   12347 <*>1.2.3.4<*> for conditions in templates; ^1.2.3.4$ to
                                    configure 1.2.3.4.
```

NOTE:
If the caret (^) is used as the first character in a pattern, only expressions discovered at the beginning of lines are matched. For example, "^ab" matches the string "ab" in the line "abcde", but not in the line "xacde".

If the dollar sign is used as the last character of a pattern, only expressions at the end of lines are matched. For example, "de$" matches "de" in the line "abcde", but not in the line "abcdex".

File Modification Test

The RPC client checks the server port specification file. It re-loads it if it has been changed, indicated by a different size or modification time, before opening a connection to an RPC server.

Name/Address Conversion

It is possible to specify either node name patterns or IP address patterns in the port specification file. Internally, the RPC clients typically use the IP address of the destination server as node identification. So the client will try to find a match to the internally used IP address in the port specification file. If the file contains name patterns, then first a name resolution has to be done for the internally used IP address (using gethostbyaddr) and then the returned list of node names and aliases will be compared with the name pattern of the port specification file. Therefore it is favorable to specify IP address patterns in the server port specification file to avoid executing the resolution step. Entries in the port specification file will never be resolved because they are always considered as patterns and are not necessarily complete hostnames or IP addresses.

File protection

It is recommended that the port specification file is protected by applying the appropriate operating system access restrictions. The file will only be read by the HP Operations services and processes and the most restrictive permission settings would be the following:

UNIX

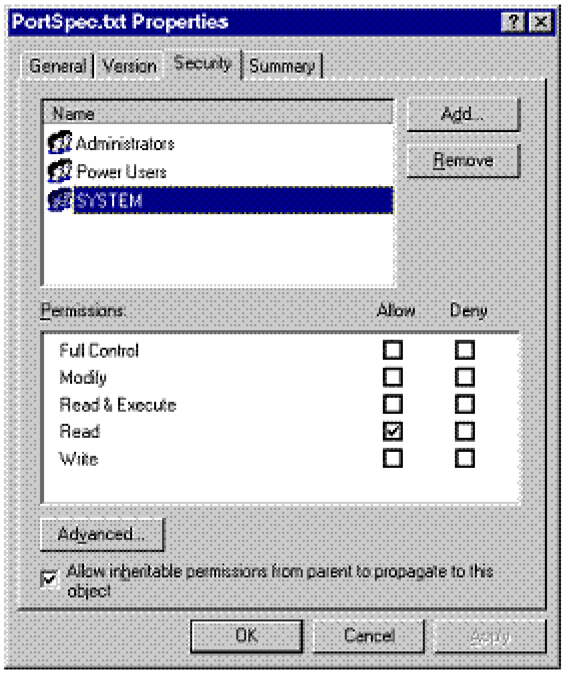
If, for example, the HP Operations agent on UNIX runs as user root, the most restrictive permission setting would be:

`-r-------- 1 root sys` *‹file›*

In the case that the HP Operations agent is not run under the user root, the file owner should be appropriately set for that user.

Windows

Allow Read for the SYSTEM account (or the HP-ITO-Account in case the Windows agent runs as HP-ITO-Account) as shown below:


PortSpec.txt Properties

General | Version | Security | Summary

Name: Administrators, Power Users, SYSTEM (selected)

Permissions	Allow	Deny
Full Control	☐	☐
Modify	☐	☐
Read & Execute	☐	☐
Read	☑	☐
Write	☐	☐

☑ Allow inheritable permissions from parent to propagate to this object

The location of the file can be defined as needed, but it is recommended to put it into a static HP Operations configuration directory and give the file an appropriately descriptive name.

Related Topics:

- Concepts of DCE RPC communication without endpoint mapper
- Configuring the HP Operations management server system
- Configuring HP Operations managed node Systems
- Configuration examples
- Verifying configuration
- Troubleshooting

- Variable reference
- Internal process handling

Configuration Examples

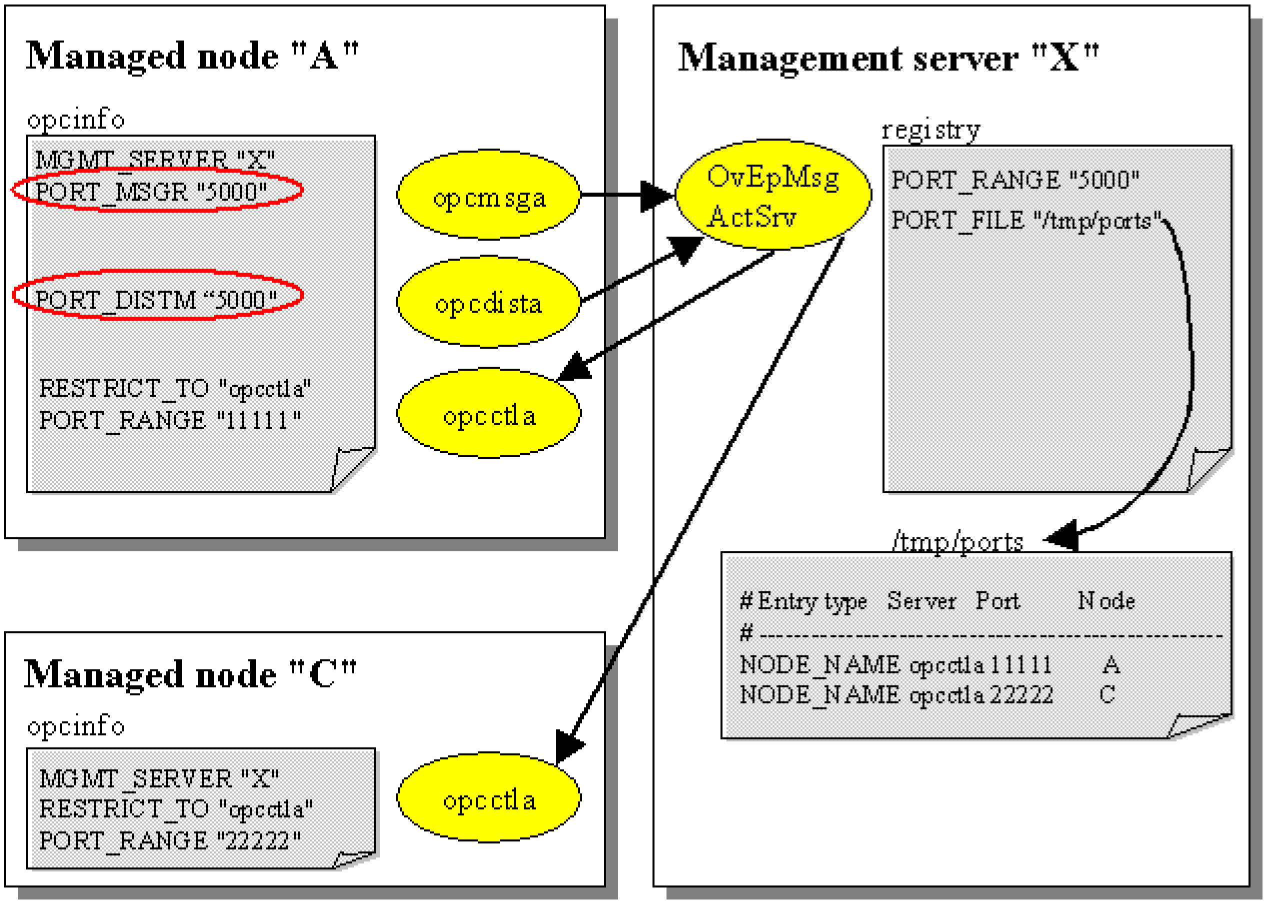
NOTE:
Keys used in the examples below are abbreviations, for example `PORT_MSGR` on the managed node stands for `OPC_COMM_PORT_MSGR` . On the management server, you can set the necessary registry keys using the Server Configuration dialog. See the Variable Reference for the exact names.

Example 1

In this example the control agents of nodes A and C do not register with the DCE endpoint mapper, therefore management server X uses entries in the port specification file to contact the control agents.

On the managed nodes, node A reads the port of its management server from the variables `PORT_MSGR` and `PORT_DISTM` in the opcinfo file. Node B still uses the endpoint mapper lookup when contacting its management server, as it has not been configured otherwise. Therefore the management server has to register its RPC server at the endpoint mapper (which is the default).

RPC clients perform the following steps when connecting to an RPC server. A key to the diagram can be found below.

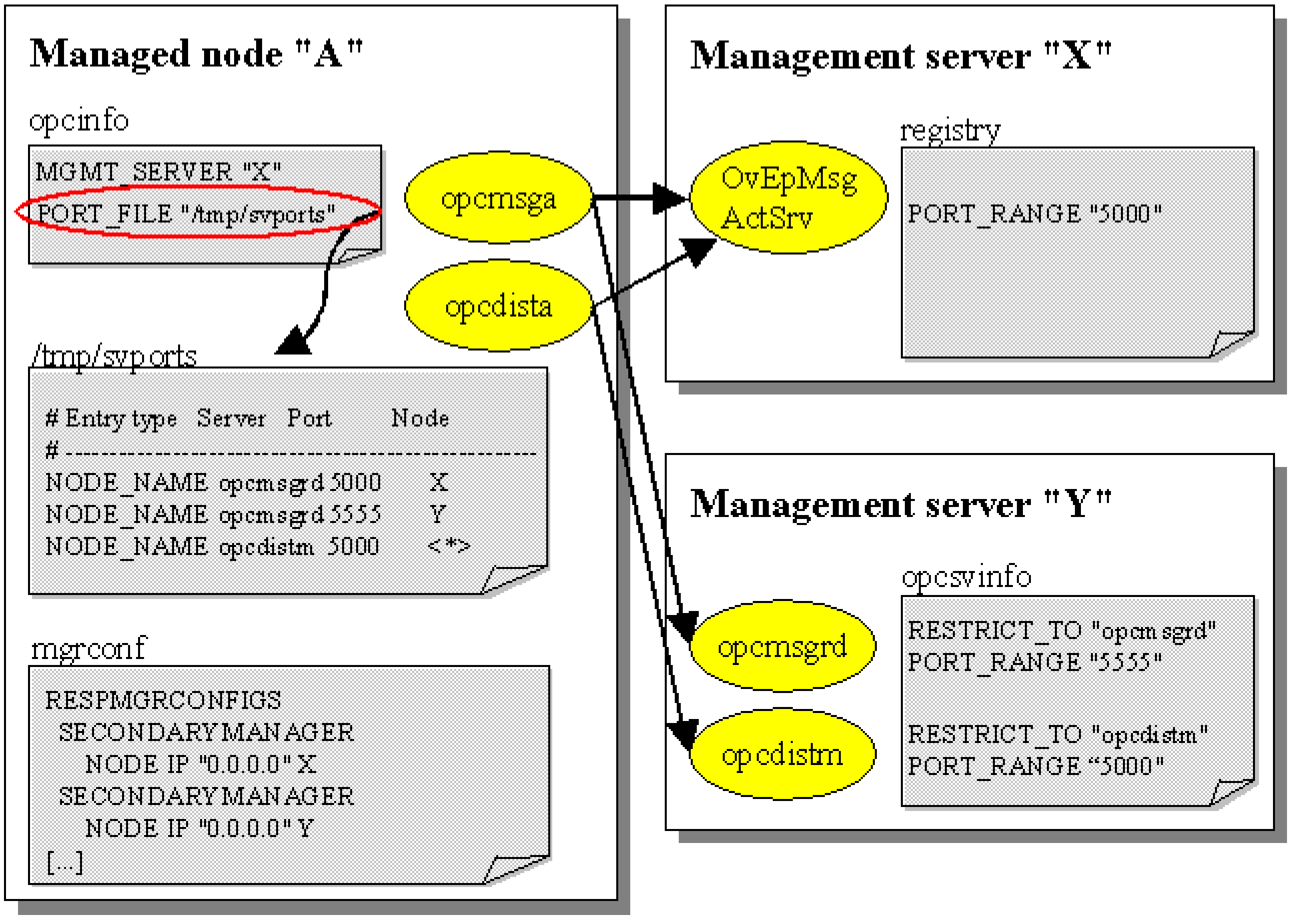


Example 2

In this flexible management scenario with two servers "X" and "Y", the RPC clients on the node A read the ports necessary to contact both management servers from the port specification file. The ports are different for management servers "X" and "Y".

"Y" is an HPOM for UNIX management server and uses the opcinfo file to specify the ports used by the server processes.

"X" is an HPOM for Windows management server and uses the registry to specify the port used by the Message and Action server process.



Related Topics:

- Concepts of DCE RPC Communication without Endpoint Mapper
- Configuring the HP Operations management server System
- Configuring HP Operations Managed Node Systems
- Server Port Specification File
- Verifying Configuration
- Troubleshooting
- Variable Reference
- Internal Process Handling

Verifying Configuration

To verify that a configuration is correct, check the following:

1. Start all HP Operations services and processes
2. Use `opcrpccp` to verify that registration of the HP Operations RPC Servers on both the managed node and the management server is correct. Make sure the correct ports are used as configured. If there is no endpoint mapper running on the destination system, this command will fail entirely (which is correct).
3. Verify that all processes are running properly using `vpstat` and `opcagt` .
4. Test Server to Agent communication by starting a tool from the HPOM for Windows tools folder on the destination managed node.
5. Test Agent to Server communication by sending test messages using opcmsg or any other mechanism generating an HP Operations message to be sent to the management server. Optionally enable tracing (Area ALL, DEBUG and debug area COMM, CONF) and check the trace file.
6. Test configuration distribution (policies and instrumentation).
7. Wait for heartbeat-polling cycles to expire - no errors should be reported. If the HP Operations Agent is stopped, the associated heartbeat-polling errors should be displayed.
8. Everything should behave as usual. There should be no communication related error messages in the HPOM for Windows GUI originating from the managed node.
9. Check the `opcerror` log files on the managed nodes for applicable entries. The utility opcragt, which was introduced with patch OVOW_00095, can also be used in environments as described in this paper. Opcragt is an excellent tool for testing the server-to-agent communication.

   NOTE:
   The utility opcragt, which was introduced with patch OVOW_00095, can also be used in environments as described in this paper. Opcragt is an excellent tool for testing the server-to-agent communication.

Related Topics:

- Concepts of DCE RPC communication without endpoint mapper
- Configuring the HP Operations management server system
- Configuring HP Operations managed node systems

- Server port specification File
- Configuration examples
- Troubleshooting
- Variable reference
- Internal process handling

Troubleshooting

Errors

The following errors, if reported after an operation, can indicate the reason of the error:

`Cannot connect to RPC service at system 'ncadg_ip_udp:15.136.120.88[22222]'. Local port configuration has been consulted - rpcd/llbd on remote system not queried. (OpC20-186)`

Communication errors always contain this form of message, when a local port configuration is used by an RPC client to connect the server. If this text is NOT part of the error message, a regular endpoint mapper lookup has been performed. The stated RPC binding contains the used protocol sequence (TCP or UDP) as well as the destination IP address and the port where the RPC server had been expected.

`Cannot find port for RPC service 'opcctla' at '15.136.120.88' in local configuration. (OpC20-187)`

No port configuration was found for the specified service (either in the server port specification file, `opcinfo` or anywhere else) and `OPC_COMM_LOOKUP_RPC_SRV` is false, that is NO endpoint mapper lookup is performed.

`The port configuration file /xxxx contains an invalid entry: '... some syntax error ...'. (OpC20-174)`

The file `/xxxx` contains bad entries. The value of `OPC_COMM_LOOKUP_RPC_SRV` is irrelevant and NO endpoint mapper lookup has been performed.

If no local port configuration for a destination service is found and `OPC_COMM_LOOKUP_RPC_SRV` is true (default), the traditional behavior of looking up the destination RPC server at the endpoint mapper applies. Then use tracing for further troubleshooting.

NOTE:
It may now take much longer for RPC calls to time out if the called RPC server is not listening - especially over UDP. The RPC client must now attempt to find a server, whereas before, the endpoint mapper was usually running and immediately returned the information about the requested RPC server and whether it was registered or not.

Tracing On Managed Nodes

The new and modified functionality contains trace and debug statements. Particularly the `DEBUG` areas `CONF` and COMM are of interest since they cover the evaluation of configurable variables and the DCE communication. The DEBUG area FILE might be interesting to track the detection of a modified server port specification file.

To enable tracing for these areas, set the following variables in the opcinfo file:

```
OPC_TRACE TRUE

OPC_TRACE_AREA ALL,DEBUG

OPC_DBG_AREA CONF,COMM
```

It is also recommended that you restrict the trace to certain processes by using the OPC_TRC_PROCS and OPC_DBG_PROCS variables and the process names, such as:

```
OPC_TRC_PROCS opcmsga

OPC_DBG_PROCS opcmsga
```

NOTE:
opcrpccp show mapping does not show RPC servers for which the OPC_COMM_REGISTER_RPC_SRV setting is FALSE . In addition, this program will fail altogether if the rpcd/dced is not running on the destination node.

RPC Servers log the following type of information, if endpoint mapper registration is enabled:

```
... opcctla(...)[DEBUG]: COMM: Register manager
... opcctla(...)[DEBUG]: CONF: Returning value 'TRUE' for key 'OPC_COMM_REGISTER_RPC_SRV'
... opcctla(...)[DEBUG]: COMM: Checking hostname '' for replacement
... opcctla(...)[INIT]: Regarding the setting of OPC_IP_ADDRESS
... opcctla(...)[DEBUG]: CONF: No value found for key 'OPC_IP_ADDRESS'
... opcctla(...)[DEBUG]: COMM: Using '15.139.88.156' as local address
... opcctla(...)[DEBUG]: COMM: Returning '15.139.88.156'
... opcctla(...)[DEBUG]: COMM: Lookup Srv
... opcctla(...)[DEBUG]: COMM: Server lookup using rpcd interface
... opcctla(...)[DEBUG]: COMM: Element lookup initialized
... opcctla(...)[DEBUG]: CONF: Returning value '13' for key 'OPC_MAX_PORT_RETRIES'
... opcctla(...)[DEBUG]: COMM: Got another element
... opcctla(...)[DEBUG]: COMM: Srv lookup using rpcd done. NumSrv = 0. rc = 0.
... opcctla(...)[DEBUG]: COMM: Register manager
... opcctla(...)[DEBUG]: COMM: rpc_ep_register for binding '0' successful
... opcctla(...)[DEBUG]: COMM: rpc_ep_register for binding '1' successful
[...]
... opcctla(...)[INT]: Entering RPC server loop ...
```

RPC Servers log the following type of information, if endpoint mapper registration is disabled:

```
... opcctla(...)[DEBUG]: COMM: Register manager
... opcctla(...)[DEBUG]: CONF: Returning value 'FALSE' for key 'OPC_COMM_REGISTER_RPC_SRV'
... opcctla(...)[DEBUG]: COMM: Register manager
... opcctla(...)[DEBUG]: COMM: Register manager
... opcctla(...)[DEBUG]: CONF: Returning value 'FALSE' for key 'OPC_COMM_REGISTER_RPC_SRV'
[...] ... opcctla(...)[DEBUG]: COMM: Entering RPC main loop ...
```

RPC clients on the managed node log the following type of information:

```
... opcmsga(...)[INIT]: Connecting message receiver on 260790428 ...
... opcmsga(...)[DEBUG]: COMM: Connecting with address: 15.139.88.156
... opcmsga(...)[DEBUG]: COMM: Getting server port for: opcmsgrd on host: '15.139.88.156'
... opcmsga(...)[DEBUG]: CONF: Returning value '/tmp/ports.tge' for key
'OPC_COMM_RPC_PORT_FILE'
... opcmsga(...)[DEBUG]: COMM: Examining external client port file /tmp/ports.tge ...
... opcmsga(...)[DEBUG]: COMM: Re-loading external client port file ...
... opcmsga(...)[DEBUG]: COMM: Activating external client port file. 0 entries
... opcmsga(...)[DEBUG]: COMM: Searching server port for: opcmsgrd at '15.139.88.156'
... opcmsga(...)[DEBUG]: CONF: Returning value '51528' for key 'OPC_COMM_PORT_MSGR'
... opcmsga(...)[DEBUG]: COMM: Got opcmsgrd server port from
opc/nodeinfo[OPC_COMM_PORT_MSGR]: 51528
... opcmsga(...)[DEBUG]: COMM: Checking hostname '15.139.88.156' for replacement
... opcmsga(...)[DEBUG]: COMM: Returning '15.139.88.156'
... opcmsga(...)[DEBUG]: COMM: Connection to non-local node. Using long timeout
... opcmsga(...)[DEBUG]: COMM: Checking server. Mgr type: 0x0
... opcmsga(...)[DEBUG]: COMM: Binding: ncadg_ip_udp:15.139.88.156[51528]
... opcmsga(...)[DEBUG]: CONF: Returning value '13' for key 'OPC_MAX_PORT_RETRIES'
... opcmsga(...)[DEBUG]: COMM: Checking whether server is listening ...
... opcmsga(...)[DEBUG]: COMM: Checking server: succeeded. st=0 rpc_rc=1
```

Related Topics:

- Concepts of DCE RPC communication without endpoint mapper
- Configuring the HP Operations management server system
- Configuring HP Operations managed node systems
- Server port specification file
- Configuration examples
- Verifying configuration
- Variable reference
- Internal process handling

Variable Reference

Management server configuration values

On the management server, you can configure all settings using the Server Configuration dialog box.

1. In the console tree, right-click **Operations Manager**, and then click **Configure→Server...**. The Server Configuration dialog box appears.
2. Click **Namespaces**, and then click **Message Action Server General**. A list of values appears.
3. After you have changed a value, restart the OvEpMessageActionServer service for the change to take effect.

Value	Type	Default	Explanation
DCE RPC server port registration	Boolean	True	Selects whether to register RPC interfaces with endpoint mapper. If set to false for the message action server, all managed nodes must have a dedicated server port configuration to be able to reach the HPOM for Windows management server. If true, managed nodes can send messages in the usual way, but managed nodes with configured RPC server ports will use those and will not perform an endpoint mapper lookup.
DCE RPC server port	Integer	0	Specifies exactly one port to be used by the RPC server. The value 0 means that the DCE runtime will assign a port.
DCE RPC server port specification file	String	none	Complete path pointing to a server port specification file as described below. Do NOT use " " when specifying the path. Example: `c:\Program Files\ports`
DCE RPC server port lookup	Boolean	True	Whether to perform an endpoint mapper lookup if no matching port for a managed node is found in the server port specification file.

Managed Node Registry Settings

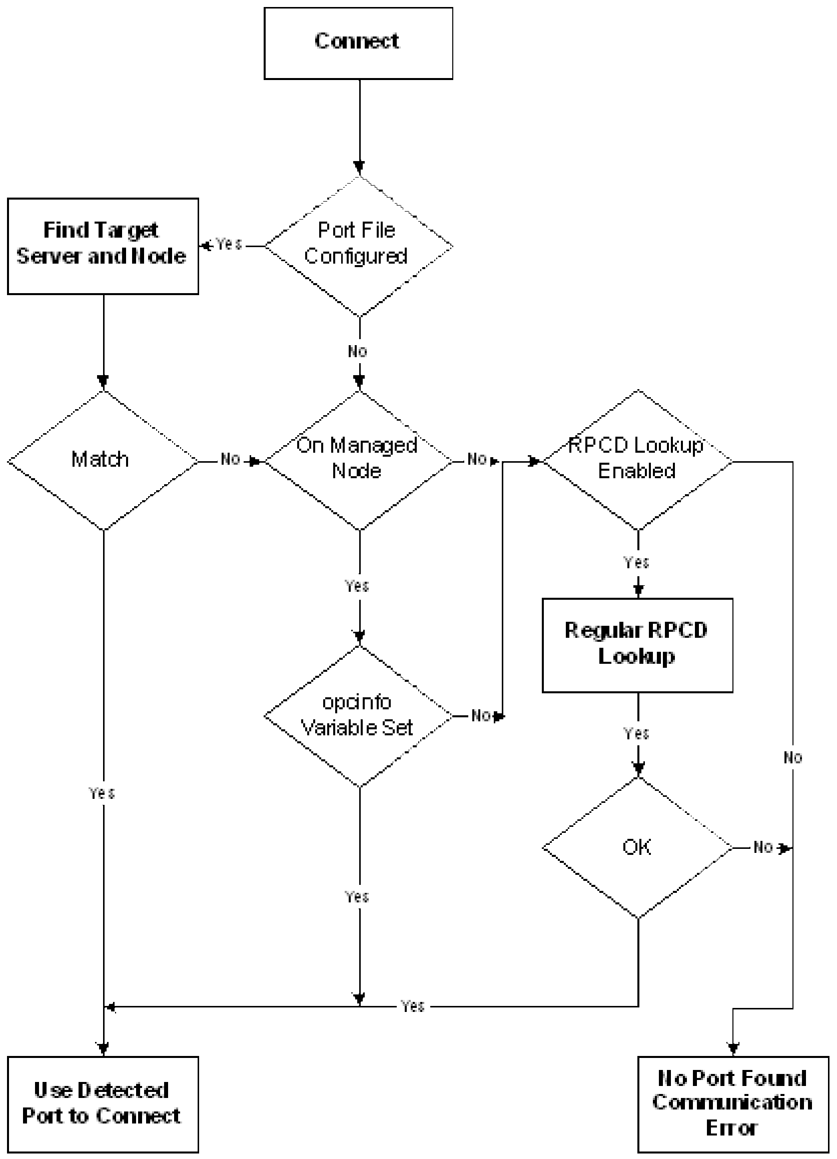
Key	Type	Default	Explanation
`OPC_COMM_REGISTER_RPC_SRV`	Boolean	TRUE	Selects whether to register RPC server interfaces with endpoint mapper.
`OPC_COMM_PORT_RANGE`	string	empty	Specifies exactly one port to be used by the RPC server. Must be set for the control agent (see page 9, how to restrict settings to individual processes).
`OPC_COMM_PORT_MSGR`	Int	0	Specifies at which port the message interface of the Message Action Server (`OvEpMsgActSrv`) is listening on the management server. For HP Operations management servers, this must be same value as used for `OPC_COMM_PORT_DISTM` .
`OPC_COMM_PORT_DISTM`	Int	0	Specifies at which port the deployment interface of the Message Action Server (`OvEpMsgActSrv`)is listening to configuration requests from the agent. For HP Operations management servers, this must be same value as used for `OPC_COMM_PORT_MSGR` . This setting is needed if the `opcinfo` file of the agent does not contain the variable setting: `OPC_NO_CFG_RQST_AT_STARTUP TRUE` .
`OPC_COMM_RPC_PORT_FILE`	String	empty	Complete path pointing to a server port specification file as described below.
`OPC_COMM_LOOKUP_RPC_SRV`	Boolean	TRUE	Whether to perform an endpoint mapper lookup if no matching port is found in the server port specification file.

Related Topics:

- Concepts of DCE RPC communication without endpoint mapper
- Configuring the HP Operations management server system
- Configuring HP Operations managed node systems
- Server Port Specification File
- Configuration examples
- Verifying configuration
- Troubleshooting
- Internal process handling

Internal Process Handling

RPC clients perform the following steps when connecting to an RPC server. A key to the diagram can be found below.



- Find destination Server and Node is the process of finding a matching entry for the destination RPC server in the server port specification file.
- Regular RPC Lookup is the process of querying the endpoint mapper on the destination system.

Some of the decisions made in the flow chart above are implemented by evaluating `opcinfo` variables on managed nodes or registry keys on the management server:

- Port file configured? Controlled with `OPC_COMM_RPC_PORT_FILE`
- Endpoint mapper lookup enabled? Controlled with `OPC_COMM_LOOKUP_RPC_SRV`
- On managed node? Will be answered with yes by all RPC clients running on the managed node. These are `opcmsga` , `opcdista` and `opcagt` .
- Opcinfo variable set? Will be answered with yes if the RPC clients want to connect to either of the following RPC servers (then, if set, the value of the according variable will be used by the RPC client as port of the destination RPC server):

RPC Server	Key
Message receiver interface of the message action server	`OPC_COMM_PORT_MSGR`
Distribution manager interface of the message action server	`OPC_COMM_PORT_DISTM`
Local Control agent (`opcctla`)	To access the local control agent, the client reads the setting used by the local RPC server, the control agent opcctla: `(OPC_RESTRICT_TO_PROCS opcctla)` `OPC_COMM_PORT_RANGE …`

Related Topics:

- Concepts of DCE RPC communication without endpoint mapper
- Configuring the HP Operations management server system
- Configuring HP Operations managed node systems
- Server port specification File
- Configuration examples
- Verifying configuration
- Troubleshooting
- Variable reference

Improved management server availability

HPOM for Windows is used to manage services and applications and therefore increase their availability for their users by detecting problems before they interfere with the normal operation of these services and applications. However, it is also desirable to improve the availability to the management server.

The following options improve the availability of the management server itself:

- For HPOM installations on a Windows 2003 Cluster, the management server supports failover from one node to another in case of a failure. For details, see the topic HPOM installed on a clustered Windows 2003 server .
- For installation on nonclustered systems, the availability of the management server can be improved by having a standby server and replicating the database periodically. For details, see Failover to backup server .

HPOM installed on a clustered Windows 2003 Server

You can install HPOM on a Windows 2003 Cluster with support for failover of the management server. Failover support means that the MS Cluster Service is able to switch the HPOM management server from one cluster node to another if there is a problem with the cluster node or the management server itself.

To install and manage an HPOM clustered installation you should be familiar with the MS Cluster Service and its concepts.

NOTE:
If you have an existing, nonclustered HPOM installation and would like to ensure its availability by clustering it, read the upgrade guide located on the HPOM installation media about how to convert your existing installation to a clustered installation. There is no way to do an in-place migration of this installation to a clustered one.

For a clustered installation of HPOM you must have a consistent installation on all cluster nodes. See the topic Check Cluster Consistency about how to check the consistency of your HPOM installation and about the impact of an inconsistent installation.

The central management tool for Windows Clusters is the Microsoft Cluster Administrator. The topic HPOM Cluster Resource Group gives an overview about the cluster resources set up during the HPOM installation.

The topic Registry Key Replication describes which server configurations are replicated between the cluster nodes.

If you would like to reinstall the agent on one of the management server nodes, see the help topic Reinstall the Management Server Agent .

Check cluster consistency

In a clustered installation of HPOM it is necessary to keep the installation consistent on all nodes, regardless of which of the physical cluster nodes it is currently running on. A consistent installation means that all nodes have the same SPIs and HPOM patches installed. A clustered installation also requires a consistent configuration of HPOM on all cluster nodes.

⚠ CAUTION:
As long as a cluster inconsistency is unresolved, HPOM cannot start or failover to the inconsistent cluster nodes. This could lead to unexpected down times of the management server.

Using the console tool Cluster Check Report , you can evaluate the current consistency status of the clustered HPOM installation. The tool also enforces cluster consistency. See the topic Cluster Consistency Enforcement for details.

During the installation of the management server, a self management policy called "VP_SM-ClusterConsistency Check" is deployed to the management serve nodes. This policy checks the cluster once a day and generates a critical message if an inconsistent cluster is discovered. For details about this inconsistency, see the message annotations and instructions.

Consistent configuration is ensured by both storing data in the database and replication registry keys. For details, see Registry Key Replication .

Cluster check report

HPOM provides a console tool, that checks to see whether all SPIs and HPOM patches are installed on all cluster nodes. This tool is located in the console tree under Tools ▭ HP Operations Manager Tools ▭ Cluster Tools . It produces a report similar to the one shown below.

```
---------------------------
 HPOM Cluster Check Report
---------------------------

 Inventory
-----------

HPOM cluster version: A.08.00
HPOM local version: A.08.00

HPOM installed on cluster nodes:
* clusternode1.domain.com
* clusternode2.domain.com

Installed SPIs:
* DBSPIMss Version 09.01.01

Installed patches:
* OVOW_00017

 Missing installations
-----------------------

On node clusternode2.domain.com:
* DBSPIMss Version 09.01.01

 Pending uninstallations
-------------------------

On node clusternode2.domain.com:
* OVOW_00151


 * end of report *
```

First, the report shows the inventory of the HPOM installation. It shows which version of HPOM is installed on the cluster and on the node running HPOM at the moment. It shows the names of the cluster nodes on which HPOM is installed and an inventory of all SPIs and patches installed on the cluster.

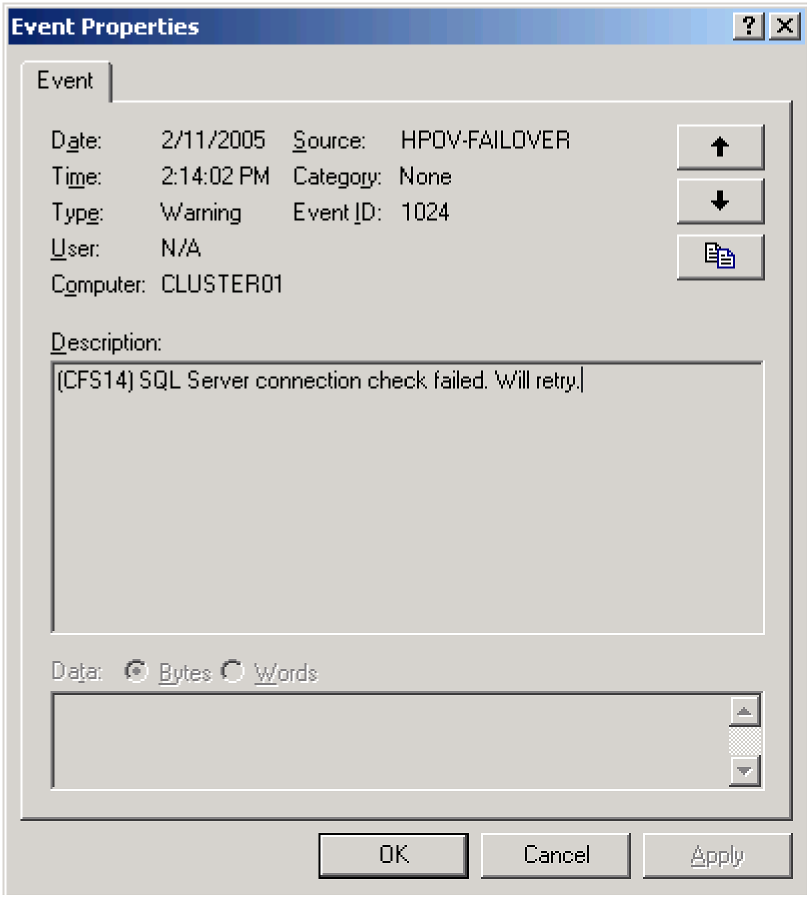
After that the report shows if there are any SPIs or patches which are not installed on all cluster nodes. If this list is empty, then the cluster is consistent. However, if there are any entries there you must install the missing SPIs/patches on these nodes as soon as possible because HPOM enforces a consistent installation and will refuse to run on the nodes in this list. This can lead to an unexpected down-time of the management server in case of a failure.

The last part of the report shows if there are any previously installed SPIs or patches which have not been uninstalled on all cluster nodes. If this list is empty, then the cluster is consistent. However, if there are any entries there you must uninstall the SPIs/patches on these nodes as soon as possible, because HPOM enforces a consistent installation and will refuse to run on the nodes in this list. This can lead to an unexpected down-time of the management server in case of a failure.

Cluster consistency enforcement

The OvOW Cluster Consistency Check cluster resource performs cluster consistency enforcement, explained in the topic HPOM Cluster Resource Group . Each time the resource is ordered to go online by the Microsoft Cluster Service (MSCS), it performs the following two checks. These checks must be performed successfully to start HPOM on that node.

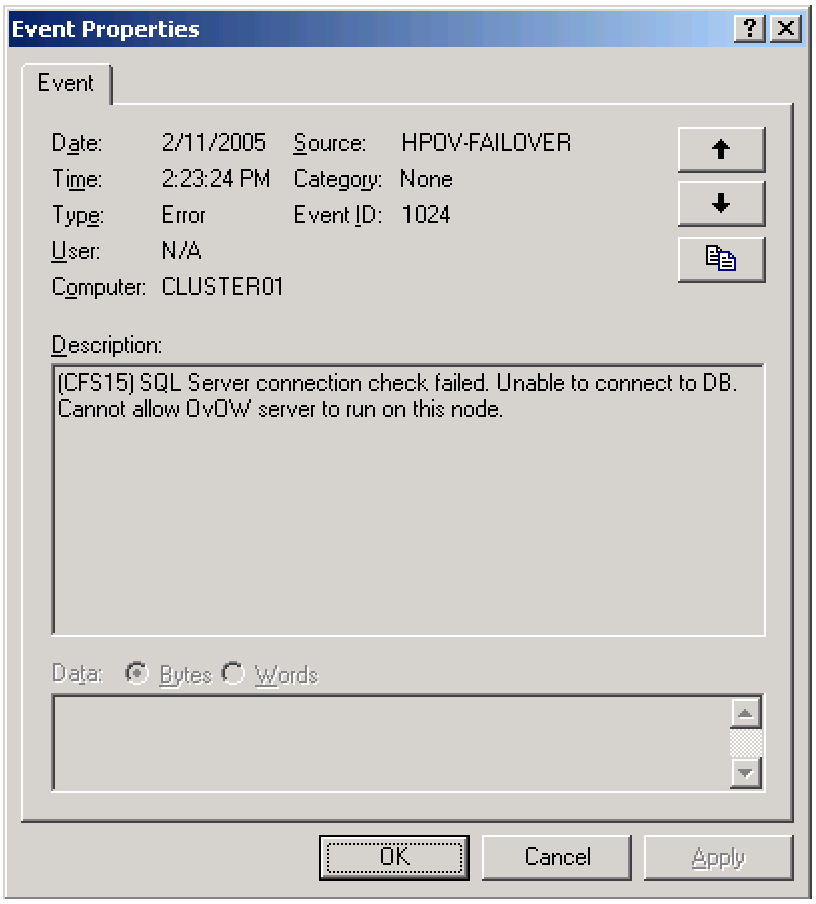
1. First, the cluster resource tries to connect to the database instance used by HPOM. This check is repeated by default 15 times, with a sleep of 20 seconds between each of these retries. Each unsuccessful try will add the following warning to the windows event log:



Date: 2/11/2005 Source: HPOV-FAILOVER
Time: 2:14:02 PM Category: None
Type: Warning Event ID: 1024
User: N/A
Computer: CLUSTER01

Description:
(CFS14) SQL Server connection check failed. Will retry.

If the consistency check is unable to connect to the database after reaching the maximum number of retries, it will fail. The MSCS will initiate a failover, depending on the cluster configuration, and write the following error message to the Windows event log:



Date: 2/11/2005 Source: HPOV-FAILOVER
Time: 2:23:24 PM Category: None
Type: Error Event ID: 1024
User: N/A
Computer: CLUSTER01

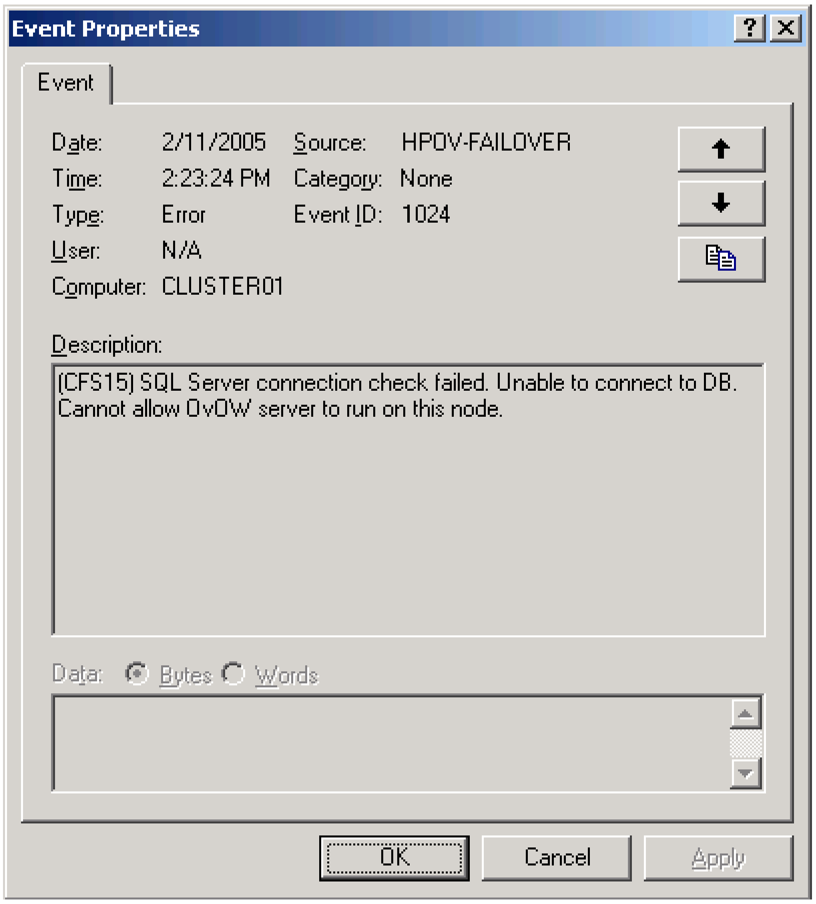
Description:
(CFS15) SQL Server connection check failed. Unable to connect to DB. Cannot allow OvOW server to run on this node.

Data: (●) Bytes ( ) Words

The number of retries and the sleep time can be configured, as explained in the topic Configure the consistency check resource for details.

2. The second test is the same cluster consistency check as done by the Cluster Check Report . If this check fails for the current node, HPOM cannot start. Depending on the MSCS configuration, HPOM will be moved to the next available node.

   If the consistency check fails, an error is written to the Windows Application Log. As shown in the illustration, attached to this event is a copy of the cluster check report, which contains the reason for the failure.



Event Properties

Event

Date:	2/11/2005	Source:	HPOV-FAILOVER
Time:	2:23:24 PM	Category:	None
Type:	Error	Event ID:	1024
User:	N/A		
Computer:	CLUSTER01		

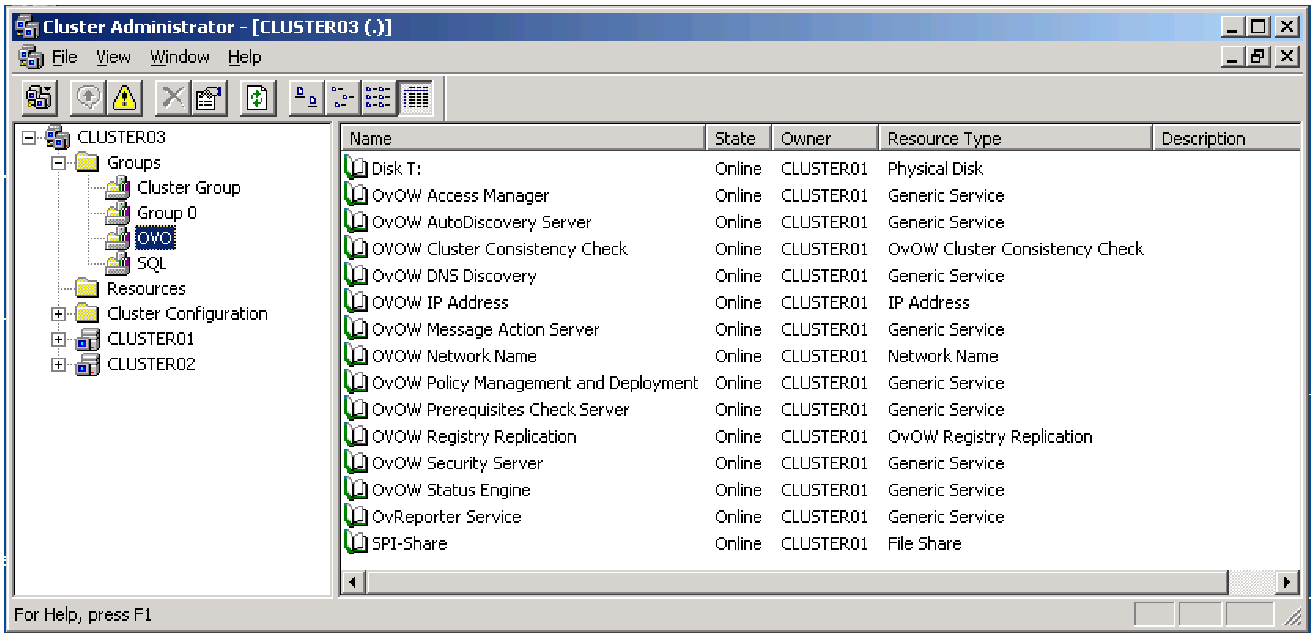
Description:

(CFS15) SQL Server connection check failed. Unable to connect to DB. Cannot allow OvOW server to run on this node.

Data: [✓ Bytes] Words

HPOM Cluster Resource Group

During the installation of HPOM on a cluster, you must select a cluster resource group for HPOM. The installation creates all cluster resources necessary to run HPOM with failover support on a Windows cluster. The following illustration shows the HPOM cluster resource group inside the MS Cluster Administrator.



Name	State	Owner	Resource Type	Description
Disk T:	Online	CLUSTER01	Physical Disk	
OvOW Access Manager	Online	CLUSTER01	Generic Service	
OvOW AutoDiscovery Server	Online	CLUSTER01	Generic Service	
OVOW Cluster Consistency Check	Online	CLUSTER01	OvOW Cluster Consistency Check	
OvOW DNS Discovery	Online	CLUSTER01	Generic Service	
OVOW IP Address	Online	CLUSTER01	IP Address	
OvOW Message Action Server	Online	CLUSTER01	Generic Service	
OVOW Network Name	Online	CLUSTER01	Network Name	
OvOW Policy Management and Deployment	Online	CLUSTER01	Generic Service	
OvOW Prerequisites Check Server	Online	CLUSTER01	Generic Service	
OVOW Registry Replication	Online	CLUSTER01	OvOW Registry Replication	
OvOW Security Server	Online	CLUSTER01	Generic Service	
OvOW Status Engine	Online	CLUSTER01	Generic Service	
OvReporter Service	Online	CLUSTER01	Generic Service	
SPI-Share	Online	CLUSTER01	File Share	

CAUTION:
After installing HPOM on a cluster, you must not change the names of the HPOM resource group or any HPOM resources. Otherwise, adding new nodes to the cluster or uninstalling HPOM will fail.

The MS Cluster Administrator can be used to change the way MSCS treats the resources. The HPOM installer creates the HPOM resources with the default settings defined by Microsoft. For details about these settings and their impact on the cluster and HPOM, consult the MS Cluster Service online help.

The cluster resource group contains the following resources:

Resource	Description
Disk <letter>	This is the shared disk used by HPOM to store that part of its data which needs to be in sync between all cluster nodes. All other resources depend on this resource; they cannot work without it.
OvOW Access Manager	This resource controls the HPOM Access Manager service. This resource depends on the consistency check resource.
OvOW AutoDiscovery Server	This resource controls the HPOM AutoDiscovery Server service. This resource depends on the consistency check resource.
OvOW cluster Consistency Check	This cluster resource ensures the consistency of the HPOM installation across all cluster nodes. See the Check Cluster Consistency Topic for details. This resource depends on the registry replication resource.
OvOW DNS Discovery	This resource controls the HPOM DNS Discovery service. This resource depends on the consistency check resource.
OvOW IP Address	This resource is the owner of the IP address of the virtual HPOM server entered during the installation. This resource depends on the shared disk resource.
OvOW Message Action Server	This resource controls the HPOM Message Action Server service. This resource depends on the consistency check resource.
OvOW Network Name	This resource is the owner of the network name of the virtual HPOM server entered during the installation. This resource depends on the IP address resource.
OvOW Policy Management and Deployment	This resource controls the HPOM PMAD service. This resource depends on the consistency check resource.
OvOW Prerequisites Check Server	This resource controls the HPOM Node Prerequisites Check Server service. This resource depends on the consistency check resource.
OvOW Registry Replication	This resource is a placeholder for the registry keys that need to be replicated between the cluster nodes to keep the HPOM configuration on all nodes in sync. See the Check Cluster Consistency and the Registry Key Replication topics for details. This resource depends on the network name resource.
OvOW Security Server	This resource controls the HPOM Security Server service. This resource depends on the consistency check resource.
OvOW Status Engine	This resource controls the HPOM Status Engine service. This resource depends on the consistency check resource.
OvReporter Service	This resource controls the HP Reporter service. This resource depends on the consistency check resource.
SPI-Share	This resource controls a Windows file share called "SPI-Share" which is used to store templates for the config file policy type. It depends on the network name resource.

In order to use HPOM on a cluster, all the above-mentioned resources need to be online. The only exception is the "OvReporter Service" in case a standalone Reporter installation is used.

Reinstall the management server agent

In a clustered HPOM installation, one agent is installed on each physical management server node. By default, the HPOM Self Management policies are deployed on each physical server node.

Because these policies are only needed on the active server node, they are controlled by the agent Application Package Manager (APM). The help topic Enable policy switching on DCE agents describes the APM configuration. The policies are enabled on the current active management server node and disabled on all other nodes.

If the agent on one of the management server nodes is removed and reinstalled, you need to recreate the cluster configuration of these policies, as described below:

To reconfigure the cluster configuration

1. Execute the tool `Add Selfmanager to APM` on the respective management server cluster node. The tool is available in the console tree under Tools → HP Operations Manager Tools → Cluster Tools .
2. To apply the configuration, restart the agent with the commands `opcagt -stop` and `opcagt -start` .

NOTE:
If you do not recreate this configuration, the HPOM Self Management policies will always be enabled on that cluster node and will report misleading errors in your HPOM installation.

Configure the consistency check resource

Two checks are performed by the cluster consistency check resource:

- Database connection check
- Consistency check for the current node

The database connection check can be configured with registry values located in the registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\OVEnterprise\Cluster\ConsistencCheck
```

Value	Data
RetryLoops of type REG_DWORD	Number of times the consistency check resource should try to connect to the database before failing. The default value is 15.
RetrySleep of type REG_DWORD	Number of seconds the consistency check resource waits between the connection retries. As the connection process itself needs some time, the absolute time between the connections is larger than the number given here and is dependent on the current system. The default value is 20.

Related Topics:

- Cluster consistency Enforcement

Registry key replication

As described in the Check Cluster Consistency topic, it is vital for the operation of HPOM in a cluster to have the HPOM installation on all cluster nodes installed and configured consistently.

Because part of the HPOM configuration is stored in the system registry, parts of it will get replicated between the cluster nodes by the OvOW Registry Replication cluster resource, as described in the topic HPOM Cluster Resource Group . Each time this resource goes online on a cluster node, it replaces the registry keys listed below with the ones existing on the last active node. This has two consequences: first, only the currently active node has an up-to-date HPOM registry, and second, it is necessary make changes in these replicated keys on the currently active node. Changes made on other nodes will be lost whenever they become active.

The following list shows the registry keys replicated by the registry replication resource. Each key, with all its values and its subkeys, will be replicated.

NOTE:
This list is not a documentation of how these keys are used by HPOM or an encouragement to change these. Refer to other parts of the documentation to see whether changing these keys is supported or not.

```
SOFTWARE\Hewlett-Packard\CoreSPIMigration
SOFTWARE\Hewlett-Packard\HP OpenView
SOFTWARE\Hewlett-Packard\OpenView\Common
SOFTWARE\Hewlett-Packard\OVEnterprise\Cluster\ConsistencyCheck
SOFTWARE\Hewlett-Packard\OVEnterprise\Cluster\MgmtServerNodes
SOFTWARE\Hewlett-Packard\OVEnterprise\Cluster\Patches
SOFTWARE\Hewlett-Packard\OVEnterprise\Cluster\Resource
SOFTWARE\Hewlett-Packard\OVEnterprise\Cluster\SPIs
SOFTWARE\Hewlett-Packard\OVEnterprise\Cluster\Upgrades
SOFTWARE\Hewlett-Packard\OVEnterprise\Log
SOFTWARE\Hewlett-Packard\OVEnterprise\Management Server\AutoDeployment
SOFTWARE\Hewlett-Packard\OVEnterprise\Management Server\DBAccess
SOFTWARE\Hewlett-Packard\OVEnterprise\Management Server\MsgActSrv
SOFTWARE\Hewlett-Packard\OVEnterprise\Management Server\Pmad
SOFTWARE\Hewlett-Packard\OVEnterprise\Plug-Ins\Self Manager
SOFTWARE\Hewlett-Packard\OVEnterprise\Security
SOFTWARE\Hewlett-Packard\The Reporter
SOFTWARE\ODBC\ODBC.INI\openview
SOFTWARE\ODBC\ODBC.INI\reporter
SYSTEM\CurrentControlSet\Services\Eventlog\OvConfigChanges
```

Failover to backup server

In a warm standby scenario, the secondary management server maintains a copy of the SQL Server database of the primary management server. The standby database on the secondary management server runs in "Recovery" (read-only) mode. To synchronize the two databases, a scheduled task policy backs up the transaction log of the active database, copies it to the secondary management server, and restores it to the standby database. This task runs at regular intervals.

In case of a failure of the primary management server, the database on the secondary management server switches from "Recovery" to "Online" mode and the HPOM services are started on the secondary management server server. Next, the Switch Management Server Tool is executed on every managed node in order to switch the agents to report to the secondary management server from now on. To connect to the secondary management server, any consoles must be restarted.

For more information, see the *HPOM High Availability Utilizing Microsoft SQL Server Log Shipping* white paper at the following location:
`%OvInstallDir%\paperdocs\en\HPOM_HA_with_SQL_Log_Shipping.pdf`

Backing up and restoring the HPOM management server

It is important that you regularly back up the data on your HPOM management server to preserve the HPOM database contents in the event of a failure and to maintain the customized configurations, policies, files, graphs, instrumentation, and reports that you have created. You may use both of the following methods:

- Full system backup (offline backup)
- Database backup (online backup)

Full system backup (offline backup)

A full system backup, including operating systems, all partitions, disks, files, and so forth is the easiest way to back up and restore your HPOM management server.

A full system backup is an offline backup. This means that the HPOM server, as well as the operating system, is shut down and the backup is started using a special boot disk or partition and special backup or imaging software.

HP recommends that you create an image/backup of your HPOM management server using a suitable solution so that you can restore your complete system, including operating system settings like IP configuration, DNS configuration, and all installed software, including HPOM for Windows.

However, such a solution is not appropriate for a daily backup, because the management server has to be shut down. Full backups can be combined with regular database backups.

In cases where you cannot use a full-system backup or imaging solution (for example, if the system where you want to restore the data uses different hardware and disks), you can also restore an HPOM server by reinstalling the operating system and HPOM software using exactly the same steps as for the original installation.

Database backup (online backup)

HPOM provides a policy and scripts that enable you to backup and restore the HPOM database. However, the database backup does not include configuration data from the registry, from other configuration files, or from the file system (for example, instrumentation). Therefore, you cannot rely on database backups alone. You many want to combine full system backups and daily database backups. If you do a full system backup of your HPOM management server including all installed patches and configurations, you can then do database backups to save data on a daily basis.

Related Topics:

- Back up the HPOM database
- Restore the HPOM database
- Back up miscellaneous data

Back up the HPOM database

The HPOM management server uses an SQL Server instance with a user-configurable name as core data repository. The HPOM installation suggests the name OVOPS as default name for this instance. The instance is also shared by other HP Software products, which create other databases in this instance. The full name of the database depends on the location of the database:

- Local database:
  If SQL Server Express is used as HPOM database, the full database instance name is `<local host>\<instance name>`.
- Remote database:
  If you are using a remote SQL Server database, the name is `<server name>\<instance name>`.
- Clustered database:
  If the SQL Server database is clustered, the instance name is `<virtual SQL server name>\<instance name>`.

Scheduled task policy VP_SM_DB_Backup

HPOM self management includes the scheduled task policy "VP_SM_DB_Backup", which schedules and performs a backup of the openview database.

After you have installed the management server, deploy the policy to the system where the HPOM database is installed. Before deploying the policy, modify it as described below:

1. Go to the policy editor.
2. Open the policy at this location:
   Policy Groups → HPOM Self Management → en → Database Server → VP_SM_DB_Backup
3. Edit the Command field:
   - If the database resides on the management server itself, you can remove `<servername>\<instancename>` from the end of the command line. In this case the SQL Server instance is looked up from the registry.
   - If the database resides on a remote server, you must replace `<servername>\<instancename>` with the SQL Server instance name of the remote database instance.
4. In the Task tab, edit the Execute as user and the Specify password fields, and add the user account and password of a user with SQL Server administrator permissions.
5. Optional: By default the database backup is scheduled for 0:30 a.m. every morning. If that time does not suit your needs, you can adopt the schedule in the Schedule tab.
6. Deploy the policy to the system where the HPOM database is installed.

Backup and restore related VB scripts

NOTE:
Modification to these scripts requires basic knowledge of the Microsoft Visual Basic script language.

The following backup and restore scripts are located in the `%OvInstallDir%\Support\` directory.

- backup_openview.vbs

  This script performs a backup of the openview database and stores the backup files in the directory `%OvDataDir%\datafiles\<instance name>\backup` .

- restore_openview.vbs

  This script restores the openview database from the backup files created by backup_openview.vbs.

Location and housekeeping of backup files

The backup files of the openview database and the openview database transaction log are written into the following directory:

`%OvDataDir%\datafiles\<instance name>\backup`

Backup files are named as follows:

- openview_dat.bak (openview database)
- openview_log.bak (openview database transaction log)

These files are overwritten on every new execution of the VP_SM_DB_Backup policy. If you want to keep them you have to ensure that they are stored on a storage device or copied into another location in the meantime.

You may change the location of the backup files by editing the VB script file `%OvInstallDir%\Support\backup_openview.vbs` .

Monitoring backup execution

The VP_SM_DB_Backup policy will send messages to the active messages browser:

- When the backup starts.
- After a successful backup. (This message also contains the backup duration time.)
- After a backup failure (critical message).

Configuration on a remote database server

If the openview database runs on a remote database server you must deploy the VP_SM_DB_Backup policy on that server instead of the management server.

Starting the openview database backup manually

If you want to execute the backup job immediately, you can trigger this with the command:

```
cscript "%OvInstallDir%\support\backup_openview.vbs "
```

NOTE:
If the HPOM database is a remote database, the SQL Server instance name must be added to the command line:
`%OvInstallDir%\support\backup_openview.vbs <servername>\<instance name>`

Related Topics:

- Restore the HPOM database

Restore the HPOM database

If you used the scheduled task policy VP_SM_DB_Backup or the tool backup_openview.vbs to create a backup of the HPOM database, you can restore it manually. Before you can start restoring the HPOM database, you must stop the management server processes.

Stop management server processes

NOTE:
The restore procedure requires that the WMI service is stopped and remains stopped until the recovery is complete. System management tools, such as HP Systems Insight Manager and the HP Operations agent itself attempt to restart WMI automatically. Make sure that these tools are stopped while restoring the database.

1. Close all open consoles first, before starting to restore the HPOM database.
2. Stop all services and processes using `vpstat -3 -r STOP` .
3. Use the service control panel (Services.MSC) to stop the Windows Management Instrumentation service. Note that there is a second service called "Windows Management Instrumentation Driver Extensions" - you do not need to stop this service (in fact you cannot stop it, the system disallows this). Alternatively, you can use the command `net stop winmgmt` to stop the WMI service.

Restore the openview database

Restore the backup files for the HPOM database backup to the same folder that they were generated in on the failed HPOM server. By default this is `<HPOM database files folder>\Backup\` .

To start the restore operation:

1. If HPOM uses a local database, start the SQL SERVER (<instance name>) service again. Otherwise make sure the remote MS SQL Server instance is running.
2. Run `cscript %OvInstallDir%\support\restore_openview.vbs` .

   NOTE:
   If your HPOM database resides on a remote database server you must add the SQL Server instance name as a parameter to the script restore_openview.vbs (for example, `cscript restore_openview.vbs TCVM08\OVOPS`).

Sample output

```
Microsoft (R) Windows Script Host Version 5.6
```

```
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

HPOM is stopped

This backup files will be used to restore the openview database:


BackupType: Full DB backup
BackupFileLocation:
E:\HPOM\Data\datafiles\OVOPS\backup\openview_dat.bak
BackupName:  openview_Full
BackupStartDate:  2006-11-09 14:55:59.000
BackupFinishDate:  2006-11-09 14:56:06.000


BackupType: Transaction log backup
BackupFileLocation:
E:\HPOM\Data\datafiles\OVOPS\backup\openview_log.bak
BackupName:  openview_Log
BackupStartDate:  2006-11-09 14:56:06.000
BackupFinishDate:  2006-11-09 14:56:06.000


Are you sure that you want to replace the current openview DB
with the contents of the backup above ? (y/n)
y

replace openview database with Restore


Restore succeeded
```

Related Topics:

- Backup the HPOM database

Back up miscellaneous data and files

The following items can be backed up in addition to the HPOM database:

Group membership

If the HPOM users are local accounts, write down which users belong to the groups HP-OVE-OPERATORS and HP-OVE-ADMINS on the HPOM management server. These group memberships must be recreated manually unless they will be restored as part of a full system recovery (from a complete server image backup).

If you used domain user groups during the HPOM installation, there is no need to back up the group membership information on the management server.

Exported messages (optional)

The default setting of the HPOM server is to delete messages in the acknowledged message browser after 30 days. It is also possible to export the message into a Comma Separated Value (CSV) file. The setting is customizable in Database Maintenance namespace in the Server Configuration dialog box.

If you want to back up the downloaded messages, (for custom reporting on messages, for example), make sure that these files are included in the backup procedure.

Custom HPOM console MMC definitions (optional)

By default the HPOM server includes only one HPOM console MMC definition file:
`%OvDataDir%\HP Operations Manager.msc`

Reconfiguring your setting in this single file is not complex or time consuming but you may want to back up the file as part of your daily backup.

If you have created additional HPOM console MMC configuration files for specific users, then these files will have been saved in a location that you have explicitly selected, for example:
`Documents and Settings\<UserName>\Desktop`

A backup of these files will save time when recovering the HPOM server, particularly if you have a number of these files - such as when multiple users are allowed to connect to the HPOM server using Terminal Services to run the HPOM console.

In most cases the HPOM console MMC configuration files will reside on the systems that host the HPOM remote console. Consideration of the backup strategy for these files should be part of a full disaster recovery plan - although in many cases, recreating the console definitions is a simple option.

HPOM patches

To successfully recover your HPOM server, you must start by recreating an identical HPOM installation with the same patch level. In some cases patches may be superseded by new patches and will then no longer be available on the hp support web page. Therefore you should include the patch files (OVOW_xxxxx.exe) in your backup to ensure that you have the relevant patch files if you need to recreate your system.

You can use `vpstat -p` to get a list of all currently installed patches. You can also include this information into your backup by redirecting the output to a file (Use `vpstat -p > vpstat.out` . Then include `vpstat.out` in your backup). This will allow you to check if you have recreated a system with the exact same patch level.

Licenses

All licenses are stored in the file `%Program Files%\Common Files\Hewlett-Packard\HPOvLIC\data\LicFile.txt` . On a clustered HPOM installation, the licenses are stored in the file `%OvShareDir%\HPOvLIC\data\LicFile.txt` .

Related Topics:

- Back up the HPOM database

Configure the database

You can set up and maintain your database in several ways:

- Maintain your database
- Implement a remote SQL Server database

Related Topics:

- Database maintenance for HPOM for Windows
- Implementing a remote Microsoft SQL server database

Maintain the database

You may find it necessary to perform certain maintenance tasks related to the openview database. Common maintenance tasks might include:

- Distributing database files and log files to different disk drives
- Maintaining acknowledged messages in the management server database
- Downloading messages
- Resizing SQL Server virtual memory usage
- DB component maintenance details
- Changing the history database default settings

NOTE:
To improve the performance of database queries, you may add your own indexes to the SQL Server database. Adding custom indexes does not violate any support agreements you may have with Hewlett-Packard.

Related Topics:

- Implementing a remote Microsoft SQL Server database
- Backing up and restoring the HPOM management server

Distribute database files and log files to different disk drives

After initial installation all components of the management server database are stored in the same directory, for example, as:

`%OvDataDir%\datafiles\<database_instance>`

The database consists of several physical files, as shown in the table:

File Name	Purpose
`openview.mdf`	Primary partition for system tables
`store.ndf`	Store tables and indexes
`pmad.ndf`	Policy Management and Deployment tables and indexes
`msgaction_dat.ndf`	Message Action Server tables
`msgaction_idx.ndf`	Message Action Server indexes
`role.ndf`	Security Server Role tables and indexes

In large production environments with many managed nodes, a considerable amount of data could be collected and stored into the database. In such environments it is advisable to distribute the database files and log files to dedicated disk drives with sufficient capacity to prevent data overflow.

To achieve better performance by utilizing parallel disk I/Os, you should distribute database files and log files to separate disk drives.

To move the database files to another disk location

You can change the location of any database file (data file or log file) using this procedure.

NOTE:
You must have SQL Server sysadmin authorization to execute this procedure.

1. Stop any console sessions.
2. Shut down the Windows Management Instrumentation (WMI) service, which stops all management server components.
3. Use the scripts for attaching and detaching the physical files with the HPOM database that were provided with the product, `attachdb.sql` and `detachdb.sql` . Scripts are located in the `%OvInstallDir%\lbin\OvOW\ServerConsoleInstall\dbscripts` directory.
4. Before you detach the openview database you must ensure that there are no more processes active in

this database.

5. Detach the openview database by running the detach script:

   ```
   osql -S<database_instance> -E -i detachdb.sql
   ```

   The operation is successful if there is no error message.

6. Copy the physical database files that you want to move to their new location.

   Make a copy of the `attachdb.sql` script, as shown:

   ```
   copy attachdb.sql myattach.sql
   ```

7. Edit the attached script `myattach.sql`, replacing the lines containing NEW_LOCATION with the actual physical location of the file.

   NOTE:
   You must specify all physical database files, even those that did not move.

8. Attach the database files. Run the command:

   ```
   osql -S<database_instance> -E -i myattach.sql
   ```

   You should see the message "Successfully attached database openview."

9. Start the OVEpStatusEngine service, which starts all management server components.

Related Topics:

- Maintain the history database
- Message download
- Resize SQL Server virtual memory usage
- DB Maintenance Component details
- Change history database default settings

Maintain acknowledged messages in the management server database

Messages that have been acknowledged are removed from the active messages browser and displayed in the acknowledged messages browser. The management server database that contains these acknowledged messages eventually reaches its capacity if regular maintenance is not performed. The DBMaint component, which is an integral part of the OvEpMessageActionServer service, performs this regular database maintenance.

The Database Maintenance namespace in the Server Configuration dialog box enables you to reconfigure the default registry settings, which specify when and how often such maintenance should be performed.

- Database maintenance time : specifies that the database should be scanned daily at 3:00 a.m. for old acknowledged messages.
- Acknowledge message retention period (in days) : specifies that messages which have been acknowledged for longer than 30 days should be deleted. To save these deleted messages, download the messages to text files using the configuration values Export acknowledged messages before deleting them and Location of the exported message files .
- Messages processed per database transaction : defines the maximum number of messages which can be processed (exported or deleted) within one transaction. After the defined number of messages has been processed, the current transaction is committed and a new transaction is started to process the next block of messages.
- Maximum database maintenance scan duration (in seconds) : defines the maximum time interval (duration) in seconds DBMaint is allowed to perform its operation. If during operation the end of the defined time interval is reached, but there are still outstanding message blocks to be processed, then DBMaint finishes the processing of the current message block by committing the current transaction. DBMaint will not start additional transactions so that it can continue to process further messages. In this case, it writes the following warning message into the Windows Application Event Log:

  (MS850) DBMaint stopped due to the maximum duration time limit being reached.

  Then DBMaint updates the table ov_ms_stats with the appropriate statistics. Messages which could not be processed (exported or deleted) due to the maximum time interval being reached will be processed the next time DBMaint is scheduled to run.

Maintenance occurs automatically and requires no further action from administrators and users.

However, as administrator, you can change these defaults if they do not meet your needs by manually changing the settings in the Server Configuration dialog box .

To edit default database maintenance configuration values

1. In the console tree, right-click Operations Manager , and then click Configure → Server... . The Server Configuration dialog box opens.
2. Click Namespace and then click Database Maintenance . A list of values appears.
3. Click the value that you want to change. Depending on the type of value, you can either type a new value, or select a value from a list that appears.
4. *Optional.* If a value appears in bold , the default value has been changed. To reset the value to its default, right-click the value, and then click Set to default .
5. Click OK .

Related Topics:

- Distribute database files and log files to different disk drives
- Message download
- Resize SQL Server virtual memory usage
- DB Maintenance Component details
- Change history database default settings

Message download

To prevent the message database from becoming too large, HPOM for Windows deletes acknowledged messages after 30 days (or a period that you specify). If you want to save the messages, you can configure HPOM for Windows to save the message information to CSV text files.

The Database Maintenance namespace in the Server Configuration dialog box enables you to reconfigure the default registry settings, which are used to set the download parameters.

Export acknowledged messages before deleting them configures whether the database maintenance scan exports acknowledged messages to the file system before it deletes them from the database. Set this value to true if you want the messages exported before they are deleted from the message database (default is false).

Location of the exported message files is a string that defines the location of the exported message files. Set this value to any drive and directory that you want. If the path does not exist, the messages are neither exported nor deleted, and an error message is written to the event log. The default path is `c:\\` . To make your CSV files go to the right directory, you need to put double back slashes for every directory, as shown:

```
C:\\Documents and Settings\\All Users\\Application Data\\HP\\HP BTO
Software\\Datafiles\\OldMessages
```

NOTE:
If you try to insert a different path, it will fail or default to the root "C:\". You must use the double back slashes, as in C:\\.

Field separator for export is a character that specifies the field separator which should be used in the exported message files for delimiting fields in a row. The default field separator character is the comma (,).

Character code page for export is a string that defines the character code page used for the exported message files. Set this value to UCS2 for 2 bytes Unicode. Set it to MULTIBYTE for mulitbyte character code page. The default value is MULTIBYTE.

Use Microsoft Excel convention for export defines whether Microsoft Excel convention is used for formatting multi-line text fields in the message files. Set this value to true if you want to process the exported files with Microsoft Excel. Set this value to false if you want to leave the formatting unchanged and do not want to use Microsoft Excel for further processing exported message files. The default value is false.

Two files are written:

- HistoryYYYY-MM-DD-HH-MMMessages.csv (view sample file)
- HistoryYYYY-MM-DD-HH-MMAnnotations.csv (view sample file)

In both cases, if no messages are available for download, no file is written.

To edit default message download configuration values

1. In the console tree, right-click Operations Manager , and then click Configure → Server... . The Server Configuration dialog box opens.
2. Click Namespace and then click Database Maintenance . A list of values appears.
3. Click the value that you want to change. Depending on the type of value, you can either type a new value, or select a value from a list that appears.
4. *Optional.* If a value appears in bold , the default value has been changed. To reset the value to its default, right-click the value, and then click Set to default .
5. Click OK

Related Topics:

- Distribute database files and log files to different disk drives
- Maintain the history database
- Resize SQL Server virtual memory usage
- DB Maintenance Component details
- Change history database default settings

Resize SQL Server virtual memory usage

At installation, the "max server memory" for the HPOM SQL Server instance (the HP Operations Manager database) is set to 50% of physical memory. This configuration is made to limit the amount of virtual memory that is consumed by SQL Server. See the related URL:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/architec/8_ar_sa_40vt.asp

If more physical memory is installed, then the maximum memory size configured for the \OVOPS instance may be increased accordingly. Use the command-line script provided in the <Support> directory for this purpose.

To change the "max server memory" setting

1. The `setmaxmem.bat` file will run the OSQL script `set_max_memory.sql` . If you have changed the username:password from <sa>, then edit the `setmaxmem.bat` file with the correct system administrator's username and password.
2. Edit the `set_max_memory.sql` file with the desired <max server memory> size.
3. Run the `setmaxmem.bat` file.

You do not need to reboot or restart SQL Server for these changes to take effect. However, if a smaller amount of virtual memory is specified than is already used, the new maximum value will not take effect until the system is reset.

Related Topics:

- Distribute database files and log files to different disk drives
- Maintain the history database
- Message download
- DB Maintenance Component details
- Change history database default settings

DB Maintenance Component details

The DB Maintenance component is an integral part of the management server. When the management server starts, the component performs the following tasks:

1. It checks the registry for the time to start maintenance and the time interval after which acknowledged messages should be purged from the history database. If these values are present, the component does nothing until the time for executing maintenance operations is reached.

   If these values are not configured, then the component is suspended until these time values are entered into the registry. (The component waits for a registry changed notification event.)

2. When the time for maintenance is reached, the component restarts and performs the following tasks:
   - Deletes all entries in the tables OV_MS_MESSAGE and OV_MS_ANNOTATION which match these criteria:
     - The message is acknowledged.
     - The time interval between the acknowledgement of the message and the current time is greater than or equal to the configured time interval for message deletion.
   - Updates the statistics in the table `OV_MS_Stats` of the management server database:
     - Start time for maintenance
     - Finish time for maintenance
     - Number of entries deleted in the table `OV_MS_message`
     - Number of annotations deleted in the table `OV_MS_Annotation`
   - Suspends activity until the time for the next maintenance cycle is reached.

Related Topics:

- Distribute database files and log files to different disk drives
- Maintain the history database
- Message download
- Resize SQL Server virtual memory usage
- Change history database default settings

Change StartTime DBMaint and DBMaintTimeSpec Settings

The default settings for database maintenance can be easily changed to meet your specific requirements. The table following the numbered steps shows the correct format for the values and some example settings for each.

To edit default database maintenance configuration values

1. In the console tree, right-click Operations Manager , and then click Configure → Server... . The Server Configuration dialog box opens.
2. Click Namespace and then click Database Maintenance . A list of values appears.
3. Click the value that you want to change. Depending on the type of value, you can either type a new value, or select a value from a list that appears.
4. *Optional.* If a value appears in bold , the default value has been changed. To reset the value to its default, right-click the value, and then click Set to default .
5. Click OK .

Value	Description

Database maintenance time	Specifies when the DBMaint component should start the maintenance cycle. The default specifies that the database is scanned daily at 3:00 a.m. for old acknowledged messages. It has the following format:
	
	Syntax:
	
	`[DayOfWeek-]HourOfDay:Minute`
	`DayOfWeek: 0 -6; Sunday = 0`
	`HourOfDay: 0 -23`
	`Minute: 0 -59`
	
	Examples:
	
	`6-20:10 (each Saturday at 8:10 p.m.)`
	`23:59 (every day at midnight)`

Acknowledged message retention period (in days)	Specifies the time interval after which acknowledged messages should be deleted. The default is 30 days.
	
	Syntax:
	
	`NumOfDays-NumOfHours-NumOfMin`
	`NumOfDays: 0 - any number.`

	`NumOfHours: 0 - 23`
	`NumOfMin: 0 - 59`
	
	Examples:
	
	`2-0-0` (purge messages after 2 days of acknowledgement).
	
	`375-23-0` (purge messages after 375 days, 23 hours of acknowledgement)

Messages processed per database transaction	Defines the maximum number of messages which can be processed (exported or deleted) within one transaction. After the defined number of messages has been processed, the current transaction is committed and a new transaction is started to process the next block of messages. Syntax: Possible values: any value > 0 Default value: 150 (process maximum 150 messages within a transaction. After processing 150 messages, start a new transaction.
Maximum database maintenance scan duration (in seconds)	Defines the maximum time interval (duration) in seconds DBMaint is allowed to perform its operation. If during operation the end of the defined time interval is reached, but there are still outstanding message blocks to be processed, then DBMaint finishes the processing of the current message block by committing the current transaction. DBMaint will not start additional transactions so that it can continue with processing further messages. In this case, it writes the following WARNING message into the Windows Application Event Log: (MS850) DBMaint stopped due to the maximum duration time limit being reached. Then DBMaint updates the table ov_ms_stats with the appropriate statistics. Messages which could not be processed (exported or deleted) due to the maximum time interval being reached will be processed the next time DBMaint is scheduled to run. Syntax: Possible values: 0 (indicates unlimited time; default) any value > 0 (indicates the number of seconds DBMaint is allowed to perform its operation). Use this configurational value if there is a danger that occasionally DBMaint could take too long (more than ~30 minutes), which could then lead to problems as it overlaps with other scheduled tasks.

Example:

1800 (Process messages maximum 30 minutes. Those messages which could not be processed within 30 minutes will be processed the next time DBMaint is scheduled to run.)

NOTE:
DBMaint history log files use local time in the file name, not Greenwich Mean Time (GMT).

To delete acknowledged messages at any time you specify

If you want to delete acknowledged messages from the database at any time you specify, rather than waiting for the regularly scheduled maintenance, you can change the configuration value Database maintenance time to the time you prefer. You can also change the default 30-day setting for the deletion of acknowledged messages by changing the setting for the configuration value name Acknowledged message retention period (in days) , as shown in the following examples. For best performance, delete no more than 1,000 messages at a time.

To delete all acknowledged messages immediately

1. Assume that the current time is 2:10 p.m.
2. Set the following registry values.

   Database maintenance time : 14:11

   Acknowledged message retention period (in days) : 0-0-0

The Database maintenance time must use time specifications in the range of 0:00-23:59. It does not understand a.m. and p.m. You should add one or two minutes to the current time.

The Acknowledged message retention period (in days) value of 0-0-0 means a time interval of zero, so it includes all acknowledged messages.

DBMaint automatically responds when any of its configuration values change.

NOTE:
Reset the values to their normal settings when this exceptional maintenance finishes.

DBMaint notifies all interested HP Operations Manager components on message deletions; running consoles register for message deletion events and update their acknowledged messages browser accordingly. Check your acknowledged messages browser to confirm that the messages have been deleted.

Related Topics:

- Distribute database files and log files to different disk drives
- Maintain the history database
- Message download
- Resize SQL Server virtual memory usage
- DB Maintenance Component details

Implementing a remote Microsoft SQL database

The HP Operations Manager for Windows management server uses a data repository, which is currently based on one of the following databases:

- Microsoft SQL Express (out-of-the-box)
- Microsoft SQL Server (optional)

Before you begin the HPOM installation, you must decide whether you want to use a local or a remote database. A Microsoft SQL Express database is always installed locally during the HPOM installation. A Microsoft SQL Server database can be installed locally or on a remote system. During the HPOM installation, the installation program asks you to enter the location and name of your database instance. You can then choose the out-of-the-box database (Microsoft SQL Express) or a previously configured instance.

You can also migrate an existing, local database to a remote SQL Server database after the installation. The database and management server then run on two separate machines. The HPOM management server communicates with the remote database server using the network.

Separating the management server from the database is desirable when the server where HPOM is installed has reached its hardware performance limit.

Related Topics:

- Install and configure the management server database
- Install and configure the remote database system
- Final configuration steps

Install and configure the management server database

To implement the remote server database, these criteria must be met:

- Both the management server system and the database server system should have a unique, fully-qualified domain name.
- Names and addresses of both systems must be resolvable at all times. HP recommends using DNS for name resolution.

Follow the instructions below to configure the management server and the system where the remote database will reside.

To install and configure the management server

1. Install HP Operations Manager and the desired Smart Plug-ins and HP Reporter as described in the Installation Guides.
2. Stop the management console.
3. Stop the following services:

   OvAutoDiscovery Server

   OvDnsDiscovery Service

   OvEpMessageActionServer

   OvEpStatusEngine

   OvEpMsmAccessManager

   OvSecurityServer

   Windows Management Instrumentation

   OVOWReqCheckSrv

   OVStoreProv

   OvowWmiPlatProf

   OvServiceLogger
4. Back up the openview database:

   ```
   cscript backup_openview.vbs
   ```

Related Topics:

- Implementing a remote Microsoft SQL database
- Install and configure the remote database system
- Final configuration steps

Install and configure the remote database system

After installing and configuring the management server, as described in the topic "Install and configure the management server database", you must then install and configure the database on the remote server.

1. Install the MS SQL Server with the instance name of your choice.
2. Create the directory `<HPOM database files directory>\backup` .
3. Copy the database backup files from the management server directory `<HPOM database files directory>\backup` to the same location on the remote database server.
4. Restore the openview database:

   ```
   cscript Restore_openview.vbs <SQL Server instance name> <full backup file location>
   <transaction log file location>
   ```

5. Now that you have copied the HP Operations Manager database to the remote server, you must set up a database backup on this server for production data protection:

   Deploy the self management policy VP_SM_DB_BACKUP to the HPOM database server. This policy will schedule and trigger the database backup of the openview database.

Related Topics:

- Implementing a remote Microsoft SQL database
- Install and configure a remote database system
- Final configuration steps

Final configuration steps

The following tasks are required to complete the implementation of your remote SQL Server database. Follow these steps for final configuration:

Edit vpstat and registry keys

1. On the management server, you can either keep the existing openview database on the management server and just disable the SQL Server (<Instance name>) database instance service, or to save disk space, you can also delete the openview database prior to stopping the database instance service:

   ```
   osql –S<management server>\<instance name> -E
   ```

   drop database openview

   go

   quit
2. Stop and disable these services:

   SQL Server (<Instance name>)

   SQL Server Agent (<Instance name>)
3. Edit the vpstat.conf file. Change to directory `%OvDataDir%\conf\vpstat` and open the configuration file `vpstat.conf` in your favorite text editor.

   Search for the section beginning with [SERVICELIST]. Add a hash at the beginning of the line ".service=VAR_SQLSERVER_INSTANCESQLSERVER_LOCAL."
4. Edit the following registry keys (for remote database access):

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\OVEnterprise\Management
   Server\DBAccess\OVOWINSTANCE
   ```
5. Reboot the management server.

The remote database configuration is now complete and ready for use with HP Operations Manager.

Remove the self-management policy VP_SM_CHK_OVODB

As the HPOM database is no longer located on the management server, you have to remove the self-management policy VP_SM_CHK_OVODB from the management server. Deploy this policy to the remote database server instead.

If the DB SPI is installed on the management server

If the *HP Operations Smart Plug-in for Databases* is installed on the management server, you must remove the SPI and its policies from the management server and redeploy the SPI and policies on the remote database server. This is required because the location of the OVOPS database instance has changed to the remote database server.

Related Topics:

- Implementing a remote Microsoft SQL database
- Install and configure a remote database system
- Install and configure the management server database

Configure managed nodes

The Node Configuration editor allows you to specify the managed nodes that you can monitor with HP Operations Manager for Windows. Operators monitor nodes and services and use the available tools to perform routine maintenance and also resolve problems that have critical business impact. Configuring a managed node is a multi-step process:

- Specify the node or group of nodes you want to manage.
- Enter details about each node's hardware and environment.
- Specify the tools available for each managed node or node group.

The managed nodes that you specify are listed in the Nodes folder of the console tree. You must configure each managed node individually.

Configure nodes using the wizard

To quickly configure a new node, use the Node Configuration wizard. This will provide the minimum amount of information you need to bring the node under management.

NOTE:
At any time you can exit the wizard by clicking Expert Mode . This returns you to the Node Configuration editor, where you can more completely specify the characteristics of this node. You cannot return to the wizard from the Node Configuration editor. When you add another node, the wizard opens again.

To open the wizard

1. Open the Node Configuration editor, if it is not already open.
2. In the Nodes tree, right-click Nodes and select New Node to open the wizard.
3. In the Base Settings dialog box, type the fully-qualified domain name in the box provided. You must supply this information to proceed. The system will discover information about this node and display it in the following screen.
4. Continue to specify the settings you want for this node as you proceed through the wizard screens.

To configure managed nodes manually

1. Open the Node Configuration editor, if it is not already open.
2. In the Configure Managed Nodes dialog box, specify the node or node group you want to manage by

copying nodes from the Discovered Nodes list to the Nodes list or by creating new nodes or node groups.

3. In the Nodes list, right-click the node you want to configure to open the Node Properties dialog box. Use the tabs to display information about the selected node and to configure it:
   - General: specify information about the environment, such as system name and owner.
   - Network: specify information such as primary node name, communications path, domain, heartbeat polling, and aliases.
   - Messages: view alias names and create new aliases.
   - System: specify information about the specific node, such as manufacturer, model, system type, operating system, and version.
   - Tools: specify tools available on the managed node.
   - Node Groups: view the parents of the selected node.
   - Outage: view outage information for the managed node.
4. Click Apply to see the effects of your changes.
5. Click OK to confirm your changes and close the dialog box.

   NOTE:
   When you bring new nodes under management, a prerequisite check runs automatically when you click Apply or OK . This opens the Prerequisite Check Component dialog box, which displays the status of the process, the names of the nodes being checked, and the results of the check, including details and recommended actions. See the help topic Check prerequisites for managed nodes for complete details.

6. Click Cancel to close the Node Configuration editor without saving your changes.

NOTE:
Special configuration may be required if you want to manage nodes over a firewall. Refer to the HP Operations Manager for Windows Firewall Configuration white paper for more information. Contact your HP BTO software representative to obtain a copy of this paper.

Automatic agent installation

The HP Operations agent is automatically installed on a managed node when you deploy a policy to the managed node. In some situations, you may want to install the agent manually. See Related Topics for details.

Managing intelligent devices

Using SNMP, you can manage intelligent SNMP devices such as printers, routers, and computers with unsupported operating systems and manage them from HPOM. See Related Topics for details.

Server and cluster failover behavior

HPOM for Windows monitors the availability of server components such as the Windows Management Interface (WMI), the server itself, and the cluster resource group (if your environment contains clusters.)

In the event of a failure of one or more of these components, some activities may be temporarily unavailable, until WMI or the server is restored. See Related Topics for details.

Related Topics:

- Specify nodes to be managed
- Configure general information for managed nodes
- Configure network information
- Configure alias names
- Configure system information for managed nodes
- Configure tools for managed nodes
- View outage information for managed nodes
- Add node to node group after upgrade from Windows 2000 to Windows 2003 management server
- View node group properties
- View policy and package inventory
- Manual installation of HTTPS agents
- Manual installation of DCE agents
- Manage intelligent devices
- Server or cluster failover behavior

Specify nodes to be managed

You can specify the nodes in your environment that you want to manage in either of two ways:

- Select nodes to be managed from the list of discovered nodes.
- Create new nodes manually.

Using either method, you configure nodes from which you want to receive:

- status information.
- event and performance-related messages.
- performance data which can be graphed for display in the console.

After you have specified the nodes to be managed, you must configure each node by specifying various required and optional properties. When configuration is complete, your selections will display as managed nodes in the console tree Nodes folder.

Related Topics:

- Create a new node
- Select nodes from discovered nodes list
- Configure general information for managed nodes
- Configure details for managed nodes
- Configure tools for managed nodes
- Create a new node group

Select nodes from discovered nodes list

To select nodes from the Discovered Nodes list

HP Operations Manager automatically discovers all the nodes in your environment, providing the nodes are up and running. You can drag and drop or cut and paste selected discovered nodes to the list of managed nodes.

1. Open the Node Configuration editor, if it is not already open.
2. From the Discovered Nodes list, select the node or node group you want to manage and drag it or copy and paste it to the Managed Nodes list. Required fields for these nodes or node groups are filled in automatically for you.

   Discovered node groups can be Domain Name System (DNS) nodes, Microsoft Windows Network nodes, or Unmanaged Nodes with Agents. Unmanaged Nodes with Agents are nodes where the agent software has been manually installed.

   NOTE:
   HPOM for Windows uses the computer's domain suffix as the default DNS domain. The domain suffix must be set properly in order for the default domain to display. Follow the steps below to specify the domain suffix.

3. To configure the node, select it from the Managed Nodes list and double-click to open the Node Properties dialog box.

To set the domain suffix:

1. On your desktop, right-click My Computer and select Properties from the shortcut menu to open the System Properties dialog box.
2. Select the Network Identification tab and click Properties to open the Identification Changes dialog box.
3. Click More to open the DNS Suffix and NetBIOS Computer Name dialog box.
4. Type in the domain suffix for your computer.
5. Click OK to close the dialog boxes and confirm your changes.

Create new nodes

You can create a new node , rather than select from the Discovered Nodes list. Perhaps the node you want to add is not included in the Discovered Nodes list and so is not available for selection. In that case, you can add a new node manually.

To add a new node

1. Open the Node Configuration editor if it is not already open.
2. From within the Configure Managed Nodes dialog box, select the Nodes folder in the managed nodes list.
3. Right-click to open the shortcut menu.
4. Select New Node to open the Node Configuration wizard which you use to configure the new node. The Base Settings page displays by default.
5. In the Base Settings page, type the fully-qualified domain name in the box provided and click Next . You must supply this information to proceed. The system will discover information about this node and display it in the following page.
6. Continue to specify the settings you want for this node as you proceed through the wizard pages, or click Expert Mode to open the Node Properties dialog box where you can specify the following settings:
   - General: Provide a required caption and an optional description.
   - Network: Provide the required primary node name and optional Communication Path and Domain information.
   - Message Identification: Display or create aliases for the selected node.
   - System: Provide the required system type, operating system, and version, and optional manufacturer, model, and owner name and phone number. If a primary node name is entered on the Network tab, the system type, OS type, and OS version are automatically filled in for you. See System type, OS type, and OS version information for details.
   - Tools: Associate a tool or tools with a node or node group.
   - Node Groups: Shows all node groups that contain the selected node.
   - Outage: View outage information for the selected node.
7. Click Apply as you finish with a tab to apply your changes.
8. Click OK to confirm your changes and close this dialog box.

9. Click Cancel to close the dialog box without saving your changes.

To delete a node

1. Select the node you want to delete in the Configure Managed Nodes dialog list.
2. Right-click to open the shortcut menu. Select Delete .

Create new node groups

A node group is a logical group of internal and external nodes managed by operators. The administrator can apply a consistent set of tools, reports and graphs, and policies to this logical group. A single node can belong to many groups.

To add a node group

1. Open the Node Configuration editor if it is not already open.
2. From within the Configure Managed Nodes dialog box, select the Nodes folder in the managed nodes list.
3. Right-click to open the shortcut menu.
4. Select New Node Group to open the Node Group Properties dialog box which you use to configure the new node group. The General tab displays by default.
5. Configure the new node group as necessary using the tabs in the Node Group Properties dialog box:
   - Node group General information properties
   - Node group Tools properties
   - Node group Reports and Graphs information properties
   - Node group Deployment properties
6. Click Apply as you finish with a tab to apply your changes.
7. Click OK to confirm your changes and close this dialog box.
8. Click Cancel to close the dialog box without saving your changes.

To delete a node group

1. Select the node group you want to delete in the Configure Managed Nodes dialog list.
2. Right-click to open the shortcut menu. Select Delete .

Check prerequisites for managed nodes

Prerequisite checking tests a node's manageability before it is brought under management. You can run a prerequisite check in the following ways:

- Automatically

  When you install an agent, the installation process automatically performs a prerequisite check, unless you clear the Run prerequisites check automatically during job execution check box in the Agent Installation dialog box.

- Manually

  You can start a prerequisite manually by selecting Run Prerequisite Check in the shortcut menu in the Configure Managed Nodes window, or by running the command line tool ovowreqcheck .

To launch the prerequisite check manually

1. Open the Node Configuration editor, if it is not already open.
2. Select a node or nodes in the managed nodes list.
3. Right-click to open the context menu.
4. Click Run Prerequisite Check .

To view the results of the prerequisite check

1. When the prerequisite check begins, the Prerequisite Check Component dialog box opens and displays the following information:
   - Status: The Status column indicates the status for each listed node.

     View status options
   - Node: The Node column displays the fully-qualified domain name or IP address of the node to be checked.
   - Result: The Result column gives a brief description of the results of the check. You can see further details by clicking on the individual node name to display information in the Description box, which displays both requirements and recommendations.

     View typical results messages
2. When the prerequisite check completes, click OK to close the dialog box.

To see if the check passed or failed

1. From the console tree, select the node on which the check was run.
2. Right-click to open the context menu and select Properties to open the Properties dialog box.
3. Select the System tab and view the Prerequisite Check Passed check box for the status of the prerequisite check.

You can get the same information in the Node Configuration editor by selecting a node in the list of managed nodes and right-clicking on a name. Select Properties from the context menu and then the System tab in the Node Properties dialog box. The results of the prerequisite check appear in this dialog.

To restart the prerequisite check

If no check is currently running, you can restart the check after you have corrected any problems encountered by the original prerequisite check.

1. In the Prerequisite Check Component dialog box, select the nodes on which you want to run the check again. You can select more than one node.
2. Right-click to open the context menu.
3. Select Restart . The prerequisite check will run again on the selected nodes.
4. To cancel the check, right-click to open the context menu and select Cancel .
5. To copy the contents of the node list, right-click to open the context menu and select Copy .

To sort the columns

If no prerequisite check is in progress, you can sort by status, node, and result columns by clicking on the column. For example, clicking the Status column arranges the nodes according to status such as passed or failed.

Requirements not checked

The following requirements must be checked manually on DCE agents:

- On Tru64 systems :

  Required patches on TruCluster

  DCE on TruCluster
- On Linux systems :

  Kernel features
- On HP-UX systems :

  All kernel parameters except for nfile and nflocks

TIP:

To run the prerequisite check on Windows Vista nodes, you must first enable the Remote Registry service on the node. (On Windows Vista nodes, this service is by default disabled.)

Related Topics:

- ovowreqcheck
- ovoreqcheckagt
- Configure System information for managed nodes

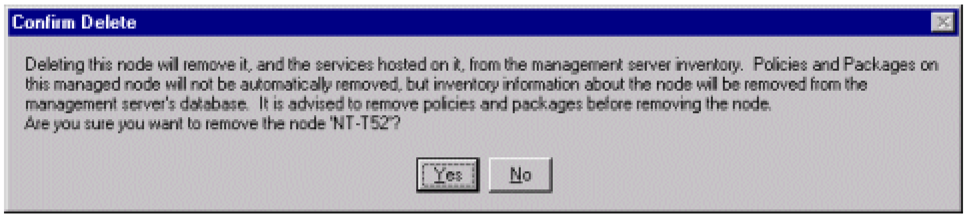
Delete, copy, and move managed nodes

You can easily delete a node from the managed nodes list. You can also move a node to another node group or copy a node to more than one node group.

NOTE:
Node groups with identical names that exist more than once at different locations in the console tree are stored on the server as a single data unit.

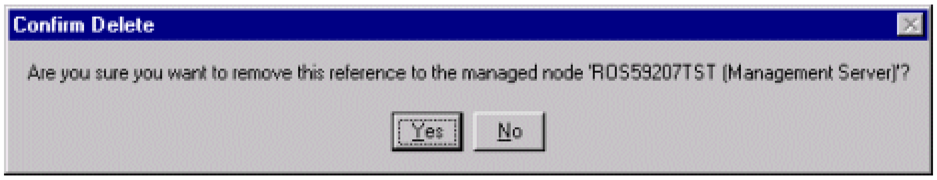
To delete a managed node

1. Open the Node Configuration editor if it is not already open.
2. In the Nodes list, either select the name of the node or node group you want to delete and press the Delete key or right-click to open the shortcut menu and select Delete . If you are deleting the only instance of the node, a message box opens to explain what happens when nodes are deleted and how to handle policies and packages on the deleted node.


Confirm Delete

Deleting this node will remove it, and the services hosted on it, from the management server inventory. Policies and Packages on this managed node will not be automatically removed, but inventory information about the node will be removed from the management server's database. It is advised to remove policies and packages before removing the node.
Are you sure you want to remove the node 'NT-T52'?

If you are deleting a copy (shortcut to the node), the following message box appears.


Confirm Delete

Are you sure you want to remove this reference to the managed node 'ROS59207TST (Management Server)'?

3. Click Yes to continue the delete operation. The selected node is removed from the list of managed nodes.
4. To cancel the delete operation, click No .

To move a managed node

1. Open the Node Configuration editor, as explained above in Step 1.

2. Right-click the name of the node you want to move to another node group to open the shortcut menu.
3. Click Cut to remove the node from the selected group.
4. Select the node group you want to move the node to.
5. Click Paste Shortcut to place the node in the selected group.

   If you decide to delete this node, you will see the same error message shown in Step 2 of To Delete a Managed Node.

To copy a node into one or more additional node groups

1. Open the Node Configuration editor.
2. Right-click the name of the node you want to copy to another node group to open the shortcut menu.
3. Click Copy .
4. Select the node group to which you want to copy the node.
5. Right-click to open the shortcut menu.
6. Click Paste Shortcut . The node appears in the second node group.

   If you decide to delete this node, you will not receive a message. You are deleting a shortcut to the node, not the node itself, which still exists in the original node group.

Change names and IP addresses

You may find it necessary to change the names and IP addresses of various system components, such as the management server or managed nodes. You can:

- Change the IP Address of a node
- Change the Fully-Qualified Domain Name (FQDN) of a node
- Change the IP address of the management server
- Change the Fully -Qualified Domain Name (FQDN) of the management server
- Change the management server in a cluster
- Change the FQDN or IP address in a flexible management environment

Change the IP address of a managed node

Follow these steps on the managed node:

1. For DCE nodes only, check to see if the OPC_IP_ADDRESS setting is set in the opcinfo or nodeinfo file. The location of this file will be different, depending on your operating system.

   NOTE:
   HPOM does not set the OPC_IP_ADDRESS setting per default. If the OPC_IP_ADDRESS is manually created, be sure to update it.

2. Change the IP address, and reboot if required by your OS.
3. If you are changing the IP address of a DCE node and a reboot is not required by the OS, stop and restart the agent using opcagt -kill and opcagt -start so that the DCE agent uses the new IP address.
4. Check the Network tab of the Node Properties dialog box for the node:

   If Notify management server if node communication address changes is selected, then no further action should be required. Check to see if the new IP address is shown in the IP address field. If not, correct the IP address.
5. If Notify management server if node communication address changes is not selected, but IP address is selected, then you need to correct the IP address manually.
6. If Domain Name(FQDN) is selected, verify that the old name resolves to the new address and leave the setting unchanged.

   NOTE:
   By default, after the management server resolves a node's domain name to an IP address, it stores this IP address in its node cache to increase performance. If Domain Name(FQDN) is selected,

but Notify management server if node communication address changes is not selected, and you then change the node's IP address, the management server might continue to use the old IP address from its cache. To rebuild the management server's node cache, restart the OvEpMessageActionServer service. (For more details, see Optimize HPOM node name resolution .)

Change the Fully-Qualified Domain Name (FQDN) of a managed node

Follow these steps on the managed node to change the FQDN. During this process you may see several error messages stating that the agent cannot read the encrypted policies. After you have changed the name, reinstall the agent packages on the node to resolve the problem.

1. Change the name of the node and reboot if required by your OS.
2. *On Windows nodes with DCE agents only.* Change the name of the node in the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\OpenView\ITO\Hostname`
3. Check the Network tab of the Node Properties dialog box for the node:

   If the old name of the node was specified in the Primary node name field, correct it.

   If Notify management server if node communication address changes is selected, then no further action should be required. Check to see if the new node name is shown in the Domain Name(FQDN) field. If not, correct the name.

   If Notify management server if node communication address changes is not selected, but Domain Name(FQDN) is selected, then you need to correct the name manually. Make sure that the new name resolves to the IP address of the node.
4. Check also the Message Identification tab to see if the old name of the node was specified there. If yes, correct it.
5. *Optional.* Change the display name of the node in the General tab.
6. *For DCE nodes only.* The agent encrypts several files using its short node name. If this changes, then the agent cannot read the encrypted files (policies, agent registration) anymore. Therefore the agent must be reinstalled.
   - On Unix nodes: reinstall the agent manually.
   - On all nodes: run All Tasks →Reinstall/Update... . In the Reinstall / Update Options dialog box, select Reinstall and Scope: All to update the policies. This also reinstalls the agent on Windows nodes.

Change the IP Address of the management server

NOTE:
The management server does not support running with dynamic IP address allocation (DCHP).

1. If the IP address of the management server is used in server-to-node communication, open the properties of the node and change the IP address in the Network tab.
2. Change the IP address, and reboot if required by your operating system.
3. After you have changed the IP address, the old license cannot be used any more. You can temporarily use the 60-day trial license. However, you should move the license to the new IP address as soon as possible. Visit http://webware.hp.com/welcome.asp (→ Move license to new server). You will need the original HP order number, the old IP address of the management server, the new IP address and the new hostname to successfully order a new license.

Change the Fully-Qualified Domain Name(FQDN) of the management server

1. Deploy a flexible management policy to the nodes from the current management server to configure the management server's new name as a secondary management server of the nodes.

   For more details, see Configure action-allowed and secondary managers .
2. Prepare all agents. To switch the name of the management server on every node, go to Tools → HP Operations Manager Tools and use the Switch Management Server tool.

   Each node must be able to resolve the new management server name. For details, see the help topic Resolve the IP address of the management server .
3. Stop all agent processes on the management server:

   ```
   opcagt -kill
   ```
4. Exit all consoles and stop all management server processes:

   ```
   vpstat -3 -r STOP
   ```
5. Open a command prompt on the management server and enter:

   ```
   ovconfchg -edit
   ```

   A text editor starts and displays the settings file.
6. Search for all occurrences of the old management server name and replace them with the new name. Then save your changes and close the text editor.
7. Use regedit to update the following registry keys. Change the old management server name to the new one:
   - `HKLM\SOFTWARE\Hewlett-Packard\OVEnterprise\ManagementServer\DBAccess\OvOWInstance`
   - `HKLM\SOFTWARE\Hewlett-Packard\OVEnterprise\Management Server\MsgActSrv\MGMT_SERVER`
   - `HKLM\SOFTWARE\Hewlett-Packard\OVEnterprise\Management Server\MsgActSrv\NAMESRV_LOCAL_NAME`
8. Change the hostname and reboot the system.

9. Start the console and connect to the new management server name.
10. Check the Network tab of the Node Properties dialog box for the management server:

    If the old name of the management server was specified in the Primary node name field, correct it.

    Check also the Message Identification tab to see if the old name of the management server was specified there. If yes, correct it.

    If Domain Name(FQDN) is selected, verify that the new name resolves to the IP address of the management server.

    Optional. Change the display name of the management server on the General tab.
11. Update the Windows DCE agent package with the new server name:

    Open a command prompt and enter:

    SetMgmtServer

    The output will be similar to this:

    ```
    setmgmtserver

    Agent package 'C:\Program Files\HP

    OpenView\\packages\Windows\OvEpMsgActAgt.FM'

    Management server name was updated from 'old_serv.rose.hp.com'

    to 'hpom_serv.rose.hp.com'

    SYSTEM account installation: enabled

    Authentication enforcement:  disabled

    Forced user account switch:  disabled
    ```
12. In the flexible management policy editor, delete the old management server name from the flexible management policy, and redeploy the policy.

Change the Fully-Qualified Domain Name (FQDN) of the management server in a cluster

1. Before performing a change in the IP address, check to see whether any node is using the variable OPC_RESOLVE_IP to get the IP address of the management server. If so, those nodes have to be updated before performing the change on the management server. If the host name of the management server changes, be sure to also switch the management server name on all managed nodes using the "Switch Management Server" tool located in Tools → HP Operations Manager

Tools

Each node must be able to resolve the new management server name. For details, see the help topic Resolve the IP address of the management server .

2. Update the following registry keys with the virtual server name on all physical systems:
   - `HKLM\Software\Hewlett-Packard\OVEnterprise\Management Server\MsgActSrv\MGMT_SERVER`
   - `HKLM\Software\Hewlett-Packard\OVEnterprise\Management Server\MsgActSrv\NAMESRV_LOCAL_NAME`
   - `HKLM\Software\Hewlett-Packard\OVEnterprise\Agent\ManagementServer`
   - `HKLM\Software\Hewlett-Packard\OpenView\Common\MgmtServerName`
3. Update the registry key `HKLM\Software\Hewlett-Packard\OpenView\Common\MgmtServerIP` with the virtual server IP address.
4. *Optional.* If HP Reporter is installed, update the server name on all physical systems:
   a. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
   b. click Namespaces , and then click Reporter Integration . A list of values appears.
   c. Update the value Server name with the virtual server name.
   d. Click Apply .
5. Start the Cluster Administrator from Administrative Tools .
6. Select the HPOM group and take it offline.
7. If the IP address, host name, or both have changed, open the HPOM IP Address resource or the HPOM Network Name resource and change the parameters of the resource accordingly.
8. Run the following command:

   ```
   SetMgmtServer.exe /servername <virtual HPOM server name>
   ```
9. In the Cluster Administrator, take the HPOM group online again.
10. If not only the virtual IP/host name changed, but also the IP/host name of the physical hosts changed, follow the steps shown in Change the Fully-Qualified Domain Name (FQDN) of a node .

Change the FQDN or IP address in a flexible management environment

- Change the management server

  The basic procedures to change the IP address or Fully-Qualified Domain Name (FQDN) of a management server in a flexible management environment are the same as for standalone servers. In addition, update the flexible management policies with the new IP address or FQDN of the management server and deploy them as necessary.

- Change the managed nodes

  If Notify management server if node communication address changes is not selected in the node's properties, you must manually update the node's properties on all servers in the environment. The easiest way to achieve this is to modify the node properties on one management server, and then use the ovpmutil tool to download and upload the node information.

Agent ID on server and node mismatch

In the event that the agent ID on the server and the node do not match, you may see error messages that refer to "agent ID mismatch", and you will notice communication problems between the managed node and the management server . This can occur if you delete a node and then add it back again.

If you know the correct agent ID, open the Node Configuration Editor and type it in the Modify Agent ID box in the Advanced Configuration dialog box. If you do not know the agent ID, you can correct an agent ID mismatch with this procedure:

To correct an agent ID mismatch

1. As a user with administrative privileges, stop the agent using this command:
   `opcagt -kill`
2. On the managed node, delete the files:
   - On AIX: `/var/lpp/OV/conf/OpC/agentid`
     and if present: `/var/lpp/OV/conf/OpC/managedNodeId.txt`
   - On HP-UX, Linux, Sun Solaris, and Tru64 UNIX: `/var/opt/OV/conf/OpC/agentid`
     and if present: `/var/opt/OV/conf/OpC/managedNodeId.txt`
   - On Windows: `%OvAgentDir%\conf\OpC\agentid`
     and if present: `%OvAgentDir%\conf\OpC\managedNodeId.txt`
3. On the managed node, delete the OPC_AGENT_ID<agentGUID> from the nodeinfo file using a text editor. The nodeinfo file is located in `conf/OpC/nodeinfo` .

   For example, you would delete the text `OPC_AGENT_ID617a0010g90f-71d7-0666-0f0899d4000` up to the carriage return symbol. Do not delete the `OPC_NODE_TYPE` that appears after the carriage return symbol or any text that follows the `OPC_NODE_TYPE` .
4. As a user with administrative privileges, restart the agent using this command:
   `opcagt -start`
5. On the management server, open the Node Configuration editor and select the node with the mismatched agent ID. Right-click the node and select Properties → General → Advanced Configuration . In the dialog box, select Modify Agent ID , then remove the agent ID.
6. Deploy any policy to the managed node. A new agent ID will be created.

Handling of messages with no or an empty agent ID

If the node name or IP address of the node matches a node that is known to the management server, messages are accepted even if they have no agent ID or if the agent ID is set to null. This may be the case if older versions of the agent are attempting to send messages to the management server. This behavior can be changed by modifying values in the Server Configuration dialog box.

In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.

In the namespace Message Action Server Message Filter , you can change the following values:

- Discard messages with empty agent ID
  If this value is set to TRUE, all messages that come from an agent that does not yet have an agent ID assigned are discarded. The default value is FALSE.
- Discard messages with no agent ID
  If this value is set to TRUE, all messages that come from an old agent (for example VPO 5.x, VPO 6.x, VPW 6.x) that has no agent ID functionality are discarded. The default value is FALSE.
- Allow actions in messages with no agent ID
  If this value is set to TRUE, all messages coming from an old agent (for example VPO 5.x, VPO 6.x, VPW 6.x) are not preprocessed. That means, the operator-initiated and the automatic commands being part of the message will not be removed. The default value is FALSE.

Related Topics:

- Change server configuration values

Resolve the IP address of the management server

The information reported on the management server and on the managed node must match. If you are using DNS in your environment, you must modify your DNS server configuration to ensure that the information matches. Follow this procedure on the management server and the managed node to find out if name resolution is properly configured.

1. On the management server, determine the DNS domain and IP of the management server by typing the following command:

   ```
   ipconfig /all
   ```

   Note the following information:

   ```
   Host Name :kurbis

   Primary DNS Suffix :veg.com

   ...

   IP Address :204.174.18.152

   ...
   ```

2. On Windows 2003 managed nodes, resolve hostname with DNS by typing these commands:

   a. Purge the DNS cache by typing:

   ```
   ipconfig /flushdns
   ```

   b. `ping kurbis`

   c. Display the DNS resolution cache by typing this command:

   ```
   ipconfig /displaydns
   ```

   Note the following information:

   ```
   ...

   scavenger

   Record Name: kurbis.veg.com

   Record Type: 1

   Time To Live: 42560

   Data Length: 4
   ```

```
Section: Answer

A (Host) Record: 204.174.18.152

...
```

3. If you are using WINS, check the following:
   - Purge the WINS cache by typing this command:

     ```
     nbtstat -R
     ```

     Successful purge and preload of the NBT Remote Cache Name Table.
   - `ping "kurbis "`

     The quotes and space behind the name are required to force resolution through WINS.
   - Display the WINS name resolution cache by typing this command:

     ```
     nbtstat -c
     ```

     Note the following information:

     ```
     Node IpAddress: [15.136.3.33] Scope Id: []
     ```

NetBIOS Remote Cache Name Table

Name	Type	Host Address	Life (sec)
kurbis <00>	UNIQUE	204.174.18.152	567

If the NetBIOS name is found and IP equals management server, then the node can resolve the IP address through WINS. If not, you must modify the configuration of your WINS server to ensure that it matches.

Add node to node group after upgrading from Windows 2000 to Windows 2003

After you upgrade a node from Windows 2000 to Windows 2003, the node does not automatically move to the Windows 2003 node group and the managed node does not appear in the service map. Follow the steps below to have the managed node appear in the service map after a platform upgrade.

To display a node in the service map after a platform upgrade:

1. Manage (Add) a Windows 2000 server node from the HPOM server console. After the node is added, it appears in the service map under "Windows 2000."
2. Upgrade the managed "Windows 2000 server" node to "Windows server 2003."
3. From the HPOM server console, launch the Node Configuration Editor .
4. Navigate to HP Defined Group → Windows → Windows 2000 →<Managed Node >
5. Select the managed node; right-click to open the context menu.
6. Select Properties to open the Node Properties dialog box.
7. Click the System tab.
8. From the Operating System list, select Windows Server 2003.
9. From the Version list, select 5.2.
10. Click OK to add the managed node to the Windows Server 2003 node group.
11. Remove the two discovery policies WINOSSPI-MSWINSys_AutoDiscovery and WINOSSPI-MSWINApp_AutoDiscovery from the managed node.
12. Go to the Configure Services dialog box and delete the old services for the node.
13. Deploy the two discovery policies WINOSSPI-MSWINSys_AutoDiscovery and WINOSSPI-MSWINApp_AutoDiscovery to the managed node.

Discover managed nodes

HP Operations Manager for Windows automatically discovers Windows and UNIX nodes in your enterprise environment and displays their names in a list of discovered domains in the Configure Managed Nodes dialog box. Click the + sign beside any domain name in the DNS folder to expand the list and see the nodes it contains.

Discovery is dynamic, so that when a domain is discovered, new nodes are automatically included in the Discovered Nodes list. When you first expand the domain, the new nodes appear in the list. The list is updated each time the Node Configuration editor is reopened. An asterisk (*) indicates required information.

To quickly locate a specific node in long lists, expand the domain (for example, DNS) in the console tree and type the name of the node you want to find. As you type, the system locates the node for you. This method works within the Configure Managed Nodes dialog box also. In either the left or right pane, expand the node list and begin typing the name of the node you want to find. It will appear selected in the list of nodes.

To configure DNS discovery

1. Select the Nodes folder in the console tree.
2. Click to open the Configure Managed Nodes dialog box. Discovered nodes display in the Discovered Nodes list.
3. To add a new domain, select the DNS folder in the Discovered Nodes list and right-click to open the shortcut menu.
4. Select New Domain to open the DNS Domain Properties dialog box.
5. In the Domain Name box, enter the name of the domain as you want it to appear in the Discovered Nodes list.
6. In the Domain Server box, enter the DNS server name for the domain you are adding.
7. Click OK to confirm your changes and close this dialog box.

NOTE:
In a DNS/ADS environment, it is possible for the Windows domain controller to register a few A records (phantom nodes) in the DNS server, which may cause phantom nodes to appear in the Configure Nodes dialog box. These are aliases; it is not recommended that you bring these under management. To manage an ADS environment, you should use ADS discovery, where these aliases will not be visible. For more detail, see the ADS discovery example .

To edit an existing domain

1. In the Discovered Nodes List , select the name of the domain you want to edit.
2. Right-click to open the shortcut menu.
3. Select Properties to open the DNS Domain Properties dialog box.
4. Edit the displayed information.
5. Click OK to save your changes and close this dialog box.

Troubleshooting

To set up HPOM to select discovered DNS nodes through the Configure Managed Nodes editor, you configure the DNS domains you want to view along with the appropriate DNS server for each domain.

If, after adding a new DNS domain, you are having trouble expanding your view of the domain in the interface, follow these steps to make sure your configuration is correct:

1. Verify that you have specified the correct DNS server.
2. Verify that the DNS server is set to Allow Zone Transfers. If you receive "Discovery failed" errors, the DNS server may not be configured to Allow Zone Transfers.
3. Run the tool nslookup as follows to verify that you can see the nodes:

   ```
   c:\>nslookup
   >server
   >ls
   ```

4. If you are unable to see the nodes through nslookup (that is, you receive "Query Refused" or "Non-existent domain"), then verify your basic DNS configuration or Active Directory configuration by following the procedures in your Windows Server documentation. The Microsoft Windows documentation provides examples of configurations that could affect your ability to view new nodes.

Related Topics:

- SNMP discovery
- ADS discovery example

SNMP discovery

In order for system type, OS type, and OS version to be automatically determined when a node is put under management, the managed node must meet following requirements:

- All managed nodes
  - An SNMP service/daemon must be running on the managed node.
  - At least the management server must have access permissions to the SNMP object "system" on the managed node.

    Some operating systems (for example, Sun Solaris) allow limiting access to the "system&quot SNMP object (for example, localhost only), which can prevent the management server from determining the information. Consult the SNMP manuals for information about a specific operating system.
- Windows managed nodes
  - Must have at least READ ONLY access to the community name "public".
  - The option "Accept SNMP packets from any host" must be enabled.
- Linux managed nodes

  Because the SNMP daemon does not give you the exact distribution version of the Linux OS, HPOM performs an additional check using a telnet protocol. After HPOM determines (through the SNMP daemon) that the system is a Linux OS, it performs a telnet emulation to obtain information from the telnet welcome string.

  You can easily change the telnet welcome string on a Linux system so that if the Configure Nodes dialog box reports that the OS version of a Linux system could not be determined, you do not need to supply it manually.

If the system type, OS type, and OS version cannot be determined automatically, you can specify it manually.

NOTE:
It is important to correctly supply the system type, OS type, and OS version information. If you do not, you may see attempts to deploy the agent packages of the wrong architecture or errors related to auto-deployment of policies of the wrong architecture.

Change the SNMP community name

By default, the management server uses the "public" community name to connect to SNMP on a target node and get the system type, OS type, and OS version information. If this community name is changed to something other than "public" on nodes that are to be managed, the management server needs to be informed about this community name in the following way:

1. Start Notepad.
2. Write this line to the empty Notepad document:

   `SNMPCommunityName=<your_community_name>`

   Replace <your_community_name> with the community name that you are using. For example, to use "private" as the community name, specify `SNMPCommunityName=Private` .
3. Choose File → Save As . In the Save As dialog box, navigate to the `%OvShareDir%\conf\DNSDisc` directory and save the file as follows:

   File name: dnsdscr.ini

   Save as types: Text documents (*.txt)

   Encoding: ANSI

To revert to using the "public" community name, do one of the following:

- Change the above line in the `dnsdscr.ini` file to:

  `SNMPCommunityName=public`
- Delete the file `dnsdscr.ini` from the `%OvShareDir%\conf\DNSDisc` directory

Manage intelligent devices

Using SNMP, you can manage intelligent SNMP devices such as printers, routers, and computers with unsupported operating systems and manage them from HPOM.

Related Topics:

- Discover managed nodes
- Manage intelligent devices

Specify IP Range for DNS discovery

HPOM provides dynamic discovery of Windows and UNIX nodes in your enterprise environment and displays their names in a list of discovered domains in the Configure Managed Nodes dialog box. Click the + sign beside any domain name in the DNS folder to expand the list and see the nodes it contains.

You can create a new list of nodes by specifying a range of IP addresses to be included in the list.

To include a range of IP addresses

1. To specify a range of IP addresses to include, select the DNS folder and right-click to open the shortcut menu.
2. Select New IP Range to open the IP Range Properties dialog box.
3. In the Name box, enter a name to identify the range of IP addresses you are creating, as you want it to appear in the Discovered Nodes list.
4. In the Enter the Range of IP Addresses to Include boxes, enter a range of IP addresses. Only nodes with the specified IP addresses will appear in the Discovered Nodes list when the item is expanded.
5. Click OK to confirm your changes and close this dialog box.

To edit the IP range

1. In the Discovered Nodes List , right-click the IP range you want to edit and select Property to edit the IP range.
2. Click OK to save your changes and close this dialog box.

Filter domain view

You can filter the view of a large domain by specifying criteria for nodes that you want to group into a logical display. Nodes that fit the criteria are grouped into a folder under the selected domain in the Configure Managed Nodes dialog box, making it easy to see which nodes belong to a particular system type, OS type, and version.

To specify filter criteria

1. Open the Node Configuration editor if it is not already open.
2. Click the + sign beside a network folder to expand it and display a list of domains.
3. Right-click a domain name to open the shortcut menu.
4. Select New Discovery Folder to open the Discovered Node Folder dialog box tab.
5. In the Display Name box, type a name for the domain filter folder.
6. In the Node name contains box, type any text that the names of the nodes you want to filter for have in common. For example, if several of your node names contained the text "murr", nodes named "murr59544", "testmurr59544", and "murr" would all be listed in the filter folder.
7. In the IP Address box, choose Include Range , Exclude Range , or Any . This IP range is different from the IP range set in the IP Range Properties dialog box and is created and modified only within this property page.
8. In the Platform tab, select the System Type you want to filter for.

   The Operating System and Version selections are only available if the HP NNM Adapter is installed on your management server. Otherwise, they are dimmed and unavailable.
9. Select one of the options for version selection. The default when an OS version other than "Any" or "Unknown" is selected is Include only the selected version of the operating system.
10. Click Apply to see the results of your changes.
11. Click OK to confirm your selections and close this dialog box.

You will see a message box saying that the system is retrieving nodes in the selected domain. A counter tallies nodes that meet the filter criteria as they are discovered. A folder with the filter name you specified appears in the Discovered Nodes pane under the selected node.

To cancel the discovery operation, right-click in the Discovered Nodes pane and select Cancel Discovery .

To delete a selected group of filtered nodes, right-click to open the shortcut menu and select Delete .

ADS discovery example

When enumerating a DNS server that serves an ADS domain, some phantom nodes seem to appear in the Configure Nodes dialog box. For example, in expanding the domain demonet.com, among the normal nodes displayed were the nodes demonet.com and gc._msdcs.demonet.com. An excerpt from the nslookup output for that DNS server is shown below:

Nodes	Record Type	IP Address
demonet.com	A	211.70.73.161
demonet.com	A	211.70.73.68
demonet.com	A	211.70.73.139
demonet.com	NS	server = gilligan.demonet.com
demonet.com	NS	server = professor.mobile.demonet.com
demonet.com	NS	server = luvie.demonet.com
demonet.com	NS	server = maryanne.demonet.com
gc._msdcs	A	211.70.73.144
gc._msdcs	A	211.70.73.68
gc._msdcs	A	211.70.73.139
gc._msdcs	A	211.70.73.161
gc._msdcs	A	211.70.73.4
maryanne	A	211.70.73.161
luvy	A	211.70.73.68
gilligan	A	211.70.73.139

Both demonet.com and gc._msdcs.demonet.com are present as "A" records in the DNS server, which means that the server treats them as normal nodes. In reality, these nodes are aliases to a real DNS node, a domain controller. The demonet.com record of type "A" is an alias to the IP of the domain controller. The gc._msdcs.demonnet.com "A" record is an alias to the IP of the node that takes care of the ADS Global Catalog (again, a domain controller.)

The situation is one of different DNS names but the same IP, which is permissible. While it is possible to bring these nodes under management, it is not recommended. The effective way to manage your ADS environment is to use ADS discovery, where these aliases are not visible.

Configuring node information

When you bring a node under management, as part of the process you must specify detailed information about that node.

Related topics:

- Configure General information for managed nodes
- Configure System configuration for managed nodes
- Configure Network configuration for managed nodes
- Configure Outage information for managed nodes
- Configure Messages configuration for managed nodes
- Configure Tools configuration for managed nodes
- View node group information

Configure general information for managed nodes

Each managed node must be uniquely identified by specifying properties for that node. Use the General tab in the Node Properties dialog box to specify details that identify each managed node. Required information is specified.

1. In the list of managed nodes, select the node you want to configure.
2. Right-click to open the shortcut menu.
3. Select Properties to open the Nodes Properties dialog box, which displays the General tab by default. The automatically generated unique ID (GUID) for the selected node appears at the top of the dialog box.
4. In the Display Name box, enter the label (display name) for the managed node. The caption appears in the Managed Nodes list. This information is required and is automatically entered if you selected the node to be managed by dragging and dropping the node name from the Discovered Nodes list to the Nodes list in the Configure Managed Nodes dialog box.
5. Enter any comments or additional information in the Description box. This information is optional.
6. Enter the name of the node owner or administrator in the Owner Name box. For example, enter the name of the critical person to contact. This information is optional.
7. In the Contact Details box, enter the owner's phone number, pager number, or email address for the person you specified in the Owner Name box. This information is optional.
8. In the Manufacturer box, enter the hardware maker's name. For example, you might enter HP. This information is optional.
9. In the Model box, enter the model name or number of the selected system. For example, you might enter Kayak XA for an HP system. This information is optional.
10. Click Advanced Configuration to open the Advanced Configuration dialog box, which displays the following information:
    - Modify Agent ID: The GUID of the agent that resides on this node. If you change this ID, you will see a message warning you that you may no longer receive messages if the agent ID is incorrect .
    - Modify Certificate State: Shows whether the HTTPS node has the certificates it requires to communicate securely with the management server. You only need to update the certificate state manually to troubleshoot certificate problems (for example, if you deploy certificates to this node manually, but network problems prevent the agent from notifying the management server of this).

      View certificate states
11. Click Apply to apply your changes.
12. Click OK to apply your changes and close this dialog box.

13. Click Cancel to close this dialog box without saving your changes.
14. Select the Network tab to continue configuring this node.

Configure network information for managed nodes

Use the Network tab of the Node Properties dialog box to specify the primary node name, communication path, and domain name. Required information is specified.

1. From the list of managed nodes in the Configure Managed Nodes dialog box, select the node you want to configure.
2. Right-click to open the shortcut menu.
3. Select Properties to open the Node Properties dialog box.
4. Select the Network tab.
5. In the Primary Node Name box, enter a unique node name. This information is required and is automatically entered if you selected the node to be managed by dragging and dropping the node name from the Discovered Nodes list to the Nodes list in the Configure Managed Nodes dialog box.

   When this information is entered or changed, the System Type , OS Type , and OS Version are automatically modified in the System tab.

   NOTE:
   A problem may arise when the primary node name entered here (for example, rednode.somedomain.hp.com) does not match the network settings on the agent node, which may not have a primary domain name set. In this case, a short node name (for example, rednode) may be filled in by the HPOM message infrastructure in the node field of operator-initiated actions. If this occurs, the operator-initiated action fails because HPOM cannot match the short name from the message to the fully-qualified name that is in the node database. HPOM reports that it does not know the node mentioned in the message.

6. Specify your preferences in the Server To Node Communications box. Check Node IP address obtained automatically (DHCP) if you do not want to use a static IP address.

   The server will not cache an IP address it once received for a system, but will do a name resolution every time a system is contacted. This means that if you specify the name of a system as the communication path value, and you have checked this box, the given name is always resolved (using an external name resolution service like DNS) if the node is contacted.

   NOTE:
   If you do not check this box, the server caches the first IP address it received for a node from a name resolution service. On subsequent contacts, the server uses the cached IP address instead of doing another external name resolution. This increases performance for nodes using a static IP address.

   Check Notify management server if node communication address changes if you want the server to be aware of IP address changes.

The agent checks each time you start the system to see if the IP address of the system you are running on has changed. If the answer is yes, this information is sent to the management server, where the communication path of the corresponding node is updated with the new IP address.

NOTE:
If the address space of the managed node is different from the address space of the management server (because they are located in different subnets), it is possible that the management server cannot use the new IP address it received from an agent. In this case, to prevent errors, do not check the Notify management server if node communication address changes check box.

If you check this box, selections below it in the dialog box appear dimmed and are unavailable.

7. If you did not make a selection in the Server to Node Communications box, click either the IP Address or Domain Name FQDN button to specify a communications path. If you select IP Address , you must enter the IP address manually.

   By default, the Domain Name (FQDN) box displays the primary node name. If you change the primary node name, the change is reflected in the Domain Name (FQDN) box. You can also enter the domain name for the node you are configuring in the space provided. The domain can be either a Windows or IP domain. This information is optional.

   If you have multiple IP addresses for a particular node and the communications path would differ from the Primary Node Name, specify the particular IP address or DNS name that you want.

8. Heartbeat Polling sends a signal to the managed node or contacts the agent on the node to check whether the node is offline.

   System Default means that the management server uses the heartbeat polling setting from the Server Configuration dialog box. This setting is valid for all managed nodes and is by default "ICMP & Agent". To override the system default for a managed node, set Polling to Custom , then specify the Ping Protocol you want to use:

   - ICMP & Agent

     With this option, the server first attempts to contact the node using ICMP packages to find out if the node is reachable. If this succeeds, it will contact the agent on the node to find out if the agent processes are running. When this fails, it will use ICMP packages again to find out if, at least, the system is alive. As soon as this succeeds, the agent is contacted again. This option is not recommended for nodes outside of a firewall because ICMP calls are usually blocked by firewalls.

   - Agent Only

     The management server does not actively contact the node with ICMP pings, but still contacts the agent on the node. This is the recommended setting for nodes outside of a firewall. The disadvantage is that in the event of a system outage, the network load is higher than with normal heartbeat monitoring because the agent connection is still being tried.

   - ICMP Only

The management server sends ping packages (using ICMP) to verify the availability of the agent. This option is not recommended for nodes outside a firewall because ICMP calls are usually blocked by firewalls.

9. Set the polling Interval in seconds. This is the interval at which the management server checks whether the managed node is offline. The minimum is polling interval is 60 seconds, and the maximum is 28800 (8 hours).
10. Clear Enable Auto Deployment if you do not want HPOM to automatically deploy policies to the node. HPOM automatically deploys policies to a node when the agent is correctly installed. For HTTPS agents, a correctly installed certificate is also required; otherwise the deployment job fails.

    By default, HPOM automatically deploys certain core policies to nodes. The core policies include autodiscovery policies that gather service information on nodes. This information is sent back to the management server to generate a service tree. (You can also automatically deploy additional groups of policies by associating policy groups with node groups and service types.)
11. Click Apply to apply your changes.
12. Click OK to apply your changes and close this dialog box.
13. Click Cancel to close this dialog box without saving your changes.
14. Select the Messages tab to continue configuring this node.

Related Topics:

- Select nodes to be managed
- Agent health checks
- Disable policy autodeployment

Configure messages information for managed nodes

In some environments, a node can be known by several names in addition to the primary node name. The Message Identification tab of the Properties dialog box displays any other names (aliases)that apply to the selected node.

To create an alias for a node

1. From the Message Identification tab, click Add to open the Add New Alias dialog box.
2. In the New Alias box, enter any other names for the node that apply to it. This allows messages to be generated using any of these names. Any aliases are listed in the Other names for node in browser messages: group box.

   If a node has multiple IP addresses, the IP addresses can also be entered in the New Alias box. This information is optional.
3. To remove an alias, select it from the list and click Remove
4. Click Apply to apply your changes without closing this dialog box.
5. Click OK to confirm your choices and close this dialog box.
6. Select the System tab to continue configuring this node.

Configure system information for managed nodes

Use the System tab of the Managed Nodes Properties dialog box to record information about the selected node's hardware, software, and environment. You need to separately configure details for each managed node. Required information includes System Type , OS Type , and OS Version , which HPOM fills in for you, if possible.

NOTE:
Information for System Type, OS Type, and OS Version is automatically entered if the node was selected by dragging or copying it from the Discovered Nodes list or if the primary node name is entered in the Network tab. This information appears when you open the node's property sheet or select Close or Apply in the Configure Managed Nodes dialog. For manually created nodes, system and OS type discovery is attempted. There is no system or OS information associated with node groups.

1. From the list of managed nodes in the Configure Managed Nodes dialog box, select the node you want to configure.
2. Right-click to open the shortcut menu.
3. Select Properties to open the Node Properties dialog box.
4. Select the System tab.
5. In the System Type box, the system type for the selected node has been automatically entered. If the value cannot be determined, the field will display "Other" and you must enter the correct information. If the node is of a type that is not supported by HPOM for Windows, this box will display a system type of Other.
6. The Agent Comm Type box displays one of two communication types (DCE or HTTPS), depending on the system type.
7. In the Operating System box, the operating system type has been automatically entered. If the value cannot be determined or is unsupported, this box displays an Operating System of Unknown.
8. In the Bit Length box, the bit length has been automatically entered. There are two choices: 32 and 64 bit.
9. The Agent Binary Format box is automatically filled depending on the system type.
10. In the Version box, the information has been automatically entered. If the value cannot be determined or is unsupported, this box remains blank. The list box shows the available versions for the selected operating system. You can select from the list or type in a version not available from the list (for example, a newly released version).
11. Check the Automatically grant certificate: box if you want the management server to automatically

grant certificate requests for this node. The HTTPS agent requires certificates, which enable it to communicate securely with the management server. The node can request these certificates from the management server.

For this property to take effect, you must also configure the management server to grant certificate requests automatically.

12. Certificate State: Shows whether the HTTPS node has the certificates it requires to communicate securely with the management server.

    ▾ View certificate states

13. Prerequisite check: Shows whether the node has passed the prerequisite check. If Passed is not selected, the prerequisite check has not yet run or has failed.
14. Click Apply to apply your changes.
15. Click Close to save your changes and close this dialog box.
16. Click Cancel to close the dialog box without saving your changes.
17. Select the Tools < tab to continue configuring this node.

Configure tools for managed nodes

As administrator, you can specify which tools are available for each managed node. Tools include applications, scripts, and commands that perform necessary tasks in your environment to resolve problems and perform routine tasks and maintenance. Available tools are contained in tool groups in the Tools folder in the HP Operations Manager for Windows console tree. To view tools, select a tools folder to display the tools it contains in the details pane.

When you first open the Tools folder in the console tree, you see the default tool groups supplied with HP Operations Manager for Windows. These might be tools provided by SPIs, applications such as Excel, or scripts that perform such tasks as generating a report or checking TCP/IP status. As administrator, you can create additional scripts and add any other tools you might need to this list to help operators take corrective actions or get additional information.

Using the Tools tab of the Node Properties dialog box, you select the tools you want to associate with this node and make available for selection from the Tools folder in the console tree.

To configure tools for managed nodes

1. From the list of managed nodes in the Configure Managed Nodes dialog box, select the node you want to configure.
2. Right-click to open the shortcut menu.
3. Select Properties to open the Node Properties dialog box.
4. Select the Tools tab. The Tools inherited from node groups box displays a list of tools that are automatically associated with this node because this node is part of one or more node groups. The list shows the name of the tool, the node group the tool is associated with, and a description of the tool.
5. To specify additional tools for this node, click Add to open the Associate Tools with Node dialog box.
6. From the list of tools displayed, select the check box for each tool you want to associate with the node. You can associate multiple tools with a node. Click OK to close the Associate Tools with Node dialog box. The tool or tools you selected are added to the list of available tools in the Tools tab of the Node Properties dialog box.
7. If you want to remove a tool, select the tool from the Tools associated with this node list and click Remove . The tool is removed from the Tools folder in the console tree and is no longer associated with the selected managed node.
8. Click Apply to apply your changes.
9. Click Close to save your changes and close this dialog box.

10. Click Cancel to close the dialog box without saving your changes.
11. Select the Node Groups tab to continue configuring this node.

As administrator, you can create new tools if needed using the Tools Configuration editor, available from the Configuration Toolbar in your console view or the View menu in the console tree.

Configure Outage information for managed nodes

Use the Outage tab of the Managed Node Properties dialog box to view outage information for a managed node. This read-only tab displays the settings established for both unplanned and scheduled outages.

A scheduled outage meets these criteria:

- Planned to happen
- Recurs at regular intervals for maintenance
- Configured by a policy

An unplanned outage is unexpected and can occur randomly.

Only administrators can put a managed node into maintenance mode. Managed nodes in maintenance mode by default do not affect the status of their parent node groups.

To view Outage properties for a selected managed node

1. Right-click the name of the managed node for which you want to display properties; this opens the context menu.
2. Click Properties to open the Properties dialog box for the selected managed node, which displays the General tab by default.
3. Select the Outage tab to view the following information:
   - Current Outage State:

     Displays the outage status of the selected managed node. This status will be either "ON " or "OFF".
   - Unplanned Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage and Heartbeat Polling During Outage . Messages can be either deleted or acknowledged. Heartbeat polling can be set to either "ON " or "OFF".
   - Scheduled Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage and Heartbeat Polling During Outage . Messages can be either deleted or acknowledged. Heartbeat polling can be set to either "ON " or "OFF".

View node group information

The Node Groups tab of the Node Properties dialog box lists the parents of this selected node. Items in the list appear dimmed; this is a read-only list for your information.

System type, OS type, and OS version information

When you configure a managed node, the system type, OS type, and OS version for the node are automatically entered when:

- You configure a managed node by dragging or copying it from the Discovered Nodes list. This is only auto-discovered when you open the node property sheet or close or apply the Configure Managed Nodes dialog box.
- The primary node name is entered in the Network tab.

If the value cannot be determined, the fields display "n/a" and you must specify the correct information. If the node is a type that is not supported by HPOM, the System Type field displays a value of "Other". Supported operating systems are shown on the Software Support Online web site support matrix .

NOTE:
It is important to correctly supply the system type, OS type, and OS version information. If you do not, you may see attempts to deploy the agent packages of the wrong architecture or errors related to auto-deployment of policies of the wrong architecture.

Related Topics:

- SNMP discovery

View policy and package inventory

The console shows an inventory of all policies and packages deployed to managed nodes. The inventory is kept on the management server.

To see the policies or packages for a particular node

1. From the console tree, right-click a node name to open the context menu.
2. Select View → Policy Inventory to display a list of deployed policies for that node in the details pane. Click a policy name to open a read-only version of the policy in the policy editor.
3. Select View →Packages Inventory from the context menu to display a list of deployed packages for the selected node in the details pane.

Configuring external nodes

Nodes for external events are used in HPOM to handle the following types of messages:

- Messages from systems without the need to configure each system in HPOM as a separate managed node. This is useful for managers in an environment that get messages forwarded from lower level managers, from another network, or from system management products such as HP Network Node Manager.
- Messages from systems without an installed agent (SNMP devices, for example).
- Messages from new systems in a certain IP subnet (created from SNMP traps), making it possible to get messages immediately without the need to set them up.

Each external node must be uniquely identified by specifying properties for that node. Use the following property sheets in the External Node Properties dialog box to specify the unique details that identify each external node. Required information is specified.

Configure external nodes using the wizard

To quickly configure a new external node, use the External Node Configuration wizard. This will provide the minimum amount of information you need to bring the external node under management.

NOTE:
At any time you can exit the wizard by clicking Expert Mode . This returns you to the External Node Configuration editor, where you can more completely specify the characteristics of this node. You cannot return to the wizard from the External Node Configuration editor. When you add another external node, the wizard opens again.

To open the wizard

1. Open the Node Configuration editor if it is not already open.
2. From within the Configure Managed Nodes dialog box, select the Nodes folder in the managed nodes list.
3. Right-click to open the shortcut menu.
4. Select New External Node to open the wizard.
5. In the General page, type the Display name in the box provided. You must supply this information to proceed. The caption appears in the External Nodes list. Enter any comments or additional information in the Description box. This information is optional.

6. Click Next to continue to specify the settings you want for this node.
7. In the Pattern page, type the pattern that you want the external node to match. The pattern you create here will be checked against the option you specify in Check pattern against .

   The pattern acts as a filter that defines the external node. For example, if the pattern is `ROS*` then all messages with node IDs that match this pattern will be associated with the external node and will only appear in the browser for that node. The first match found determines to which node the message belongs. When an incoming message matches a node pattern, the evaluation stops and the message appears in the browser for that node.

   Examples:

   - To match fully-qualified domain names

     To include all IP names in the domain, "deu.hp.com", use the following example: `<*>.deu.hp.com`

   - To match an IP address range

     For example, to set up an IP address range to match all addresses between 15.136.123.5 (including) and 15.136.123.72 (excluding), use the following example: `15.136.123.<5 -le [<#>] -lt 72>`

   - To match a specific string

     To match a specific string, remember to enclose the string with the (^) and ($) characters in the following manner: `^STRING$` . This pattern matches only `STRING` and excludes similar strings such as `STRING1` or `FIRSTSTRING` .

   NOTE:
   Click the button to the right of the Pattern box to help you enter variables and operators for numeric comparison. Click Test Pattern... to open the Test Pattern dialog box where you can create a pattern and enter strings to immediately compare your pattern.

8. Specify what the pattern should match in the Check pattern against group box. Choose from Fully Qualified Domain Name , IP Address , or Node Name . This selection clarifies any possible ambiguity in the pattern. For example, a pattern like `*15*` could refer to an IP address like `15.1.2.2` , or to a node name like `ROS15test` . By selecting IP Address or Node Name , you avoid any confusion about the meaning of the pattern.

   - Fully Qualified Domain Name

     If Fully Qualified Domain Name is the pattern choice, the filter will ignore the IP address for this pattern.

   - IP Address

     If IP Address is the pattern choice, only valid IP strings are evaluated.

   - Node Name

     If Node Name is selected, the filter matches the node name.

9. Click Next to proceed to the wizard's Order page.

10. Select Check Before Managed Nodes to have external nodes evaluated before managed nodes. This can be faster, because it avoids some of the checking that occurs with configured node names. When this box is checked, messages with node IDs that match the pattern you specified in the previous page are associated with the external node. If this box is not checked, the managed node takes precedence.
11. The two lists display all external nodes that have been created. The list order shows the order of evaluation set for incoming messages. The node being edited can be moved from one list to the other by using the Check Before Managed Nodes check box. Use the Move Up and Move Down buttons to change this order. The node at the top of the list will be evaluated first. After the first node has matched, the message is assigned to that node and no further evaluation takes place.
12. Click Finish to complete the configuration and close the wizard.

To configure an external node

1. Open the Node Configuration editor if it is not already open.
2. In the Nodes list, right-click the external node you want to configure to open the External Node Properties dialog box. Use the tabs to display information about the selected node and to configure it:
   - General: provide a required caption and an optional description.
   - Details: provide the owner information and specify the node match pattern.
   - Order: displays the order of evaluation for the existing external nodes.
   - Outage: displays the current outage state.
3. Click Apply as you finish with a tab to apply your changes.
4. Click OK to confirm your changes and close this dialog box.
5. Click Cancel to close the dialog box without saving your changes.

Configure General information for external nodes

Each external node must be uniquely identified by specifying properties for that node. Use the General tab in the External Node Properties dialog box to specify details that identify each external node. Required information is specified.

To configure general information for external nodes

1. In the list of external nodes, select the node you want to configure.
2. Right-click to open the shortcut menu.
3. Select Properties to open the External Nodes Properties dialog box, which displays the General tab by default. The automatically generated unique ID (GUID) for the selected node appears at the top of the dialog box.
4. In the Display Name box, enter the label (display name) for the external node. The caption appears in the External Nodes list. This information is required.
5. Enter any comments or additional information in the Description box. This information is optional.
6. Click Apply to apply your changes.
7. Click OK to apply your changes and close this dialog box.
8. Click Cancel to close this dialog box without saving your changes.
9. Select the Details tab to continue configuring this node.

Related Topics:

- Configure Details information for external nodes
- Configure Outage Information for external nodes
- Configure Order information for external nodes

Configure Details information for external nodes

Use the Details tab to provide contact information for the selected external node.

To configure details information and pattern-matching options for external nodes

1. Type the owner's name and contact details in the boxes provided.
2. Type the pattern that you want the external node to match in the Pattern field. The pattern you create here will be checked against the option you specify in Check pattern against .

   The pattern acts as a filter that defines the external node. For example, if the pattern is `ROS*` then all messages with node IDs that match this pattern will be associated with the external node and will only appear in the browser for that node. The first match found determines to which node the message belongs. When an incoming message matches a node pattern, the evaluation stops and the message appears in the browser for that node.

   Examples:

   - To match fully-qualified domain names

     To include all IP names in the domain, "deu.hp.com", use the following example: `<*>.deu.hp.com`
   - To match an IP address range

     For example, to set up an IP address range to match all addresses between 15.136.123.5 (including) and 15.136.123.72 (excluding), use the following example: `15.136.123.<5 -le [<#>] -lt 72>`
   - To match a specific string

     To match a specific string, remember to enclose the string with the (`^`) and ($) characters in the following manner: `^STRING$` . This pattern matches only `STRING` and excludes similar strings such as `STRING1` or `FIRSTSTRING` .

   NOTE:
   Click the button to the right of the Pattern box to help you enter variables and operators for numeric comparison. Click Test Pattern... to open the Test Pattern dialog box where you can create a pattern and enter strings to immediately compare your pattern.
3. Specify what the pattern should match in the Check pattern against group box. Choose from Fully Qualified Domain Name , IP Address , or Node Name . This selection clarifies any possible ambiguity in the pattern. For example, a pattern like `*15*` could refer to an IP address like `15.1.2.2` , or to a node name like `ROS15test` . By selecting IP Address or Node name , you avoid any confusion

about the meaning of the pattern.

- Fully Qualified Domain Name

  If Fully Qualified Domain Name is the pattern choice, the filter will ignore the IP address for this pattern.

- IP Address

  If IP Address is the pattern choice, only valid IP strings are evaluated.

- Node Name

  If Node Name is selected, the filter matches the node name.

4. Click Apply to apply your changes.
5. Click OK to confirm your changes and close this dialog box.
6. Select the Order tab to continue configuring this node.

Related Topics:

- Configure Order information for external nodes
- Configure General information for external nodes
- Configure Outage information for external nodes

Configure Order information for external nodes

Use the Order tab to specify the order of evaluation of external and managed nodes. The order of evaluation is partly determined by the pattern settings you make in the Details tab.

To specify the order of evaluation

1. In the Details tab, you will have specified the pattern you want to match.
2. Select the Check Before Managed Nodes check box to have external nodes evaluated before managed nodes. This can be faster, because it avoids some of the checking that occurs with configured node names. When this box is checked, messages with node IDs that match the pattern you specified in the Details tab are associated with the external node. If this box is not checked, the managed node takes precedence.
3. The two lists display all external nodes that have been created. The order of the two list shows the order of evaluation set for incoming messages. Nodes are evaluated in descending order; the node at the top of the list has the highest priority and will be evaluated first. You can change the order of evaluation using the Move Up and Move Down buttons. After the first node has matched, the message is assigned to that node and no further evaluation takes place.
4. Select the Outage tab to view outage information for the external node.

Related Topics:

- Configure Outage information for external nodes
- Configure General information for external nodes
- Configure Details information for external nodes

Configure Outage information for external nodes

Use the Outage tab of the External Node Properties dialog box to view outage information for an external node. This read-only tab displays the settings established for both unplanned and scheduled outages.

A scheduled outage meets these criteria:

- Planned to happen
- Recurs at regular intervals for maintenance
- Configured by a policy

An unplanned outage is unexpected and can occur randomly.

Only administrators can put an external node into maintenance mode.

To view Outage properties for a selected external node

1. Expand the Nodes folder in the console tree, then select HP Defined Groups → External to display a list of any external nodes that have been configured for your environment.
2. Right-click the name of the external node for which you want to display properties; this opens the context menu.
3. Click Properties to open the Properties dialog box for the selected external node, which displays the General tab by default.
4. Select the Outage tab to view the following information:
   - Current Outage State:

     Displays the outage status of the selected external node. This status will be either "ON " or "OFF".
   - Unplanned Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage . Messages can be either deleted or acknowledged.
   - Scheduled Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage . Messages can be either deleted or acknowledged.

Related Topics:
- Configure General information for external nodes
- Configure Details information for external nodes
- Configure Order information for external nodes

Cluster support in HPOM

A cluster is a group of computers that work together to run a common set of applications; the client and the application see only a single system. The computers in the cluster are connected by cables and programmatically connected by cluster software.

If one of the nodes in a cluster is out of service due to failure or maintenance, another node in the cluster becomes available to provide service. This process is called failover.

HP Operations agents are cluster-aware, which means that the agent recognizes that it is running on a cluster system. (In a cluster, the agent is installed on each of the nodes in the cluster.) The agent supports switching application monitoring packages between the physical nodes to monitor the application where it really runs.

For a list of platforms and versions for which HP Operations Manager for Windows cluster-awareness support is available, see the support matrix on the Software Support Online web site.

Related Topics:

- Manage cluster-aware applications
- Customize cluster state mappings
- Configure HTTPS agents running under non-administrative accounts
- Server or cluster failover behavior
- TruCluster system information

Manage cluster-aware applications

The configuration described below will create a scenario where policies that you designate are enabled on the node in the cluster where the application is currently running, and disabled on all other nodes in the cluster.

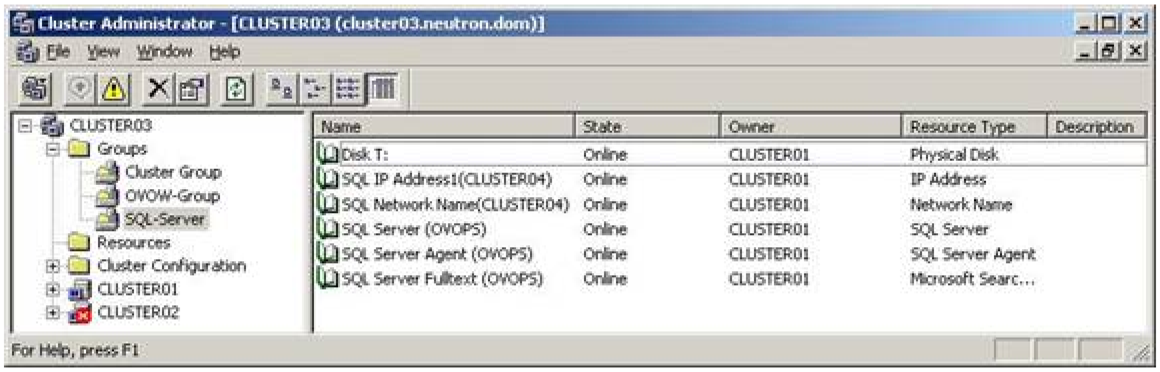
Select the type of agent that you want to configure, and the procedure below will change to reflect the necessary steps.

HTTPS agent

To monitor a cluster-aware application

1. *Optional.* For up-to-date information on supported cluster environments, see the support matrix at HP Software Support Online.
2. Make sure that the resource group which contains the resource being monitored contains both a network name and an IP address resource.

   The figure below shows the Microsoft Cluster Administrator window with the SQL-Server resource group shown in detail. This group contains the required network name (CLUSTER04) and IP address resources.

   - CLUSTER03
     - Groups
       - Cluster Group
       - OVOW-Group
       - SQL-Server
     - Resources
     - Cluster Configuration
     - CLUSTER01
     - CLUSTER02

Name	State	Owner	Resource Type	Description
Disk T:	Online	CLUSTER01	Physical Disk	
SQL IP Address1(CLUSTER04)	Online	CLUSTER01	IP Address	
SQL Network Name(CLUSTER04)	Online	CLUSTER01	Network Name	
SQL Server (OVOPS)	Online	CLUSTER01	SQL Server	
SQL Server Agent (OVOPS)	Online	CLUSTER01	SQL Server Agent	
SQL Server Fulltext (OVOPS)	Online	CLUSTER01	Microsoft Searc...	

3. Create an XML file that describes the application instances, and name it `apminfo.xml` .

   This file is used to define the resource groups that will be monitored and to map application instances to resource groups.

   Format

   DTD

   Example

4. Save the completed `apminfo.xml` file on each node in the cluster in the following directory:
   - Windows DCE agent

     <install_dir `>\Installed Packages\{790C06B4-844E-11D2-972B-080009EF8C2A}\conf\OpC`
   - UNIX DCE agent

     `/var/opt/OV/conf/OpC`
5. Save the completed `apminfo.xml` file on each node in the cluster in the following directory:

   `$OvDataDir/conf/conf/`
6. Create an XML file that describes the policies to be cluster-aware. The file name must have the format <*name_of_cluster-aware_application*> `apm.xml` . <*name_of_cluster-aware_application*> must be identical to the content of the <Name> tag in the `apminfo.xml` file.

   ▼ Format

   ▼ Example
7. Save the completed <*name_of_cluster-aware_application*> `apm.xml` file on each node in the cluster in the following directory:

   `$OvDataDir/bin/instrumentation/conf`
8. Write policies to monitor the application on the cluster.
9. Assign a category to the policies. For more information about categories, see Add categories to a policy .
10. Create a category directory for the category you defined earlier. Copy the XML files generated in step 5 to this directory.

    The management server automatically deploys the XML files to the node whenever it deploys policies of this category. The files are placed in the following directories on the managed nodes:
    - Windows DCE agent

      <install_dir `>\Installed Packages\{790C06B4-844E-11D2-972B-080009EF8C2A}\bin\Instrumentation\conf`
    - UNIX DCE agent

      `/var/opt/OV/bin/instrumentation/conf`
11. Ensure that the physical nodes where the resource groups reside are all managed nodes.
12. Deploy the policies listed in <*name_of_cluster-aware_application*> `apm.xml` and the monitors to all the physical nodes in the cluster.
13. *Optional.* For some physical nodes, for example for multihomed hosts, the standard hostname may be different from the name of the node in the cluster configuration. If this is the case, the agent cannot correctly determine the current state of the resource group. Configure the agent to

use the hostname as it is known in the cluster configuration:

a. Obtain the name of the physical node as it is known in the cluster configuration:

   `ovclusterinfo -a`

b. Configure the agent to use the name of the node as it is known in the cluster configuration:

   `ovconfchg -ns conf.cluster -set CLUSTER_LOCAL_NODENAME` *<name>*

   Replace *<name>* with the name of the node as reported in the output of `ovclusterinfo -a`.

14. Restart the agent:
    a. Stop the agent: **`ovc -stop AGENT`**
    b. Restart the agent: **`ovc -start AGENT`**
15. Restart the agent:
    a. Stop the agent: **`opcagt -kill`**
    b. Restart the agent: **`opcagt -start`**

Customize cluster state mappings

The agent checks the state of the resource group on a cluster node to decide whether the policies have to be enabled or disabled. By default, if a state maps to "online", then a policy is enabled; if it maps to "offline" or "unknown", then it is disabled.

You may want to modify the mapping, for example, to treat the cluster state "partial" as "online". Use the tool ovconfchg on all HTTPS agents in a cluster to change the mapping of cluster states. To change the mapping on DCE agents running Microsoft Cluster Server, you must configure parameters in a nodeinfo policy.

To customize cluster state mappings on HTTPS agents

`ovconfchg -ns conf.cluster.RGState.<`*`cluster_software`*`> -set <`*`cluster_state`*`> <`*`state`*`>`

For example, to map the Microsoft Cluster Server state "ClusterGroupPartialOnline" to "online", enter:
`ovconfchg -ns conf.cluster.RGState.MSCS -set ClusterGroupPartialOnline online`

In the example above, the HTTPS agent has been modified to treat a cluster group in a partial online state as online rather than the default value of offline, and as such when the resource group is in a partially online state the policies continue to be enabled.

▼ Default cluster state mappings

To customize cluster group state mappings on DCE agents running Microsoft Cluster Server

By default only the state "ClusterGroupOnline" is treated as online. But it is possible to map other states using the following nodeinfo parameters:

- OPC_APM_HANDLE_GROUP_AS_ONLINE

  This nodeinfo parameter allows you to define which cluster group states are treated as online.

  Syntax:

  ```
  OPC_APM_HANDLE_GROUP_AS_ONLINE <state1>, <state2>
  ```

  Example:

  ```
  OPC_APM_HANDLE_GROUP_AS_ONLINE ClusterGroupOnline, ClusterGroupPartialOnline,
  ClusterGroupFailed
  ```

- PC_APM_HANDLE_PARTIAL_AS_ONLINE (deprecated)

This nodeinfo parameter allows you to define that the cluster group state "ClusterGroupPartialOnline" is handled as online.

Syntax:

```
OPC_APM_HANDLE_PARTIAL_AS_ONLINE TRUE
```

This nodeinfo variable is still available for backward compatibility but should not be used in future. Instead use:

```
OPC_APM_HANDLE_GROUP_AS_ONLINE ClusterGroupOnline, ClusterGroupPartialOnline
```

If both variables, OPC_APM_HANDLE_GROUP_AS_ONLINE and OPC_APM_HANDLE_PARTIAL_AS_ONLINE, are used, the variable OPC_APM_HANDLE_GROUP_AS_ONLINE has the higher priority and OPC_APM_HANDLE_PARTIAL_AS_ONLINE is ignored.

Cluster group states in Microsoft Cluster Server

To customize cluster node state mappings on DCE agents running Microsoft Cluster Server

By default, the state "ClusterNodeUp" is considered as online. But it is possible to map other states using the following nodeinfo parameter:

- OPC_APM_HANDLE_NODE_AS_ONLINE

  This nodeinfo parameter allows you to define which cluster node states are treated as online.

  Syntax:

  ```
  OPC_APM_HANDLE_NODE_AS_ONLINE <state1>, <state2>
  ```

  Example:

  ```
  OPC_APM_HANDLE_NODE_AS_ONLINE ClusterNodeUp, ClusterNodeJoining
  ```

Cluster node states in Microsoft Cluster Server

Related Topics:

- Manage cluster-aware applications
- Node Info Policy Type

Configure HTTPS agents running under non-administrative accounts

By default, HTTPS agents regularly check the status of the resource group. On UNIX and Linux nodes, the agents use cluster application-specific commands, which can typically only be run by root users. (On Windows nodes, the agents use APIs instead of running commands.)

If you change the user of an HTTPS agent, the agent may no longer have the permissions required to successfully execute cluster commands. In this case you must configure the agent to use a security program (for example, sudo or .do) when issuing cluster commands.

To configure non-root HTTPS agents to use a security program

1. On the physical cluster nodes, log in as root and open a shell prompt. Ensure that the PATH variable contains the path to the agent commands.

   - On HP-UX, Solaris, or Linux, type export PATH=/opt/OV/bin:$PATH and then press Enter .
   - On AIX, type export PATH=/usr/lpp/OV/bin:$PATH and then press Enter .
   - On Tru64, type export PATH=/usr/opt/OV/bin:$PATH and then press Enter .

2. To stop the agent, type the following command:

   `ovc -kill`

3. To configure the agent to use a security program, type the following command:
   ovconfchg -ns ctrl.sudo -set OV_SUDO <*security_program* >

   Replace <*security_program* > with the name of the program you want the agent to use, for example `/usr/local/bin/.do` .

4. To start the agent, type the following command:

   `ovc -start`

Cluster commands

If you are using a configuration file to specify which users can run which commands, you must add the cluster commands listed in the table below to this file.

For example, if you use sudo and the agent runs as opc_op, add the following line to the sudoers file to allow the opc_op user to run HP Serviceguard commands without the need for a password:

```
opc_op ALL=(ALL) NOPASSWD: /usr/sbin/cmviewcl,/usr/sbin/cmgetconf
```

Cluster application	Cluster commands

AIX Cluster	`/usr/es/sbin/cluster/clstat`
	`/usr/es/sbin/cluster/utilities/clRGinfo`
	`/usr/es/sbin/cluster/utilities/clgetip`

HP Serviceguard	`/usr/sbin/cmviewcl` `/usr/sbin/cmgetconf`
Microsoft Cluster Server	The agent uses APIs instead of commands.
Red Hat Enterprise Linux 3	`/usr/sbin/redhat-config-cluster-cmd` `/usr/sbin/clustat`
Red Hat Enterprise Linux 4	`/sbin/cman_tool` `/usr/sbin/clustat`
Red Hat Enterprise Linux 5	`/usr/sbin/cman_tool` `/usr/sbin/clustat`

Sun Cluster	`/usr/cluster/bin/scha_cluster_get`
	`/usr/cluster/bin/scha_resource_get`
	`/usr/cluster/bin/scha_resourcegroup_get`

TruCluster	`/usr/sbin/clu_get_info`

Veritas Cluster	`/opt/VRTSvcs/bin/haclus`
	`/opt/VRTSvcs/bin/hasys`
	`/opt/VRTSvcs/bin/hagrp`
	`/opt/VRTSvcs/bin/hares`

Related Topics:

- Change the user of an HTTPS agent on a UNIX or Linux node
- ovconfchg

Server or cluster failover behavior

HPOM for Windows monitors the availability of server components such as the Windows Management Interface (WMI), the server itself, and the cluster resource group (if your environment contains clusters.)

In the event of a failure of one or more of these components, some activities may be temporarily unavailable, until WMI or the server is restored. Depending on the cause of the failure and your specific environment, you may experience different results depending on your activity at the time of the failure. Some activities may be interrupted, as described below. You will see a message explaining that a service or component is temporarily unavailable and you have the option to retry the action.

In a cluster environment, services should fail over to another server in the cluster so that you can continue to work almost without interruption.

Node Configuration Editor

If a failure occurs while you are bringing nodes under management, the following actions occur:

- The OK and Apply buttons in the editor are disabled.
- You cannot expand folders in the discovered nodes list.
- Operating system discovery will not take place in the managed nodes list. Prerequisite checking cannot take place because it requires the discovery of a system and OS type. You can manually set the System and OS type and then manually run the prerequisite check.
- If the problem is a cluster offline event, you cannot add reports and graphs to a node or node group.

You can continue to work in the console to perform the following tasks:

- Bring nodes under management
- Change node locations in the managed nodes list.
- Use the node and node group property sheets.

However, you cannot apply your changes and close the editor until WMI or the server are available again.

Service Type Editor

If you are performing an operation that requires network access to the server at the time of a failure, you will see a message explaining that your changes will be lost. You can return to the Service Type dialog box or exit.

Tool Configuration Editor

If you are performing an operation that requires network access to the server at the time of a failure, you will see a message explaining that your changes will be lost. You can return to the Tool Configuration dialog box or exit.

Apply Tool dialog box

If you are in the process of running a tool when the server connection is lost, any tools that are currently running will stop. You will see a "server down" message in the Tool Output details pane and the Rerun button will be unavailable. When the server connection is restored, the Rerun button will again be available.

Policy management server

If you are working with the policy management service at the time of a failover, you will see a message explaining that the connection to the PMAD server is lost. The console will reconnect to the PMAD server automatically. When the server connection is restored, right-click Policy management and select Refresh from the context menu.

TruCluster system information

This topic summarizes the features, notes, and recommendations for monitoring applications on a TruCluster system, and explains how to relocate the monitoring of a single-instance application during failover.

Notes and recommendations

- You must create a node group for your TruCluster system and add the TruCluster members to this node group.
- Installing or uninstalling the agent software must be done on one TruCluster member; the software is added to or deleted from the other TruCluster members automatically.
- Deploy instrumentation and policies to all TruCluster members.

Relocating the monitoring of a single-instance application during failover

Cluster Application Availability (CAA) is used to start a single-instance application on an individual TruCluster member and relocate it during failover to another cluster member. You can use CAA to relocate application monitoring during a failover.

For further information on CAA, see the *Cluster Highly Available Applications* manual in the Tru64 UNIX TruCluster documentation set. Chapter 2, *Using CAA for Single-Instance Application Availability* , is particularly useful.

For further information on TruCluster system administration, see the Cluster Administration manual in the Tru64 UNIX TruCluster documentation set. Tru64 UNIX documentation is available online at the following URL: `http://h30097.www3.hp.com/docs/pub_page/doc_list.html`

To make the application a highly available CAA resource

1. Create the CAA resource profile and action script for the application, either through the SysMan Menu or by using the caa_profile command.

   This step creates the `/var/cluster/caa/profile/.cap` and `/var/cluster/caa/scripts/.scr files` , respectively.

   a. Test the action script.

   b. Validate the resource profile.

   c. Register the resource with CAA.

   d. Start the resource.

2. From the HPOM server, assign the template that you created to monitor your application to all TruCluster members.
3. From the HPOM server, distribute this template to all cluster members.
4. After the initial distribution of the template, use the opctemplate -d command to disable the template for all cluster members on which the application does not run.

   ```
   opctemplate -d
   ```

   This step is required only after the initial template distribution. On subsequent template distributions, the template state is maintained on the managed TruCluster members.
5. Edit the application's action script, which has three main routines: start, stop, and check.
   a. Enable the template in start routine with the `opctemplate -e` command.
   b. Disable the template in stop routine with the `opctemplate -d` command.

▼ View example

Related Topics:

- Cluster support in HPOM

Optimize HPOM node name resolution

Several enhancements have been implemented to improve the performance of HPOM for Windows node name resolution. The speed of the node name resolution has a significant impact on the following areas:

- Message filter (directly impacts flow of new incoming messages)
- Health check
- Actions

These enhancements will help resolve (or at least reduce) the following performance related problems:

- Slow initialization of the HPOM node cache during the startup phase of the OvEpMessageActionServer service (30-60 minutes). During the initialization, the message flow is disabled.
- Slow name resolution if using agentless nodes.
- Significant delays for new incoming messages due to slow node name resolution.
- Slow health check, slow processing of ICMP replies sent from managed nodes.

Managed environments that do not make use of DHCP will experience the largest benefits. Here, an optional memory based cache can be configured which drastically reduces the number of the Windows API calls `gethostbyname()` and `gethostbyaddr()` . However, environments using DHCP can also take advantage of a modified ip-address verification which tries to avoid unnecessary reverse node name resolution.

To make use of the enhancements, you must configure several values using the Server Configuration dialog box. If the values are not set, then the enhancements will not take effect. See the Related Topics for enhancement details.

Related Topics:

- Avoid unnecessary name resolution for agentless nodes
- Reducing the number of Windows API calls gethostbyname() and gethostbyaddr()
- Configure the number of retries for gethostbyname() and gethostbyaddr()
- Avoid unnecessary reverse name resolution

Avoid unnecessary name resolution for agentless nodes

Certain environments make use of "agentless nodes". These nodes are not physical nodes, meaning they do not have an IP address or an agent id. They are only used to associate messages to them which originate from non-IP network elements or from other HP BTO Software products such as HP Internet Services.

They are configured in the HPOM node bank by using an arbitrary name. Although agentless nodes do not have an IP address and an agent ID, the MessageActionServer per default tries to resolve them when initializing its node cache and each time they appear in a message. This unnecessary name resolution has a significant impact on startup time and on message throughput.

Currently HPOM node configuration does not allow to explicitly define a node as "agentless". Therefore the following convention is used to define agentless nodes. A node set up in the node database qualifies for being recognized as a agentless node when it meets the following prerequisites:

- Must not have an IP address and the name given should not be DNS resolvable at any time.
- Must not have an agent id
- Must not be configured for DHCP

Additionally agentless nodes must match a user-configured value pair for the following properties:

- System type
- Operating system

For example, consider nodes as being agentless nodes if they do not have an IP address, do not have an agent id, are not configured for DHCP, and have the system type "other" and operating system type "unknown".

If a node has been qualified as being a agentless node based on the property settings, then no name resolution will be performed during production and messages associated with the node will pass the message filter without further checks.

The value pair for agentless nodes is defined by values in the Node Cache Settings namespace in the Server Configuration dialog box. After you have changed a value, restart the OvEpMessageActionServer service for the change to take effect. The values, along with their possible settings, are shown below:

Value	Possible Settings	Default Setting
Node system type for agentless nodes	KNOWN, UNKNOWN, ANY, TURNOFF	TURNOFF
Node OS type for agentless nodes	KNOWN, UNKNOWN, ANY, TURNOFF	TURNOFF

Node system type for agentless nodes specifies how to qualify agentless nodes in dependency of their system types. If the value is set to "KNOWN", then system type must have been configured to a known system type (for example, Pentium Compatible, Power PC family, SPARC Family, PA-RISC Family, Itanium compatible) and must not have been configured as "Other".

If the value is set to "UNKNOWN", then system type must have been configured as "Other". If the value is set to "ANY" then any node qualifies independently of the System Type setting.

Node OS type for agentless nodes specifies how to qualify agentless nodes in dependency of their operating systems. If the value is set to "KNOWN", then the operating system must have been configured to a known operating system and must not have been configured as "Unknown". Supported operating systems are shown on the Software Support Online web site support matrix .

If the value is set to "UNKNOWN" then system type must have been configured as "Unknown". If the value is set to "ANY" then any node qualifies independently of the operating system setting.

NOTE:
For qualifying agentless nodes, both values must be set. If none or only one of the values are set, then no agentless nodes will qualify.

The following examples show how the registry names described above are interpreted while evaluating whether nodes matching the prerequisites (no IP address, no agent id, not setup as DHCP node) qualify as agentless nodes or not.

Node system type for agentless nodes	Node OS type for agentless nodes	Description
UNKNOWN	UNKNOWN	Node qualifies as a agentless node if system type has been defined as "other" and operating system has been defined as "unknown".
KNOWN	UNKNOWN	Node qualifies as a agentless node if the system type is not set to "other" and the operating system is unknown.
ANY	ANY	Node qualifies as a agentless node independently of system type and operating system.

Related Topics:

- Change server configuration values

Reduce the number of Windows API Calls **`gethostbyname()`** and **`gethostbyaddr()`**

To drastically reduce the number of the Windows API calls `gethostbyname()` and `gethostbyaddr()` , an HPOM user-configurable name server cache has been added. It caches the parameter and the returned results of these functions. Whenever a name resolution or reverse name resolution is required for a node, it will be first checked whether the required information is already available in the cache.

If the information is there, then it will be used instead of calling `gethostbyname()` or `gethostbyaddr()` . When the maximum number of the cache entries is reached, the entry with the lowest reference count number will be swapped to make space for a new entry. The strategy is to keep those entries which are mostly referenced in the cache. Also the parameter and results of unsuccessful `gethostbyname()` and `gethostbyaddr( )` calls are kept in the cache. This helps quickly detect nodes which are not resolvable because they might have been deleted from DNS but still send messages as an agent is installed there.

NOTE:
Caching is used only if none of the managed nodes are configured for DHCP in the HPOM node configuration. If there are DHCP nodes configured, then the name server cache will be ignored independently of whether it is enabled or disabled.

You can configure the name server cache with two values in the Node Cache Settings namespace in the Server Configuration dialog box: Turn on node cache and Node cache size . After you have changed a value, restart the OvEpMessageActionServer service for the change to take effect.

Registry Name	Possible Values	Effect
Turn on node cache	TRUE, FALSE	The default value is TRUE. When set to TRUE, the name server cache is enabled and will be used.
Node cache size	Any value greater than 100.	Defines the maximum number of entries in the name server cache. It should be set to a minimum value of 100. The maximum value depends on the available main memory and the number of managed nodes. Recommended setting: Number of managed nodes + 100. NOTE: the name server cache is located in the address space of `OvEpMsgActSrv.exe` , so there might be

		an increased usage of the virtual memory of this process. Default value=100

Related Topics:

- Change server configuration values

Configure the number of retries for **gethostbyname()** and **gethostbyaddr()**

Independent of whether name server caching is enabled or disabled, the Windows API functions gethostbyname() and gethostbyaddr() will be used at some points. Under certain conditions (network problem, DNS problems) these calls might fail. This would then be interpreted as "node name could not be resolved". There is however a very low probability that a name or IP address could be successfully resolved if these calls would be retried more often.

By default, the maximum number of retries is 3. So if the first call fails, there is a maximum of 2 more retries. Such retries however have a significant impact on the time it takes to decide whether a node name is resolvable or not. If the assumption can be made that the network and DNS is reliable and is properly configured, then the number of retries can be set to the minimum value of 1. In case there are some doubts on the reliability of the network and DNS, the number should be set to 2 or 3.

The number of overall retries for the Windows API calls gethostbyname() and gethostbyaddr() is configured by the value Name resolution retries in the Node Cache Settings namespace in the Server Configuration dialog box. After you have changed a value, restart the OvEpMessageActionServer service for the change to take effect.

Value	Value Range	Effect
Name resolution retries	1-3	Specifies the maximum number of retries in case of unsuccessful gethostbyname() and gethostbyaddr() function calls. Default value=3

Related Topics:

- Change server configuration values

Avoid unnecessary reverse name resolution

By default, HPOM performs reverse name resolution for incoming messages that contain the IP address and optionally the node name of the managed node, but no agent ID.

Such messages are typically triggered by SNMP traps and are proxied. The default reverse name resolution process calls the function `gethostbyaddr()` by using the IP address stored in the arriving message. Then it takes the returned node name and verifies whether it really corresponds to the name which is stored in the message.

If no node name is stored in the message, then it crosschecks whether the node name returned by the `gethostbyaddr()` call is known in the HPOM node configuration. If the returned node name is a short name, then it calls `gethostbyname()` to get the long host name (FQDN). Then it checks whether the long host name is known in the HPOM node configuration. These multiple level of checks are used to cover all kind of theoretical corner cases which might show up very rarely in well-managed environments.

The vast majority however will never hit such corner cases, so in most environments the reverse name resolution could be avoided safely if the following conditions match:

- If the name of a managed node is changed, then this change is immediately performed manually in the HPOM node configuration by using the HPOM management console.
- If the IP address of a non-DHCP managed node is changed and the IP address was used for configuring the primary node name or the communication path then this change is immediately performed manually in the HPOM node configuration by using the HPOM management console.

Avoiding unnecessary reverse name resolution can be configured by the value Reverse name resolution method in the Node Cache Settings namespace in the Server Configuration dialog box.

Value	Possible Settings	Effect
Reverse name resolution method	1	Reverse name resolution is always performed if no agent ID but IP address is contained in an incoming message. This is the default value.
	2	Reverse name resolution will be avoided if possible.

If specifying the value 2 for the registry name Reverse name resolution method then HPOM verifies the IP address differently from the default way to avoid reverse name resolution if possible, as follows: if the message contains only the IP address, but no node name, then check whether the IP address is known in the HPOM configuration. If yes, then the message can pass. If the message contains both the IP address and node name, then first check whether the pair IP address plus node name can be found in the HPOM node configuration. If yes, then the message can pass.

Otherwise HPOM performs a reverse name resolution.

NOTE:
If DHCP is not used and name server caching is enabled, then reverse name resolution (if required) will be fast in any case.

Related Topics:

- Change server configuration values

Configuring Agents

An agent is a deployment package that enables you to manage nodes . After you deploy an agent to a node, it enables you to collect data, discover services, monitor events, and run actions and commands that control the node.

Before HPOM for Windows 8.00, management servers and agents communicated using DCE/RPC. HPOM 8.10 includes an agent that communicates using HTTPS. Although the DCE agent is currently still supported, you are encouraged to use the HTTPS agent for new nodes. If nodes already exist in your environment that have DCE agents, consider migrating to the HTTPS agent.

The HTTPS agent offers the following benefits:

- Secure communication based on the HTTPS protocol. All communications between management servers nodes is strongly encrypted.
- Policy , message , and action security. Policies, messages, and actions all contain signatures, which management servers and nodes create and check using certificates. If a malicious user attempts to tamper with a signed policy, message, or action, the signature becomes invalid.
- Simplified firewall configuration. Nodes and management servers accept all inbound communications to a single port, so it is simpler to configure firewalls and proxies.
- Multiple management servers can deploy policies to the same node.

Remote agent installation

You can install agents remotely when you create a new node. Alternatively, when you deploy a policy to a node , the management server checks that the correct agent packages exist on the node. If the agent packages are not already installed, or this policy requires an later version of the agent, the management server automatically deploys the latest agent packages on the node.

The management server can automatically deploy HTTPS and DCE agents to nodes that have a supported Windows operating system. In addition, if you configure a suitable secure shell client on the management server, it can also automatically deploy HTTPS agents to nodes that have a supported UNIX or Linux operating system.

If you are using HTTPS agents, you can specify default settings that you want the management server to apply when it automatically installs the agent on specific nodes.

If you do not want the management server to install agents automatically, you can disable this feature.

To automatically deploy agent packages, the management server creates a deployment job. Before the job starts, the management server requests the credentials of a user who has administrative access to the node.

NOTE:
You can also start agent deployment by dragging the Operations-agent deployment package to a node or node group.

Related Topics:

- Install agents remotely
- Configure agent deployment to UNIX and Linux nodes
- Configure HTTPS agent installation defaults
- Disable automatic agent installation
- Deploy deployment package

Install agents remotely

If you have the correct permissions and network access to remote nodes, you can install agents onto the nodes from the console. To remotely install agents, the management server requires the credentials of a user who has administrative access to the node. The following table shows the specific permissions required, according to the node's operating system, and the type of agent.

	HTTPS agent	DCE agent
Windows operating systems	■ Write access to the admin$ share (the user must be part of the local administrators group) ■ Read access to the registry ■ Permission to log on as a service (this is only required if you select User/Password in the Set Credentials list)	■ Write access to the registry ■ Write access to the admin$ share (the user must be part of the local administrators group) ■ Write access to the Program Files folder ■ Permission to create DCOM (Distributed Component Object Model) connections
UNIX or Linux operating systems	Permission to log in to SSH on the node for file transfers and to execute installation commands	Not applicable. Only manual installation is possible.

Remote agent installation is supported from management servers that belong to an Active Directory domain. If your management server is in a Windows workgroup environment, install agents manually instead.

To install agents remotely

1. *Optional.* For up-to-date information on supported agent platforms, see the support matrix at HP Software Support Online.
2. Open the Agent installation dialog box if it is not already open.
3. *Optional.* In the list of nodes, select or clear the Deploy check box to indicate the nodes that you want to install the agent on.

   If you select the Deploy check box for nodes that have a UNIX or Linux operating system, the Credentials dialog box appears. Type the Username of a user who has permission to install software on the node. Type the password in Password and Repeat Password .
4. *Optional.* Specify the credentials that the management server uses to deploy the agent. Right-click a node in the list, and then click Set Credentials . Click one of the following commands:

- PMAD user . The management server attempts to deploy the agent as the user under which the policy management and deployment (PMAD) service runs (called HP-OVE-Deleg-User by default).

  You can only use this for nodes with a Windows operating system. The nodes can belong to the same domain as the management server, a trusted domain, or a workgroup. If you are installing a DCE agent to a Windows node, this is the only command available.

  NOTE:
  The PMAD user does not by default have administrative access to the node. For more information, see Start Windows node security setup .

- Impersonate user . The management server attempts to deploy the agent using the credentials that you are currently logged in to Windows with.

  You cannot use impersonation for nodes in untrusted domains or workgroups.

  You can use impersonation only if the PMAD user is trusted for delegation in Active Directory, unless your console runs directly on the management server. For more details on delegation, refer to the Active Directory documentation that Microsoft provides.

- User/Password . The Credentials dialog box appears. Type the Username of a user who has permission to install software on the node and their Password . Click OK . This automatically selects the Deploy check box.

  For nodes with a UNIX or Linux operating system, this is the only command available .

  You can also use this command to install the HTTPS agent to Windows nodes that belong to the same domain as the management server, a trusted domain, or a workgroup. The user must have permission to log on as a service. (You can assign this permission in the local Windows security settings on the node, or a group policy object in Active Directory.)

  For Windows nodes specify the username in one of the following formats:

  - domain\user (for an Active Directory user)
  - computer_name\user (for a local user on the node)

5. *Optional.* Select or clear the Run prerequisites check automatically during job execution check box. If a node does not meet any of the prerequisites, the agent installation fails for that node. If you are sure that the nodes meet all prerequisites, you can clear this check box so that the agent installation jobs complete more quickly. (This check box is dimmed if an administrator disables all prerequisite checking using the Server Configuration dialog.)
6. Click OK . The management server creates an agent installation job for each of the nodes that you selected. To view the progress of these jobs, click Deployment jobs in the console tree. If an agent installation job fails, right-click the job, and then click Error description .

NOTE:

The management server does not create a deployment job if the agent package is already in the node's package inventory. If a node's package inventory contains the agent package, but you are sure that the agent is not actually installed on a node, you can force the management server to remove the agent package from the inventory in the Uninstall package from... dialog. For more information, see Remove package from node .

Related Topics:

- Install agents in trusted domains
- Check prerequisites for managed nodes
- Deployment jobs
- Generic server configuration
- Manual agent installation

Configure agent deployment to UNIX and Linux nodes

If you want to deploy HTTPS agents to nodes that have a supported UNIX or Linux operating system, you must configure a suitable secure shell client on the management server . The secure shell client must provide secure file transfer and remote command execution functionality.

By default, HPOM does not configure a secure shell client as part of the management server. You must configure your own. For convenience, third-party software (PuTTY) is available in the following folder:

<*install_dir* >`\contrib\OVOW\PuTTY`

If you choose to use this third-party software, copy the files `PLINK.EXE` , `PSCP.EXE` and `runplink.cmd` to any folder that is included in your PATH environment variable. For example, if you installed the management server into `c:\Program Files\HP\HP BTO Software` , you could copy the files into the following folder:

`c:\Program Files\HP\HP BTO Software\bin`

DISCLAIMER:
PuTTY is not HP software. It is provided "as is" for your convenience. You assume the entire risk relating to the use or performance of PuTTY.

After you install a suitable secure shell client, you must configure the management server to use the client. You must configure the following attributes in the namespace `depl.mechanisms.ssh` :

- COPY
  This attribute defines a command that the management server can use to copy files to remote UNIX and Linux nodes.
- EXEC
  This attribute defines a command that the management server can use to execute commands on remote UNIX and Linux nodes.

You can view and change these attributes using the commands `ovconfget` and `ovconfchg` from a command prompt. You must configure appropriate values for whichever secure shell client you install. By default, the attributes have the following values:

```
[depl.mechanisms.ssh]
COPY=pscp -q  -batch -pw <passwd> <sourcefile> <user>@<host>:<targetfile>
EXEC=runplink.cmd <passwd> <user> <host> <command>
```

The management server substitutes variables in angle brackets (< >) with actual values when it calls the command.

The default value for the COPY attribute configures the management server to call pscp to copy files to nodes.

The default value for the EXEC attribute configures the management server to call `runplink.cmd` , which is a command that calls PLINK.EXE to execute commands on nodes. When any of the PuTTY commands connects to a remote node for the first time, it requests verification of the node's SSH host key. The command `runplink.cmd` approves this request silently. PuTTY stores the node's key automatically in the Windows registry to use for future connections. If you need to verify SSH host keys manually, you can reconfigure the EXEC attribute.

To configure manual verification of SSH host keys:

1. Log in to the management server as a user with administrative rights, and open a command prompt.
2. Type **`ovconfchg -edit`** . A text editor appears, which enables you to edit attributes.
3. Find the namespace `[depl.mechanisms.ssh]` , and replace the EXEC attribute with a call to PLINK.EXE. The value should then be as follows:
   ```
   EXEC=plink -ssh -batch -2 -pw <passwd> <user>@<host> <command>
   ```
4. Save the file, and then close the text editor. The new configuration becomes active immediately.

After you do this, you must verify each node's SSH host key before you can deploy the HTTPS agent. You can do this by, for example, using the following `plink` command, which prompts you to verify the SSH host key:

```
plink -ssh <user name>@<node name> hostname
```

Related Topics:

- ovconfget
- ovconfchg

Configure HTTPS agent installation defaults

If you are using HTTPS agents, you can specify default settings that you want the management server to apply when it automatically installs the agent on specific nodes . You can specify default values for any attributes that you would otherwise set using `ovconfchg` on the node.

NOTE:
You can also use these settings for manual HTTPS agent installations by creating an agent profile.

To configure HTTPS agent installation defaults

1. Copy the file:

   <*share_dir* >`\conf\PMAD\agent_install_defaults.cfg.sample`

   to:

   <*share_dir* >`\conf\PMAD\agent_install_defaults.cfg`
2. Open the new file in a text editor.
3. On a new line, type the namespace that the parameter you want to set belongs to. Enclose the namespace in brackets []. For example, to specify a default value for a parameter in the sec.cm.client namespace, add the following line to the file:

   `[sec.cm.client]`
4. *Optional.* After the namespace and on a new line, specify the node or nodes to which the management server should apply the default setting. Type a pattern that matches one or more node names or IP addresses. Use standard HPOM pattern syntax. For example:

   - `node1.example.com` matches any node with a name that contains the string node1.example.com
   - `example.com$` matches any node with a name that ends with example.com
   - `^192.168.<<#> -lt 10>` matches any node with an IP address in the range 192.168.0.0 to 192.168.9.255

   Type a colon (:) at the end of the pattern.

   NOTE:
   To specify a default setting for all nodes, omit the pattern and colon.
5. Type the attribute and default value separated by an equal sign (=). If you specified a node pattern,

continue on the same line, after the colon (:). Otherwise, if the setting applies to all nodes, create a new line after the namespace. For example, to set the CERTIFICATE_DEPLOYMENT_TYPE attribute to MANUAL, type the following:

```
CERTIFICATE_DEPLOYMENT_TYPE = MANUAL
```

6. *Optional.* You add further lines under the namespace, and also add other namespaces.

Example HTTPS agent installation defaults

The follow example shows a file that configures different proxies depending on a node's IP address. It also sets the certificate deployment to manual if a node's name does not contain example.com.

```
[bbc.http]
^192.168.<*>.<*> : PROXY = proxyA.example.com:8088
^10.<*>.<*>.<*> : PROXY = proxyB.example.com:8088

[sec.cm.client]
![example.com] : CERTIFICATE_DEPLOYMENT_TYPE = MANUAL
```

Related Topics:

- Manually install an HTTPS agent with a profile
- Pattern-matching details

Configure DCE agent installation defaults

During the management server installation the default settings for DCE agents (for example the name of the management server) are automatically stored in the DCE agent package. The management server to applies these defaults when it remotely installs the DCE agent on a node .

You can use the `SetMgmtServer` command to configure the following default settings in the DCE agent package:

- Management server . To change the name of the management server in the DCE agent package, use the following command:

  ```
  SetMgmtServer /servername <name>
  ```

- Agent account . In previous releases, two local user accounts (HP ITO Account and opc_op account) were created on the node during agent installation and random passwords were generated. Since the OVO for Windows 7.20 release, the DCE agent installation does not create accounts on the node. The default settings now configure the agent to run under the Local System account.

  You can change the default setting in the DCE agent package, so that agents run under a specific user account. To do this, use the following command:

  ```
  SetMgmtServer /user <user > /password <encrypted password >
  ```

  You create the encrypted password with the tool `opcpwcrpt` . For more details, see Change the default user of DCE agents on Windows nodes .

  Alternatively, you can change the default setting in the DCE agent package to use the HP ITO Account instead. This may be useful if you are upgrading a DCE agent that already runs under the HP ITO Account. To change this default settings to use the HP ITO Account, use the following command:

  ```
  SetMgmtServer /system /off
  ```

  If you then install the DCE agent on Windows nodes, the agent installation creates the HP ITO account on the node and adds it to the Administrators group. The installation also assign additional user rights directly to the HP ITO account.

  HPOM also provides tools for managing the default DCE agent account. The tools are available in the console under Tools → HP Operations Manager Tools → Agent Account .

- Force agent account change . By default, when the management server reinstalls or upgrades the DCE agent on a node, the existing user account settings remain unchanged for that node. To force the agent installation to use account settings from the agent package, use the following command:

  ```
  SetMgmtServer /force /on
  ```

- Authentication: By default, the DCE agent does not require a password when running actions and tools under a different user account. To require passwords for actions and tools run under a different user account, use the following command:

  ```
  SetMgmtServer /auth /on
  ```

  For more details, see Security authentication module .

- Rename Perl . The Perl interpreter that is included with the agent package is installed with the name perl.exe by default. You can change this default to ovperl.exe to avoid collisions with existing Perl installations. To change this default setting in the DCE agent package, use the following command:

  ```
  SetMgmtServer /renperl /on
  ```

The full usage of SetMgmtServer is shown below:

```
Usage: SetMgmtServer.exe []
   [/servername <name>]
   [-d]
   [-s|/system [/on|/off|/default]]
   [/forced [/on|/off|/default]]
   [/auth [/on|/off|/default]]
   [/renperl [/on|/off|/default]]
   [/current]
   [/user <user>]
   [/password <encrypted password>]

   Updates the name of the management server in the agent installation <pkg>.

     /servername <name>       Sets management server name. By default current management server

     -d                       Removes the NetBIOS domain name from the
                              fully qualified hostname (if included).

     -s                       Installs agents under the Local System account.

     /system /on              Turns on Local System account installation.
     /system /off             Turns off Local System account installation.
     /system /default         Sets the System account installation to default (on).

     /forced /on              Turns on forced switch of the user account.
     /forced /off             Turns off forced switch of the user account.
     /forced /default         Sets forced switch of the user account back to default
                              (off).

     /auth /on                Turns on authentication.
     /auth /off               Turns off authentication.
```

```
/auth /default          Sets the authentication to default (off).

/renperl /on            Enforces the agent installation to rename to ovperl.exe.
/renperl /off           Sets the agent installation to use perl.exe.
/renperl /default       Sets the name of the Perl interpreter to default (off).

/current                Shows the settings in the given package or on the system.

/user <user>            Sets the user name the agent should be installed under.
/password <pwd>         Sets an encrypted password that should be used for the user.
                        NOTE: Encrypt the password with the command
                              opcpwcrpt.exe (in %OvInstallDir%\bin\OpC\install)
                        The specified password is ignored when installing the
                        agent under the Local System account.
```

You can combine several options at the same time. For example:

- To set the user to the Local System account for all DCE agent installations and upgrades, use:

  `SetMgmtServer /system /on /forced /on`

- To set the user to the HP ITO Account for all DCE agent installations, use:

  `SetMgmtServer /system /off /forced /on`

- To set the user to the Local System account for new DCE agent installations, but keep the existing user for upgrades, use:

  `SetMgmtServer /system /on /forced /off`

NOTE:
When you start `SetMgmtServer` the command writes the settings of the `/system` , `/auth` , `/forced` , and `/renperl` switches into the registry. These registry settings are used the next time you start `SetMgmtServer` , unless you specify other settings. For example, if you use the following command:

```
SetMgmtServer /system ON
```

then the existing settings for in the registry and could result in the following:

```
SetMgmtServer /system ON /auth OFF /forced OFF
```

The `-s` option is only valid for the current call of `SetMgmtServer` and is not written to the registry. The next time you call `SetMgmtServer` without the `-s` option, the registry settings are used.

NOTE:
The `SetMgmtServer` tool changes only the installation defaults in the DCE agent package on the management server. To apply changes to nodes where the DCE agent is already installed, you must

redeploy the agent.

Related Topics:

- Reinstall all

Install agents in trusted domains

You can remotely install agents on nodes in other Active Directory domains if the correct trust relationships exist. You must provide the credentials of a user who has administrative access to the node. Typically, this user is a member of the Domain Admins group in the trusted domain, or a member of the Local Administrators group on the node.

To install agents remotely, open the Agent installation dialog box.

The Agent installation dialog box provides the following options to set the credentials:

- PMAD user . To use this option, you must give the PMAD user administrative access to the node in the trusted domain. For more information, see Start Windows node security setup .
- Impersonate user . To use this option, you must log in to Windows with the credentials of user who has administrative access to the node in the trusted domain. The user must also be a member of the HPOM administrators group (called HP-OVE-ADMINS by default). This group is created when the management server is installed, and may either be a domain group, or a local group on the management server. If the HPOM administrators group is a domain group, you may need to change the group scope to universal before you can add a user from a trusted domain. For more details, refer to the Active Directory documentation that Microsoft provides.
- User/Password . To use this option, give the credentials of the user who has administrative access to the node in the trusted domain. This option is available only if you are installing the HTTPS agent. Specify the username in one of the following formats:
  - domain\user (for an Active Directory user)
  - computer_name\user (for a local user on the node)

NOTE:
To avoid communication problems, ensure that nodes in trusted domains can resolve the name of the management server. For more details, see Resolve the IP address of the management server .

Related Topics:

- Trust relationships between Active Directory domains

Specify folders for remote agent installation

When you install an HTTPS agent, the installation creates two main folders on the node :

- Installation folder
  This folder contains the agent software (for example, the executable files that collect data, discover services, and monitor events).
- Data folder
  This folder contains the data that the agent receives and creates (for example, policies, instrumentation, and performance data).

The following table lists the default installation and data folders for the HTTPS agent, which vary according to the node's operating system.

Operating system	Installation folder	Data folder
Windows Server 2008 Windows Vista	`%ProgramFiles%\HP\HP BTO Software\`	`%AllUsersProfile%\HP\HP BTO Software`
Windows Server 2003 Windows XP		`%AllUsersProfile%\Application Data\HP\HP BTO Software`
HP-UX Solaris Linux	`/opt/OV/`	`/var/opt/OV/`
AIX	`/usr/lpp/`	`/var/opt/OV/`
Tru64	`/usr/opt/OV/`	`/usr/var/opt/OV/`

On new nodes that run a Windows operating system, you can install the HTTPS agent into different folders. It is not possible to specify different folders in any other scenario:

- Remote or manual agent installations on UNIX and Linux nodes always use the default folders.
- Upgrades of existing HTTPS agents always use the same folders as the existing agent.

NOTE:
To specify a different data folder on Windows nodes you need HTTPS agent version 8.53 or higher. If necessary, download the latest agent patch from the HP Software Support web site.

To specify folders for remote agent installation

1. Prepare the agent installation defaults file, as described in the topic Configure HTTPS agent installation defaults .

2. On a new line in the agent installation defaults file, type the following namespace:

   ```
   [pmad]
   ```

3. Under the `pmad` namespace add the following parameters on separate lines:

   ```
   INSTALL_DIR="<installation_folder >"
   DATA_DIR="<data_folder >"
   ```

4. *Optional.* Specify the node or nodes to which the management server should use the folder settings. Prefix each parameter with a pattern that matches one or more node names or IP addresses. Use standard HPOM pattern syntax.

   Type a colon (:) at the end of the pattern.

NOTE:
The management server applies the settings in the `pmad` namespace to remote agent installations only. To specify folders for manual agent installations, see Specify folders for manual agent installation .

Example

```
[pmad]
^192.168.<*>:INSTALL_DIR="C:\HP Operations Agent"
^192.168.<*>:DATA_DIR="C:\HP Operations Agent Data"
```

The above example specifies the following folders for remote agent installation on nodes with IP addresses that begin 192.168.*

- Installation folder: `C:\HP Operations Agent`
- Data folder: `C:\HP Operations Agent Data`

On nodes with any other IP addresses, the management server installs the agents using the default folders.

Related Topics:

- Pattern-matching details

Disable automatic agent installation

By default, the management server automatically installs or updates agents when you deploy policies. If you deploy a policy that requires a newer version of the agent than is currently installed on the node , the management server automatically creates a job to first deploy the latest agent packages to the node. You can disable this feature if, for example, you prefer to install agents manually.

If you disable this feature, a policy deployment job fails if the policy requires a version of the agent that is newer than the current version on the node. In this case, you must deploy the agent packages from the console, or you can install the latest agent on a node manually.

To disable automatic agent installation

1. In the console tree, right-click Operations Manager , and then click Configure→ Server... . The Server Configuration dialog box appears.
2. Select the Expert Mode check box.
3. Click Namespace , and then click Policy Management and Deployment . A list of values appears.
4. Set the value of Automatic agent upgrade to False.
5. Click OK .

NOTE:
To re-enable automatic agent upgrade, set Automatic agent upgrade back to True .

Related Topics:

- Deploy deployment package
- Manual agent installation
- Change server configuration values

Deploy agent hotfixes

HP Software Support may provide you with hotfixes for the HTTPS agent to address specific change requests. Agent hotfixes normally include several updated files, which you can install on affected agents immediately (without having to wait for the next version of the agent package to become available).

HPOM provides several tools that enable you to deploy and manage agent hotfixes remotely from the management server. These tools enable you to perform the following tasks:

- Deploy agent hotfixes
- List installed agent hotfixes
- Remove agent hotfixes
- Roll back agent hotfixes

To deploy agent hotfixes

1. Extract the hotfix files to a temporary folder on the management server called `c:\temp\hotfix` (or, if necessary, any other temporary folder).
2. Launch the tool **Tools** → **Hotfix Deployment - HP OvEaAgt** → **Copy Hotfix** .

   The tool copies the hotfix files from the temporary folder `c:\temp\hotfix` to a target folder on the management server. If you extracted the hotfix files to any other folder, specify the folder in the tool parameters before you launch the tool.

   NOTE:
   The Copy Hotfix tool creates a target folder based on information in the hotfix files. If previous hotfixes already exist in the target folder, the tool overwrites the files. However, hotfixes are cumulative, and contain all the fixes for a particular version of the agent.

3. Launch the tool **Tools** → **Hotfix Deployment - HP OvEaAgt** → **Select Hotfix** . In the Edit Parameters dialog box, select the nodes or node groups to which you intend to deploy the hotfixes, and then click **Launch...** .

   The Select Hotfix tool creates a configuration file for each combination of operating system, binary format, and agent version. If you want to deploy only a subset of available hotfixes, you can edit the configuration files.

4. Launch the tool **Tools** → **Hotfix Deployment - HP OvEaAgt** → **Deploy Hotfix** . In the Edit Parameters dialog box, select the nodes or node groups to which you want to deploy the hotfixes, and then click **Launch...** .

   The Deploy Hotfix tool copies the hotfixes to each node and starts an installation script on the node.

The Tool Status dialog box opens, and shows the results of the hotfix deployment. The following log files also contain details of the results:

- On the management server: `%OvAgentDir%\log\Agt_Hotfix_Install.log`
- On nodes with a Windows operating system: `%OvDataDir%\log\hotfix_inst.log`
- On nodes with a UNIX or Linux operating system: `/var/opt/OV/log/hotfix_inst.log`

NOTE:
If more recent hotfixes already exist on a node, the tool does not deploy the currently selected hotfixes to that node.

To list installed agent hotfixes

1. Launch the tool Tools → Hotfix Deployment - HP OvEaAgt → List Inventory .
2. In the Edit Parameters dialog box, select the nodes or node groups for which you want a list of installed hotfixes, and then click Launch... . The Tool Status dialog box opens, and shows the inventory of each node that you selected.

To remove agent hotfixes

You can remove all deployed hotfixes from nodes to restore the agent to the latest installed version.

1. Launch the tool Tools → Hotfix Deployment - HP OvEaAgt → Remove Hotfix .
2. In the Edit Parameters dialog box, select the nodes or node groups from which you want to remove the hotfixes, and then click Launch... . The Tool Status dialog box opens, and shows the results of the hotfix removal.

To roll back agent hotfixes

You can roll back the most recently deployed hotfixes to restore previously installed hotfixes. If no previous hotfixes exist on a node, or you have already rolled back to the previous hotfixes, the tool restores the agent to the latest installed version.

1. Launch the tool Tools → Hotfix Deployment - HP OvEaAgt → Rollback Hotfix .
2. In the Edit Parameters dialog box, select the nodes or node groups on which you want to roll back the hotfixes, and then click Launch... . The Tool Status dialog box opens, and shows the results of the hotfix roll back.

Related Topics:

- Configure tool details

Start Windows node security setup

Before HPOM for Windows 8.00, to deploy an agent to nodes with the Windows operating system, you had to add a domain group (called HP-OVE-GROUP by default) to the node's local administrators group. You could do this manually or using the Windows Node Security Setup dialog box. On the management server , the policy management and deployment (PMAD) service ran under an account that was a member of this domain group, and therefore had administrative access the nodes.

HPOM now enables you to install the HTTPS agent using the credentials that you are currently logged in to Windows with. This is called impersonation, because the PMAD service runs under it's own user account (called HP-OVE-Deleg-User by default), but uses your credentials to access to the nodes. (This requires that the PMAD user is trusted for delegation in Active Directory, unless your console runs directly on the management server.)

Alternatively, you can now also install the HTTPS agent using the credentials of a user who already has access to the node. For example, you can specify the user name and password of the node's local administrator.

Therefore, it is no longer necessary to add a domain group to the node's local administrators group. Nevertheless, you can still give the PMAD user administrative access to nodes. This may be useful if you need to install DCE agents on Windows nodes, or so that console users who do not otherwise have administrative access can install agents.

HPOM enables you to add the PMAD user to the nodes' local administrators group in the following ways:

- Start Windows node security setup for specific nodes
- Start Windows node security setup automatically for new nodes

Alternatively, if you are installing DCE agents on Windows nodes, HPOM starts the Windows node security setup automatically.

NOTE:
Windows node security setup can add the PMAD user to nodes in the same domain as the user that you are currently logged in to Windows with. For nodes in untrusted domains or workgroups, you must manually create the PMAD user in the nodes' local administrators group. The management server uses pass-through authentication to access these nodes. Therefore, you must ensure that the name and password of user that you create are identical to those that the PMAD service runs under.

To start Windows node security setup for specific nodes

1. Log in to Windows with an account that has administrative rights on the nodes, and open the console.
2. In the console tree, click Tools → HP Operations Manager Tools . A list of tools appears in the

details pane.

3. Right-click Windows Node Security Setup and then click All Tasks →Launch Tool... . The Edit Parameters dialog box opens.
4. Select the check boxes for the nodes and node groups that you want to configure, and then click OK . The Windows Node Security dialog box opens.

To start Windows node security setup for automatically for new nodes

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Select the Expert Mode check box.
3. Click Namespaces , and then click Policy Management and Deployment . A list of values appears.
4. Set the value of Enable the old node security setup dialog for nodes to True . The Windows Node Security Setup dialog box opens automatically when you install an agent remotely on a Windows node.

The Windows Node Security Setup dialog box displays the following information:

- Node: lists each new or selected node.
- Status: displays a status of failed, succeeded, in progress, or waiting for each node.
- Note: displays a message explaining the reported status.

If the attempt to add the user fails for any node, click the node, and then click Details . An error message appears, which explains the cause of the failure and suggests actions to correct the problem. Examples of the problems that can occur are as follows:

- You do not have sufficient rights to add the user. Even if the user already exists on the node, you cannot detect this without administrative rights on the node.
- The node may not be reachable, because of a problem with the network or operating system.
- The node may not be reachable, because the network properties are incorrectly configured in the Node Properties dialog box.

Related Topics:

- Install agents remotely
- Install agents in trusted domains
- Configure network information for managed nodes

Manual agent installation

In some situations, you may want to install an agent on a node manually, without using the management server . For example, you may not be able to install software on a remote system because of security restrictions like firewalls.

In addition, it is not possible to remotely deploy DCE agents to node that have a UNIX or Linux operating system. Therefore, the only option is to install agents on these systems manually.

The procedures for manual agent installation are different, depending on whether you want to use the HTTPS agent or the DCE agent. Although the DCE agent is currently still supported, you are encouraged to use the HTTPS agent for new nodes.

Related Topics:

- Manual installation of HTTPS agents
- Manual installation of DCE agents

Manual installation of HTTPS agents

In some situations, you may want to install an HTTPS agent on a system manually, without using the management server . HTTPS agent installation packages are available, which you can copy or transfer to a system. You can then start the agent installation locally. You must use the correct installation package for the node's architecture and operating system. The command `ovpmutil` enables you to download the correct installation package. You can also find installation packages in the following folder on the management server:

`%OvShareDir%Packages\HTTPS \`

Subfolders below this contain different packages for each supported agent platform. (For up-to-date information on supported agent platforms, see the support matrix at HP Software Support Online.) The subfolders below `%OvShareDir%Packages\HTTPS \` conform to the following structure:

<*os type* >\<*os vendor* >\<*os versions* >\<*agent binary format* >\Operations-agent\<*agent version* >\<*os bit length* >\https

NOTE:
By default `%OvShareDir%` on the management server is `\Documents and Settings\All Users\Application Data\HP\HP BTO Software\shared\` . An alternative path may have been configured during installation.

The HTTPS agent packages contain subpackages for all supported locales. This means, for example, that you can download an agent package from a management server that runs in the English locale, and then install it on a node that runs in the Japanese locale.

If the node does not exist in the console, or does not have any custom configuration, you can manually install an agent using the default configuration. Otherwise, if you have already created the node in the console and configured it, you can copy its profile to the system to use during the installation process. You can also preinstall HTTPS agents, and then configure and start them at a later time.

When you install an HTTPS agent, the installation creates two main folders on the node (an installation folder, and a data folder). On new nodes that run a Windows operating system, you can manually install the HTTPS agent into folders that you specify.

Every node that you manage with an HTTPS agent requires a certificate. If you are manually installing HTTPS agents, you should also plan certificate deployment before you begin.

Related Topics:

- Manually install a default HTTPS agent
- Manually install an HTTPS agent with a profile
- Preinstall an HTTPS agent
- Specify folders for agent installation and data

- Configuring Certificates

Manually install a default HTTPS agent

You can install the HTTPS agent software manually, instead of deploying it to the node using the management server . You can install an HTTPS agent with the default configuration in either of the following situations:

- the node does not already exist on the management server
- the node already exists on the management server, but has a default configuration

To manually install a default HTTPS agent

1. Log in to the node as a user with administrative rights.
2. *Optional.* Check that the node meets all prerequisites for the agent.
3. Copy the HTTPS agent installation packages into a temporary directory on the node. The agent packages are available on the management server. Different packages are available to support different platforms. You can obtain the correct package using the command `ovpmutil` on the management server.

   If the node already exists on the management server, and the node's system properties are correct, use the following parameters:

   `ovpmutil dnl pkg Operations-agent [`<*package version* >`] [/d `<*target directory* >`] /pnn` <*primary node name* >

   Otherwise, you must specify the system properties on the command line, using the following parameters:

   `ovpmutil dnl pkg Operations-agent [`<*package version* >`] [/d `<*target directory* >`] /ost` <*OS type* > `/osv` <*OS version* > `/abf` <*agent binary format* >

   The package version and target directory parameters are optional:

   <*package version* >
   : Specify the version of the agent package that you want to download.

   /d <*target directory* >
   : Specify the directory to which you want to download the agent package. If you omit this parameter, the command downloads the packages to the current directory.

   For example:
   ```
   ovpmutil dnl pkg Operations-agent /pnn node1.example.com
   ovpmutil dnl pkg Operations-agent 8.50.10 /d c:\temp /ost Solaris /osv 9 /abf SPARC
   ```

TIP:
To check which agent packages are available, and find values for the package version, OS type, OS version, and agent binary format parameters, view the deployment packages report. In the console tree, right-click Deployment packages , and then click View→ Package details .

4. Open a command prompt and navigate to the temporary directory that you copied the installation packages into. The directory contains opc_inst , which you use to start the installation.

   - On Windows operating systems:
     cscript opc_inst.vbs -srv <*management_server_host_name* > -cert_srv <*certificate_server_host_name* >
   - On UNIX and Linux operating systems:
     i. Add permission to execute opc_inst (for example, using the command chmod u+x opc_inst).
     ii. **./opc_inst -srv <*management_server_host_name* > -cert_srv <*certificate_server_host_name* >**

   This installs the agent software and sends a certificate request to the certificate server that you specify. If it does not already exist, you must configure the node in the console and ensure that the node receives a certificate.

Related Topics:

- Configure managed nodes
- Configuring Certificates
- ovoreqcheckagt

Manually install an HTTPS agent with a profile

If you need to manually install an HTTPS agent with a custom configuration, you can create an agent profile. An agent profile contains a list of settings for an agent on a specific node . For example, you can use an agent profile to configure the communication ports and certificate handling for a node.

To manually install an HTTPS agent with a profile

1. Create an HTTPS node using the console, but do not deploy the agent packages to the node.
2. Open a command prompt and use the following command to generate an agent profile for the node:

   `ovpmutil dnl prf [/d` *<target directory* `>] [/fqdn` *<fully qualified domain name* `>] | [/ip` *<ip address* `>] | [/nodeid` *<object id* `>]`

   If you omit the `/d` option, the command downloads the packages to the current directory. You only need to specify one of the command line options `/fqdn` , `/ip` , or `/nodeid` . The command generates a file on the management server with a name in the format `<node>.profile` , where `<node>` matches the command line option you choose.

   For example, the following command generates the profile for a node with the host name `node100.example.com` and saves it in on the management server in the file `c:\temp\node100.example.com.profile`

   ```
   ovpmutil dnl prf /d c:\temp /fqdn node100.example.com
   ```
3. The profile contains a list of settings for the node. If the file `%OvShareDir%conf\PMAD\agent_install_defaults.cfg` exists, the profile includes defaults from this file. If necessary, edit the profile with a text editor. You can specify default values for any attributes that you would otherwise set using ovconfchg on the node.
4. Log in to the node as a user with administrative rights.
5. *Optional.* Check that the node meets all prerequisites for the agent.
6. Create a temporary folder on the node, and then copy the profile file from the management server to this temporary folder.
7. Copy the HTTPS agent installation packages into a temporary directory on the node. The agent packages are available on the management server. Different packages are available to support different platforms. You can obtain the correct package using the command `ovpmutil` on the management server.

   If the node already exists on the management server, and the node's system properties are correct, use the following parameters:

`ovpmutil dnl pkg Operations-agent [<`*package version*`>] [/d <`*target directory*`>] /pnn <`*primary node name*`>`

Otherwise, you must specify the system properties on the command line, using the following parameters:

`ovpmutil dnl pkg Operations-agent [<`*package version*`>] [/d <`*target directory*`>] /ost <`*OS type*`> /osv <`*OS version*`> /abf <`*agent binary format*`>`

The package version and target directory are optional:

<*package version*>
: Specify the version of the agent package that you want to download.

/d <*target directory*>
: Specify the directory to which you want to download the agent package. If you omit this option, the command downloads the packages to the current directory.

For example:
```
ovpmutil dnl pkg Operations-agent /pnn node1.example.com
ovpmutil dnl pkg Operations-agent 8.50.10 /d c:\temp /ost Solaris /osv 9 /abf SPARC
```

TIP:
To check which agent packages are available, and find values for the package version, OS type, OS version, and agent binary format parameters, view the deployment packages report. In the console tree, right-click **Deployment packages**, and then click **View** → **Package details**.

8. Open a command prompt and navigate to the temporary directory that contains the installation packages and agent profile. The directory contains `opc_inst`, which you use to start the installation as follows:
   - On Windows operating systems:
     `cscript opc_inst.vbs -configure <`*profile_filename*`>`
   - On UNIX and Linux operating systems:
     i. Add permission to execute `opc_inst` (for example, using the command `chmod u+x opc_inst`).
     ii. `./opc_inst -configure <`*profile_filename*`>`

   This installs the agent and starts it with the configuration settings from the profile. You must ensure that the node receives a certificate.

Related Topics:

- Configure managed nodes
- Configure HTTPS agent installation defaults
- ovoreqcheckagt
- Configuring Certificates

Preinstall an HTTPS agent

You can preinstall the HTTPS agent software on a system without configuring or starting the agent. You then can start the agent at a later date, using either the default configuration or a custom configuration.

To preinstall an HTTPS agent

1. Log in to the node as a user with administrative rights.
2. *Optional.* Check that the node meets all prerequisites for the agent.
3. Copy the HTTPS agent installation packages into a temporary directory on the node. The agent packages are available on the management server . Different packages are available to support different platforms. You can obtain the correct package using the command `ovpmutil` on the management server.

   If the node already exists on the management server, and the node's system properties are correct, use the following parameters:

   `ovpmutil dnl pkg Operations-agent [`<*package version* >`] [/d `<*target directory* >`] /pnn` <*primary node name* >

   Otherwise, you must specify the system properties on the command line, using the following parameters:

   `ovpmutil dnl pkg Operations-agent [`<*package version* >`] [/d `<*target directory* >`] /ost` <*OS type* > `/osv` <*OS version* > `/abf` <*agent binary format* >

   The package version and target directory parameters are optional:

   <*package version* >
   : Specify the version of the agent package that you want to download.

   /d <*target directory* >
   : Specify the directory to which you want to download the agent package. If you omit this parameter, the command downloads the packages to the current directory.

   For example:
   ```
   ovpmutil dnl pkg Operations-agent /pnn node1.example.com
   ovpmutil dnl pkg Operations-agent 8.50.10 /d c:\temp /ost Solaris /osv 9 /abf SPARC
   ```

   TIP:
   To check which agent packages are available, and find values for the package version, OS type, OS version, and agent binary format parameters, view the deployment packages report. In the console tree, right-click Deployment packages , and then click View→ Package details .

4. Open a command prompt and navigate to the temporary directory that contains the installation packages. The directory contains `opc_inst`, which you use to start the installation as follows:

   - On Windows operating systems type **`cscript opc_inst.vbs -no_start`** and then press Enter.
   - On UNIX and Linux operating systems:
     i. Add permission to execute `opc_inst` (for example, using the command `chmod u+x opc_inst`).
     ii. Type **`./opc_inst -no_start`** and then press Enter.

To start a preinstalled HTTPS agent with the default configuration

Open a command prompt on the node and navigate to the directory `%OvInstallDir%bin\OpC\install`. The directory contains `opcactivate`, which you use to start the agent as follows:

- On Windows operating systems:
  `cscript opcactivate.vbs -srv <`*`management_server_host_name`*`> -cert_srv <`*`certificate_server_host_name`*`>`
- On UNIX and Linux operating systems:
  `./opcactivate -srv <`*`management_server_host_name`*`> -cert_srv <`*`certificate_server_host_name`*`>`

This starts the agent and sends a certificate request to the certificate server that you specify. If it does not already exist, you must create the node in the console and ensure that the node receives a certificate.

To start a preinstalled HTTPS agent with a custom configuration

1. Create the profile on the management server and copy it to the node.
   a. Open a command prompt on the management server and use the following command to generate an agent profile for the node:

      `ovpmutil dnl prf [/d <`*target directory*`>] [/fqdn <`*fully qualified domain name*`>] | [/ip <`*ip address*`>] | [/nodeid <`*object id*`>]`

      You only need to specify one of the command line options `/fqdn`, `/ip`, or `/nodeid`. The command generates a file with a name in the format `<node>.profile`, where `<node>` matches the command line option you choose.

      For example, the following command generates the profile for a node with the host name `node100.example.com` and saves it in the file `c:\temp\node100.example.com.profile`

      ```
      ovpmutil dnl prf /d c:\temp /fqdn node100.example.com
      ```

   b. The profile contains a list of settings for the node. If the file `%OvShareDir%conf\PMAD\agent_install_defaults.cfg` exists, the profile includes defaults from this file. If necessary, edit the profile with a text editor. You can specify default values for any attributes that you would otherwise set using ovconfchg on the node.

2. Open a command prompt on the node and navigate to the directory `<`*`install_dir`*`>\bin\OpC\install`. The directory contains `opcactivate`, which you use to start the agent as follows:

   - On Windows operating systems:
     `cscript opcactivate.vbs -configure <`*`profile_filename`*`>`
   - On UNIX and Linux operating systems:
     `./opcactivate -configure <`*`profile_filename`*`>`

   This starts the agent with the configuration settings from the profile. You must ensure that the node receives a certificate.

Related Topics:

- Configure managed nodes
- ovoreqcheckagt
- Configuring Certificates

Specify folders for manual agent installation

When you install an HTTPS agent, the installation creates two main folders on the node :

- Installation folder
  This folder contains the agent software (for example, the executable files that collect data, discover services, and monitor events).
- Data folder
  This folder contains the data that the agent receives and creates (for example, policies, instrumentation, and performance data).

The following table lists the default installation and data folders for the HTTPS agent, which vary according to the node's operating system.

Operating system	Installation folder	Data folder
Windows Server 2008 Windows Vista	`%ProgramFiles%\HP\HP BTO Software\`	`%AllUsersProfile%\HP\HP BTO Software`
Windows Server 2003 Windows XP		`%AllUsersProfile%\Application Data\HP\HP BTO Software`
HP-UX Solaris Linux	`/opt/OV/`	`/var/opt/OV/`
AIX	`/usr/lpp/`	`/var/opt/OV/`
Tru64	`/usr/opt/OV/`	`/usr/var/opt/OV/`

On new nodes that run a Windows operating system, you can install the HTTPS agent into different folders. It is not possible to specify different folders in any other scenario:

- Remote or manual agent installations on UNIX and Linux nodes always use the default folders.
- Upgrades of existing HTTPS agents always use the same folders as the existing agent.

NOTE:
To specify a different data folder on Windows nodes you need HTTPS agent version 8.53 or higher. If necessary, download the latest agent patch from the HP Software Support web site.

To specify agent installation and data folders

1. Prepare the manual agent installation as described in one of the following topics, but do not start the installation script:
   - Manually install a default HTTPS agent
   - Manually install an HTTPS agent with a profile
   - Preinstall an HTTPS agent
2. Open a command prompt and navigate to the temporary folder that contains the installation packages. The folder contains `opc_inst.vbs` , which you use to start the installation as follows:
   - Default HTTPS agent :
     `cscript opc_inst.vbs -srv <`*`management_server_host_name`*` > -cert_srv <`*`certificate_server_host_name`*` > -inst_dir <`*`installation_folder`*` > -data_dir <`*`data_folder`*` >`
   - HTTPS agent with a profile:
     `cscript opc_inst.vbs -configure <`*`profile_filename`*` > -inst_dir <`*`installation_folder`*` > -data_dir <`*`data_folder`*` >`
   - HTTPS agent preinstallation:
     `cscript opc_inst.vbs -no_start -inst_dir <`*`installation_folder`*` > -data_dir <`*`data_folder`*` >`

   The script installs the agent into the installation and data folders that you specify. If you preinstall an agent, you must later start the agent. In all cases, you must ensure that the node receives a certificate.

NOTE:
The command-line options are available for manual agent installations only. To specify folders for remote agent installations, see Specify folders for remote agent installation .

Related Topics:

- Configuring Certificates
- Environment variables

Manual installation of DCE agents

Before HPOM for Windows 8.00, management servers and agents communicated using DCE/RPC. HPOM 8.10 includes an agent that communicates using HTTPS. Although the DCE agent is currently still supported, you are encouraged to use the HTTPS agent for new nodes.

However, in some situations, you may want to install a DCE agent on a system. You can deploy DCE agents from the management server , or manually, without using the management server.

DCE agent installation packages are available, which you can copy or transfer to a system. You can then start the agent installation locally. The DCE agent packages are available on the management server in the folder `%OvShareDir%Packages\` . This folder contains subfolders for the supported operating systems and node architectures. You must copy the correct installation package for the node's operating system and architecture.

NOTE:
By default `%OvShareDir%` on the management server is `\Documents and Settings\All Users\Application Data\HP\HP BTO Software\shared\` . An alternative path may have been configured during installation.

For up-to-date information on supported agent platforms, see the support matrix at HP Software Support Online.

The instructions for deploying DCE agents manually are provided in read-me files with the installation packages. You can copy these along with the package to the node for reference when performing the installation. The files are also available here:

- Read-me file for DCE agents on Windows
- Read-me file for DCE agents on HP-UX
- Read-me file for DCE agents on Linux
- Read-me file for DCE agents on OpenVMS
- Read-me file for DCE agents on Solaris
- Read-me file for DCE agents on Tru64

Related Topics:

- Manual installation of HTTPS agents
- Remote agent installation

Read-me File for HP Operations DCE Agents on Windows

Revised March 2008

Contents

Supported Platforms

Platform support information changes regularly. For up-to-date information, see the support matrix at HP Software Support Online.

Hardware Requirements

Before installing the agent on a supported Windows system, make sure that the system meets the following hardware requirements:

- Agent Processes - At least 15 MB memory for agent processes.
- Local Drive - 40 MB hard disk space required for installation.

> NOTE:
> Up to 40MB may be required for the performance database, depending on the configured collections. The actual disk space used depends on which packages and policies are installed on the managed node and the amount of performance data collected.

Software Requirements

The following section contains requirements for Windows managed nodes Windows 2000, XP, Windows Server 2003, and Novell NetWare.

You can run without NetBIOS as part of your Windows 2000 and XP managed node networking environments as long as Active Directory Service (ADS) client software is installed.

Running the management server in a Microsoft Cluster HA package is possible. You should note the following requirements and limitations when installing on a node in a cluster:

- Management server software runs only on that node.
- Management server software cannot be run as an HA package; it cannot switch to another cluster system.

Restrictions if the Default Local System Account is Not Used

Observe the following restrictions if you are installing to an Active Directory domain controller. If your agent is running under the Local System account, you can disregard these instructions.

If you want to install agents on Active Directory domain controllers (DC) and if you cannot use the default Local System account (your agent is running as an HP ITO account) follow the sequence described:

1. Install the agent software on the Active Directory DC with the Primary Domain Controller (PDC) emulator FSMO role before you attempt to install the agent on any additional DC.
2. Wait for the Active Directory domain controller account information to be replicated to the other DCs. It may take some time for the account information to be published to the other DCs. The installation of the agent software will only succeed if the account information has been replicated.
3. Install the agent software on the domain controllers.

SNMP Requirements for all Windows Managed Nodes

All Windows managed nodes must meet these SNMP requirements to collect SNMP events and discover node properties:

- SNMP service must be installed and running.
- Must have at least READ_ONLY access to the community name public (For further details, refer to the section System types in the online help.)
- Option Accept SNMP packets from any host enabled.

Windows 2000 Managed Node Software

Windows 2000 Professional, Server Edition, Advanced Server Edition or Data Center Edition, Service Pack 1, 2, 3, and 4. Windows 2000 without any service packs is not supported.

Recommended: Windows 2000 SP4.

Microsoft recommends that users accept all critical updates from Windows Update, available from the top of your Start menu. Click Windows Update to reach the Microsoft Windows Update web page, where you can scan for recent updates of critical and lesser severity. A list of critical updates is returned and you can select those you want to download. HP Operations Manager for the Windows operating system was tested with Microsoft's critical updates.

Windows XP Managed Node Software

Microsoft recommends that users accept all critical updates from Windows Update, available from the Admin dialog. Select All Programs -> Windows Update to reach the Microsoft Windows Update web page, where you can scan for recent updates of critical and lesser severity. A list of critical updates is returned and you can select those you want to download. HPOM for Windows was tested with Microsoft's critical updates.

Windows Server 2003 Managed Node Software

All Windows Server 2003 managed nodes must meet the requirements for the SNMP service for all Windows managed nodes, as described above.

Microsoft recommends that users accept all critical updates from Windows Update, available from the Admin dialog.

To reach the Microsoft Windows Update web page, where you can scan for recent updates of critical and lesser severity, select:

All Programs -> Windows Update

A list of critical updates is returned and you can select those you want to download. HPOM for Windows was tested with Microsoft's critical updates.

Agent Software Directories

HPOM for Windows provides a set of binary files for Windows managed nodes, located in the following directory on the management server.

```
%OvShareDir%\Packages\Windows\Windows_manual\<language>
```

For example:

```
C:\Documents and Settings\All Users\Application Data\HP\HP BTO
Software\shared\Packages\Windows\Windows_manual\en
```

NOTE:

To manually install the agent on a non-English language system, use the Japanese language version of the agent for Japanese systems, and the English language version for all other languages (for example, Spanish, Korean, or Chinese).

Agent Installation Procedure

Although the agent is usually installed using the management server console, there are some circumstances when installing the agent manually on the Windows node is advantageous or essential, for example:

- When there is no network connection available between the management server and the managed node system.
- Preinstalling agents if you prepare many computers at a central location.
- Installing the agent on a computer behind a firewall.

The HPOM package is a mechanism for deploying an agent without using an HPOM management server.

This installation installs the agent and the configfile policy type (which is used by some Smart Plug-ins).

Installation Types

- New

  A first time install. No previous manual agent package is installed on the target machine.

- Maintenance

  Reinvoking the install program for a manual agent package that is already installed places the installer in "maintenance mode". In this mode, three operations are available (as selected in the maintenance mode screen displayed when the install program starts). These modes are:

  - Modify - used to reconfigure the agent. The management server host name and agent service account can be changed.

- Repair - reinstalls the manual agent package using the existing configuration. Useful if you suspect that a file is missing or damaged.
- Remove - uninstalls the manual agent package.

Maintenance mode is invoked by either rerunning setup.exe or selecting the Change button for the manual agent in Add/Remove Programs.

- Upgrade

  Similar to a Modify install, except that existing agent data is preserved during the upgrade process. During installation, the install program determines automatically if an upgrade is needed.

NOTE:

You can deploy, remove, or reinstall packages on a node from the console with the credentials that you log into the console with.

Alternatively, if the console users do not have permission to add and remove programs on the managed node, you can add the policy management and deployment (PMAD) user to the local administrators group. By default, in an Active Directory environment, the management server installation creates the PMAD user with the name HP-OVE-Deleg-User (but the administrator who installs the management server can specify a different user name).

You can add the PMAD user to the local administrators group using Computer Management in Windows. Alternatively, you can start the Windows Node Security Setup on the management server. Log in to the management server as a user with administrative rights on the node (for example, a member of the domain administrators group) and then start `OveConfig.exe` from a command prompt:

<installation directory >\bin\OveConfig.exe *<management server host name* > *<unique ID of node* >

To copy the unique ID of a node:

1. Right-click the node in the console tree, and then click **Properties** .
2. In the **General** tab, select the value of **Unique ID** , including the braces ({}).
3. Right-click the selected ID, and then click **Copy** .

Installing the Agent

1. Check to ensure that the managed node meets all hardware and software requirements specified above. Update the node as necessary before continuing.

   NOTE:

Prerequisites can be checked automatically with the ovoreqcheckagt tool, with the command:

```
Windows_manual\<language>\ovoreqcheckagt.exe
```

For detailed information about the ovoreqcheckagt tool, refer to the HPOM for Windows online help.

2. On the management server, go to the directory:

   ```
   %OvShareDir%\Packages\Windows\Windows_manual\<language>
   ```

   Copy the directory to your managed node system or to some portable media.

3. On the node, start `setup.exe` .

4. In the Destination Folders screen you can use the default destination folder or select a destination folder where you want to install the agent. The destination location you select here will be used as the default folder for other HP BTO Software products and once set cannot be changed without first uninstalling all HP BTO Software products using it. This includes the agent.

   The default destination folders are:

   <*ProgramFilesFolder* >`\HP OpenView\`

   and

   <*ProgramFilesFolder* >`\HP OpenView\Data`

   where <*ProgramFilesFolder* > is the default program files parent folder on the target machine.

5. In the Agent Configuration screen, set the host name of the agent's management server and the name of the account that the agent service will use.

   In Hostname , type the fully qualified name of the management server that you want to manage this node, for example, "potato.veg.com".

   In the Run Agent service as radio group, select either HP ITO Account or LocalSystem .

   In previous releases, the two local user accounts (HP ITO Account and opc_op account) were created on the node and random passwords were generated. Since the OVO for Windows 7.20 release, the HP ITO Account and opc_op account are no longer created on the node. Instead, the Local System account is used as the default.

   If you are upgrading from a previous release that used the HP ITO Account, you can continue to use this account. Upgrades keep the existing user on the node. See the HP Operations Manager for Windows Installation Guide for further details on the HP ITO Account and Local System accounts.

   NOTE:

After the agent is installed, you can change the management server host name and agent service account by invoking Modify mode (see Installation Types).

6. Click Install to begin the installation.
7. Check the status of the completed installation operation by viewing the install log file. Log files reside in <*TempFolder* > (where <*TempFolder* > is the name of the target machine's "temp" folder. This is typically:

   `C:\Temp` or `C:\Documents and Settings\<`*user name*`>\Local Settings\Temp`

   A log file is created for every install operation (see Installation Types below) using the following naming rules:

   - AgentInstallResults.txt - new, modify, and repair installations
   - AgentUnInstallResults.txt - uninstallations
   - AgentUpgradeResults.txt - upgrade installations

   Any previous log file is saved by appending a number to its file name before the new log file is created.
8. If you want to check to see if the agent is running, open a command shell and enter:

   ```
   opcagt -status
   ```

   To restart the agent, enter:

   ```
   opcagt -start
   ```

Performing An Unattended (Silent Mode) Installation

You can install agents in unattended mode by using a configuration file and some command line parameters. In addition, you can use the configuration file in an interactive installation to offer different default values in the install wizard. In the configuration file you can specify all the configuration values that you can specify during an interactive install.

The default configuration file is named AgentConfig.ini and resides in the same location as the installation files. The installation uses this file automatically. You can also specify an alternative configuration file on the install command line (see below). For instructions on how to edit configuration parameters read the information contained in the AgentConfig.ini file.

There are three methods for launching an unattended install:

- setup.exe
- msiexec.exe
- the MSI file (using the .msi default action)

Command Line Options

- `/qr` - runs the install in "reduced" UI mode. No user input needed. Only modeless status dialogs are displayed.
- `/qb` - runs in install in "basic" UI mode. Similar to reduced mode, but adding a "+" to the option (/qb+) displays a modal dialog at the end of the installation session.
- `/qn` - runs the install with no visual indicators. Adding a "+" to the option (/qn+) displays a modal dialog at the end of the installation session.
- `CONFIGFILE=`*<config filename >*
  - use to specify a configuration file other than the default. *<config filename >* is the fully qualified name of the configuration file.

Unattended Install Examples

- Run an install completely silently. This example shows all three launch methods.

  ```
  setup.exe /v"/qn"
  ```

  or

  ```
  "HP Operations Manager Manual Agent.msi" /qn
  ```

  or

  ```
  msiexec.exe /I "HP Operations Manager Manual Agent.msi" /qn
  ```

- Run an install completely silently, and display a modal dialog signifying that the installation is complete.

  ```
  setup.exe /v"/qn+"
  ```

- Run an install showing a progress dialog, and display a modal dialog signifying that the installation is complete.

  ```
  setup.exe /v"/qb+"
  ```

- Run a completely silent install and use the custom configuration file "`c:\temp\myconfig.ini`" .

  ```
  setup.exe /v"/qn CONFIGFILE=c:\temp\myconfig.ini"
  ```

Install the Agent Using a Third-Party Tool

It is possible to use the third-party tool NetInstall to perform the manual installation of the Windows agent. If you own the NetInstall software (which is not provided with HPOM for Windows) you can use it to install the Windows agent providing you meet the preconditions described below and the correct patches are applied.

If you do not want to accept the defaults in the AgentConfig.ini file, you need to change the file in these areas: (You must at least modify the management server host name section.)

- Installation program location: Enter the desired install directory name and path.
- Management server host name: Specify the management server you want to use. (Starts out blank, so must be modified.)
- Installation data location : Specify the data directory to use.

You will find the AgentConfig.ini file with the rest of the agent package in the following folder on the management server:

`%OvShareDir%\Packages\Windows\Windows_manual\en`

To use NetInstall to manually install the Windows agent, follow these steps:

1. Change a copy of the AgentConfig.ini file.
2. Follow the NetInstall Guide instructions on setting up the `HP Operations Manager Manual Agent.msi` file.
3. Refer to the NetInstall documentation for further instructions to configure and deploy the manual agent installation.

Operating the Agent

When the computer is connected to the network, it will begin to broadcast its presence to the management server. Use the procedure below to begin to manage the node.

Managing A Windows Node With A Manually Installed Agent

To manage a Windows node with a manually installed agent, complete the following steps:

1. In the console tree, right-click a node and select Configure Nodes . The Configure Managed Nodes window opens.
2. In the Configure Managed Nodes window, expand the Unmanaged Nodes with Agents folder and locate the node that you manually added.
3. Click the node in the Unmanaged Nodes with Agents folder, drag it to the Nodes folder, and drop it into the chosen folder.
4. Select OK .
5. In the console tree, right-click the new node and select: All Tasks -> Synchronize inventory -> Packages . This updates the package inventory to reflect the version of the agent.
6. *Optional* . Start the Windows Node Security Setup, (see above).

Managing A Non-English Windows Node

After performing the installation procedure above, you need to indicate the character set that the node is using by adding an entry to the nodeinfo file on the node.

1. Open the file `%ovagentdir%\conf\opc\nodeinfo` in a text editor.
2. Add the line:

   OPC_NODE_CHARSET <*character set* >

   Where <*character set* > is one of the following:

   - Traditional Chinese: `big5`
   - Simplified Chinese: `gb2312`
   - Japanese: sjis or acp932
   - Korean: eucKR
   - Central European languages, for example Czech: `acp1250`
   - Cyrillic: `acp1251`
   - Hebrew: `acp1255`
   - Others: `utf8` or `ascii`

   If you use utf8 or ascii as codeset in the nodeinfo file, the data is not converted but is sent "as is" to the management server. Therefore the data must be in a codeset the management server can handle.

   With OVO for UNIX, the management server cannot communicate with a managed node that uses the codeset utf8. If the managed node reports to both an HPOM for Windows and an OVO for UNIX management server, use ascii instead of utf8.

NOTE:
If for some reason the node does not appear under unmanaged nodes, you can add it as if it were a new node.

Read-me File for HP Operations DCE Agents on HP-UX

Revised August 2007

Contents

Supported Platforms

Platform support information changes regularly. For up-to-date information, see the support matrix at HP Software Support Online.

Hardware Requirements

Before installing the DCE agent on a supported HP-UX system, make sure that the system meets the following hardware requirements:

- Disk Space - 45 MB (plus 45 MB required temporarily during software installation).
- Additional swap space - None
- Additional RAM - None

Patches and Software Requirements

HP-UX 11.11 Patches

On systems with the HP-UX 11.11 operating system, make sure that the system has the following software:

- Patches

PHCO_24402	libc cumulative header file patch
PHCO_26124	libc cumulative patch
PHNE_25625	ONC/NFS General Release/Performance Patch
PHSS_24638	HP aC++_AA runtime libraries (aCC A.03.33)
PHNE_31247	Cumulative ARPA Transport patch

- Required dependencies

PHCO_24777	mountall cumulative patch

Important Kernel Parameters

The following table gives values for kernel parameters on HP-UX managed nodes.

Parameter	Description	Minimum Value
nfile	Maximum number of open files	20 (see note)
semmns	Required semaphores	20
shmmax	Maximum shared memory	None required
msgmni	Message queues	None required
nflocks	File locks	10

NOTE:
The nfile number depends on several factors. Normally a value of 20 per process is sufficient. However, the more logfiles that are configured for the logfile encapsulator, the more file descriptors are needed. Normally, one logfile requires one file descriptor. Any actions that result in processes being started on the managed node need additional file descriptors.

General Software Requirements

Before installing the DCE agent, be sure the following software is installed on the supported HP-UX 11.x systems that you want to manage.

- Operating System

  Any required patches listed above.

- Kernel or System Parameters

  As listed above. You can verify and change the system parameters using the SAM tool. If monitoring performance metrics with the embedded performance component, increase the value of the kernel parameter max_thread_proc to at least (Number_of_Policies x 2).

- Internet Services

  SD package: InternetSrvcs.INETSRVCS-RUN

- LAN/9000

  SD package: Networking.NET-RUN

- SNMP Agent for MIB Monitoring

  SD Package for HP-UX 11.x and higher: OVSNMPAgent

- Native Language Support (NLS) Package

  SD package: OS-Core.NLS-AUX

Agent Software Directories

HPOM for Windows provides sets of binary files for HP-UX managed nodes, located in the directories shown below (one for directory for PA-RISC platforms and the other for Itanium IA64, native 32-bit mode).

- PA-RISC

  `%OvShareDir%\Packages\HPUX\11PA`

  For example:
  `C:\Documents and Settings\All Users\Application Data\HP\HP BTO Software\shared\Packages\HPUX\11PA`

- Itanium IA64, native 32-bit mode

  `%OvShareDir%\shared\Packages\HPUX\11IA`

  For example:
  `C:\Documents and Settings\All Users\Application Data\HP\HP BTO Software\shared\Packages\HPUX\11IA`

Agent Installation Procedure

The DCE agent can be installed on computers running HP-UX using the procedure below.

NOTE:
If the installed agent is from OVO for UNIX version 7.0 or greater, and you only want to receive messages from, and carry out actions on the managed node (that is, you don't want to deploy policies on the node), you can achieve this without reinstalling the agent. See the online help topic *Communicate with nodes installed by OVO for UNIX* for more information.

1. Ensure that you are logged in as a user with appropriate administrative rights.
2. Uninstall any existing OVO for UNIX agent software:

   ```
   swremove ITOAgent
   ```

   NOTE:
   If you are installing a newer agent version than one already installed, uninstalling of the existing agent is optional. If you uninstall the agent software, all configuration and performance data are lost. If you prefer, you can upgrade the agent without first uninstalling the earlier software version. This preserves configuration and performance data on the node.

3. Copy the files `opc_pkg.Z` , `comm_pkg.Z` , `perf_pkg.Z` , `svcdisc_pkg.Z` , `opc_inst` , `ovoreqcheckagt` , `ovoreqcheckagt.awk` , and `ovoreqcheck.cfg` from the management server to the `/tmp` directory on the computer to be managed. These files are located in: `%OvShareDir%\Packages\<supported operating system>\<supported version>\<supported version>` directory exists only if different versions have different packages.

   NOTE:
   If you use ftp to copy the files, remember to use binary mode.

4. Make the `opc_inst` and `ovoreqcheckagt` scripts executable with the commands:

   ```
   chmod +x /tmp/opc_inst
   ```

   ```
   chmod +x /tmp/ovoreqcheckagt
   ```

5. Check to ensure that the managed node meets all hardware and software requirements. Update the node as necessary before continuing.

   NOTE:
   Prerequisites can be checked automatically with the `ovoreqcheckagt` tool, with the command:

   ```
   /tmp/ovoreqcheckagt -det -agt_comm_type DCE -agt_bin_format [IPF32 | PA-RISC]
   ```

   For detailed information about the ovoreqcheckagt tool, refer to the HPOM for Windows online help.

6. Install the agent using the command:

   ```
   /tmp/opc_inst
   ```

   NOTE:
   If you install a newer agent version over one already installed, you must stop the agent with the command:

   ```
   opcagt -kill
   ```

   before starting the install script `opc_inst` .

7. Start the agent with the command:

   ```
   /opt/OV/bin/OpC/install/opcactivate -s <VP_mgt_server> -cs <server_codeset> -cn
   <agent_codeset> -sv
   ```

   NOTE:
   The server and agent code set on an English system is usually iso88591.

For non-English nodes reporting to an HPOM for Windows management server, set <server_codeset> to utf8, and set the <agent_codeset> to one of the following:

Language	nodeinfo value	Agent locale
Traditional Chinese	big5	zh_TW.big5
Simplified Chinese	gb2312	zh_CH.hp15CN
Japanese - SJIS	sjis	ja_JP.SJIS
Japanese - EUC	euc	ja_JP.eucJP
Korean	eucKR	ko_KR.eucKR
Central European languages, for example Czech	iso88592	cs_CZ.iso88592
Cyrillic	iso88595	ru_RU.iso88595
Hebrew	iso88598	iw_IL.iso88598
Others	utf8 or ascii	C

If you use utf8 or ascii as codeset in the nodeinfo file, the data is not converted but is sent "as is" to the management server. Therefore the data must be in a codeset the management server can handle.

An OVO for UNIX management server cannot communicate with a managed node that uses the codeset utf8. If the managed node reports to both an HPOM for Windows and an OVO for UNIX management server, use ascii instead of utf8.

8. Add the node to the HPOM console.
9. To synchronize the package inventory on the management server so that it includes the agent package, right-click the node in the console tree, and then click All Tasks -> Synchronize inventory -> Packages .
10. Deploy instrumentation to the node.

Read-me File for HP Operations DCE Agents on Linux

Revised January 2007

Contents

Supported Platforms

Platform support information changes regularly. For up-to-date information, see the support matrix at HP Software Support Online.

Hardware Requirements

Before installing an agent on a supported Linux system, make sure that the system meets the following hardware requirements:

- Disk Space - 70 MB (plus 70 MB required temporarily during software installation).

  NOTE:
  The agent must be installed on a second extended (ext2) file system.

- Additional swap space - None
- Additional RAM - 20 MB

Patches and Software Requirements

Important Kernel Features

The following kernel features must be enabled:

CONFIG_NET	Networking support
CONFIG_BINFMT_ELF	Kernel support for ELF binaries
CONFIG_SYSVIPC	System V IPC
CONFIG_INET	TCP/IP networking
CONFIG_NETDEVICES	Network devices support
CONFIG_EXT2_FS or CONFIG_REISERFS_FS	Second extended file system support or Reiser file system support
CONFIG_PROC_FS	Proc file system support

General Software Requirements

Use the following command to obtain a list of installed packages and the architecture of those packages:

```
rpm -qa --qf "%{NAME} %{VERSION} %{RELEASE} %{ARCH}\n "
```

The following packages must be installed:

- All Linux distributions
  - bash
  - gawk
  - DCE RPC - Delivered with the agent packages.
- RHEL 3

  uname -a = i686:

Name	Version	Release	Arch
bash	2.05b	>= 29.0.3	i386
gawk	3.1.1	>= 9	i386
glibc	2.3.2	>= 95.30	i686
compat-libstdc++	7.3	>= 2.96.128	i386
rpm	4.2.3	>= 13	i386
rpm-libs	4.2.3	>= 13	i386

Due to a bug in the "rpm" and "rpm-libs" in v4.2.2 (corruption of the RPM database when you install OPC packages) you have to upgrade to v4.2.3 (Taroon Update 4)

uname -m = ia64:

Installation of i386 rpm on ia64 fails. In the file `/etc/rpm/macros` , add the line : `%_autorelocate_path /emul/ia32-linux` (Bugzilla Bug 137452)

Name	Version	Release	Arch
bash	2.05b	>= 41.4	ia64
gawk	3.1.1	>= 9	ia64
glibc	2.3.2	>= 95.37	ia64
rpm	4.2.3	>= 13 or 24_nonptl	ia64
rpm-libs	4.2.3	>= 13 or 24_nonptl	ia64
ia32el	1.3	>=1.EL3	ia64
compat-libstdc++	7.3	>= 2.96.128	i386
glibc	2.3.2	>= 95.37	i686
redhat-lsb	1.3	>= 3.1.EL3	i386

uname -m = x86_64:

Name	Version	Release	Arch
bash	2.05b	>= 29.0.3 or 41.4	x86_64
gawk	3.1.1	>= 9	x86_64
glibc	2.3.2	>= 95.27 or 95.33	x86_64
rpm	4.2.3	>= 10 or 21_nonptl	x86_64
rpm-libs	4.2.3	>= 10 or 21_nonptl	x86_64
glibc	2.3.2	>= 95.33	i686
compat-libstdc++	7.3	>= 2.96.128	i386

- RHEL 4

  uname -m = i686:

Name	Version	Release	Arch
bash	3.0	>= 19.2	i386
gawk	3.1.3	>= 10.1	i386
glibc	2.3.4	>= 2	i686
compat-libstdc++-296	2.96	>= 132.7.2	i386

  uname -m = x86_64:

Name	Version	Release	Arch
bash	3.0	>= 19.2	x86_64
gawk	3.1.3	>= 10.1	x86_64
glibc	2.3.4	>= 2.9	x86_64
glibc	2.3.4	>= 2.9	i686
compat-libstdc++-296	2.96	>= 132.7.2	i386

NOTE:
Due to a bug in the "ia32el" package (version <= 1.2-4.EL3.1), the rpcd process hangs. You must upgrade to ia32el, which is available with RHEL 4.0 Update 2. DCE agents are not supported with the initial release of RHEL 4.0 or with RHEL 4.0 Update 1. For more information, refer to document ID OV-

EN020032 at HP Software Support Online.

- SLES 9

  uname -m = ia64:

Name	Version	Release	Arch
bash	2.05b	>= 305.6	ia64
gawk	3.1.3	>= 210.1	ia64
glibc	2.3.3	>= 98.28	ia64
ia32el	5.3	>=2.13	ia64
bash-x86	9	>= 200407011228	ia64
glibc-x86	9	>= 200407011233	ia64
compat-x86	9	>= 200407011228	ia64

NOTE:
Due to a bug in the "ia32el" package (version 4.4-1.2), the rpcd process hangs. You must upgrade to IA-32 EL 5.3.5352/ 5.3.109.44.24. The DCE agent runs with this version (available from Novell's support web site). For more information, refer to document ID OV-EN020032 at HP Software Support Online.

uname -m = x86_64:

Name	Version	Release	Arch
bash	2.05b	>= 305.9	x86_64
gawk	3.1.3	>= 210.1	x86_64
glibc	2.3.3	>= 98.38	x86_64
glibc-32bit	9	>= 200412131610	x86_64
compat-32bit	9	>= 200407011229	x86_64

The following packages are optional:

- SNMP Daemon (optional)

  To provide the management server with sufficient information to automatically determine the node type of the Linux managed node, the SNMP daemon (snmpd) should be running when you install the software remotely from the management server. After you finish the installation, the daemon must be running if you want to use MIB variable monitoring.

Agent Software Directories

HP Operations Manager for the Windows operating system provides a set of binary files for Linux managed nodes, located in the directory shown below.

```
%OvShareDir%\Packages\Linux
```

For example:

```
C:\Documents and Settings\All Users\Application Data\HP\HP BTO Software
\shared\Packages\Linux
```

Agent Installation Procedure

The agent can be installed on computers running Linux using the procedure below.

NOTE:
If the installed agent is from OVO for UNIX version 7.0 or greater, and you only want to receive messages from, and carry out actions on the managed node (that is, you don't want to deploy policies on the node), you can achieve this without reinstalling the agent. See the online help topic *Communicate with agents installed by OVO for UNIX* for more information.

1. Ensure that you are logged in as a user with appropriate administrative rights.
2. Before beginning the uninstall steps, run the command:

   ```
   opcagt -kill
   ```

3. Uninstall any existing OVO for UNIX agent software:

   ```
   rpm -e OPCSVCDISC OPCPERF OPCJRE OPC OPCCOMM
   ```

   ```
   rpm -e dce
   ```

   NOTE:
   If you are removing a version of the agent earlier than 7.0, use the command:

   ```
   rpm -e OPC dce
   ```

   To find out which version of the agent is installed, check the value of the parameter `OPC_INSTALLED_VERSION` in the `/opt/OV/bin/OpC/install/opcinfo` file on the node.

4. Copy the files `opc_pkg.Z` , `comm_pkg.Z` , `perf_pkg.Z` , `svcdisc_pkg.Z` , `opc_inst` , `ovoreqcheckagt` , `ovoreqcheckagt.awk` , and `ovoreqcheck.cfg` from the management server to the `/tmp` directory on the computer to be managed. These files are located in:

   `%OvShareDir%\Packages\<`*supported operating system* `>`

   NOTE:
   If you use ftp to copy the files, remember to use binary mode.

5. Make the `opc_inst` and `ovoreqcheckagt` scripts executable with the commands:

   ```
   chmod +x /tmp/opc_inst
   ```

   ```
   chmod +x /tmp/ovoreqcheckagt
   ```

6. Check and ensure that the managed node meets all hardware and software requirements. Update the node as necessary before continuing.

   NOTE:
   Prerequisites can be checked automatically with the ovoreqcheckagt tool, with the command:

   ```
   /tmp/ovoreqcheckagt -det -agt_comm_type DCE -agt_bin_format [x64 | x86]
   ```

   For detailed information about the ovoreqcheckagt tool, refer to the HPOM for Windows online help.

7. Install the agent using the command:

   ```
   /tmp/opc_inst
   ```

8. Start the agent with the command:

   `/opt/OV/bin/OpC/install/opcactivate -s <`*management_server* `> -cs <`*server_codeset* `> -cn <`*agent_codeset* `> -sv`

   NOTE:
   The server and agent code set on an English system is usually iso88591.

   For non-English nodes reporting to an HPOM for Windows management server, set <*server_codeset* > to utf8, and set <*agent_codeset* > to one of the following:

Language	nodeinfo value	Agent locale
Traditional Chinese	big5	zh_TW
Simplified Chinese	gb2312	zh_CN
Japanese - SJIS	sjis	ja_JP.SJIS
Japanese - EUC	euc	ja_JP
Korean	See Others below	See Others below
Central European languages, for example Czech	iso88592	cs_CZ
Cyrillic	iso88595	ru_RU
Hebrew	iso88598	iw_IL
Others	utf8 or ascii	C

If you use utf8 or ascii as codeset in the nodeinfo file, the data is not converted but is sent "as is" to the management server. Therefore the data must be in a codeset the management server can handle.

An OVO for UNIX management server cannot communicate with a managed node that uses the codeset utf8. If the managed node reports to both an HPOM for Windows and an OVO for UNIX management server, use ascii instead of utf8.

9. Add the node to the HPOM console.
10. To synchronize the package inventory on the management server so that it includes the agent package, right-click the node in the console tree, and then click All Tasks -> Synchronize inventory -> Packages .
11. Deploy instrumentation to the node.

Read-me File for HP Operations DCE Agents and SPI on OpenVMS

Revised January 2007

Contents

Supported platforms

Platform support information changes regularly. For up-to-date information, see the support matrix at HP Software Support Online.

Software requirements

Before you install the DCE agent and SPI on an OpenVMS system, ensure that the system has the following OpenVMS patches. The patches are available at the services web site:

http://www.itrc.hp.com/service/patch/mainPage.do

HP recommends that you also install the DCE Denial of Service Security Patch (DCE_030_SSRT3608 V1.0).

Version 7.3-2 Alpha

- VMS732_SYS V11.0
- VMS732_UPDATE V9.0

Version 8.2 Alpha

- VMS82A_SYS V6.0
- VMS82A_UPDATE V5.0
- VMS82A_PERFAPI V1.0

Version 8.2-1 Integrity

- VMS821I_SYS V3.0
- VMS821I_UPDATE V5.0
- VMS821I_PERFAPI V1.0

Agent software directories

HP Operations Manager for the Windows operating system provides binary files for OpenVMS managed nodes, located in the directories shown below.

- For Alpha servers:
  `%OvShareDir%\Packages\OpenVMS\Alpha`
- For Integrity servers:
  `%OvShareDir%\Packages\OpenVMS\Itanium`

For example:
`C:\Documents and Settings\All Users\Application Data\HP\HP BTO Software\shared\Packages\OpenVMS\Alpha`

Configuring the management server

Before you can manage OpenVMS nodes, you must configure your HPOM for Windows management server as follows:

1. On the management server, extract the compressed files, which contain the VMS .PCSI files that you need:
   - `opc_pkg.Z`
   - `opc_spi.Z`

   The files are located in the directories shown below:
   - For Alpha nodes:
     `%OvShareDir%\Packages\OpenVMS\Alpha`
   - For Integrity nodes:

```
%OvShareDir%\shared\Packages\OpenVMS\Itanium
```

2. Open a command prompt and navigate to the directory that contains the .PCSI files.
3. FTP the .PCSI files to the each OpenVMS node by entering the following commands:

   `ftp` <*your OpenVMS node* >
   `SYSTEM`
   <*password* >
   `ftp> cd SYS$UPDATE:` (the colon is required)
   `ftp> bin`
   `ftp> put hp-`<*architecture* >`ovoagents-`<*kit* >`.pcsi`
   `ftp> put hp-`<*architecture*> `vmsspi-`<*kit* >`.pcsi`

   NOTE:
   Replace <*architecture* > and <*kit* > with the architecture and version number from the file name, for example, `hp-i64vms-ovoagents-v0200-22-1.pcsi` .

Installing the OpenVMS DCE agent

Follow the steps in this section after you have made the appropriate updates to your management server (see Configuring the management server).

To install the OpenVMS DCE agent:

1. Log on to the OpenVMS node with system account.
2. Set your default directory to SYS$UPDATE, which contains the .PCSI files that were copied from management server:

   ```
   $ SET DEFAULT SYS$UPDATE
   ```

3. Use the `PRODUCT INSTALL` command to install the agent:

   ```
   $ PRODUCT INSTALL OVOAGENTS
   ```

   The installation procedure moves most files to subdirectories under `SYS$SPECIFIC:[OVO]` . Several files are moved to standard system directories such as `SYS$STARTUP` and `SYS$MANAGER` . One file, the shareable image `OVO$LIBOPC_R.EXE` , is moved to `SYS$SYSROOT:[SYSLIB]` .

4. To start the agent, type the following commands. When prompted, type the name of your management server.

   ```
   $ @SYS$MANAGER:OVO$CONFIG
   ```

   ```
   $ @SYS$STARTUP:OVO$STARTUP
   ```

Performing additional configuration on the management server

Follow the steps in this section *after* you install the agent on the nodes that you want to manage (see Installing the OpenVMS DCE agent). Refer to the appropriate online help sections if you need additional instructions.

1. In the console, add the OpenVMS nodes and clusters, on which you already installed the agent.
2. To synchronize the package inventory on the management server so that it includes the agent package, right-click the node in the console tree, and then click All Tasks -> Synchronize inventory -> Packages .
3. Deploy appropriate policies to each OpenVMS node. In the console tree, the OpenVMS policies are available within the following policy groups:

   Policy management > Policy groups > OpenVMS_policies
   Policy management > Policy groups > OpenVMS_SPI_policies
4. Add the message group OpenVMS to the user roles of users who monitor messages from the OpenVMS nodes.

Installing the Smart Plug-in (SPI) for OpenVMS

Follow the steps in this section *after* you complete all the steps above.

NOTE:
You must install the agent software on the selected managed node *before* you install the SPI (see Installing the OpenVMS DCE agent).

To install the files in the SPI, follow the numbered steps below:

1. Log in to the system account.
2. Set your default directory to `SYS$UPDATE` :

   ```
   $ SET DEFAULT SYS$UPDATE
   ```
3. Use the `PRODUCT INSTALL` command to install the kit:

   ```
   $ PRODUCT INSTALL VMSSPI
   ```

   The distribution procedure moves most files to subdirectories under `SYS$SPECIFIC:[OVO]` . Several files are moved to standard system directories such as `SYS$STARTUP` and `SYS$MANAGER` .
4. Start the OSSPI:

   ```
   $ @SYS$STARTUP:VMSSPI$STARTUP
   ```

NOTE:

To customize the SPI configuration files, refer to the *HP Smart Plug-in (SPI) for OpenVMS User's Guide* . This guide is available in postscript and pdf formats and the files are located in `SYS$SPECIFIC:[OVO]` with the file names:

- `VMSSPI_USER_GUIDE.PS`
- `VMSSPI_USER_GUIDE.PDF`

Read-me File for HP Operations DCE Agents on Solaris

Revised January 2007

Contents

Supported Platforms

Platform support information changes regularly. For up-to-date information, see the support matrix at HP Software Support Online.

Hardware Requirements

Before installing the agent on a supported Solaris system, make sure that the system meets the following hardware requirements:

- Disk Space - 65 MB (plus 65 MB required temporarily during software installation).
- Additional swap space - None
- Additional RAM - None

Patches and Software Requirements

The following patches are required for Sun Solaris managed nodes. They are available from the www.sunsolve.sun.com web site.

NOTE:
The Sun Microsystems download site may contain more recent patches than those listed in this document. Patches with higher version numbers than those listed should be usable, but they have not been tested with the agent. Note that Sun only allows downloading of their latest version.

Solaris 8 Patches

109147-09	SunOS 5.8: Linker patch
108434-03	SunOS 5.8: Shared library patch for C++
108827-11	SunOS 5.8: libthread patch

Solaris 9 Patches

None required.

Problems Caused by Missing OS Patches for Sun Solaris

If the operating system patches for Sun Solaris are missing, the following problems occur:

- Patch Versions

  If version -04 or -05 of patch 101327 is installed, the agent installation fails on Sun Solaris managed nodes with the following message:

  tar xof...core dump

  To solve this problem, do one of the following:

  - Install patch version -06 (or later).
  - Deinstall the old patch.

  To check which patches are currently installed on Sun Solaris systems, enter:

  ```
  showrev -p
  ```

- Multi-processor Patch

  Be sure you have the following patches installed. The monitor agent process (opcmona) may hang on a multi-processor Solaris managed node if the following patches are not installed.

  For Solaris 8, use the following recommended patches:

108827-11

Important Kernel Parameters

The following table gives values for kernel parameters on Solaris managed nodes.

Parameter	Description	Minimum Value
nfile	Maximum number of open files.	20 (see note)
semmni	Number of semaphore identifiers	30
semmns	Number of semaphores in system	200 or greater
semmsl	Maximum number of semaphores per ID	100

You can check and change the kernel parameters by editing the `/etc/system` file.

NOTE:
The minimum value of 20 for nfile depends on several factors. Normally a value of 20 per process is sufficient. However, the more logfiles that are configured for the logfile encapsulator, the more file descriptors are needed. Normally, one logfile requires about one file descriptor. Any actions that result in processes being started on the managed node need additional file descriptors.

General Software Requirements

- Communication Software

  HPOM supports the DCE RPC communication type. If none of the supported DCE packages is installed (or running) on the managed node, then the HPlwdce (HP Lightweight DCE runtime version 1.1) is installed and configured during agent installation. The supported DCE packages are listed below:

OS	DCE
Solaris 8	IBM DCE 3.1, HPlwdce, DASCOM DCE 1.1
Solaris 9	IBM DCE 3.1, HPlwdce, DASCOM DCE 1.1

- ARPA/Berkeley Services
- MIB

  The MIB monitoring functionality of HPOM requires the snmpd provided by HP, or SNMP-based, MIB-I (RFC 1156) or MIB-II (RFC1158) compliant agent software.

Agent Software Directories

HP Operations Manager for the Windows operating system provides a set of binary files for Solaris managed nodes, located in the directory shown below.

```
%OvShareDir%\Packages\Solaris
```

For example:
```
C:\Documents and Settings\All Users\Application Data\HP\HP BTO
Software\shared\Packages\Solaris
```

Agent Installation Procedure

The DCE agent can be installed on computers running Solaris using the procedure below.

NOTE:
If the installed agent is from OVO for UNIX version 7.0 or greater, and you only want to receive messages from, and carry out actions on the managed node (that is, you don't want to deploy policies on the node), you can achieve this without reinstalling the agent. See the online help topic *Communicate with nodes installed by OVO for UNIX* for more information.

1. Ensure that you are logged in as a user with appropriate administrative rights.
2. Stop all agents running on the managed node with the command:

   ```
   /opt/OV/bin/OpC/opcagt -kill
   ```

3. Uninstall any existing OVO for UNIX agent software:

   ```
   /usr/sbin/pkgrm OPC OPCCOMM OPCPERF OPCSVCDIS HPlwdce
   ```

   NOTE:
   If you are installing a newer agent version than one already installed, uninstalling of the existing agent is optional. If you uninstall the agent software, all configuration and performance data are lost. If you prefer, you can upgrade the agent without first uninstalling the earlier software version. This preserves configuration and performance data on the node.

   NOTE:
   If you are removing a version of the agent earlier than 7.0 use the command:

   ```
   /usr/sbin/pkgrm HPlwdce OPC
   ```

   To find out which version of the agent is installed, check the value of the parameter

OPC_INSTALLED_VERSION in the `/opt/OV/bin/OpC/install/opcinfo` file on the node.

4. Copy the files `opc_pkg.Z` , `comm_pkg.Z` , `perf_pkg.Z` , `svcdisc_pkg.Z` , `opc_inst` , `ovoreqcheckagt` , `ovoreqcheckagt.awk` , and `ovoreqcheck.cfg` from the management server to the `/tmp` directory on the computer to be managed. These files are located in:

   `%OvShareDir%\Packages\<`*supported operating system* `>`

   NOTE:
   If you use ftp to copy the files, remember to use binary mode.

5. Make the opc_inst and ovoreqcheckagt scripts executable with the commands:

   ```
   chmod +x /tmp/opc_inst
   ```

   ```
   chmod +x /tmp/ovoreqcheckagt
   ```

6. Check to ensure that the managed node meets all hardware and software requirements. Update the node as necessary before continuing.

   NOTE:
   Prerequisites can be checked automatically with the ovoreqcheckagt tool, with the command:

   ```
   /tmp/ovoreqcheckagt -det -agt_comm_type DCE -agt_bin_format [SPARC | x86]
   ```

   For detailed information about the ovoreqcheckagt tool, refer to the HPOM for Windows online help.

7. Install the agent using the command:

   ```
   /tmp/opc_inst
   ```

   NOTE:
   If you install a newer agent version over one already installed, you must stop the agent with the command:

   ```
   opcagt -kill
   ```

   before starting the install script opc_inst .

8. Start the agent with the command:

   `/opt/OV/bin/OpC/install/opcactivate -s <`*VP_mgt_server* `> -cs <`*server_codeset* `> -cn <`*agent_codeset* `> -sv`

   The server and agent code set on an English system is usually iso88591. For non-English nodes reporting to an HPOM for Windows management server, set <*server_codeset* > to utf8, and set <*agent_codeset* > to one of the following:

Language	nodeinfo value	Agent locale
Traditional Chinese	big5	zh_TW.big5
Simplified Chinese	gb2312	zh
Japanese - SJIS	sjis	ja_JP.PCK
Japanese - EUC	euc	ja
Korean	eucKR	ko
Central European languages, for example Czech	iso88592	cs_CZ.ISO8859-2
Cyrillic	iso88595	ru_RU
Hebrew	iso88598	he
Others	utf8 or ascii	C

If you use utf8 or ascii as codeset in the nodeinfo file, the data is not converted but is sent "as is" to the management server. Therefore the data must be in a codeset the management server can handle.

An OVO for UNIX management server cannot communicate with a managed node that uses the codeset utf8. If the managed node reports to both an HPOM for Windows and an OVO for UNIX management server, use ascii instead of utf8.

9. Add the node to the HPOM console.
10. To synchronize the package inventory on the management server so that it includes the agent package, right-click the node in the console tree, and then click All Tasks -> Synchronize inventory -> Packages .
11. Deploy instrumentation to the node.

Read-me File for HP Operations DCE Agents on HP Tru64 UNIX

Revised January 2007

Contents

Supported Platforms

Platform support information changes regularly. For up-to-date information, see the support matrix at HP Software Support Online.

Hardware Requirements

Before installing the DCE agent on a supported HP Tru64 UNIX system, make sure that the system meets the following hardware requirements:

- Disk Space - 50 MB (plus 50 MB required temporarily during software installation).

  NOTE:
  For TruCluster systems, this disk space is required on each node.

- Additional swap space - None

- Additional RAM - None

Patches and Software Requirements

Before installing the DCE agent, be sure the following software is installed on the HP Tru64 UNIX managed node you want to manage.

HP Tru64 TruCluster Software

DCERTS410 DCE Runtime Services v4.1 and DCERTS420 DCE Runtime Services v4.2 are the only valid DCE runtime kits for TruCluster systems. You must configure DCE on each TruCluster member individually.

The following patch must be installed on systems in a TruCluster environment in order for clu_add_member to propagate the agents properly when adding a TruCluster member.

You can download the latest version from the following web pages:

- TruCluster 5.1A:

  http://www.zk3.dec.com/dupatchwww/v50pats.html

Using the dupatch utility, install patch number:

TCRPAT00024600520 (00246-00)

NOTE:
LCore and the DCE agent cannot coexist on a HP Tru64 UNIX system. For LCore, `/usr/op/OV` and `/usr/var/opt/OV` are shared directories. All other directories below `/usr/var/opt/OV` except for "shared" are CDSLs. For the DCE agent, `/usr/opt/OV` and `/usr/var/opt/OV` are CDSLs. No coexistence is possible.

Important Kernel Parameters

The following table gives values for kernel parameters on HP Tru64 UNIX managed nodes.

Parameter	Description	Minimum Value
nfile	Maximum number of open files.	20 (see note)
semmns	Required semaphores.	20
shmmax	Maximum shared memory.	None required.
msgmni	Message queues.	None required.
nflocks	File locks.	10

NOTE:
This number depends on several factors. Normally a value of 20 per process is sufficient. However, the more logfiles that are configured for the logfile encapsulator, the more file descriptors are needed. Normally, one logfile requires one file descriptor. Any actions that result in processes being started on the managed node need additional file descriptors.

If monitoring performance metrics with the embedded performance component, and agent runs as non-root user, increase the value of the kernel parameter max_threads_user to:

default + (Number_of_Templates * 2)

General Software Requirements

- Basic Networking Services

  OSFCLINET4xx Basic Networking Services

- DCE Runtime Kit

DCE Runtime Kit	HP Tru64 UNIX System	OS Version
DCERTS 430 DCE	TruCluster or single	V5.1B
Runtime Services V4.3 system		

  NOTE:
  HP Operations Manager for the Windows operating system supports DCE versions supplied with the HP Tru64 UNIX operating system. DCE has to be installed separately as an optional product.

  For a TruCluster system, you must configure DCE on each TruCluster member individually.

- Japanese Base System

IOSJPBASE4xx Japanese Base System. This system is only for managed nodes running HP Tru64 UNIX in a Japanese environment.

- Package: OSFINCLUDE440

  OSFINCLUDE440 Standard Header Files package is required for building executables on HP Tru64 UNIX nodes.

- Additional Prerequisite Packages

  Install the following packages from the Associated Products Volume 1 disk, supplied with your HP Tru64 UNIX operating system.

  - IOSWWBASE<version_id>
  - IOSWWBINUCS<version_id>

  Where version_id is:

  520 for 5.1 A

  These files are located at:

  <*cdrom_mount_point* >`Worldwide_Language_Support/kit`

  These files can be installed with the command:

  `cd <`*cdrom_mount_point* `>/Worldwide_Language_Support/kit \ setld -1 pwd <`*package* `>`

Agent Software Directories

HPOM for Windows provides binary files for HP Tru64 UNIX managed nodes which are located in the directory shown below:

```
%OvShareDir%\Packages\Tru64\5.1A
```

For example:

```
C:\Documents and Settings\All Users\Application Data\HP\HP BTO
Software\shared\Packages\Tru64\5.1A
```

Agent Installation Procedure

NOTE:
Before installing new agents on any HP Tru64 UNIX node, be sure that old agents are uninstalled from the managed node. Also make sure that after old agents are uninstalled, neither of the following directories are present on the managed node:

```
/usr/opt/OV
/var/opt/OV
```

The DCE agent can be installed on computers running HP Tru64 UNIX using the procedure below. For TruCluster systems, install the agent software on any one TruCluster member; the agent software is propagated automatically to all the other TruCluster members. After verifying the installation, start the remaining TruCluster members.

NOTE:
If the installed agent is from OVO for UNIX version 7.0 or greater, and you only want to receive messages from and carry out actions on the managed node (that is, you don't want to deploy policies on the node), you can achieve this without reinstalling the agent. For more information, refer to the online help topic *Communicate with nodes installed by OVO for UNIX* .

Installing the Agent

1. Ensure that you are logged in as a user with appropriate administrative rights.
2. Uninstall any existing OVO for UNIX agent software:

   ```
   setld -d OPCAGT000 OPCCOMMAGT000 OPCPERFAGT000 OPCSVCDISCAGT000
   ```

   NOTE:
   If you are removing a version of the agent earlier than 7.0 use the command:

   ```
   setld -d OPC000
   ```

   To find out which version of the agent is installed, check the value of the parameter OPC_INSTALLED_VERSION in the `/usr/opt/OV/bin/OpC/install/opcinfo` file on the node.
3. Copy the files `opc_pkg.Z` , `comm_pkg.Z` , `perf_pkg.Z` , `svcdisc_pkg.Z` , `opc_inst` , `ovoreqcheckagt` , `ovoreqcheckagt.awk` , and `ovoreqcheck.cfg` from the management server to the `/tmp` directory on the computer to be managed. These files are located in:

   `%OvShareDir%\Packages\<`*supported operating system* `>\<`*supported version* `>\`

   <*supported version* > directory exists only if different versions have different packages.

   NOTE:
   If you use ftp to copy the files, remember to use binary mode.
4. Make the opc_inst and ovoreqcheckagt scripts executable with the commands:

```
chmod +x /tmp/opc_inst

chmod +x /tmp/ovoreqcheckagt
```

5. Check and ensure that the managed node meets all hardware and software requirements. Update the node as necessary before continuing.

   NOTE:
   Prerequisites can be checked automatically with the ovoreqcheckagt tool, with the command: `/tmp/ovoreqcheckagt -det -agt_comm_type DCE -agt_bin_format Alpha` . The `ovoreqcheckagt` tool checks the prerequisite against data in the `ovoreqcheck.cfg` file. By default, this file is placed in the local directory but you can specify an alternative location using the `-cfg` <*config* > option. For detailed information about the `ovoreqcheckagt` tool, refer to the HPOM for Windows online help.

6. Install the agent using the command:

   `/tmp/opc_inst <`*arguments* `>`

   All arguments passed to the `opc_inst` command (except `-h`) are passed to the `opcactivate` command. This command also starts the managed node; for a TruCluster system, this starts all the TruCluster members.

   On TruCluster systems, the agent software is propagated to the other TruCluster members at this time.

7. If no arguments are passed to the command above, start the agent with the command:

   `/usr/opt/OV/bin/OpC/install/opcactivate -s <`*management_server* `> -cs <`*server_codeset* `> -cn <`*agent_codeset* `> -sv`

   NOTE:
   For TruCluster systems, you must start all TruCluster members individually.

8. Add the node to the HPOM console.

   For TruCluster systems, create a node group for the TruCluster system on the management server. Add all the TruCluster members to this node group.

9. To synchronize the package inventory on the management server so that it includes the agent package, right-click the node in the console tree, and then click All Tasks -> Synchronize inventory -> Packages .

   For TruCluster systems, synchronize the package inventory for the TruCluster node group, or for each TruCluster member.

10. Deploy instrumentation to the node.

For TruCluster systems, deploy instrumentation to the TruCluster node group or to each TruCluster member.

11. Deploy policies to the node.

    For TruCluster systems, deploy policies to the TruCluster node group or to each TruCluster member.

    NOTE:
    The server and agent code set on an English system is usually iso88591.

    For non-English nodes reporting to an HPOM for Windows management server, set the <*server_codeset* > to utf8 and set the <*agent_codeset* > to one of the following:

Language	nodeinfo value	Agent locale
Traditional Chinese	big5	zh_HK.big5
Simplified Chinese	See Others below.	See others below.
Japanese - SJIS	sjis	ja_JP.SJIS
Japanese - EUC	euc	ja_JP
Korean	eucKR	ko_KR.eucKR
Central European languages, for example Czech	iso88592	cs_CZ
Cyrillic	iso88595	ru_RU
Hebrew	iso88598	iw_IL
Others	utf8 or ascii	C

    If you use utf8 or ascii as codeset in the nodeinfo file, the data is not converted but is sent "as is" to the management server. Therefore the data must be in a codeset the management server can handle.

    An OVO for UNIX management server cannot communicate with a managed node that uses the codeset utf8. If the managed node reports to both an HPOM for Windows and an OVO for UNIX management server, use ascii instead of utf8.

Verifying the Installation

After the assignment and distribution of the policies, verify the installation by executing the shell script:

```
/usr/opt/OV/bin/OpC/utils/submit.sh
```

and examining the messages in the message browser.

For TruCluster systems, perform verification for each TruCluster member.

Starting Subsequent TruCluster Members

It may be necessary to start TruCluster members that were either down during the installation and were subsequently brought up or were added after the agent software installation. The agent software is propagated automatically to the TruCluster member. Use these steps to start them:

1. On the managed node (that is, the TruCluster member), run the following command:

   `/usr/opt/OV/bin/OpC/install/opcactivate -s <`*management_server*` > -cs <`*server_codeset*` > -cn <`*agent_codeset*` > -sv`

2. In the HPOM console, add the TruCluster member to the node group for the TruCluster system.

3. Deploy instrumentation to the node.

Deinstalling Agents on a TruCluster System

For TruCluster systems, you need to deinstall the agents on only one TruCluster member. The agent software is deinstalled on all other TruCluster members automatically. To deinstall the agents, follow these steps:

1. Stop all HPOM agents running on the managed node.

   For TruCluster systems, you need to stop the agents on all TruCluster members.

2. Deinstall the agent software from the managed nodes by entering:

   ```
   setld -d OPCAGT000 OPCCOMMAGT000 OPCPERFAGT000 OPCSVCDISCAGT000
   ```

Agent migration

This version of HPOM introduces new agent packages that communicate using HTTPS. If you have existing nodes in your environment, you can continue to use the previous agent packages that communicate using DCE. Alternatively you can migrate from DCE agents to HTTPS agents.

Although all the functionality of the DCE agent is supported on the HTTPS agent, you can nevertheless, migrate from the HTTPS agent back to the DCE agent if necessary.

If you have nodes with 64 bit architectures and operating systems, you can also migrate to native 64 bit agent packages if appropriate.

Related Topics:

- Migrate from DCE to HTTPS
- Migrate from HTTPS to DCE
- Migrate from 32 bit to 64 bit

Migrate from DCE to HTTPS

If you have DCE agents installed on nodes in your environment, you can migrate to the HTTPS agent. For example, you may need to deploy a new policy that requires the HTTPS agent.

When you migrate a node from the DCE agent to the HTTPS agent, the management server removes all existing policies, instrumentation and packages from that node. After it deploys the HTTPS agent packages, it then redeploys the policies, and the packages and instrumentation that the policies require. If the node's policy inventory contains policies that are not available on the management server, the management server cannot redeploy these policies to the HTTPS agent. This can happen, for example, if the policy exists only on a different management server, or if the policy exists on the node but has been deleted from the management server.

When you migrate a DCE agent to an HTTPS agent, the installer migrates the following data:

- If the DCE agent has an ID, it remains the same on the HTTPS agent. (On the HTTPS agent, the agent ID is also known as the core ID. Other HP BTO Software may also create a core ID on the node, which the HTTPS agent then also uses.)
- The configuration settings in the opcinfo file. On the HTTPS agent, you can configure the settings using `ovconfchg` . The installer migrates the settings to the `eaagt` namespace on the HTTPS agent.
- The service discovery configuration settings INSTANCEDELETIONTHRESHHOLD and ACTION_TIMEOUT from OvJavaAgent.cfg. The installer migrates the settings to the `agtrep` namespace on the HTTPS agent.
- Data collected by the embedded performance component.

CAUTION:
When you migrate from a DCE agent to an HTTPS agent, you lose the following data from the DCE agent:

- Unsent messages in the message buffer.
- Custom settings and data from Smart Plug-ins.
- Integrations with other HP Software products.
- Custom service discovery configurations.
- Other custom settings specific to your organization.
- Obsolete settings from the opcinfo file.

If appropriate, back up this data and reconfigure it after you migrate to the HTTPS agent.

To migrate from a DCE agent to an HTTPS agent

1. In the console tree, right-click **Nodes** , and then click **Configure** → **Nodes...** . The Configure Managed

Nodes dialog appears.

2. Right-click the node and then click Change agent . The Change Agent dialog appears.
3. Click Comm Type and then select HTTPS .
4. *Optional.* Select the Auto Grant check box if you want the management server to automatically grant the certificate for this node. (This only happens if you also configure the management server to grant certificate requests automatically.)
5. *Optional.* Specify the credentials that the management server uses to deploy the agent. Right-click a node in the list, and then click Set Credentials . Click one of the following commands:
   - PMAD user . The management server attempts to deploy the agent as the user under which the policy management and deployment (PMAD) service runs (called HP-OVE-Deleg-User by default).

     You can only use this for nodes with a Windows operating system. The nodes can belong to the same domain as the management server, a trusted domain, or a workgroup.

     NOTE:
     The PMAD user does not by default have administrative access to the node. For more information, see Start Windows node security setup .

   - Impersonate user . The management server attempts to deploy the agent using the credentials that you are currently logged in to Windows with.

     You cannot use impersonation for nodes in untrusted domains or workgroups.

     You can use impersonation only if the PMAD user is trusted for delegation in Active Directory, unless your console runs directly on the management server. For more details on delegation, refer to the Active Directory documentation that Microsoft provides.

   - User/Password . The Credentials dialog box appears. Type the Username of a user who has permission to install software on the node and their Password . Click OK . This automatically selects the Deploy check box.

     For nodes with a UNIX or Linux operating system, this is the only command available . You can also use this command to install the HTTPS agent to Windows nodes that belong to the same domain as the management server, a trusted domain, or a workgroup.
6. *Optional.* Select or clear the Run prerequisites check automatically during job execution check box. If you are sure that the nodes meet all prerequisites, you can clear this check box so that the agent installation jobs complete more quickly. (This check box is dimmed if an administrator disables all prerequisite checking using the Server Configuration dialog.)
7. Click OK . The management server creates all the necessary deployment jobs.
8. While the change agent deployment job is in progress, the node requests a certificate. Policy and package deployment jobs start for each node only after this request is granted. If you did not configure

automatic granting of certificates, you must grant the certificates manually. Otherwise, the policy and package deployment jobs fail.

Related Topics:

- ovconfchg
- Configuring Certificates
- Generic server configuration
- Restart job with new options

Migrate from HTTPS to DCE

Where possible you should deploy the HTTPS agent on your managed nodes. Nevertheless, you can migrate from the HTTPS agent to the DCE agent if necessary.

When you migrate from the HTTPS agent to the DCE agent, the management server removes all existing policies, instrumentation and packages from that node. After it deploys the DCE agent packages, the management server then redeploys the policies, and the packages and instrumentation that the policies require. However, if any of the policies require the HTTPS agent, the deployment jobs for those policies will fail.

It is not possible to deploy the DCE agent to nodes with a UNIX or Linux operating system using the console. You must install the DCE agent manually on these nodes.

CAUTION:
When you migrate from an HTTPS agent to a DCE agent, you lose all data that the DCE agent stores locally, including performance data and custom configuration settings. If appropriate, back up this data and reconfigure it after you migrate to the DCE agent.

NOTE:
The HTTPS agent enables multiple management servers to configure the same node. This is not possible with the DCE agent. If other management servers are currently managing the node, they do not receive a notification when you migrate to the DCE agent. If deployment jobs exist on other management servers for this node, they will fail. Ensure that the node is removed from other management servers before or after migrating to the DCE agent. You can then add the node again with the communication type set to DCE.

To migrate from an HTTPS agent to a DCE agent

1. In the console tree, right-click Nodes , and then click Configure→ Nodes... . The Configure Managed Nodes dialog appears.
2. Right-click the node and then click Change agent . The Change Agent dialog appears.
3. Click Comm Type and then select DCE .
4. *Optional.* Specify the credentials that the management server uses to deploy the agent. Right-click a node in the list, and then click Set Credentials . Click one of the following commands:
   - PMAD user . The management server attempts to deploy the agent as the user under which the policy management and deployment (PMAD) service runs (called HP-OVE-Deleg-User by default).

     The node can belong to the same domain as the management server, a trusted domain, or a workgroup.

NOTE:
The PMAD user does not by default have administrative access to the node. For more information, see Start Windows node security setup .

- Impersonate user . The management server attempts to deploy the agent using the credentials that you are currently logged in to Windows with.

  You cannot use impersonation for nodes in untrusted domains or workgroups.

  You can use impersonation only if the PMAD user is trusted for delegation in Active Directory, unless your console runs directly on the management server. For more details on delegation, refer to the Active Directory documentation that Microsoft provides.

5. *Optional.* Select or clear the Run prerequisites check automatically during job execution check box. If you are sure that the nodes meet all prerequisites, you can clear this check box so that the agent installation jobs complete more quickly. (This check box is dimmed if an administrator disables all prerequisite checking using the Server Configuration dialog.)
6. Click OK . The management server creates all the necessary deployment jobs.

Related Topics:

- Manual installation of DCE agents
- Generic server configuration
- Restart job with new options

Migrate from 32 bit to 64 bit

Agent packages in 64 bit format are available, which you can deploy to nodes that have 64 bit architectures and operating systems. You can also deploy agent packages in 32 bit format to some 64 bit nodes. For more information, see the support matrix at HP Software Support Online.

If you have 32 bit agent packages installed on nodes that have 64 bit architectures and operating systems, you can migrate to the 64 bit packages. When you do this, the management server removes all existing policies, instrumentation and packages from that node. After it deploys the 64 bit agent packages, it then redeploys the policies, and the packages and instrumentation that the policies require.

To migrate from a 32 bit agent to a 64 bit agent

1. In the console tree, right-click Nodes , and then click Configure→ Nodes... . The Configure Managed Nodes dialog appears.
2. Right-click the node and then click Change agent . The Change Agent dialog appears.
3. Click Binary Format and then select an option. Agent packages in 64 bit format may not be available for all platforms.
4. *Optional.* Select the Auto Grant check box if you want the management server to automatically grant the certificate for this node. (This only happens if you also configure the management server to grant certificate requests automatically.)
5. *Optional.* Specify the credentials that the management server uses to deploy the agent. Right-click a node in the list, and then click Set Credentials . Click one of the following commands:
   - PMAD user . The management server attempts to deploy the agent as the user under which the policy management and deployment (PMAD) service runs (called HP-OVE-Deleg-User by default).

     You can only use this for nodes with a Windows operating system. The nodes can belong to the same domain as the management server, a trusted domain, or a workgroup.

     NOTE:
     The PMAD user does not by default have administrative access to the node. For more information, see Start Windows node security setup .

   - Impersonate user . The management server attempts to deploy the agent using the credentials that you are currently logged in to Windows with.

     You cannot use impersonation for nodes in untrusted domains or workgroups.

     You can use impersonation only if the PMAD user is trusted for delegation in Active Directory, unless your console runs directly on the management server. For more details on delegation, refer to the

Active Directory documentation that Microsoft provides.

- User / Password . The Credentials dialog box appears. Type the Username of a user who has permission to install software on the node and their Password . Click OK . This automatically selects the Deploy check box.

  For nodes with a UNIX or Linux operating system, this is the only command available . You can also use this command to install the HTTPS agent to Windows nodes that belong to the same domain as the management server, a trusted domain, or a workgroup.

6. *Optional.* Select or clear the Run prerequisites check automatically during job execution check box. You can clear this check box to save time if you are sure that the nodes meet all system requirements. (This check box is dimmed if an administrator disables all prerequisite checking using the Server Configuration dialog.)
7. Click OK . The management server creates all the necessary deployment jobs.

Related Topics:

- Configuring Certificates
- Generic server configuration
- Restart job with new options

Manual agent deinstallation

If you no longer need to manage a node using HPOM, you can deinstall the agent from it. You can easily do this by removing the Operations-agent package from the node using the console. This also removes any policies from the node.

You can also deinstall the agent manually if necessary, for example if the node no longer exists in the console. The procedure for manual agent installation depends on whether the node has the HTTPS agent or the DCE agent. If the node has the DCE agent, the procedure depends on whether the node has a Windows operating system, or a UNIX or Linux operating system.

Related Topics:

- Remove package from node
- Deinstall an HTTPS agent manually
- Deinstall a DCE agent from a Windows node
- Deinstall a DCE agent from a UNIX or Linux node

Deinstall an HTTPS agent manually

You can deinstall an HTTPS agent by removing the Operations-agent package from node in the console. However, you can also deinstall the agent manually if necessary.

To deinstall an HTTPS agent manually

1. Log in to the node as a user with administrative rights.
2. Open a command prompt and use ovc to stop the agent as follows:
   - On Windows operating systems:
     `ovc -stop AGENT`
   - On UNIX and Linux operating systems:
     `/opt/OV/bin/ovc -stop AGENT`
3. Use `opc_inst` to start the deinstallation as follows:
   - On Windows operating systems:
     `cscript <`*`install_dir`*` >\bin\OpC\install\opc_inst.vbs -r`
   - On UNIX and Linux operating systems:
     `/opt/OV/bin/OpC/install/opc_inst -r`
4. *Optional.* Check the following log file for deinstallation errors:
   - On Windows operating systems:
     <*data_dir* >`\log\opc_inst.log`
   - On UNIX and Linux operating systems:
     `/var/opt/OV/log/opc_inst.log`
5. *Optional.* The management server does not receive notification when you deinstall the agent manually. Therefore, if the node still exists on the management server, the package and policy inventory is incorrect.

   You can delete the node from the management server using the Configure Managed Nodes dialog. Alternatively, to clear the inventory, you can force the management server to remove the agent package in the Uninstall package from... dialog.

NOTE:
The HTTPS agent includes several components, which appear in Add or Remove Programs on nodes that run a Windows operating system. Dependencies between these components make it difficult to remove the components using Add or Remove Programs. HP recommends that you remove the HTTPS agent

using `opc_inst.vbs` , as the above procedure describes.

Related Topics:

- Delete, copy, and move managed nodes
- Remove package from node
- Manual agent installation

Deinstall a DCE agent from a Windows node

To manually deinstall an existing DCE agent on a Windows computer:

1. On the node, remove the agent using Add/Remove Programs .
2. Remove the PMAD user (which has the name HP-OVE-Deleg-User by default) from the node's local administrators group.

   NOTE:
   If the agent was originally installed from an earlier version of the management server, the PMAD user may not be part of the node's local administrator group. You may need to remove a group account (which has the name HP-OVE-GROUP by default) instead.

3. *Optional.* The management server does not receive notification when you deinstall the agent manually. Therefore, if the node still exists on the management server, the package and policy inventory is incorrect.

   You can delete the node from the management server using the Configure Managed Nodes dialog. Alternatively, to clear the inventory, you can force the management server to remove the agent package in the Uninstall package from... dialog.

Related Topics:

- Delete, copy, and move managed nodes
- Remove package from node
- Manual agent installation
- Deinstall a DCE agent from a UNIX or Linux node

Deinstall a DCE agent from a UNIX or Linux node

To deinstall an existing DCE agent from a UNIX or Linux node :

1. Run the `opcagt -kill` command.
2. Run the appropriate command to deinstall the agent:
   - Linux operating system:

     To find out which version of the agent is installed, check the value of the parameter OPC_INSTALLED_VERSION in the `/opt/OV/bin/OpC/install/opcinfo` file on the node.

     If you are removing a version of the agent earlier than 7.0, use the command:
     `rpm -e OPC dce`

     If you are removing the agent A.07.10 on a Linux operating system, deinstall in two steps:

     i. `rpm -e OPSCVCDISC OPCPERF OPCJRE OPC OPCCOMM`

     ii. `rpm -e dce`

     Otherwise, use the command:
     `rpm -e OPCSVCDISC OPCPERF OPCJRE OPC OPCCOMM dce`
   - AIX operating system: `installp -ug OPC OPCCOMM OPCPERF OPCSVCDISC`
   - HP-UX operating system: `swremove ITOAgent`
   - Solaris operating system: `/usr/sbin/pkgrm OPC OPCCOMM OPCPERF OPCSVCDIS HPlwdce`
   - Tru64 operating system: `setld -d OPCAGT000 OPCCOMMAGT000 OPCPERFAGT000 OPCSVCDISCAGT000`
3. *Optional.* The management server does not receive notification when you deinstall the agent manually. Therefore, if the node still exists on the management server, the package and policy inventory is incorrect.

   You can delete the node from the management server using the Configure Managed Nodes dialog. Alternatively, to clear the inventory, you can force the management server to remove the agent package in the Uninstall package from... dialog.

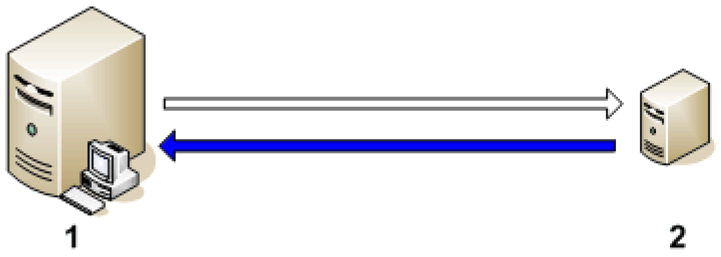
Related Topics:

- Delete, copy, and move managed nodes
- Remove package from node
- Manual agent installation
- Deinstall a DCE agent from a Windows node

Configuring HTTPS communication through firewalls

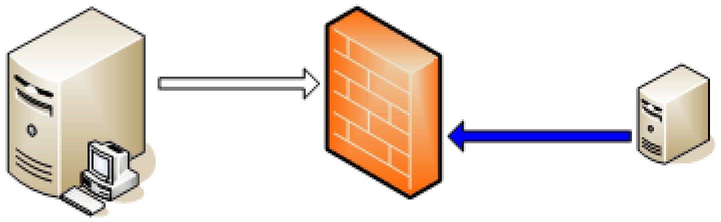
Management servers and nodes communicate with each other over the network. For nodes that have the HTTPS agent, this communication uses the HTTPS protocol. The figure below shows the network connections between management servers and nodes as follows:

- The management server (1) opens connections to nodes (2) , for example to deploy policies and instrumentation, for heartbeat polling, or to launch actions.
- Nodes (2) open connections to the management server (1) , for example to send messages, and action responses.



When a management server or node opens a new connection, the operating system allocates the local port for the connection. On the other side of the connection, management servers and nodes both have communication brokers, which listen on port 383 for incoming connections. So by default, all connections have a local port assigned by the operating system and the destination port is 383.

If you have management servers and nodes on different networks that are separated by a firewall, the firewall may block connections between them, as the figure below shows. This prevents you from managing the nodes, because, for example, management servers cannot deploy policies and nodes cannot send messages.



If a firewall blocks HTTPS connections, you can reconfigure communication between management servers and nodes in several ways. The HPOM configuration you choose to implement depends mainly on the configuration of your network.

- If your network allows HTTPS connections through the firewall in both directions, but with certain restrictions, the following configuration options are possible in HPOM to accommodate these restrictions:
  - If your network allows only certain proxy systems to open connections through the firewall, you can

    redirect HPOM communication through these proxies.

    - If your network allows inbound connections to only certain destination ports, but not to port 383, you can configure alternate communication broker ports.
    - If your network allows outbound connections from only certain local ports, you can configure HPOM to use specific local ports.

- If your network allows only outbound HTTPS connections from the management server across the firewall, and blocks inbound connections from nodes, you can configure a reverse channel proxy.

NOTE:
In an environment with multiple management servers, you can also configure the management servers to communicate with each other through firewalls. The configuration is the same as for communication between management servers and nodes.

Related Topics:

- Configuring two-way communication
- Configuring outbound-only communication
- Server-based flexible management

Configuring two-way communication

If your network allows HTTPS connections through the firewall in both directions, but with certain restrictions, the following configuration options are possible in HPOM to accommodate these restrictions:

- If your network allows only certain proxy systems to open connections through the firewall, you can redirect HPOM communication through these proxies.
- If your network allows inbound connections to only certain destination ports, but not to port 383, you can configure alternate communication broker ports.
- If your network allows outbound connections from only certain local ports, you can configure HPOM to use specific local ports

Related Topics:

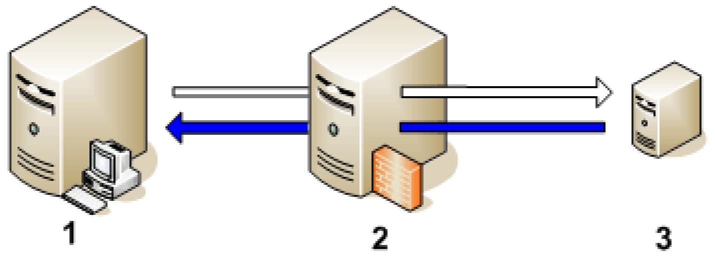
- Redirect HTTPS communication through proxies
- Configure communication broker ports
- Configure local communication ports

Redirect HTTPS communication through proxies

Overview

You can redirect connections from management servers and nodes that are on different networks through a proxy. The figure below shows connections between a management server and node through a proxy as follows:

- The management server (1) opens connections to the proxy (2) , for example to deploy policies and instrumentation, for heartbeat polling, or to launch actions. The proxy opens connections to the node (3) on behalf of the management server, and forwards communication between them.
- The node (3) opens connections to the proxy (2) , for example to send messages, and action responses. The proxy opens connections to the management server (1) on behalf of the node.

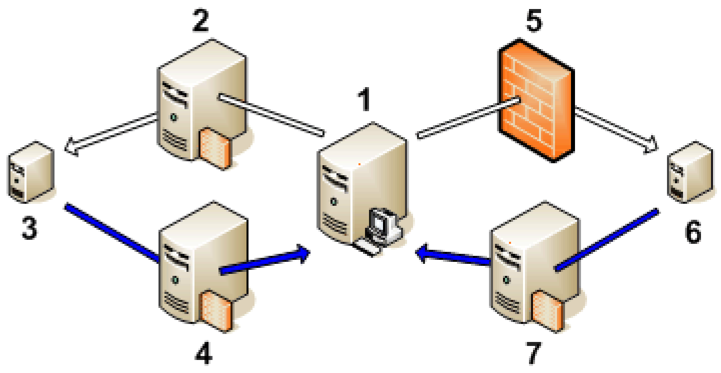


You can also redirect communication through proxies in more complex environments as follows:

- Each management server and node can use different a proxy server to communicate with each other.
- You can configure management servers and nodes to select the correct proxy according to the host they need to connect to.

The figure below shows connections between a management server and nodes through multiple proxies as follows:

- The management server (1) opens connections to a proxy (2) . The proxy opens connections to the node (3) on behalf of the management server.
- The node (3) opens connections to a different proxy (4) . The proxy opens connections to the management server (1) on behalf of the node.
- The network allows management server (1) to make outbound HTTP connections directly through the firewall (5) to another node (6) . (The nodes (3, 6) are on different networks.)
- The firewall (5) does not allow inbound HTTP connections. Therefore, node (6) opens connections to the management server through a proxy (7) .



PROXY parameter syntax

You redirect outbound HTTPS communication through proxies by setting the PROXY parameter in the **bbc.http** name space on the management servers and nodes. You can configure this parameter in the following ways:

- Configure the values in the HTTPS agent installation defaults. This is recommended if you need to configure proxies for large numbers of nodes. You must plan and configure the installation defaults before you create or migrate your nodes.
- Use ovconfchg or ovconfpar at a command prompt.

The value of the PROXY parameter can contain one or more proxy definitions. Specify each proxy in the following format:

<*proxy_hostname* **>:<***proxy_port* **>+(<***included hosts* **>)-(<***excluded hosts* **>)**

Replace <*included_ hosts* > with a comma-separated list of hostnames or IP addresses to which the proxy enables communication. Replace <*excluded hosts* > with a comma-separated list of hostnames or IP addresses to which the proxy cannot connect. Asterisks (*) are wild cards in hostnames and IP addresses. Both <*included_ hosts* > and <*excluded hosts* > are optional.

To specify multiple proxies, separate each proxy with a semicolon (;). The first suitable proxy in the list takes precedence.

Example PROXY parameter values

To configure a node to use `proxy1.example.com` port `8080` for all outbound connections, you would use the following value:

```
proxy1.example.com:8080
```

To configure a management server to use `proxy2.example.com:8080` to connect to any host with a hostname that matches `*.example.com` or `*example.org` except hosts with an IP address in the range

192.168.0.0 to 192.168.255.255, you would use the following value:

```
proxy2.example.com:8080+(*.example.com,*.example.org)-(192.168.*.*)
```

To extend the above example to use `proxy3.example.com` to connect to `backup.example.com` only, you would use the following value:

```
proxy3.example.com:8080+(backup.example.com);
proxy2.example.com:8080+(*.example.com,*.example.org)-(192.168.*.*)
```

In the above example, `proxy3.example.com:8080+(backup.example.com)` must be first, because the include list for `proxy2.example.com` contains `*.example.com`.

To redirect HTTPS communication through proxies using **ovconfchg**

1. Log in to the management server or node as a user with administrative rights and open a command prompt or shell.
2. On nodes that run a UNIX or Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type **`export PATH=/opt/OV/bin:$PATH`** and then press Enter.
   - On AIX, type **`export PATH=/usr/lpp/OV/bin:$PATH`** and then press Enter.
   - On Tru64, type **`export PATH=/usr/opt/OV/bin:$PATH`** and then press Enter.
3. Specify the proxies that the node should use. You can specify different proxies to use depending on the host that the agent wants to connect to. Type the following command:

   `ovconfchg -ns bbc.http -set PROXY <`*`proxy`*`>`

NOTE:
When you use the command ovconfchg on a management server that runs in a cluster, add the parameter -ovrg server.

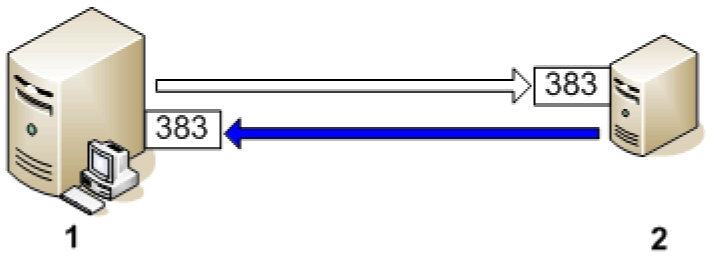
Related Topics:

- Configure HTTPS agent installation defaults
- ovconfchg
- ovconfpar

Configure communication broker ports

Overview

Management servers and nodes that have HTTPS agents both include communication brokers that listen for inbound connections on port 383, as the figure below shows. The communication broker on a management server (1) handles all inbound connections from nodes that have HTTPS agents, and also from other management servers. The communication broker on a node that has the HTTPS (2) agent handles all inbound connections from management servers.



You can configure any communication broker to listen on a port other than 383. If you do this, you must also configure the other management servers and nodes in the environment, so that their outbound connections are destined for the correct port. For example, if you configure a node's communication broker to listen on port 5000, you must also configure the management server so that it connects to port 5000 when it communicates with this node.

PORTS parameter syntax

You configure communication broker ports by setting the PORTS parameter in the **`bbc.cb.ports`** name space on all management servers and nodes that communicate with each other. You can configure this parameter in the following ways:

- Configure the values in the HTTPS agent installation defaults. This is recommended if you need to configure communication broker ports for large numbers of nodes. You must plan and configure the installation defaults before you create or migrate your nodes.
- Use `ovconfchg` or `ovconfpar` at a command prompt.

The values must contain one or more host names or IP addresses and have the following format:

`<`*host*`>:<`*port*`>[,<`*host*`>:<`*port*`>] ...`

The <*host*> can be either a domain name or IP address. For example, to configure the communication broker port to **`5000`** on a management server with the host name `manager1.emea.example.com`, use the following command on the management server itself, and also any other management servers and nodes that open connections to it:

ovconfchg -ns bbc.cb.ports -set PORTS manager1.emea.example.com:5000

If you need to configure communication broker ports on multiple systems, you can use wildcards and ranges, as follows:

- You use a wildcard at the start of a domain name by adding an asterisk (*). For example:
  - `*.emea.example.com:5000`
  - `*.test.com:5001`
  - `*:5002`
- You can use wildcards at the end of an IP address by adding up to three asterisks (*). For example:
  - `192.168.1.*:5003`
  - `192.168.*.*:5004`
  - `10.*.*.*:5005`
- You can replace one octet in an IP address with a range. The range must be before any wildcards. For example:
  - `192.168.1.0-127:5006`
  - `172.16-31.*.*:5007`

If you specify multiple values for the PORTS parameter, separate each with a comma (,). For example:

ovconfchg -ns bbc.cb.ports -set PORTS *.emea.example.com:5000,10.*.*.*:5005

When you specify multiple values using wildcards and ranges that overlap, the management server or node selects the port to use in the following order:

- Fully qualified domain names.
- Domain names with wildcards.
- Complete IP addresses.
- IP addresses with ranges.
- IP addresses with wildcards.

For example, if you configure communication broker ports on all management servers and nodes with the following command:

ovconfchg -ns bbc.cb.ports -set PORTS *.emea.example.com:6000,10.*.*.*:6001,manager1.emea.example.com:6002,10.0-127.*.*:6003

the following ports are used:

- Host name: **`node1.asia.example.com`**
  IP address: **`10.127.1.1`**
  Communication broker port: **`6003`** .
- Host name: **`manager1.emea.example.com`**
  IP address: **`10.1.1.1`**
  Communication broker port: **`6002`** .
- Host name: **`node1.test.com`**
  IP address: **`192.168.1.1`**
  Communication broker port: **`383`** .

To find out which port is currently configured, type the following command:

```
bbcutil -getcbport <host >
```

TIP:
To organize settings for many communication broker ports, you can add parameters of any name in the **`bbc.cb.ports`** name space. The value of any parameter in the name space is evaluated. The PORTS parameter is optional.

To configure communication broker ports using **ovconfchg**

1. Open a command prompt on the management server or node.
2. On nodes that run a UNIX or Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type **`export PATH=/opt/OV/bin:$PATH`** and then press Enter .
   - On AIX, type **`export PATH=/usr/lpp/OV/bin:$PATH`** and then press Enter .
   - On Tru64, type **`export PATH=/usr/opt/OV/bin:$PATH`** and then press Enter .
3. Specify the communication broker ports by typing the following command:

   ```
   ovconfchg -ns bbc.cb.ports -set PORTS <host >:<port >[,<host >:<port >] ...
   ```

NOTE:
When you use the command ovconfchg on a management server that runs in a cluster, add the parameter -ovrg server

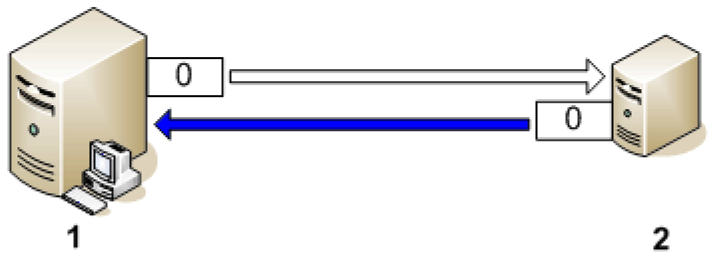
Related Topics:

- Configure HTTPS agent installation defaults
- ovconfchg
- ovconfpar

Configure local communication ports

Overview

Management servers open outbound connections to nodes that have HTTPS agents, and nodes that have the HTTPS agent open outbound connections to management servers, as the figure below shows. A management server (1) opens these connections, for example, to deploy policies and instrumentation, for heartbeat polling, and to launch actions. An HTTPS agent (2) opens these connections, for example, to send messages and action responses.



By default, management servers and nodes use local port 0 for outbound connections, which means that the operating system allocates the local port for each connection. Typically, the operating system will allocate local ports sequentially. For example if the operating system allocated local port 5055 to an Internet browser, and then the HTTPS agent opens a connection, the HTTPS agent receives local port 5056.

However, if a firewall restricts the ports that you can use, you can configure management servers and nodes to use a specific range of local ports instead.

CLIENT_PORT parameter syntax

You configure local communication ports by setting the CLIENT_PORT parameter in the **`bbc.http`** name space on the management server or node. You can configure this parameter in the following ways:

- Configure the values in the HTTPS agent installation defaults. This is recommended if you need to configure local communication ports for large numbers of nodes. You must plan and configure the installation defaults before you create or migrate your nodes.
- Use `ovconfchg` or `ovconfpar` at a command prompt.

The value must be a range of ports in the following format:

<lower port number >-<higher port number >

For example, if the firewall only allows outbound connections that originate from ports 5000 to 6000 you would use the following value:

```
5000-6000
```

To configure local communication ports using **ovconfchg**

1. Open a command prompt on the management server or node.
2. On nodes that run a UNIX or Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type **`export PATH=/opt/OV/bin:$PATH`** and then press Enter .
   - On AIX, type **`export PATH=/usr/lpp/OV/bin:$PATH`** and then press Enter .
   - On Tru64, type **`export PATH=/usr/opt/OV/bin:$PATH`** and then press Enter .
3. Specify the range of local ports that the management server or node can use for outbound connections by typing the following command:

   ovconfchg -ns bbc.http -set CLIENT_PORT <*lower port number* **>-<***higher port number* **>**

NOTE:
When you use the command ovconfchg on a management server that runs in a cluster, add the parameter -ovrg server .

Related Topics:

- Configure HTTPS agent installation defaults
- ovconfchg
- ovconfpar

Configure multihomed nodes

By default, when a node has several IP addresses, the agent uses them as follows:

- The communication broker accepts incoming connections on all IP addresses.
- The agent opens connections to the management server using the first network interface that it finds.
- The embedded performance component accepts incoming connections on all IP addresses.

You can configure the HTTPS agent to always use a specific IP address. You do this by configuring the parameters in the following table using `ovconfchg` or `ovconfpar` at a command prompt:

Namespace	Parameter	Value
bbc.cb	SERVER_BIND_ADDR	The IP address that you want the communication broker to listen on for incoming connections.
bbc.http	CLIENT_BIND_ADDR	The IP address that you want the agent use on the node end of connections to the management server.
coda.comm	SERVER_BIND_ADDR	The IP address that you want the embedded performance component to listen on for incoming connections. This can either be the same IP address as the SERVER_BIND_ADDR parameter in the bbc.cb namespace, or you can set this value to localhost. If you set the value to localhost, the embedded performance component to accepts incoming connections only through the communication broker.

You can configure the DCE agent to always use a specific IP address by setting the OPC_IP_ADDRESS parameter in a node info policy, which you then deploy to the node.

Related Topics:

- ovconfchg
- ovconfpar
- Node Info Policy Type

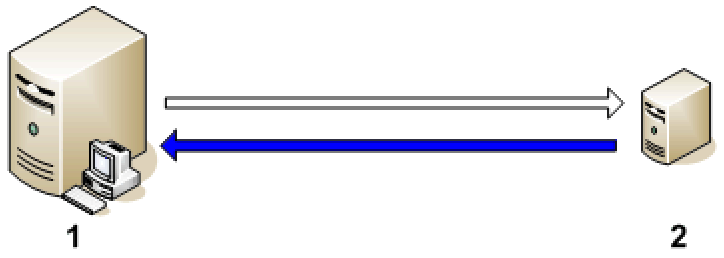
Configuring outbound-only communication

Overview

Management servers and nodes communicate with each other over the network. Normally, management servers open outbound network connections to nodes and nodes open inbound network connections to management servers.

The figure below shows the network connections where there is no firewall that blocks inbound HTTPS connections to the management server as follows:

- The management server (1) opens outbound connections to agents, for example to deploy policies and instrumentation, for heartbeat polling, or to launch actions.
- Agents (2) open inbound connections to the management server, for example to send messages, actions responses, or to launch remote actions.



If a firewall blocks inbound HTTPS connections from a node to a management server, the node cannot communicate with the management server properly. To enable proper communication, you configure an HTTPS agent to act as a reverse channel proxy (RCP).

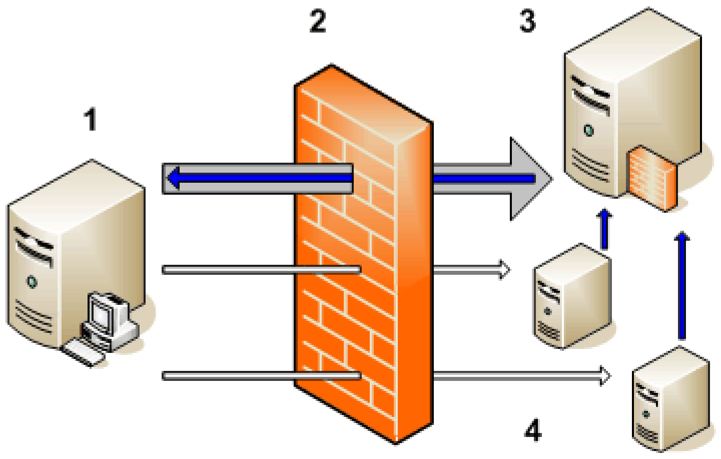
An RCP handles communication between management servers and nodes, so that they do not need to communicate with each other directly. An RCP can run on the managed node that it serves, or on a separate system that serves multiple managed nodes. The RCP is on the same side of the firewall as the node or nodes that it serves.

Outbound-only communication through one firewall

The figure below shows the network connections where there is a firewall that blocks inbound HTTPS connections to the management server as follows:

- The management server (1) makes an outbound connection through the firewall (2) to an RCP (3) . This connection is called a reverse administration channel. The management server maintains the reverse administration channel, so that the RCP never needs to make an inbound connection to the management server.

- Agents (4) open connections to the RCP, instead of the management server. The RCP (3) forwards the agents' communications to the management server using the reverse administration channel.
- The management server (1) also makes outbound connections directly to agents (4) .



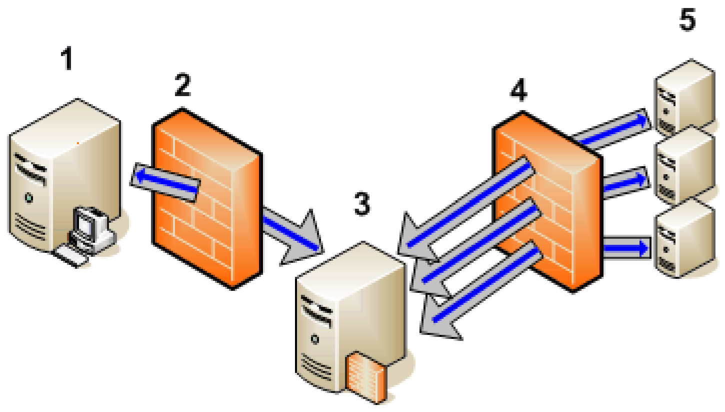
To configure outbound-only communication in this scenario, you must:

1. Configure the RCP, so that it listens for incoming connections.
2. Configure the management server, so that it opens the reverse administration channel to the RCP.
3. Configure the agents, so that they use the RCP for their outbound connections to the management server.

Outbound-only communication through two firewalls

The figure below shows the network connections where there are two firewalls. One firewall blocks inbound connections to the management server. The other firewall blocks inbound connections to the nodes.

- The management server (1) opens a reverse administration channel through the firewall (2) to the RCP (3) . The management server maintains the reverse administration channel, so that the RCP never needs to make an inbound connection to the management server.
- Each agent (5) opens a reverse administration channel through the firewall (4) to the RCP (3) . The agents maintain these connections, so that the RCP never needs to make inbound connections to the agents.
- The management server (1) and agents (5) open outbound connections to the RCP, instead of directly to each other. The RCP (3) forwards the these communications to the using the reverse administration channel.



To configure outbound-only communication in this scenario, you must:

1. Configure the RCP, so that it listens for incoming connections.
2. Configure the management server, so that it opens a reverse administration channel to the RCP.
3. Configure the management server, so that it uses the RCP as a proxy for its outbound connections to agents.
4. Configure the agents, so that they each open a reverse administration channel to the RCP.
5. Configure the agents, so that they use the RCP for their outbound connections to the management server.

Related Topics:

- Configure a reverse channel proxy
- Configure reverse administration channels
- Forward outbound connections through a reverse channel proxy

Configure a reverse channel proxy

Before you can configure a system as a reverse channel proxy (RCP), you must install the HTTPS agent software and add the node to the console. You can deploy the HTTPS agent software automatically from the console, or install it manually. You must also configure the node's certificates.

To configure a reverse channel proxy

1. Log in to the node as a user with administrative rights and open a command prompt or shell.
2. On nodes that run a UNIX or Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type **`export PATH=/opt/OV/bin:$PATH`** and then press Enter .
   - On AIX, type **`export PATH=/usr/lpp/OV/bin:$PATH`** and then press Enter .
   - On Tru64, type **`export PATH=/usr/opt/OV/bin:$PATH`** and then press Enter .
3. Set the port that agents and management servers can connect to. Type following command:

   ovconfchg -ns bbc.rcp -set SERVER_PORT <*port_number* **>**

   NOTE:
   Ensure that the port number you specify is not already in use by any other software on the system.

4. Register the RCP component so that ovc starts, stops and monitors it. Type the following commands:
   a. **ovcreg -add <***install_dir* **>/newconfig/DataDir/conf/bbc/ovbbcrcp.xml**
   b. **ovc -kill**
   c. **ovc -start**

NOTE:
After you configure the management server to establish a connection with this RCP you can check that the connection exists, by typing the following command:

ovbbcrcp -status
The command shows details of the reverse channel connection.

Related Topics:

- Remote agent installation
- Manual agent installation

- Configuring Certificates
- ovconfchg
- ovbbcrcp

Configure reverse administration channels

NOTE:
When you use the command ovconfchg in the following procedure, add the parameter `-ovrg server` only if your management server runs in a cluster.

To configure a reverse administration channel

1. Log in to the management server or agent as a user with administrative rights and open a command prompt.

2. Enable outbound-only communication. By default, this is disabled. To change this, type the following command:

   ```
   ovconfchg [-ovrg server] -ns bbc.cb -set ENABLE_REVERSE_ADMIN_CHANNELS true
   ```

3. Specify the reverse channel proxies (RCPs) that to which you want to open reverse administration channels. You must specify RCPs in the following format:

   <*host* >:<*port* >[,<*OvCoreID* >]

   For example, if a management server must connect to port 50000 on `rcp1.example.com` you specify the RCP with:

   ```
   rcp1.example.com:50000,9fcc7062-0472-751c-1236-84372bec342d
   ```

   If you specify the optional OvCoreID, the server checks that the RCP has that OvCoreID. You can specify the RCPs at the command prompt or in a file:

   - To specify the RCPs at the command prompt, type the following command:

     ```
     ovconfchg [-ovrg server] -ns bbc.cb -set RC_CHANNELS <rcp >[;<rcp >]
     ```

     Separate each RCP with a semicolon (;).

   - To specify the RCPs in a file:

     i. Create a text file that specifies each RCP on a separate line

     ii. *Optional.* Add comments on lines that begin with the number sign (#).

     iii. Save the file in the folder <*data_dir* >`\conf\bbc` .

     iv. Type the following command:

ovconfchg [-ovrg server] -ns bbc.cb -set RC_CHANNELS_CFG_FILES <*file name* >

You must also type the same command if you later change the contents of the file. The server reads the file only after you use `ovconfchg` .

4. *Optional.* Configure whether the server should automatically retry failed reverse administration channel connections. By default, the server does not retry failed connections. To change the default, type the following command:

   ovconfchg [-ovrg server] -ns bbc.cb -set RETRY_RC_FAILED_CONNECTION TRUE

5. *Optional.* Set the maximum number of attempts that the server should make to reconnect to a failed reverse administration channel connection. By default, this is set to -1 (infinite). To change the default, type the following command:

   ovconfchg [-ovrg server] -ns bbc.cb -set MAX_RECONNECT_TRIES <*number of tries* >

6. *Optional.* Configure the management server to generate a warning message about failed reverse administration channel connections. By default, the management server does not generate this message. To change the default, type the following command:

   ovconfchg [-ovrg server] -ns bbc.cb -set RC_ENABLE_FAILED_OVEVENT TRUE

   However, if you set `RETRY_RC_FAILED_CONNECTION` to `TRUE` , the management server attempts to reconnect the failed connection without generating the message.

7. *Optional.* Configure the number of minimum and maximum number of worker threads for connections to RCPs. The communication broker can use multiple worker threads to enhance the performance of connections to RCPs.

   By default, the maximum number of worker threads is 1 and the minimum number of worker threads is 0. If the system has sufficient resources, you can increase the number of worker threads. To change the defaults, type the following commands:

   ovconfchg [-ovrg server] -ns bbc.cb -set RC_MAX_WORKER_THREADS <*number* >

   ovconfchg [-ovrg server] -ns bbc.cb -set RC_MIN_WORKER_THREADS <*number* >

8. *Optional.* To check that the reverse administration channel is open, type the following command:

   ovbbccb -status

   The output lists all open reverse administration channels. If any of the reverse administration channel connections has the state `FAILED` , you can attempt to restore the connections by typing the following command:

   ovbbccb -retryfailedrcp [-ovrg server]

NOTE:

The following parameters and command are only available after you deploy agent version 8.60 or higher to the management server:

- `RETRY_RC_FAILED_CONNECTION`
- `RC_ENABLE_FAILED_OVEVENT`
- `RC_MAX_WORKER_THREADS`
- `RC_MIN_WORKER_THREADS`
- `ovbbccb -retryfailedrcp`

Related Topics:

- ovconfchg
- ovbbccb

Forward outbound connections through a reverse channel proxy

To forward outbound connections through a reverse channel proxy

1. Log in to the node or management server as a user with administrative rights and open a command prompt or shell.
2. On nodes that run a UNIX or Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type **`export PATH=/opt/OV/bin:$PATH`** and then press Enter .
   - On AIX, type **`export PATH=/usr/lpp/OV/bin:$PATH`** and then press Enter .
   - On Tru64, type **`export PATH=/usr/opt/OV/bin:$PATH`** and then press Enter .
3. Specify the RCP to use for outbound connections. You can specify different RCPs to use depending on the destination host. Type the following command:

   ovconfchg -ns bbc.http -set PROXY <*rcp* >[;<*rcp* >]

   Separate each RCP with a semicolon. Specify each <*rcp* > in the following format:

   <*rcp_hostname* >:<*rcp_port* >+(<*included hosts* >)-(<*excluded hosts* >)

   Replace <*included_ hosts* > with a comma-separated list of valid destination hostnames or IP addresses for the RCP. Replace <*excluded hosts* > with a comma-separated list of hostnames or IP addresses that system should not use the RCP to connect to. Asterisks (*) are wild cards in hostnames and IP addresses.

   NOTE:
   <*excluded_hosts* > must always contain the hostname and the fully qualified domain name of the RCP.

   For example, to configure an agent to use `rcp1.example.com:50000` to connect to any host with a hostname that matches `*.example.com` or `*example.org` except hosts with an IP address in the range 192.168.0.0 to 192.168.255.255, you would type the following command:

   ```
   ovconfchg -ns bbc.http -set PROXY rcp1.example.com:50000+(*.example.com,*.example.org)-
   (192.168.*.*,rcp1.example.com,rcp1)
   ```

4. *Optional.* For an agent, specify the OvCoreID of the management server that the RCP should connect this agent to. This is useful if the RCP cannot resolve the hostnames of management servers because of firewalls. When an agent attempts to open a connection to a management server, the RCP can use the OvCoreID instead of the hostname to select the correct reverse administration channel. You can either specify the management server's OvCoreID directly, or specify a command that returns the

OvCoreID.

- To specify the management server's OvCoreID directly, type the following command:

  ovconfchg -ns bbc.http -set TARGET_FOR_RC <*management server OvCoreID* **>**

- To specify a command that returns the management server's OvCoreID, type the following command:

  ovconfchg -ns bbc.http -set TARGET_FOR_RC_CMD <*command* **>**

5. To restart the message agent, type **ovc -restart opcmsga** and then press Enter .

Agent users

By default, on nodes with a Windows operating system, both HTTPS and DCE agents run under the Local System account, which is built-in to Windows. On nodes with a UNIX or Linux operating system, the HTTPS agent runs under the root account. If necessary, you can configure HTTPS agents to run under a different user account after you install the agent. For DCE agents on Windows nodes, you can change the user in the installation defaults on the management server before you install the agent.

The agent starts automatic or operator-initiated commands under the user account that the agent itself is currently running under. However, you can configure an HTTPS agent to start commands under a different user account.

Related Topics:

- Change the default user of DCE agents on Windows nodes
- Change the user of an HTTPS agent on Windows nodes
- Change the user of an HTTPS agent on UNIX and Linux nodes
- Change the default user for commands

Change the default user of DCE agents on Windows nodes

By default on managed nodes with a Windows operating system, the DCE agent runs under the Local System account. However, you can configure the DCE agent installation defaults so that the agent runs under a different user account. For example, you may want the agent to run under an account with fewer permissions to the Local System account. Alternatively, you may want the agent to run under a domain account that gives the agent permission to access remote systems.

You must test whether the user account has appropriate rights to run the agent and manage the node correctly. You assign these user rights in the local Windows security settings on the node, or a group policy object in Active Directory. The user rights that you assign depend on your requirements. You may, for example, consider assigning the following user rights:

- Access this computer from the network.

  Required for installation.
- Act as part of the operating system.

  Required to run actions as a user other than the agent user.
- Log on as a service.

  Required to run the agent as a service.
- Adjust memory quotas for a process (also called Increase quotas in some versions of Windows)

  Required for a full switch user with password to get network access when executing tools.
- Manage auditing and security log.

  Required during the execution of actions.
- Replace a process-level token.

  Required for the user switch in the action agent.
- Shut down the system.

  Required to shutdown the managed node.
- Additional rights for the management tasks that you need to perform. For example:
  - If you want be able to monitor a log file using a policy , the agent user must have permission to read that log file.
  - If you want to be able to start a program using an automatic command, the agent user must have permission to start that program.

NOTE:
This procedure changes only the installation defaults in the DCE agent package on the management server . To apply changes to nodes where the DCE agent is already installed, you must redeploy the agent. The redeployed agent runs under the new user account. Remove the old agent user account

manually if you no longer need it.

To change the default user of DCE agents

1. Log in to the management server with an account that is a member of the HPOM administrators group. Open a command prompt.
2. Type **`cd "%OvInstallDir%\bin\OpC\install"`** and then press Enter .
3. To encrypt the DCE agent user's password type **`opcpwcrpt <`***password* **`>`** and then press Enter . Copy the output.
4. Type **`SetMgmtServer /user <`***user* **`> /password <`***encrypted password* **`>`** and then press Enter .
   - Replace <*user* > with the name of the user, for example AgentUser . The name must not contain spaces.

     You can specify the name of a existing domain user, but do not specify the domain (for example, do not specify DOMAIN\account or account@domain). The domain user must belong to the same domain as the node, and no local user with the same name must exist on the node.

     If you specify a user that does not exist, the agent installation creates a local user with the specified name on each node. The new user is a member of the local Administrators group.
   - Replace <*encrypted password* > with the output from `opcpwcrpt` , which you copied.

     CAUTION:
     If the specified account already exists on a node, but the default password in the agent package does not match, the agent installation removes the existing account and recreates it with the same name but a different internal user ID.

Related Topics:

- Configure DCE agent installation defaults
- Change the user of an HTTPS agent on a Windows node

Change the user of an HTTPS agent on a Windows node

By default on nodes with a Windows operating system, the HTTPS agent runs under the built-in Local System account. However, you can configure the HTTPS agent to run under a different user account. For example, you may want the agent to run under an account with fewer permissions than the Local System account. Alternatively, you may want the agent to run under an account that has permission to access remote systems over the network.

You must test whether the user account has appropriate rights to run the agent and manage the node correctly. You assign these user rights in the local Windows security settings on the node, or a group policy object in Active Directory. The user rights that you assign depend on your requirements. The user account may, for example, need the following user rights:

- User rights to run the agent:
  - Log on as a service
  - Manage auditing and security log
- User rights to manage the node:
  - Shut down the system

    This allows the agent to shut down the system (for example, when a user starts the shutdown tool in the console).
  - Debug programs

    This allows the agent to collect information about processes, and to kill processes (for example, when a user starts the list processes or kill process tool in the console).
- User rights to allow the agent to start commands and tools as a user other than the agent user:
  - Act as part of the operating system.
  - Adjust memory quotas for a process (also called Increase quotas in some versions of Windows)
  - Replace a process-level token.
- Permissions for registry entries:
  - `HKEY_LOCAL_MACHINE/Software/Hewlett-Packard/OpenView`

    The user must have full control for this registry key and all child objects.
  - `HKEY_LOCAL_MACHINE/Software/Microsoft/WindowsNT/CurrentVersion/Perflib`

    The user must have permission to read this registry key for the agent to access performance data.

The following procedure assigns the above user rights to a user group that you specify. You may need to assign additional rights for the management tasks that you need to perform. For example:

- If you want be able to monitor a log file using a policy, the agent user must have permission to read that log file.
- If you want to be able to start a program using an automatic command, operator-initiated command, tool, or scheduled task, the agent user must have permission to start that program.

  Additionally, you must set the parameter OPC_PROC_ALWAYS_INTERACTIVE=NEVER in the eaagt namespace. You can configure this parameter in the HTTPS agent installation defaults or using ovconfchg or ovconfpar at a command prompt. After you set this parameter, processes that the agent starts do not have access to the default desktop. This setting applies to logfile encapsulator pre-processing and scripts that the monitor agent invokes.
- Some Smart Plug-ins may require additional configuration or user rights when the agent runs under a user account that does not have administrative rights. For more details, see the documentation for individual Smart Plug-ins.

To change the user of an HTTPS agent

1. *Optional.* Create a new user for the agent to run under.
2. *Optional.* Create a new group, and add the user as a member of this group.
3. On the node, open a command prompt, and type the following command:

   cscript "%OvInstallDir%\bin\ovswitchuser.vbs"-existinguser <*DOMAIN\USER* **> -existinggroup <***GROUP* **> -passwd <***PASSWORD* **>**

   - Replace <DOMAIN\USER> with the domain and user name, for example EXAMPLE\AgentUser . For a local user, specify just the user name, for example AgentUser .
   - Replace <GROUP> with the name of a group that the user belongs to, for example AgentGroup . The command gives this group full control of all files in the agent data directory (%OvDataDir%), and also full control of all installed packages. If you previously started the command and specified a different group, the command removes control of the files for the previous group.
   - Replace <PASSWORD> with the user's password.

   NOTE:
   The command assigns the user rights required for basic agent functionality at group level, not to the individual user. Therefore, take care when you select the group to use. It is advisable to create a new group specifically for the agent user, and add the agent user as a member.

4. Type the following commands:

   a. **ovc -kill**

b. **`ovc -start`**

The control service and agent processes now run as the user that you specified.

Related Topics:

- Change the default user of DCE agents on Windows nodes
- Configure HTTPS agent installation defaults
- ovconfchg
- ovconfpar

Change the user of an HTTPS agent on a UNIX or Linux node

By default on nodes with a UNIX or Linux operating system, the HTTPS agent runs under the root account. However, you can configure the HTTPS agent to run under a different user account. For example, you may want the agent to run under an account with fewer permissions than the root account. Alternatively, you may want the agent to run under an account that has permission to access remote systems over the network.

You must test whether the user account has appropriate rights to run the agent and manage the node correctly. You may need to assign additional rights for the management tasks that you need to perform. For example:

- If you want to be able to monitor a log file using a policy, the agent user must have permission to read that log file.
- If you want to be able to start a program using an automatic command, operator-initiated command, tool, or scheduled task, the agent user must have permission to start that program.
- Some Smart Plug-ins may require additional configuration or user rights when the agent runs under an alternative user. For more details, see the documentation for individual Smart Plug-ins.

To change the user of an HTTPS agent

1. *Optional.* Create a new user for the agent to run under.
2. *Optional.* Create a new group, and add the user as a member of this group.
3. On the node, log in as root and open a shell prompt. Ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type `export PATH=/opt/OV/bin:$PATH` and then press Enter .
   - On AIX, type `export PATH=/usr/lpp/OV/bin:$PATH` and then press Enter .
   - On Tru64, type `export PATH=/usr/opt/OV/bin:$PATH` and then press Enter .
4. To stop the agent, type the following command:

   ovc -kill
5. To change the agent user, type the following command:

   ovswitchuser.sh -existinguser <*USER* > -existinggroup <*GROUP* >

   - Replace <*USER* > with the user name, for example AgentUser . The command modifies the system's boot scripts so that the agent runs under the user that you specify.

- Replace <*GROUP* > with the name of a group that the user belongs to, for example AgentGroup . The command gives this group full control of all files in the agent data directory, and also full control of all installed packages. If you previously started the command and specified a different group, the command removes control of the files for the previous group.

  The group ID flag is set on the agent's data directories*.* This flag means that the group that you specify will also own any new files and subdirectories in the agent's base directories.

NOTE:
The command assigns the user rights required for basic agent functionality at group level, not to the individual user. Therefore, take care when you select the group to use. It is advisable to create a new group specifically for the agent user, and add the agent user as a member.

6. HTTPS agents include communication brokers that listen for inbound connections from management servers on port 383 by default. However, on UNIX and Linux nodes, non-root users cannot open ports in the range 0 to 1023. Therefore, you must configure the communication broker on the node to listen on a different port (above 1023). You must also configure the management servers that connect to the node, so that their outbound connections are destined for the correct port.

   You configure communication broker ports by setting the PORTS parameter in the bbc.cb.ports name space. You can configure this parameter in the following ways:

   - Configure the values in the HTTPS agent installation defaults. This is recommended if you need to configure communication broker ports for large numbers of nodes. You must plan and configure the installation defaults before you create or migrate your nodes.
   - Use `ovconfchg` or `ovconfpar` at a command prompt.

   The value must contain one or more host names or IP addresses and have the following format:

   <*host* >:<*port* >[,<*host* >:<*port* >] ...

   For example, to configure the communication broker port to 5000 on a node with the host name node1.emea.example.com, use the following command on the node itself, and also any management servers that open connections to it:

   ```
   ovconfchg -ns bbc.cb.ports -set PORTS node1.emea.example.com:5000
   ```

7. To start the agent, type the following commands:

   a. **su <*USER* >**

   b. **ovc -start**

   The control service and agent processes now run as the user that you specified.

Related Topics:

- Configure communication broker ports
- Configure HTTPS agent installation defaults
- ovconfchg
- ovconfpar

Change the default user for commands

By default, the agent starts automatic or operator-initiated commands under the user account that the agent itself is currently running under. However, you can configure an HTTPS agent to start commands under a different user account. You do this by setting the OVO_STD_USER parameter in the `eaagt` name space on the nodes. You can configure this parameter in the following ways:

- Configure the values in the HTTPS agent installation defaults. This is recommended if you need to configure the user for large numbers of nodes. You must plan and configure the installation defaults before you create or migrate your nodes.
- Use `ovconfchg` or `ovconfpar` at a command prompt.

Specify the value of OVO_STD_USER in the format <*user*>/|<*encrypted password*>

- Replace <*user*> with the name of the user. For a domain user, specify the domain and user name, for example `EXAMPLE\AgentUser`. For a local user, specify just the name, for example `AgentUser`.
- Replace <*encrypted password*> with output from the command `opcpwcrpt` <*password*>. You can start this command from a command prompt on the management server.

It is also possible to use the OVO_STD_USER when you configure or launch a tool. Specify the user name $OVO_STD_USER and leave the password blank.

You must test whether the user account has appropriate rights to run commands and tools correctly.

CAUTION:
If the agent fails to start a command or tool as the OVO_STD_USER, the agent may start the command or tool under the same user account that the agent is currently running under. This can happen, for example, if you specify an incorrect user or password.

Related Topics:

- Configure HTTPS agent installation defaults
- ovconfchg
- ovconfpar

Troubleshooting HTTPS agents

The following steps help to troubleshoot problems with nodes that use the HTTPS agent. This assuming that agent is correctly installed. If this is not the case, see Deployment troubleshooting or Manual installation of HTTPS agents for more information.

- If you install an HTTPS agent on a computer on which another HP BTO Software product already exists, the values of any existing configuration parameters remain unchanged. In some cases, you many need to complete additional steps before the agent can run. For example:
  - If the node's core ID is already set, you may need to check that node's core ID is correct on the management server. Right-click the node in the console tree, and then click Properties... . The node properties dialog appears. In the General tab, click Advanced Configuration .
  - If the certificate server is already set, you may need to change it or manually install certificates for the HPOM management server (see Configuring Certificates).

  Alternatively, to force the agent installation to replace the values of any existing configuration parameters, preinstall the agent without starting it (see Preinstall an HTTPS agent), and then activate the agent with the `-force_config_mode` option as follows:

  - On Windows operating systems:
    `cscript opcactivate.vbs -srv <`*`management_server_host_name`*` > -cert_srv <`*`certificate_server_host_name`*` > -force_config_mode`
  - On UNIX and Linux operating systems:
    `./opcactivate -srv <`*`management_server_host_name`*` > -cert_srv <`*`certificate_server_host_name`*` > -force_config_mode`

- Check the agent installation log file. After you install or upgrade an HTTPS agent, a log file is in available in:
  `<`*`data_dir`*` >\log\opc_inst.log.`

  After you deinstall an HTTPS agent, a log file is available in the following location:

  - On nodes that run a Windows operating system:
    `%SYSTEMROOT%\temp\opc_inst.log`
  - On nodes that run a UNIX or Linux operating system:
    `/var/tmp/opc_inst.log` .

- Check whether the management server can resolve the node's host name to an IP address, and whether the node can also resolve the management server's host name. The operating system normally provides a suitable command such as `nslookup` or `dig` .

  On nodes that run the Solaris operating system, the HPOM provides the command `ovgethostbyname` for this purpose.

- Check whether the management server can connect to the node, and whether the node can connect to the management server. The following command enables you to do this:

  ```
  bbcutil -ping [<hostname>|<ip>][:<port>]] [count]
  ```

  For example, to check whether a node can connect to manager1.example.com by sending 10 packets, open a command prompt on the node and type the following command:

  ```
  bbcutil -ping manager1.example.com 10
  ```

  If the connection is successful, the command returns `status=eServiceOK`.

  To run the command on a management server that is part of a cluster, you must also specify the resource group name by adding the -ovrg option.

  ```
  bbcutil -ovrg <resource> -ping [<hostname>|<ip>][:<port>]] [count]
  ```

- Check the status of the communication process, using the following command:

  ```
  ovbbccb -status
  ```

  Alternatively, to check the status of communication processes on a remote system, use the following command:

  ```
  bbcutil -status [<hostname>|<ip>][:<port>]]
  ```

  If the processes are not running, restart communication processes using the following command:

  ```
  ovc -restart ovbbccb
  ```

  If the communication process does not start successfully:

  - Check the log file <*data_dir*>`\log\System.txt` for error messages.
  - Type `ovbbccb -nodaemon -verbose` and then press Enter. This attempts to start the communication process, and displays details of any errors.

- Check that the HTTP communication is possible between the node and management server. Open the following location in a web browser on a node or management server:

  `http://<`*`hostname`*`>:<`*`port`*`>/Hewlett-Packard/OpenView/BBC/`

  By default, the communication process listens for connections on port 383. You can check which port the communication process is listening on using the following command:

  `bbcutil -getcbport <`*`hostname`*`>`

  If HTTP communication is possible, the HTTP Communication Information Modules page opens.

- Check that the node communicates with the correct management server. Check the management server using the following command on the node:

  ```
  ovconfget sec.core.auth MANAGER
  ```

  The command must return the correct management server hostname.

  Check that the node has the correct management server core ID. Get the core ID using the following command on the management server:

  ```
  ovcoreid [-ovrg server]
  ```

  The `-ovrg` option is only necessary if the management server is part of a cluster. The command returns a core ID, which must match the core ID that the following command returns on the node:

  ```
  ovconfget sec.core.auth MANAGER_ID
  ```

- Check that the node communicates with the correct certificate server using the following on the node command:

  ```
  ovconfget sec.cm.client CERTIFICATE_SERVER
  ```

  The command must return the host name of the management server that acts as the certificate authority. If the command returns an incorrect value, the command `ovcert -list` should confirm that the node has no certificates. If this is the case, you can set the correct value using the following command:

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

  You can then restart the agent using the following command `ovc -restart`

- Check that the node has received the certificates that it requires using the following command:

  ```
  ovcert -check
  ```

  If the certificates are missing, ensure that the node receives them. For more information, see Configuring Certificates .

- Check that the node's core ID is correct on the management server. Get the core ID using the following command on the node:

  ```
  ovcoreid
  ```

  The core ID should exactly match the agent ID on the management server. To check this, right-click the node in the console tree, and then click Properties... . The node properties dialog appears. In the General tab, click Advanced Configuration . The Advanced Configuration dialog appears, which enables you to check that the IDs match, and change the agent ID on the management server if necessary.

- Check that the messages that you expect to arrive from the node are not being suppressed. Right-click Operations Manager , and then click Configure→ Server... . The Server Configuration dialog appears. Check the settings in the Message Suppression tab. For more information, see Duplicate message suppression .

- The management server does not start to automatically deploy policies until the node's package inventory contains the agent package. Normally, this happens automatically. To view the node's package inventory, right-click the node in the console tree, and then click View →Package Inventory .

  If the agent package is missing from the node's package inventory, but the agent is installed on the node, synchronize the node inventory manually. Right-click the node, and then click All Tasks→Synchronize inventory →Packages . The management server adds the HTTPS agent package to its node inventory, and starts automatic policy deployment (if enabled).

Configuring Certificates

Nodes that you manage with HTTPS agents require certificates. Certificates enable nodes to communicate securely with the management server and other nodes. Therefore it is essential that certificates are safely deployed to nodes to ensure that all subsequent communication is secure.

A management server can issue certificates to nodes, acting as a certificate authority. Each node needs the following certificates from the management server:

- A unique *node certificate* . The node can identify itself to its management server and other nodes by sending them its node certificate.
- A copy of the management server's *trusted certificate* . A node only allows accepts communication from a management server if it has the trusted certificate for that management server.
- In an environment with multiple management servers, a copy of the trusted certificates for all other management servers.

You can ensure that a node obtains these certificates in the following ways:

- Request certificates automatically
- Request certificates with an installation key
- Deploy certificates manually

You can configure the management server to handle certificate requests in the following ways:

- Map certificate requests to nodes
- Grant certificate requests automatically
- Grant or deny certificate requests manually

You can maintain the certificate authorities on management servers in the following ways:

- Configure trusted certificates for multiple management servers
- Backup and restore certificates

Related Topics:

- Configure HTTPS agent installation defaults

Request certificates automatically

Nodes that you manage with HTTPS agents require certificates. Certificates enable nodes to communicate securely with the management server and other nodes.

When you deploy an HTTPS agent to a node using the console, the node requests certificates automatically from the management server. The node encrypts the certificate request using a key, which is embedded in the agent software. This is more secure than sending the request unencrypted, but does not provide full security.

The request must then be granted on the management server. You can configure this to happen automatically or manually. After this happens, the management server sends the certificates to the node. If the management server denies the certificate request you can send another using the following command on the managed node:

```
ovcert -certreq
```

In a highly secure environment, you should disable automatic certificate requests. Do this by setting the certificate deployment type to manual in the HTTPS agent installation defaults. You then need to either request the certificates with installation key or deploy the certificates manually.

Related Topics:

- Configure HTTPS agent installation defaults
- Deploy certificates manually
- Request certificates with an installation key

Request certificates with an installation key

You can use installation keys to encrypt certificate requests. You generate an installation key on the management server , and then transfer it to the node manually.

Requesting certificates with an installation key is more secure than using standard certificate requests. An installation key is unique, and you can use it to encrypt only one certificate request. If you use standard certificate requests, all nodes encrypt all requests using the same key, which is embedded in the agent software.

Also, if you request certificates with an installation key, you ensure that the node's private key never leaves the node to which it belongs. This is not the case when you install certificates manually, because you generate the node's private key and certificate on the management server and then copy it to the node.

Before you request certificates with an installation key, ensure that the HTTPS agent is running on the node. Normally, the agent sends a certificate request the first time it starts. If you then request a certificate with an installation key, the new certificate request overwrites the original certificate request on the management server. You can suppress the first certificate request by setting the parameter CERTIFICATE_DEPLOYMENT_TYPE=manual in the sec.cm.client namespace using the HTTPS agent installation defaults.

To request certificates with an installation key

1. Log in to the management server with an account that is a member of the HPOM administrators group. Open a command prompt.
2. Use `ovowcsacm` to generate an installation key. The syntax for this command is:

   `ovowcsacm -genInstKey [-file <`*`file_name`*` >] [-pass <`*`password`*` >]`

   Specify the options as follows:

Option	Description
`-genInstKey`	Specifies that you want to generate an installation key.

`-file <`*`file_name`*` >`	*Optional.* The name of the file into which the command generates the installation key. If you omit this option, the command creates a file in the following directory:
	
	`\<`*`data_dir`*` >\shared\server\certificates`
	
	The default file name has the following format:
	
	`CertificateIK_<`*`management_server_name`*` >_<`*`universally_unique_id`*` >`

`-pass <`*`password`*` >`	*Optional.* A password that the command uses to encrypt the installation key. You need this password when you later request the certificates from the node. If you omit this option, the command prompts you for a password.

3. Log in to the node with the same account used to install the node. Open a command or shell prompt.
4. Securely transfer the generated file to the node. The installation key is valid for any node.
5. On nodes that run a UNIX or Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type **`export PATH=/opt/OV/bin:$PATH`** and then press Enter .
   - On AIX, type **`export PATH=/usr/lpp/OV/bin:$PATH`** and then press Enter .
   - On Tru64, type **`export PATH=/usr/opt/OV/bin:$PATH`** and then press Enter .
6. Use `ovcert` to request a certificate from the management server. The syntax for this command is:

   ```
   ovcert -certreq -instkey <file_name >
   ```

   The command prompts you for the password that you specified when you generated the installation key. The node then uses the installation key from the file to encrypt a certificate request, which it then sends to the management server.
7. The request must then be granted on the management server. You can configure this to happen automatically or manually. After this happens, the management server sends the certificates to the node.

Related Topics:

- Configure HTTPS agent installation defaults
- Grant certificate requests automatically
- Grant or deny certificate requests manually

Deploy certificates manually

You can generate certificates for nodes on the management server , and then transfer them to the nodes manually. This avoids sending certificates over the network before fully secure HTTPS communication. For example, if you do not want to transmit certificates over the network, you can export them to a file on a disk and take the disk to the node.

Normally, the agent sends a certificate request to the management server the first time it starts. You can suppress this certificate request by setting the parameter CERTIFICATE_DEPLOYMENT_TYPE=MANUAL in the sec.cm.client namespace using the HTTPS agent installation defaults.

To deploy certificates manually

1. Log in to the management server with an account that is a member of the HPOM administrators group. Open a command prompt.
2. Use ovowcsacm to generate a certificate. The syntax for this command is:

   `ovowcsacm -issue -name <node_name > [-file <file_name >] [-coreid <OvCoreId >] [-pass <password >]`

   Specify the options as follows:

Option	Description
`-issue`	Specifies that you want a certificate for a node.
`-name <node_name >`	The primary name of the node to generate a certificate for. The node must already exist in the console.
`-file <file_name >`	*Optional.* The name of the file into which the command generates the certificates. If you omit this option, the command creates a file in the following directory: `%OvShareDir%server\certificates` The default file name has the following format: `<node_name >-<OvCoreId >.p12` .
`-coreid <OvCoreID >`	*Optional.* The OvCoreID, which uniquely identifies the node, is used to generate the certificates. If you omit this option, the command generates an ID for the node. You need to specify the OvCoreID if the node currently exists in the console, and the HTTPS agent is already installed on the node. To find an existing node's OvCoreID: a. In the console tree, right-click the node, and then click **Properties** . The node properties dialog appears. b. In the General tab, click **Advanced Configuration** . The Advanced

	Configuration dialog appears, which shows the ID that you need.
`-pass <`*`password`*` >`	*Optional.* A password that the command uses to encrypt the certificate data. You need this password when you later import the certificates on the node. If you omit this option, the command prompts you for a password.

3. If the HTTPS agent is not already installed on the node, install it. If you manually install the agent, use a profile. This ensures that the agent uses the same OvCoreID that ovowcsacm generated on the management server.
4. Log in to the node with the same account used to install the node. Open a command or shell prompt.
5. Securely transfer the generated file to the node.
6. On nodes that run a UNIX or Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type **`export PATH=/opt/OV/bin:$PATH`** and then press Enter .
   - On AIX, type **`export PATH=/usr/lpp/OV/bin:$PATH`** and then press Enter .
   - On Tru64, type **`export PATH=/usr/opt/OV/bin:$PATH`** and then press Enter .
7. If the agent is running on the node, type **`ovc -stop`** and then press Enter . This stops the agent processes on the node.
8. Use `ovcert` to import the certificates from the generated file. The syntax for this command is:

   ```
   ovcert -importcert -file <file_name >
   ```

   The command prompts you for the password that you specified when you generated the certificates. Type the password and press Enter . The command then notifies the management server that the certificates are installed and the management server updates the node's certificate state.

   If the management server does not receive the notification for any reason, you must update the node's certificate state manually, as follows:

   a. In the console-tree, right-click the node, and then click Properties . The Node properties dialog appears.
   b. In the General tab, click Advanced Configuration . The Advanced Configuration dialog appears.
   c. Select the Modify Certificate State check box, and then select Installed in the list of certificate states.
   d. Click OK .

   NOTE:
   If the node's OvCoreID does not match the OvCoreID in the certificate, you see a warning on the node that the common name field in the certificate does not match the OvCoreID of the system. If

the node is new (you are not reinstalling or migrating the agent on an existing node), you can change the node's OvCoreID as follows:

   a. Copy the certificate's common name field from the warning.

   b. Type **ovcoreid -set <***common name field* **> -force** and then press Enter .

For example, for the following warning:

```
WARNING: The common name field (CN) in the certificate
         '89aea662-b9e6-7527-148d-8a612e083f23' does not match the OvCoreId
         '8b2ae5c2-b99c-7527-0263-cf9a16f2aace' of the system.
```

the command would be:

```
ovcoreid -set 89aea662-b9e6-7527-148d-8a612e083f23 -force
```

9. On the node, type **ovc -start** and then press Enter . This restarts the agent processes.
10. Securely delete any copies of the file that contains the certificates. Depending on how you generate and transfer the file, you may, for example, have copies in the following locations:
    - on the management server
    - on a floppy disk, CD, or other portable media
    - on the node
11. *Optional.* If you enabled automatic policy deployment for the node, the policy deployment jobs may have failed before you installed the certificate. To restart a failed job:
    a. In the console tree, expand Policy management→Deployment jobs .
    b. Right-click the failed job, and then click All Tasks→Restart job .

Related Topics:

- Create new nodes
- Manually install an HTTPS agent with a profile
- Configure HTTPS agent installation defaults
- Configure general information for managed nodes

Map certificate requests to nodes

When the management server receives a certificate request, it attempts to automatically identify the node that the request comes from using the host name. This automatic mapping fails in the following situations:

- There is no node with the same host name as the system that the request came from.
- There is a node with the same host name, but the node has an OvCoreId that is different to the OvCoreId in the certificate request.

If the management server fails to map a certificate request to a node, and you know that the node already exists in the database, you can map the certificate request to the node manually. If a node does not already exist in the database, you must configure it before you can map the certificate request. The node is listed under Unmanaged Nodes with Agents in the Configure Managed Nodes dialog.

You can unmap certificate requests if you need to, and then map them to a different node.

As well as mapping certificate requests in the console, you can also map and unmap them from the command prompt.

To map certificate requests to nodes

1. In the console tree, click Certificate Requests . A table of certificate requests appears in the details pane.
2. Right-click an unmapped certificate request, and then click All Tasks →Map to Node . The Map Certificate Request to Node dialog appears.
3. Navigate the node tree and click the node that you want to map the certificate request to.
4. Click Map Node .
5. Read the warning message that appears, and then click OK or Cancel . If you click OK, the management server updates the node's OvCoreID with the OvCoreID from the certificate request. The certificate request then becomes pending and can be granted either automatically or manually.

To unmap certificate requests

1. In the console tree, click Certificate Requests . A table of certificate requests appears in the details pane.
2. Right-click a mapped certificate request, and then click All Tasks →Unmap . You can then map the certificate request to a different node if appropriate.

To map certificate requests from the command prompt

1. Log in to the management server with an account that is a member of the HPOM administrators group. Open a command prompt.
2. Use `ovowcsa -listpending` to get a list of certificate requests.
3. Map a certificate request by specifying the request ID or host name with the following command:

   `ovowcsa -map` <*request_ID* | *request_host_name* >[=<*node_host_name* | *OvCoreID* >] [-force]

   The following restrictions apply:

   - If you omit the node_host_name and `OvCoreId` , the command attempts to map the certificate request using the `request_host_name` . This is useful if you configured a node with that host name after the original attempt to map the certificate request failed.
   - If there is more than one certificate request with the same `request_host_name` , you must specify the `request_ID` .
   - If the host name in the certificate request is different to the node's host name, add the `-force` option.
   - If the node has an OvCoreID that does not match the OvCoreID in the certificate request, the mapping fails. Add the `-force` option to map the certificate request.

Related Topics:

- Configure managed nodes

Grant certificate requests automatically

Nodes that you manage with HTTPS agents require certificates. Certificates enable nodes to communicate securely with the management server and other nodes. Unless you install the certificates manually, the node requests them from the management server. Before you can start to manage the node, you need to grant this certificate request.

You can configure the certificate server to automatically grant the following types of certificate request:

- Certificate requests that are encrypted with a valid installation key.
- Certificate requests from nodes that the management server recently deployed the HTTPS agent to.
- Certificate requests from specific nodes that you flag (by selecting Automatically grant certificate in the system properties of the node).

To grant certificate requests automatically

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog appears.
2. Click Namespaces , and then click Agent Certificate Autogranting . A list of values appears.
3. Set the value of Enable autogranting to True . The management server grants all certificate requests that were made manually using an installation key.

   If you set this value to false, the management server does not grant any certificate requests automatically.
4. Configure one or more of the following options to automatically grant certificate requests from recently deployed or flagged nodes:
   - Certificate requests from nodes with a recently deployed HTTPS agent

     To automatically grant certificate requests from nodes to which the management server recently deployed the HTTPS agent:

     i. Set the value of Enable autogranting by time interval to True .

     ii. Set the value of Time interval for autogranting to a number of seconds. The management server automatically grants certificate requests that arrive this number of seconds after the start of the agent deployment job or sooner.
   - Certificate requests from flagged nodes

     To automatically grant certificate requests from specific nodes that you flag:

     i. Set the value of Enable autogranting of flagged nodes to True .

ii. Set the flag to automatically grant certificate requests when you create each node.

Alternatively, to set the flag for an existing node:

TIP:
To change the default value of this flag, set the value of Default for the 'Automatically grant certificate' flag of new nodes in the Server Configuration dialog.

iii. Combination mode for recently deployed or flagged nodes

Set the value of Combination mode of autogranting conditions to AND or OR . Use AND to automatically grant any certificate request that not only arrives in the specified time but also comes from a flagged node. Use OR if certificate requests must meet just one of these criteria.

For the AND operation to function properly, set both Enable autogranting by time interval and Enable autogranting of flagged nodes to true.

iv. Click OK .

v. Use the Windows service manager to restart the OvSecurityServer service.

Related Topics:

- Configure system information for managed nodes
- Request certificates with an installation key
- Request certificates automatically

Grant or deny certificate requests manually

Nodes that you manage with HTTPS agents require certificates. Certificates enable nodes to communicate securely with the management server and other nodes. Unless you install the certificates manually, a node requests the certificates from the management server. Before you can start to manage the node, you need to grant this certificate request.

You can view a table of the certificate requests that the management server has received, and manually grant or deny requests that are pending. Alternatively, you can grant or deny a pending certificate request from the command prompt.

To grant or deny certificate requests manually

1. In the console tree, click **Certificate Requests** . A table of certificate requests appears in the details pane.
2. Right-click a pending certificate request, and then click one of the following:
   - **All Tasks** → **Grant** - The management server send certificates to the node.
   - **All Tasks** → **Deny** - The management server informs the node that its certificate request is denied.
   - **All Tasks** → **Discard** - The management server deletes the certificate request without informing the node.
3. If a confirmation message appears, read the message and then click **OK** or **Cancel** .

To grant certificate requests manually from the command prompt

1. Log in to the management server with an account that is a member of the HPOM administrators group. Open a command prompt.
2. Use `ovowcsa -listpending` to get a list of pending certificate requests. To obtain the list in a different format, add the `-format` option and any combination of the following letters:
   - r - request ID
   - h - host name
   - i - IP address
   - o - OvCoreID
   - m - mapped host name
   - p - platform

- t - time received
- s - show header line

3. Grant or deny a certificate request by specifying the request ID or host name with one of the following commands:

    - `ovowcsa -grant` <*request ID* | *host name* > - The management server send certificates to the node.
    - `ovowcsa -deny` <*request ID* | *host name* > - The management server informs the node that its certificate request is denied.
    - `ovowcsa -discard` <*request ID* | *host name* > - The management server deletes the certificate request without informing the node.

Configure trusted certificates for multiple management servers

Every node that has the HTTPS agent has a certificate, which it uses identify itself to management servers . Every management server also has a certificates, which it uses to identify itself to nodes. Nodes receive their certificates from the certificate authority on their primary management server.

In an environment with multiple management servers, you must configure each secondary management server to trust certificates that primary management servers issued. This involves exporting the trusted certificate from every primary management server to every secondary management server. You must also update the nodes' trusted certificates, so that the nodes trust the secondary management servers.

To configure trusted certificates for multiple management servers

1. On every management server, export the trusted certificate to a file using the following command:
   `ovcert -exporttrusted -ovrg server -file` *<file>*

   The command generates a file with the name that you specify.

2. Copy each file to every other management server, and import each trusted certificate using the following command:
   `ovcert -importtrusted -ovrg server -file` *<file>*

3. For every management server, update the trusted certificates on existing nodes:
   a. In the console tree, click Tools→HP Operations Manager Tools→ Certificate Management .
   b. In the details pane, double-click Update trusted certificates . A dialog box opens, which lists nodes and services.
   c. Select the nodes on which to update the trusted certificates. To update trusted certificates on all existing nodes, select the Nodes check box, and then click Launch... . The Tool Status dialog box opens and shows progress. (The tool fails for any nodes that have the DCE agent.)

      Alternatively, you can select check boxes for individual nodes or node groups, but always include the management server node in your selection.

Any new nodes that you create will receive all the trusted certificates when they receive their node certificate.

Related Topics:

- Scalable Architecture for Multiple Management Servers

Backup and restore certificates

It is important to keep a backup of the certificates for the management server on which the certificate server runs. Otherwise, if you lose or corrupt the management server's certificates, you would need to reissue every node certificate.

You can backup the certificates to files, which you should remove from the management server and store securely. You need to backup three certificates from management server:

- Management server certificate (the certificate authority certificate)
- Trusted certificate (or several trusted certificates if you imported more from other management servers)
- Node certificate (because the management server is also a node)

To backup certificates

1. Backup the management server certificate using the following command:
   `ovcm -exportcacert -file <`*`file`*`> [-pass <`*`pass_phrase`*`>]`

   The command generates a file with the name that you specify. If you specify the -pass option and the pass phrase contains spaces, surround it with quotation marks (""). If you omit the -pass option, the command prompts you for a password.

2. Backup the trusted certificate(s) to a file using the following command:
   `ovcert -exporttrusted -ovrg server -file <`*`file`*`>`

   The command generates a file with the name that you specify.

3. Find the alias of the node certificate using the following command:
   `ovcert -list -ovrg server`

   The alias of the node certificate is the long sequence of characters, which appears under the heading "Certificates". For example:

```
+---------------------------------------------------------+
| Keystore Content (OVRG: server)                         |
+---------------------------------------------------------+
| Certificates:                                           |
|     cdc7b5a2-9dd6-751a-1450-eb556a844b55 (*)            |
+---------------------------------------------------------+
| Trusted Certificates:                                   |
|     CA_cdc7b5a2-9dd6-751a-1450-eb556a844b55 (*)         |
+---------------------------------------------------------+
```

Backup the node certificate using the following command:

`ovcert -exportcert -file <`*file*`> -alias <alias> [-pass <`*pass_phrase*`>]`

The command generates files with the names that you specify. If you specify the -pass option and the pass phrase contains spaces, surround it with quotation marks (""). If you omit the -pass option, the command prompts you for a password.

To restore certificates

1. Restore the management server certificate using the following command:
   `ovcm -importcacert -file <`*file*`> [-pass <`*pass_phrase*`>]`

   The command restores the certificate from a file with the name that you specify. To restore the certificate, you need the pass phrase that was used to create the file. If you specify the -pass option and the pass phrase contains spaces, surround it with quotation marks (""). If you omit the -pass option, the command prompts you for the password.

2. Restore the trusted certificate using the following command:
   `ovcert -importtrusted -file <`*file*`>`

3. Restore the node certificate using the following command:
   `ovcert -importcert -file <`*file*`> [-pass <`*pass_phrase*`>]`

   The command imports the certificates from the file.

Configure tools

The Tool Configuration editor allows you to create the tools that operators can apply to managed nodes and services. By applying the available tools, operators resolve problems and perform routine tasks to maintain systems and services that have critical business impact. Tools can be executed on nodes or services.

For services, a tool is executed on nodes configured in the Service Configuration editor as nodes on which the tool runs. If no such nodes have been configured, the tool runs on the node where the service is hosted.

You can assign tools to specific user roles so that only users assigned to that role can apply the tools you specify. By assigning tools to a specific service type, only the tools you assign to that service type are available from the associated service.

The tools that you create are listed in tool groups in the HP Operations Manager for Windows Tools folder of the console tree, which also displays the default tool groups supplied with HP Operations Manager for Windows. You can create subfolders for sets of tools so that tools can be organized according to type of problem, operating system, function, or other convenient category. For example, you might group editing tools in one folder and performance tools in another.

One tool can be included in several folders. If you edit the tool, the changes go to all instances of the tool. To delete a tool, you must remove it from every location where it occurs.

You must configure each tool individually. An asterisk (*) indicates required information.

To configure a tool or tool group

1. Open the Tool Configuration editor, if it is not already open.
2. Select the name of the tool or tool group you want to configure and right-click to open the shortcut menu.
3. Select Properties to open the Properties or Group Properties dialog box. Tool properties dialog boxes display the Tool icon . Tool group properties dialog boxes display this Tool Folder icon . The dialog box title reflects the selection you made in step 2. Use the tabs to configure:
   - General: information that identifies and describes the tool.
   - Details: Tool Properties dialog box only. Provide details about the type of tool, its parameters, where it will be executed, and whether operators have permission to change the tool parameters. If the tool is a script, enter the script text.
   - Target: Tool Properties dialog box only. Specify the node you want to associate with this tool. (Optional).
   - Nodes: Tool Properties dialog box only. Displays lists of the nodes and node folders associated

with this tool.

4. When you finish your configuration, click Apply to see the effects of your changes.
5. Click OK to confirm your changes, close the Tools Properties dialog box, and return to the Configure Tools dialog box. The tool you configured appears in the list of tools in the details pane.
6. Click Apply to apply your changes without closing the dialog box.
7. Click OK to confirm your changes and close the Configure Tools dialog box. The new tool or folder appears in the list of tools in the console tree.
8. Click Cancel to cancel the creation of the new tool or folder.

To specify tools for services, use the Service Configuration Editor.

Server or cluster failover behavior

HPOM for Windows monitors the availability of server components such as the Windows Management Interface (WMI), the server itself, and the cluster resource group (if your environment contains clusters.)

In the event of a failure of one or more of these components, some activities may be temporarily unavailable, until WMI or the server is restored. See Related Topics: for details.

Related Topics:

- Create a new tool or tool group
- Configure general information for tools
- Configure tool details
- Configure targets for tools
- Configure tools for user roles
- Select tools for service types
- Server or cluster failover behavior

Create a new tool or tool group

You can create a new tool or tool group, rather than select from the list of existing tools. Perhaps the tool you want to add is not included in the Tools list and so is not available for selection. In that case, you can add a new tool or group of tools manually.

To add a new tool or tool group

1. Open the Tool Configuration Editor, if it is not already open.
2. Right-click to open the shortcut menu.
3. Select either New Tool or New Tool Group . The Tool Properties dialog box automatically opens so that you can begin configuring the new tool.
4. Configure the new tool or tool group as necessary using the tabs in the Tool Properties or Tool Group Properties dialog box. An asterisk (*) indicates required information. You may see a message prompting you to supply required information.
5. Click Cancel to close the properties dialog box without saving your changes. If you choose Cancel , your new tool or tool group will not be created.
6. Click Apply to apply your changes without closing the properties dialog box.
7. Click OK to confirm your changes, close this dialog box, and return to the Configure Tools dialog box. Your new tool or tool group appears in the list of tools.
8. Click Apply to apply your changes without closing the dialog box.
9. Click OK to confirm your changes and close the Configure Tools dialog box. The new tool or folder appears in the list of tools in the console tree.

To delete a tool

1. If you want to delete a tool, select it in the Configure Tools dialog box list.
2. Right-click to open the shortcut menu and select Delete or press the Delete key. A message prompts you to confirm the delete operation.

Related Topics:
- Configure general information for tools
- Configure tool details
- Configure targets for tools
- Configure tools for user roles

- Select tools for service types
- User accounts for tools

Configure general information for tools

Use the General tab in the Tool Properties dialog box to specify details that identify each tool. An asterisk (*) identifies required information.

To configure general information for tools

1. Open the Tool Configuration editor, if it is not already open.
2. From the tools displayed in the selection tree, select the tool you want to configure.
3. Right-click to open the shortcut menu.
4. Select Properties to open the Tool Properties dialog box. Properties include:
   - General information about the selected tool, such as the display name of the tool.
   - Details about the specific tool you selected, such as the type of tool, the executable name, and script details, if any.
   - Target nodes on which the tool will run.
   - Nodes and node folders associated with this tool.

   The General tab displays by default.
5. In the Display Name box, enter the name for the tool as you want it to appear in the Tools list. This information is required.
6. Enter any comments or additional information in the Description box. This information is optional.
7. Select Show tool in message context to make this tool available in the context of all messages.
8. Select the Details tab to continue configuring this tool. You will see a message prompting you for required information.

Related Topics:
- Configure Tools
- Configure tool details
- Configure targets for tools
- Edit configured tools
- Configure tools for user roles
- Select tools for service types

Configure tool details

Use the Details tab of the Tool Properties dialog box to record information about the new tool. You must separately configure details for each new tool you create.

To configure tool details

1. Open the Tool Configuration Editor, if it is not already open.
2. Right-click the tool you want to configure to open the shortcut menu.
3. Select Properties to open the Tool Properties dialog box.
4. Select the Details tab.
5. In the Command Type list box, select a tool type. A tool type can be an executable, a URL, a VB script, a Jscript, Perl script, or a Windows Scripting Host (WSH) script.

   NOTE:
   VB scripts, Jscripts, Perl scripts, and WSHscripts run only on the management server or on a managed node. If .vbs has already been associated with the scripting engine, you can run the executable without a preface. For example: `filename.vbs` .

6. Your selection of a command type determines the options available within the rest of the dialog box. Click the down arrows for details on configuring each command type.

   Executable Command Type

   URL Command Type

   VB script, JScript, Perl script, or WSHscript Command Type
7. Click Apply to apply your changes.
8. Click OK to apply your changes and close the Tools Properties dialog box.
9. Click Cancel to close this dialog box without saving your changes.
10. Select the Target tab to continue configuring this tool.

Related Topics:
- Configure Tools
- Configure general information for tools
- Configure targets for tools
- Edit configured tools
- Configure tools for user roles

- Select tools for service types

Configure target nodes for tools

The tools you create can be used in several ways. Tools can be:

- Run only on the nodes you specify.
- Configured to permit users to choose the nodes on which the tool will run.
- Run in the context of a service (execute on the node that hosts that service).
- Restricted to user roles to which specific users are assigned.
- Restricted to a specific service type for which only the tools you assign to that service type are available from the associated service.

To specify nodes, use the Target tab in the Tool Properties dialog box to select the nodes on which you want the tools to run. You must also specify Node List in the Execute On list. This configures the tool to run on all the nodes on the Predefined Node List .

If you prefer to allow your users to determine where a tool is to be run, you must choose Selected Node in the Execute On list. When the tool is executed, a list appears from which users choose the location (service or node) where the tool will run.

To configure nodes for tools

1. Open the Tool Configuration editor, if it is not already open.
2. Right-click the tool to be configured to open the shortcut menu.
3. Select Properties to open the Tool Properties dialog box.
4. Select the Target tab.
5. In the Execute On list, select the target node or nodes on which the tool will run. Click the Execute On down arrow below to view options and target locations. Security options vary depending on the target location you select. See the *HP Operations Manager for Windows Installation Guide* for details.

   Execute On management server

   Select management server if you want the tool to run only on the server. As administrator, you can specify a user name, password, or both be required from the user.
6. In the User Name box, specify the ID under which the tool executes. For example, you might enter *root* .
7. Check the Run as Agent User check box if you want to set the User Name to $AGENT_USER. This parameter is later replaced with the name of the agent user account specified on the server. The account on the management server is initially set to Local System.
8. If a password is required, specify in the Password box the password for the special user authorized in Step 6. The administrator specifies these passwords during the process of tool configuration.

9. In the Verify Password box, retype the specified password to confirm it.
10. To add nodes to the Predefined Node List , you must have Node List selected in the Execute on box. Click Add to open the Add nodes or node groups dialog box. Select one or more nodes or node groups to associate with the tool you are configuring. Click OK to close the Add nodes or node groups dialog box and add your selections to the Predefined Node list in the Target tab.

    Click Remove to delete selected systems from the Target tab.
11. When you finish configuring the tool, click Apply in the Tool Properties dialog box to apply your changes.
12. Click OK to close the Tool Properties dialog box and save your changes.
13. Click Apply in the Configure Tools dialog box to save your new tool configuration. Click OK to save your changes and close the dialog box.

Related Topics:

- Configure Tools
- Configure general information for tools
- Configure tool details
- Edit configured tools
- Configure tools for user roles
- Select tools for service types

View nodes and node groups associated with this tool

The Nodes tab of the Properties dialog box for tools displays a list of the node groups and nodes associated with the selected tool.

This information is read-only.

Add nodes or node groups

You can select the nodes on which a tool runs by adding one or more selected nodes or node groups to the Predefined Node List in the Tool Properties Target tab. The advantage of adding a node group is that the Predefined Node List is updated automatically whenever the node group changes.

NOTE:
You must also specify Node List in the Execute On list in the Target tab.

To add nodes or node groups

1. In the Tool Properties dialog box Target tab, click Add to open the Add nodes or node groups dialog box.
2. Select one or more nodes or groups of nodes.
3. Click OK to close the Add nodes or node groups dialog box. The nodes and node groups you selected appear in the Predefined Node List in the Target tab.
4. Click Cancel to close the dialog box without saving any changes.

Variables for tools

The following variables can be used in the Parameters box in the Configure Tools dialog box, Details tab.

- Management server variables
- Message variables
- Node variables
- Node group variables
- Service variables
- Environment variables

Management server variables

- $OPC_MGMTSV

  Returns the name of the management server. $OPC_MGMTSV can also be used in automatic and operator-initiated commands.

  Availability: From all actions except automatic and operator-initiated commands.

Message variables

Some variables return TRUE or FALSE, depending on the existence of a specific message attribute. For example, if an automatic action is defined, TRUE is returned. Otherwise FALSE is returned.

If an attribute is empty, an empty string is returned. If you use an attribute that does not exist, it is treated like part of a normal string, which means no evaluation happens and the string remains unchanged. The data returned from variables is exactly the same type as that shown in the Message Properties dialog box. The indexing for word extraction from strings and for access to specific annotations starts with 1, not with 0.

All message variables are replaced per message. This means that if you select two messages, you will receive two requests; each of them might have different values for the parameters.

- $OPC_CUSTOM[name]

  The same as $OPC_MSG.CMA[name]. Returns the value of the custom message attribute name. For example, the `$OPC_CUSTOM[device]` variable could return the value Lan .

- $OPC_MSG.ACTIONS.AUTOMATIC

  Indicates whether an automatic action is defined. Sample output: `TRUE`

- $OPC_MSG.ACTIONS.AUTOMATIC.COMMAND

  Returns the script or program, including its parameters, performed as an automatic action for the selected message. Sample output: `dist_del.sh 30 warning`

- $OPC_MSG.ACTIONS.AUTOMATIC.NODE

  Returns the node on which an automatic action has been performed for the selected message. Sample output: `kernighan.c.com`

- $OPC_MSG.ACTIONS.AUTOMATIC.STATUS

  Returns the current status of the message's automatic action. The variable can return running, failed, or successful. Sample output: `successful`

- $OPC_MSG.ACTIONS.OPERATOR

  Indicates whether an operator-initiated action is defined. Sample output: TRUE

- $OPC_MSG.ACTIONS.OPERATOR.COMMAND

  Returns the script or program, including its parameters, performed as an operator- initiated action for the selected message. Sample output: `ps -ef`

- $OPC_MSG.ACTIONS.OPERATOR.NODE

  Returns the node on which an operator-initiated action has been performed for the selected message. Sample output: `kernighan.c.com`

- $OPC_MSG.ACTIONS.OPERATOR.STATUS

  Returns the current status of the message's operator-initiated action. The variable can return running, failed, or successful. Sample output: `successful`

- $OPC_MSG.ANNOTATIONS

  Indicates whether annotations exist for a message. Returns TRUE if at least one annotation exists for a message. Otherwise FALSE is returned. Sample output: TRUE

- $OPC_MSG.ANNOTATIONS[n]

  Returns the nth annotation. Sample output:

  ```
  Performed Message Correlation;
  Message Key Relation:
  Message 59d06840-ac4f-71d5-1f67-0f887e320000
  with condition id
  fe00fa34-9e34-71d5-143e-0f887e320000 ackn'ed
  0 messages.
  ```

- $OPC_MSG.APPLICATION

  Returns the name of the application related to the selected message. Sample output: `/usr/bin/su(1) Switch User`

- $OPC_MSG.CMA[name]

  Same as $OPC_CUSTOM[name].

- $OPC_MSG.CREATED

  Returns the date and time the message was created on the managed node. Sample output: `09/18/01 18:08:08`
- $OPC_MSG.DUPLICATES

  Returns the number of duplicate messages that have been suppressed. Sample output: `17`
- $OPC_MSG.GROUP

  Returns the message group to which the selected message belongs. Sample output: `Security`
- $OPC_MSG.INSTRUCTIONS

  Returns the text of the instruction. Sample output: `Available space on the device holding the (root) filesystem is less than the configured threshold. This may lead to ...`
- $OPC_MSG.LAST_RECEIVED

  Returns the date and time when the last duplicate message was received on the management server. Sample output: `09/16/01 03:17:23`
- $OPC_MSG.MSG_KEY

  Returns the message key that is associated with a message. Sample output: `my_appl_down:kernighan.c.com`
- $OPC_MSG.MSG_ID

  Returns the unique identification number for the selected message. Sample output: `217362f4-ac4f-71d5-13f3-0f887e320000`
- $OPC_MSG_IDS

  Returns the message IDs (UUIDs) of all the messages currently selected. IDs are separated by space. Sample output: `85432efa-ab4a-71d0-14d4-0f887a7c0000 a9c730b8-ab4b-71d0-1148-0f887a7c0000`
- $OPC_MSG.NO_OF_ANNOTATIONS

  Returns the number of annotations of a message. Sample output: `3`
- $OPC_MSG.NODE

  Returns the managed node from which the selected message was issued. Sample output: `kernighan.c.com`
- $OPC_MSG_NODES

  Returns the names of all nodes on which the events that generated currently selected messages took place. The names are separated by spaces. Duplicate nodes are ignored. Sample output: `kernighan.c.com richie.c.com`
- $OPC_MSG.OBJECT

  Returns the object which was affected by, detected, or caused the event. Sample output: `CPU`
- OPC_MSG.ORIG_TEXT

  Returns the original text of the selected message. Sample output: `SU 09/18 18:07 + 6 root-spooladm`

- $OPC_MSG.ORIG_TEXT[n]

  Returns the nth word in the original text of the message. Sample output: `the`

- $OPC_MSG.OWNER

  Returns the owner of the selected message. Sample output: `opc_op`

- $OPC_MSG.RECEIVED

  Returns the date and time the message was received on the management server. Sample output: `09/18/01 18:08:10`

- $OPC_MSG.SERVICE

  Returns the service name that is associated with the message. Sample output: `VP_SM:Agent:ServicesProcesses@@kernighan.c.com`

- $OPC_MSG.SEVERITY

  Returns the severity of the message. This can be Unknown, Normal, Warning, Minor, Major, or Critical. Sample output: `Normal`

- $OPC_MSG.SOURCE

  Returns the name of the application or component that generated the message. Sample output: `Message:opcmsg(1|3)`

- $OPC_MSG.TEXT

  Returns the complete text of the selected message. Sample output: `The following configuration information was successfully distributed: Templates (OpC30-814)`

- $OPC_MSG.TEXT[n]

  Returns the nth word in the text of the message text. Sample output: `following`

- $OPC_MSG.TIME_OWNED

  Returns the date and time when the message was acknowledged. Sample output: `09/18/01 18:11:10`

- $OPC_MSG.TYPE

  Returns the message type of the message. Sample output: `ECS`

View Details

Node variables

- $OPC_NODES

  Returns the primary node name.

  Availability: From all actions except automatic and operator-initiated commands.

  - If the tool is launched from the **Nodes** folder in the console tree, this variable is replaced with the primary node name of the nodes that are selected at the time the tool runs.

- If the tool is launched from the Service folder in the console tree, this variable is replaced with the primary node name of the nodes that have been associated with the selected service and the selected action. If this list is empty, the primary node name of the node the service is hosted on is used.
- If the tool is launched from the Tools folder in the console tree, a node and service selector displays, allowing you to select the nodes and services to use to replace this variable.

- $OPC_NODEID

  Returns the node identifier (GUID) of a node.

  Availability: From all actions except automatic and operator-initiated commands.

  - If the tool is launched from the Nodes folder in the console tree, this variable is replaced with the GUIDs of the nodes that are selected at the time the tool runs.
  - If the tool is launched from the Service folder in the console tree, this variable is replaced with the GUIDs of the nodes that have been associated with the selected service and the selected action. If this list is empty, the GIUID of the node the service is hosted on is used.
  - If the tool is launched from the Tools folder in the console tree, a node and service selector displays, allowing you to select the nodes and services to use to replace this variable.

Node Group variables

- $OPC_NODEGROUP_ID

  Returns the identifier (GUID) of a node group.

- $OPC_NODEGROUP_LABEL

  Returns the display name of a node group.

Service variables

- $OPC_SERVICE_NAME

  Returns the ServiceName of a service.

  Availability: Supports and fills in the name of the selected service. If a service is not selected, or an action is launched from a Node or Node Group, then these values will be NULL.

- $OPC_SERVICE_LABEL

  Returns the Caption of a service.

  Availability: Supports and fills in the caption of the selected service name. If a service is not selected, or an action is launched from a Node or Node Group, then these values will be NULL.

Environment variables

- $OPC_ENV(variable name)

  Used to retrieve environment variables from the console that launched the action.

  Example: $OPC_ENV(PATH): substitutes the variable with the PATH environment variable from the console machine.

  Availability: From the actions launched from configured tools. Not available from the automatic or operator-initiated commands running on the console.

Tool security

Security for tools focuses on the user account under which a tool is launched. When you configure a tool, you can specify if a user account will be used or not and if a password is required or not. The following sections explain the requirements and consequences of such configurations:

- Tool target location details and requirements
- Security authentication module
- Allow tools to run on Windows 2003 Server nodes
- Change the password for multiple tools

User accounts for tools

As an HPOM administrator, you can specify the account a tool runs under when it executes, according to the following requirements. Requirements may vary depending on the target location.

HPOM management server, managed node, or node list

If the target location for the tool is the management server, a managed node, or the node list, you can:

- Specify both the user name and password.

  NOTE:
  On target nodes with a UNIX operating system, the agent checks only the first eight characters of the password before it runs the tool. If you specify a password that is longer than eight characters, the agent ignores the extra characters.

- Specify a user name and leave the password blank.

  This uses the security authentication module opcauth.dll (part of the HP Operations agent package) to authenticate the login for the tool. This form of authentication has the following advantages:

  - Because no password is required, you do not have to update the configuration of tools using the specified account if the password changes. For example, an administrator can create an account (opc_op), but if opc_op changes its password, the administrator might not know the new password. In this case, HPOM allows the administrator to launch the tool as opc_op using the security authentication module (opcauth.dll).

    However, the administrator would not be able to interactively log in using this account. The administrator can start the tool using the opcauth.dll, but he cannot log on to the system as opc_op, because this would require a password which he does not know.
  - Supplying the user name but not a password allows an administrator to set up a tool to run under a special account without needing to know the account's actual password.
  - If you are using a local account that has a different password on each node, a single tool definition works for all of them.

- Leave both the user name and password blank.

  If you leave the user name blank, you must also leave the password blank.

  - All limitations mentioned above for an empty password also apply when both the user name and password are blank.
  - On Windows nodes, the tool runs as the user logged in to the console. The user you are logged in as

must be a domain user.

The machine on which you want to run the tool must recognize the account name. This takes place if the machine you are logged in on is in the same domain as the target node. If not, the domain of the target node must trust the domain of the user account used when executing the tool.

- On UNIX nodes, the tool runs under the user account that the agent itself is currently running under.

HPOM management console

If the target location for the tool is the HPOM management console, the tool runs as the user logged on to that console.

- User name: cannot specify
- Password: cannot specify

Related Topics:

- Security authentication module
- Allow tools to run on Windows 2003 server nodes
- Change the password for multiple tools
- Change the default user for commands

Security authentication module

The security authentication module (opcauth.dll) can be used to authenticate the login for a tool. This is automatically used for all tools that have a user name but no password specified. When you launch a tool and specify only the user, HPOM does not ask for a password or check for one.

The security authentication module is installed by default on a node as part of the HP Operations agent package.

Deploy the security authentication module to domain controllers

If the tool uses domain accounts, then you must configure the domain controllers of the target node’s domain as managed nodes and deploy the HP Operations agent package to the domain controllers. The agent installation makes the security authentication module available on the domain controllers.

If there are several domain controllers in a domain, and depending on the availability of these domain controllers, the Windows authentication system will contact different domain controllers. For this reason the security authentication module must be available on all domain controllers of that domain. The security authentication module on the domain controllers authenticates the login for any machine when the tool launches. However, in this case, the account that the tool runs as does not have network credentials.

NOTE:
You cannot run tools using domain accounts without passwords if none of the domain controllers are available.

If the tool uses local accounts, no further action is necessary because the security authentication module is automatically deployed together with the HP Operations agent package, which is required to execute tools on the managed node.

Exclude the security authentication module from the agent installation

To exclude the security authentication module from the agent installation:

- HTTPS agents

  In the HTTPS agent installation defaults file, set the `INSTALL_OPCAUTH` option to `false` , for example:

  ```
  [eaagt]
  INSTALL_OPCAUTH = false
  ```

  NOTE:
  When you install an HTTPS agent manually, the HTTPS agent installation defaults file is not used and therefore the security authentication module is not registered.

- DCE agents

  Use the `SetMgmtServer` tool with the `/auth /on` option, for example:

  ```
  SetMgmtServer /auth /on
  ```

Disable the security authentication module after the agent installation

To disable the security authentication module after the agent has been installed:

- HTTPS agents
  a. Manually remove the following registry entry:
     `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\MSV1_0\Auth130`
  b. Reboot the system after the key has been removed.
- DCE agents

  Manually remove the `opcauth.dll` file on the node where the tool user is authenticated:
  `%SYSTEMROOT%\opcauth.dll`

If you remove this DLL, then a password will always be required to run a tool under the specified account. If you remove this DLL on all domain controllers, you cannot do a switch user without a password for domain accounts.

Related Topics:

- Allow tools to run on Windows 2003 server nodes
- Configure HTTPS agent installation defaults
- Configure DCE agent installation defaults

Allow tools to run on Windows 2003 Server nodes

With Windows 2003 Server, Microsoft introduced a new security feature using a local security policy called "Allow log on locally." This logon right determines which users can interactively log on to a system. On the Windows 2003 server, the Domain Users group and the Domain Admins group are not included in this new policy.

Existing tools that ran correctly on Windows 2000 might fail on Windows 2003 Server. This means that a tool set up to run as a domain user will fail with the following error message:

```
Logon attempt for user (ov-excl\joec) failed. The account name may be unknown or the
password was wrong.
```

To allow tools to run on Windows 2003 Server nodes

The problem can be solved in one of the following ways:

- Add the domain user to the Windows 2003 local security policy "Allow log on locally."

  Go to **Start** → **Administrative Tools** → **Local Security Policy** . Then select **Local Policies** → **User Right Assignment** and right-click **Allow log on locally** to open the properties of the **Allow log on locally** setting.

- Add the user to the local Administrators group.

Change the password for multiple tools

As administrator, you may configure many tools to run under a particular account, with a password specified for security reasons. When it becomes necessary to change that password, you can use the Change Password utility to change the password for all the tools that use that account name. You can also change the name of the account the tool runs under. The utility is available from the HP Operations Manager Tools tools group in the Tools folder in the console tree.

To change the user name, password, or both for all instances of a tool

1. In the console tree, select the tool for which you want to change the user name, password, or both.
2. Open the Tool Configuration editor and be sure these settings are correct:

   Details pane: The Parameters box must contain the parameter $OPC_MGMTSV. When you run the tool, the name of the management server is supplied for this variable.

   Target pane: The Execute on: box must specify console.
3. From the console tree, select Tools → HP Operations Manager Tools . In the Details pane, select the tool Modify Tools login / password .
4. Double-click the tool name or right-click to open the shortcut menu and select All Tasks → Launch Tool to open the Update User / Password dialog box.
5. In the Update User / Password dialog box, enter the old user name and the new user name.
6. Enter the new password and confirm it.
7. Click Apply to confirm your changes.
8. Click OK to confirm your changes and close this dialog box.
9. To test the changed user name, return to the Tool Properties dialog box Target pane. The new user name should appear in the User Name box.

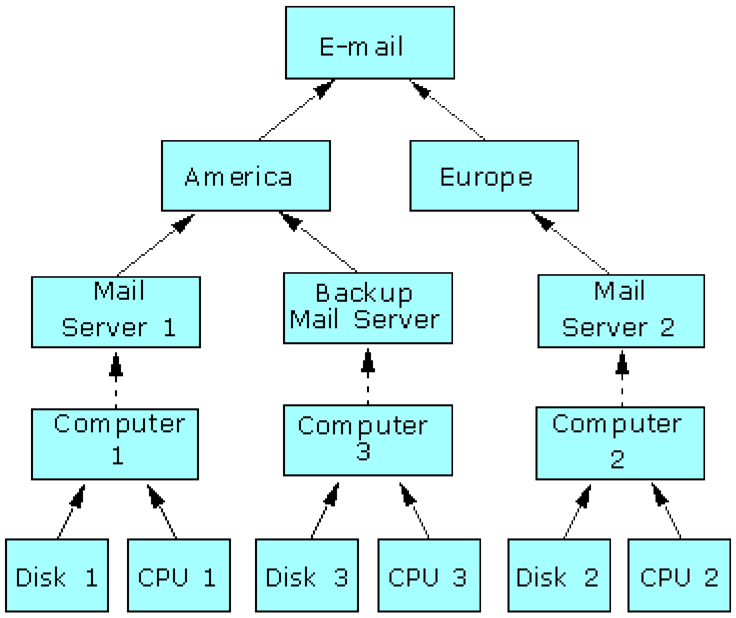
Introduction to the Service Editor

The Service Editor allows you to define how services in the service hierarchy are dependent on each other, and allows you to define rules that evaluate the severity based on the state of the contributing services. Although you can use the Service Editor to create an entire service hierarchy, it is recommended that you use the Discover Services command to create a basic hierarchy, which you can then fine tune. You can use the Service Editor to modify the status propagation so that it is tuned for your environment. You can also build on the SPI service hierarchy, using it as a building block in a larger hierarchy that you create.

1. Learn about the service hierarchy .
2. Learn how to plan a service hierarchy .
3. Learn about Service IDs .
4. Sketch out a plan for your service hierarchy that includes the high-level services that you want to monitor, for example, printing, e-mail, internet access and so on, noting the service hosting for each service.
5. Make sure that all nodes needed for the service hierarchy are configured managed nodes.
6. Execute the command Tools → Microsoft Windows → Discover Services .
7. Use the Service Editor to add folder-based services for all high-level services in your service hierarchy. Accept the default value propagation and calculation values. You can change these values later if you need to.
8. For each node, add a dependency from the Windows OS SPI hierarchy to all services that reside on that node.
9. Create folder-based services for any special applications or processes are monitored by your policies. Include these services in the service hierarchy. Ensure that the service IDs match those in the policies.

Service hierarchy overview

A service hierarchy is a logical organization of the services you provide, each higher level covering a wider or more general service area than the next lower level. The picture below shows a simple service hierarchy. The top level service is *E-mail* . This service is dependent on two lower level services, *America* and *Europe* , which in this case represent e-mail services on two continents. These services depend on other services, down to the lowest level services, in this case, the hard disks and CPUs of specific computers. This sample is small but a service hierarchy can include many more services and relationships. (Click the boxes and arrows below for more information about this service hierarchy example.)



Service Hierarchy Terms

The terms below are used in the help to describe the relationships between the various services in a service hierarchy.

- The terms superordinate and subordinate describe the hierarchical relationship between two services in the service hierarchy. A superordinate service depends on one or more subordinate services, and uses the statuses of its subordinates to calculate its own status. All services to which a service propagates its status are considered to be that service's superordinates. A service can be both superordinate and subordinate at the same time.
- Dependency describes the relationship between a superordinate service and a subordinate service. The superordinate *depends* on the services of the subordinate to function properly, and also *depends* on the status of the subordinate services to calculate its own status.

- A contained by link describes the relationship of a subservice which *cannot* exist without a superordinate service. In the example above, the service CPU 1 cannot exist if its superordinate service Computer 1 does not exist. If Computer 1 is removed from the service hierarchy, CPU 1 cannot exist in the service hierarchy.
- A used by link describes the relationship of a subservice which *can* exist without the superordinate service. In the example above, the service Computer 1 can exist if its superordinate service Mail Server 1 does not exist. If Mail Server 1 is removed from the service hierarchy, Computer 1 will remain in the service hierarchy.

Related Topics:
Planning your service hierarchy

Plan your service hierarchy

It is helpful to draft your service hierarchy before you start using the Service Editor. When planning your service hierarchy, keep the following questions in mind:

- Which IT services do you provide? Which ones do you want to monitor?
- Who are the customers of your services? Which organizations, departments, or lines of business?
- How can you logically group the services you provide? Which services are used by other services?
- How do problems in one service affect another? Which status propagation rule should you apply?
- How do you evaluate the severity of a problem? Which status calculation rule should you apply?
- Which tools should be assigned to each service?

Related Topics:

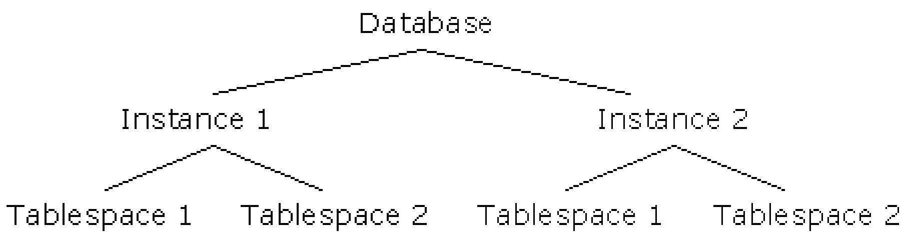
- Service Hierarchy
- Design effective service IDs

Design effective service IDs

Service IDs are unique identifiers or strings that you can choose freely when you define a component service. They are important because you use them when defining message attributes to indicate which messages match which services.

Note, however, that you don't have to create a new policy or rule for each service that you monitor. You can devise a structured naming schema for your services, and use HP Operations' predefined variables to construct the service IDs. This makes it possible to keep your policies generic, while still specifically identifying each individual service.

Consider the following example: your IT company is managing several database installations for different customers. You know that each database installation can have several instances, and that each instance has several tablespaces which you want to monitor. Your service hierarchy draft might look similar to this one:



When you know this general layout, you can begin think about creating service IDs. You will want to use the same HP Operations policy to monitor all the tablespaces, so you need to come up with a naming schema that allows you to reuse your policies, while still providing service IDs that are unique for every instance of the service. In order to do this, think of what makes each service unique and then compose the service ID with this information. In the example above, the customer name, the instance name, tablespace name, and the system name where the database is installed would uniquely identify each tablespace service. A service ID that contained exactly this information could look like this:

```
company.instance_name.tablespace_name.system.com
```

NOTE:
Service IDs can contain a maximum of 2048 characters. Service IDs cannot contain the following characters: ' (apostrophe), " (inch mark), \ (backslash), ` (grave accent), ´ (acute accent).

Although you could type this information directly into the service ID box for each policy, you would then need a different copy of the policy for each customer site. Instead of hardcoding this information, you can use the following methods to include this information in the Service ID.

HP Operations automatically includes the name of the system on which the message originated as a property of every Service ID, so it is not necessary for you to include this information in the Service ID. The company name, instance name and tablespace name of the particular service instance can be found by using variables. For example, if the policy was monitoring entries in a database log file, a log file like this might exist:

Sample log file entry:

```
Error Number: 110 tablespace_1 for instance_1 full in database Smith_Inc
Error Number: 110 tablespace_2 for instance_1 full in database Jones_Inc
```

In order to match Error Number 110 and to assign parts of this message to variables, you could type the following in the log file line box for the policy that monitors the log file:

```
^Error Number: 110 <instance_#.instance> for <tablespace_#.tablespace> full in database
<*.customer>
```

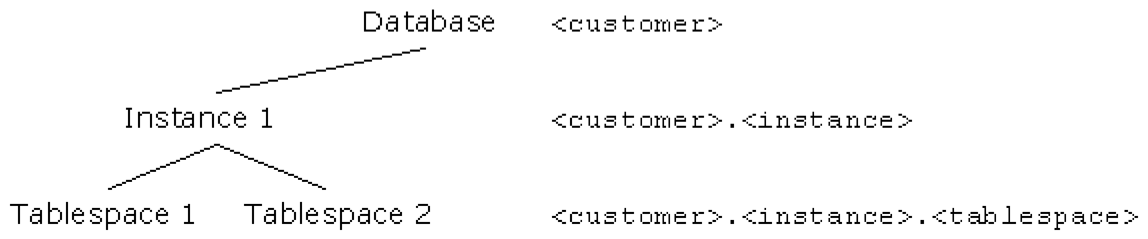
Then, in the Message Attributes tab, you would use the following entries in the Service ID and Hosted on boxes:

Service ID: `<customer>.<instance>.<tablespace>`
Hosted on: `<$MSG_NODE_NAME>`

For the first line of the sample log file, this would resolve to a service ID that looks like this:

```
Smith_Inc.instance_1.tablespace_1
```

This is the service ID that you would type in the service ID box for the component service that represents this tablespace for customer Smith Inc. The figure below shows a variable-based naming schema for all services in the example service hierarchy.



Related Topics:

- HP Operations Policy Variables
- Examples of pattern-matching in rule conditions
- Details of pattern-matching expressions
- User-defined Variables

Change status display mode

You can use the following methods to influence the display of status information in the console:

- Disable user-specific status calculation
- Enable operational status calculation

Disable user-specific status calculation

You can disable user-specific status calculation to ensure that all users see the same status for the services that are assigned to them. If user-specific status calculation is disabled, all users see the global service status, even if not all messages that contribute to that status are available to them.

A user's map view displays only the services that are assigned to the user through user roles. User roles also determine the nodes, and therefore the messages a user is responsible for. Because the status of the services in a user's map view is calculated based on the user's subordinate services and messages, it is user-specific. Another user with a different set of responsibilities may see a different status for the same services. If you want all users to see the same status for all assigned services, you can disable user-specific status calculation.

NOTE:
When you disable user-specific status calculation for services, you also change the way the status of node groups is calculated and displayed.

To disable user-specific status calculation

1. In the console tree, right-click Operations Manager , and then click Configure Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Status Engine . A list of values appears.
3. Set the value for Enable user-specific status calculation to False .
4. Click Apply , then OK to save your changes and close the dialog box.
5. Restart the `OvEpStatusEngine` service.

Enable operational status calculation

Use operational status calculation to see how a service hierarchy would look if all owned messages were acknowledged; in other words, if the problems were solved.

The status of a service is determined by the status of its subordinate services and by the status of the messages for that service. In general, when you acknowledge all messages for a service, the status of the service changes to reflect the status of its subordinate services.

You can change the role of messages in status calculation by enabling operational status calculation. In operational status calculation, the ownership state of messages determines how the messages influence status calculation, so that owned messages are considered acknowledged and do not affect the severity status of a service.

NOTE:
Enabling operational status calculation affects all consoles that connect to that management server. Console users may not notice that operational status calculation is enabled because there is no visual indication in the console, only the behavior changes.

To enable operational status calculation

1. In the console tree, right-click **Operations Manager** , and then click **Configure Server...** . The **Server Configuration** dialog box opens.
2. Click **Namespaces** , and then click **Status Engine** . A list of values appears.
3. Set the value for **Show operational status** to **Global** .
4. Click **Apply** , then **OK** to save your changes and close the dialog box.
5. Restart the `OvEpStatusEngine` service.

Related Topics:

- Introduction to the Service Editor

Service type overview

Service type is a property that defines the structure that a particular service hierarchy is allowed to have. For example, the service type *storage* , is allowed to contain the service type *disk* , but *disk* cannot contain *storage* . You can set the service type only when you are adding a new service. After the service is created, you cannot change it.

If you are editing a service hierarchy provided by a Smart Plug-in, you can choose one of several preconfigured service types when you add a new service. If you want to add services that do not fit the predefined types of a Smart Plug-in, or if you are creating your own service hierarchy, you can use the default service type Generic Service .

Related Topics:

- Service hierarchy overview

Add component service

Component services are the building blocks of your service hierarchy . They are the names and icons that you see in the service map view. In the service hierarchy example , DISK, CPU, Computer, Mail Server, and so on, are all service components.

1. Open the Service Editor
2. Select Add Service .
3. Select the Service type . Selections you make here are listed under Services in the Service Editor main dialog box.
4. If you want to see SPI service types in the Service Type list, check the box Show SPI Service Types .
5. Indicate the Service hosting. Service hosting refers to the managed node on which the service runs. The managed node you select for Hosted On is the node where service actually resides. Some services are abstractions, which do not actually exist on any node. Such services are Virtual Services . After the service hosting is set for a service, it cannot be changed.
6. Type a Display Name . This is the name that you will see in the service map view.
7. Type a Service ID , a string that will uniquely identify this service within the context of the hosting node. This string is the key that maps messages to services. All messages which should be used to calculate the severity of this service must contain this service ID as a part of the message attributes. You can use a GUID to ensure that number is unique, though since GUIDs are somewhat difficult to work with, you can also use any string that you know is unique. See Design effective Service IDs , to learn more about options for creating service IDs.

   NOTE:
   Service IDs can contain a maximum of 2048 characters. Service IDs cannot contain the following characters: ' (apostrophe), " (inch mark), \ (backslash), ` (grave accent), ´ (acute accent).

   If you select the Hosted On option, when HPOM creates the new service ID, the unique ID of the node the service is hosted on is appended to the name you have assigned. For example, if you give your new service the service ID of "service1.testlab" HPOM will add to that name the unique ID of the node hosting the service. You can see this by selecting the new service in the service map and clicking Properties . In the General tab, the Service ID box displays the name you assigned plus the unique ID for the node.

   CAUTION:
   Take care when assigning service IDs. After the ID is assigned, it cannot be changed. If you need to change a service ID, you must delete the component service and recreate it with the new ID.

8. Type a Description for this component service. The description is only visible when you edit the service.
9. Select Finish .

NOTE:
You can use cut-and-paste to move a component service if the new parent service type is identical to the old one.

Related Topics:

- Delete a service
- Add dependency

Select node for service hosting

If you indicate that a new service is hosted on some node, a selection window will appear. Select the node on which the service is hosted, and select OK . After the service hosting is set for a service, it cannot be changed.

Related Topics:

- Add component service

Configure general service properties

You can edit some basic properties of each service using this tab:

- You can change the Icon that is used to represent this service in the service map view, or you can click the Default button to refer to the icon assigned to the service type of the service.
- You can edit the Display Name . This is the name that is visible in the service map view.
- You can edit the Description for this component service. The description is only visible when you edit the service.

To configure general service properties:

1. Open the Service Editor
2. Select the service that you want to configure.
3. Select Properties .
4. Select General .

Related Topics:

- Add component service
- Custom icons for service types

Configure reports and graphs

Configure the reports and graphs you want to associate with a particular service using the Reports and Graphs tab of the Service dialog box. Settings you make here apply to the specific instance.

NOTE:
Associating a report with a service type does not create a new report, but allows you to launch a default report (usually the " All Systems" report).

To configure reports and graphs for service types

1. Open the Service Editor
2. Select the service for which you want to configure reports or graphs.
3. Select Properties .
4. Select the Reports and Graphs tab.
5. Click the Add button to open the Select Report Family or Category dialog box.
6. Select the report family and category you want to associate with this service and click OK to close the dialog box. The name of the family and category you selected appear in the Reports box.
7. Click the Add button to open the Select Graph Family or Category dialog box.
8. Select the graph family and category you want to associate with this service and click OK to close the dialog box. The name of the family and category you selected appear in the Graphs box.
9. In the Filter Value box, you can type a filter string if you want to filter all but a specific instance of the metric being graphed. For example, if you were graphing the metric BYCPU_ID, you could graph only CPU 1 with the filter value `BYCPU_ID = @@PARAMETER1` . Refer to the Metric Selection window of the Graph Configuration dialog box to see a list of valid metrics.
10. Click Apply to apply your changes without closing the dialog box.
11. Click OK to confirm your changes and close the service editor.

Related Topics:

- Configure Reports and Graphs for Service Types

Configure the status calculation

The status calculation refers to the calculation that is performed to determine the status that is assigned to a service. This status is calculated from the severity of the messages assigned to that service, and from the statuses of any subservices on which the service is dependent. For every status calculation you need to define:

- The Description of the rule is a short sentence that identifies the rule. The description is important, because it is what you will see in the rule list.
- Calculation rule

It is important to understand the principles of status calculation to understand how the propagation rules and weight factors that you select will affect the status of a service. See How is severity calculated for an interactive tutorial.

To configure Status Calculation

1. Open the Service Editor
2. Select the service that you want to configure.
3. Select Properties .
4. Select Status Calculation .
5. In the Rule name box, set the rule that the service you are editing uses when performing status calculation.

NOTE:
To edit a calculation rule or create a new rule, select Calculation Rules in the Service Configuration editor.

Related Topics:

- Configure general service properties
- Choose propagation rules

Configure calculation rules

The Calculation Rule lets you indicate the threshold value that is used to determine the status of a service. You can choose either most critical , single threshold , or multiple threshold . All three rules have the same functional principle: the highest level severity with a rating that crosses a threshold which you specify is the severity of the service.

- Most critical calculation rule
  In the case of most critical, the threshold for all severities is zero (0).
- Single threshold calculation rule
  Single threshold allows you to choose one threshold for all severities.

  NOTE:
  If you want, you can use the Set-to Value to set the severity to a specific level when the threshold is crossed, instead of accepting the severity that crossed the threshold.

- Multi-threshold calculation rules
  Multiple thresholds allow you to set a different threshold for each severity.

For the single and multi-threshold calculation rules, you can choose to perform the calculations as a percentage or a value. If you choose a percentage, then the threshold's values will be evaluated as percentages. A value threshold interprets the thresholds as integers. In most cases, a percentage threshold type is recommended because it is less rigid than a value threshold type, and therefore allows you to make changes in the number of subservices without having to change the calculation rule.

To configure calculation rules:

1. Open the Service Editor
2. Select the service you want to edit.
3. Select Properties .
4. Select Status Calculation .
5. Edit the calculation rule in the Calculation Rule functionality group.

Related Topics:
An introduction to status calculation

Configure superordinates

The services shown in the Superordinate tab are services which depend on the service you are editing. Here, you can change the way that the service you are editing propagates its status to each superordinate.

To configure superordinates:

1. Open the Service Editor
2. Select the service that you want to configure.
3. Select Properties .
4. Select Superordinates .
5. Double-click the superordinate service that you want to edit to view the General , Rule , and Association tabs.

   In the General tab, you can select the rule that will applied to the status of the service when it is included in the calculation of the superordinate.

   The Rule tab is read-only and displays the Propagation Rule that the service you are editing uses when propagating its status to a particular superordinate.

   In the Association tab, you can examine some of the superordinate and subordinate service properties.
6. By right-clicking in the Weight column, you can change the weight that the service you are editing has for each superordinate. By changing the weight you make the service more or less important than the other services that contribute to the superordinate. See the severity calculation tutorial for an interactive example of how weight affects the status calculation .

NOTE:
To edit a rule or create a new rule, select Propagation Rules in the Service Configuration editor.

Related Topics:

- Configure subordinates
- Propagation Association

General shared status propagation rule properties

In the General properties dialog for shared status propagation rule , you type the name of the rule, and a short description of the rule— both of which are visible in all status propagation dialog boxes.

Related Topics:

- Edit shared propagation rules

Choose propagation rules

The statuses of most services in the service hierarchy are calculated from the messages associated with the service and from the subservices on which the service is dependent. Status propagation refers to how a service represents its status to its superordinate services.

Rule Type
You can choose to create a Simple Rule , where the propagation is always the same, regardless of the status of the service, or a Severity Based Rule , where the propagation changes, depending on the status of the service.

Simple/Default Rule
You can create a Simple/Default Rule regardless of which rule type you choose. If the propagation rule type is Simple, then this rule will be used in all cases with two exceptions (noted below). Use a severity-based rule if you want to propagate normal as some other status. If the propagation rule type is Severity Based, then this rule will be used as the default for any severity levels for which you have not specified another rule. You can choose one of four Simple/Default Rules:

- Unchanged : Propagate the status with no change. For example, a status of Warning equals Warning. (This is a reasonable default value to use. If you are new to status calculation, consider starting with this value and editing it later, if necessary.)
- Ignore: The status of the subservice is not considered when calculating the status of the dependent service. This status propagation allows you to include a service in the service hierarchy without allowing it to influence the status calculation .
- Propagate Fixed As: The status of the subservice is always considered to have a certain status, regardless of the actual status. For example, with status propagation set to fixed:warning , a subservice with a status of minor is interpreted as warning. If the status of the subservice changes to critical, the status will still be interpreted as warning. Exception : the status normal is always propagated as normal and will not be change by this setting.
- Propagate Relative : To propagate the status at a fixed level higher or lower than what the status really is. For example, if you select INCREASE BY 2, then when the subservice has a status of warning, a status of major is propagated. Exception : the status normal is always propagated as normal and will not be change by this setting.

Severity Based Rule Propagation
Severity Based Rule Propagation allows you to override the Simple/Default rule for specific severities. For example, if your Simple rule is INCREASE BY 2, and the Severity Based Rule Propagation for all severities is set to use default , all severities (except normal) will be increased by two. If, however, you change the Severity Based Rule Propagation for Warning to Critical, then a severity of warning will be propagated as critical, normal will be propagated as normal and all others will be increased by two. Use default overrides the Simple/Default rule and allows the unchanged severity to be propagated.

To configure propagation rules:

1. Open the Service Editor
2. Select the service that you want to configure.
3. Select **Properties** .
4. Select **Superordinates** or **Subordinates** .
5. Double-click the service where you want to change the propagation rule.
6. Select **Rule** .

Related Topics:

- Configure the status calculation

Propagation Association

This tab provides an overview of the propagation between the current service and the selected superordinate service. For both services, the system displays service type , service id , display name , hosted on and description information. This summary provides information only; the properties cannot be changed in this tab.

To View the Propagation Association:

1. Open the Service Editor
2. Select the subservice where you want to view the propagation association.
3. Select Properties .
4. Select Superordinates
5. Double-click the service where you want to view the propagation association.
6. Select Association .

Related Topics:

- Edit shared propagation rules

Configure subordinates

The services shown in the Subordinate tab are services on which the service you are editing depend. Here, you can change the way that all subordinates propagate their status to the service you are editing.

Configure subordinates:

1. Open the Service Editor
2. Select the service that you want to configure.
3. Select Properties .
4. Select Subordinates .
5. Double-click the subordinate service that you want to edit to view the General and Rule tabs.

   In the General tab, you can select the rule that will applied to the status of the subservice when it is included in the calculation of the superordinate.

   The Rule tab is read-only and displays the Propagation Rule that the service you are editing uses when propagating its status to a particular superordinate.
6. By right-clicking in the Weight column, you can change the weight each subordinate is given when calculating the severity of the service you are editing. By changing the weight you make the subservice more or less important than the other subservices that contribute to the service you are editing. See the severity calculation tutorial for an interactive example of how weight affects the status calculation .

NOTE:
To edit a rule or create a new rule, select Propagation Rules in the Service Configuration editor.

Related Topics:

- Configure superordinates
- Propagation Association

Add tools

Assigning tools to services lets you run commands on the managed node that hosts the service. By assigning tools to services, you do not need to first locate the node in the node tree before running the command.

To add a tool:

1. Open the Service Editor
2. Select the service to which you want to assign a tool.
3. Select **Properties** .
4. Select **Tools** .
5. Select **Add** .
6. When the Tools window appears, select the tool groups, or specific tools that you want to assign to the service.
7. Select **OK** .

Related Topics:

- Configure Tools

View outage information

Use the Outage tab of the Service Editor to view outage information for a service. This read-only tab displays the settings established for both unplanned and scheduled outages.

A scheduled outage meets these criteria:

- Planned to happen
- Recurs at regular intervals for maintenance
- Configured by a policy

An unplanned outage is unexpected and can occur randomly.

Only administrators can put a service into maintenance mode. Services in maintenance mode by default do not affect the status of their parent services.

To view outage information for a selected service

1. Open the Service Editor
2. Select the service for which you want to view outage information.
3. Select Properties .
4. Select the Outage tab to view the following information:
   - Current Outage State:

     Displays the outage status of the selected service. This status will be either "ON " or "OFF".
   - Unplanned Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage . Messages can be either deleted or acknowledged.
   - Scheduled Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage . Messages can be either deleted or acknowledged.
5. Click OK .

Related Topics:

- Schedule an outage for a service
- Put services into unplanned outage mode

Add dependency

When you assign a service to be the subordinate of another service, you are adding a dependency. After adding a dependency, remember to check the status calculation and propagation rules of the two services to ensure that you get the status that you expect in the service view.

1. Open the Service Editor
2. Select the service which should get a new dependency (that is, the superordinate service).
3. Click Add Dependency . The Add Dependency Relationships dialog box appears.
4. Select the service on which the first service will be dependent (that is, the subordinates). You can select more than one service if you want to create more than one dependency.
5. Click OK .

NOTE:
You can also create dependencies with a drag-and-drop operation in the Service Editor.

Related Topics:

- Add component service

Create or edit shared calculation rules

A shared calculation rule can be defined once, and then used throughout the service hierarchy . While shared calculation rules can save you time, take care to use them only in identical situations, because changes to the rule affect all components that use the rule.

Create or edit a shared calculation rule:

1. Open the Service Editor
2. Select the Calculation Rules button.
3. Select the rule you want to edit.
4. Select New or Edit .
5. Enter the name of the shared calculation rule. This name is visible in all status calculation dialog boxes.
6. Enter the description of the shared calculation rule. This description is visible in all status calculation dialog boxes.
7. Enter the calculation rule .

Related Topics:

- Configure the status calculation
- Introduction to the Service Editor

Edit shared propagation rules

A shared propagation rule can be defined once, and then used throughout the service hierarchy . While shared propagation rules can save you time, take care to use them only in identical situations, because changes to the rule affect all dependencies that use the rule.

To edit shared propagation rules:

1. Open the Service Editor
2. Select the Propagation Rules button.
3. Select the rule you want to edit.
4. Select Edit .
5. Edit the name of the rule.
6. Edit the Description for the rule. This description is visible in all Status Propagation dialog boxes.
7. Edit the Propagation rule .

Related Topics:

- Choose propagation rules
- Introduction to the Service Editor

Service report

Use the tool Service Report to get information about the service hierarchy configuration on the management server .

To create and view a service hierarchy report:

1. In the console tree, select Tools , Reporting .
2. To view the entire service hierarchy, right-click General Service Report and select All Tasks → Launch Tool... . If you want to view only a portion of the service hierarchy, right-click Specialized Service Report , select All Tasks →Launch Tool... , and select the a portion of the hierarchy in the selection window that appears.
3. Wait for the xml report to be generated (this takes about 20 seconds, or longer, depending on the speed of your computer and the size of your managed environment). To display the report in the web browser, you must manually launch the tool "View Service Report." You can choose one of the following four views for the service data from the pull-down menu in the top-right corner of the browser:
   - View all services
   - View services by name
   - View services by caption
   - View selected service hierarchy

For information about additional ovconfreporter options, type `ovconfreporter -?` at a command prompt.

Edit a Service

When you edit a service, you can change the propagation rules of the service to its superordinates, or of any of this service's subordinates. You can also associate tools with the service or adjust the status calculation .

To edit a service

1. Open the Service Editor
2. Select the service that you want to edit.
3. Right-click to open the context menu
4. Select **Properties** .
5. Make the changes you want to make in one of these tabs. See the individual help topics for details on what can be edited and what is read-only information.
   - General
   - Status Calculation
   - Superordinates
   - Subordinates
   - Tools
6. After making the changes, select **OK** .

Related Topics:

- Introduction to the Service Editor

Delete a service

1. Open the Service Editor
2. Select the service that you want to delete.
3. Select the Delete button. Note that all component services contained by the service you delete are also deleted. Subordinates that are dependencies will remain.

Related Topics:

- Edit a Service
- Add component service

Configure Service Types

You can specify properties for service types, used when an instance of a service is created, using the Configure Service Types dialog box.

A service type is similar to a template; you associate a service type with specific reports, graphs, tools, and deployment packages. That service type is then used when an instance of the associated service is created. Any tools, reports, graphs, and deployment packages associated with the service type are associated with every instance of that service that has been or will be created. The service type assures that these properties are applied globally to all services of that type.

NOTE:
Associating a report with a service type does not create a new report, but allows you to launch a default report (usually the " All Systems" report).

To configure a service type, use the Properties dialog box. The title of this dialog box will change, depending on the service type you select in the Configure Service Types dialog box tree. If you select Application Services, for example, the Properties dialog box will be titled Application Services Properties .

Server or cluster failover behavior

HPOM for Windows monitors the availability of server components such as the Windows Management Interface (WMI), the server itself, and the cluster resource group (if your environment contains clusters.)

In the event of a failure of one or more of these components, some activities may be temporarily unavailable, until WMI or the server is restored. See Related Topics: for details.

Related Topics:

- Application services properties
- Configure General application services properties
- Configure reports and graphs for service types
- Configure tools for service types
- Configure deployment for service types
- Server or cluster failover behavior

Specify service type properties

Use the Properties dialog to associate icons, reports, graphs, tools, and deployment packages with a specific service type. This service type is used as a template during the creation of individual instances of the associated service. Properties you configured for the service type will be associated with each instance of the service.

To configure service types

1. Open the Service Type Configuration Editor if it is not already open.
2. In the Configure Service Types dialog box, select the service type you want to configure.
3. Click Properties to open the Properties dialog box.
4. Specify your settings in these tabs:
   - General
   - Reports and Graphs
   - Tools
   - Deployment
5. Click Apply to apply your changes without closing the dialog box.
6. Click OK to confirm your changes and close the dialog box.

Related Topics:
- Configure service types
- Configure general properties for service types
- Configure reports and graphs for service types
- Configure tools for service types
- Configure deployment for service types

Configure general information for service types

Configure general information you want to associate with a particular service type using the General tab of the Properties dialog box. Settings you make here apply to every service of this type.

A service type is used as a template for individual instances of an associated service. You can override the Icon property of a service type for a specific instance of a service usng the Configure Services editor.

To configure general information for service types

1. Open the Service Type Configuration Editor if it is not already open.
2. Select the General tab.
3. In the Icon box, enter the name of the icon that you want to represent this service type. Click Browse to open the Choose Icon dialog box, a standard Windows file selection dialog box that lists icon (.ico) files.
4. Select the icon you want to use and click Open . The icon you select here appears on the General tab. If you change to another directory to select an icon, you will see a message asking you to copy the icon to the proper directory. If you want the same icon to appear in another machine's installation of the console, it must be selected in the same manner on that machine. If the icon is not already present, it must be manually copied to that machine before you can select it. You can also create your own custom icon if desired.
5. In the Display Name box, enter the name you want to display for this service type.
6. In the Description box, enter information about the service type you are configuring. This information is optional.
7. Click Apply to apply your changes without closing the dialog box.
8. Click OK to confirm your changes and close the dialog box.

Related Topics:
- Application service properties
- Configure service types
- Configure reports and graphs for service types
- Configure tools for service types
- Configure deployment for service types

Configure reports and graphs for service types

Configure the reports and graphs you want to associate with a particular service type using the Reports and Graphs tab of the Properties dialog box. Settings you make here apply to every service of this type.

A service type is used as a template for individual instances of an associated service. The reports and graphs you specify for the service type will be available from all associated services. If you want to specify additional reports and graphs available for an individual instance of a service, you can use the Configure Services editor.

NOTE:
Associating a report with a service type does not create a new report, but allows you to launch a default report (usually the " All Systems" report).

To configure reports and graphs for service types

1. Open the Service Type Configuration Editor if it is not already open.
2. Select the Reports and Graphs tab.
3. Click the Add button to open the Select Report Family or Category dialog box.
4. Select the report family or category you want to associate with this service type and click OK to close the dialog box. If you selected a report category, the name of the family and category will appear in the Reports box. If you selected a report family, only the name of the family will appear.
5. Click the Add button to open the Select Graph Family or Category dialog box.
6. Select the graph family or category you want to associate with this service type and click OK to close the dialog box. If you selected a graph category, the name of the family and category will appear in the Graphs box. If you selected a graph family, only the name of the family will appear.
7. Click Apply to apply your changes without closing the dialog box.
8. Click OK to confirm your changes and close the dialog box.

Related Topics:
- Application service properties
- Configure General application services properties
- Configure tools for service types
- Configure deployment for service types

Configure tools for service types

Configure the tools you want to associate with a particular service type using the Tools tab of the Properties dialog box. Settings you make here apply to every service of this type.

A service type is used as a template for individual instances of an associated service. The tools you specify for the service type will be available from all associated services. If you want to specify additional tools available for an individual instance of a service, you can use the Configure Services editor.

To configure tools for service types

1. Open the Service Type Configuration Editor if it is not already open.
2. Select the Tools tab.
3. Click the Add button to open the Select Tools dialog box.
4. Select the tools you want to associate with this service type and click OK to close the dialog box. The names of the tools you selected appear in the Tools list of the Tools tab.
5. Click Apply to apply your changes without closing the dialog box.
6. Click OK to confirm your changes and close the dialog box.

Related Topics:

- Configure General application services properties
- Configure reports and graphs for service types
- Configure deployment for service types

Configure deployment for service types

Configure deployment of policy groups you want to associate with a particular service type using the Deployment tab of the Properties dialog box. Settings you make here apply to every service of this type.

A service type is used as a template for individual instances of an associated service. The policy groups you specify for the service type are those that are deployed when services of that type are created.

To configure deployment of policy groups for service types

1. Open the Service Type Configuration Editor if it is not already open.
2. Select the Deployment tab. Policies currently associated with this service type appear in the Policy Group list. Check the associated box if you want to automatically deploy the policy.
3. Click the Add button to open the Select Policy Groups dialog box.
4. Select the policy groups you want to associate with this service type and click OK to close the Select Policy Group dialog box. The names of the policy groups you selected appear in the Policy Groups box on the Deployment tab of the Properties dialog box.
5. To remove a policy group from the list, select the group name and click Remove .
6. Click Apply to apply your changes without closing the dialog box.
7. Click OK to confirm your changes and close this dialog box.

Related Topics:

- Application service properties
- Configure General application services properties
- Configure reports and graphs for service types
- Configure tools for service types
- Disable policy autodeployment

Select report family or category for service types

Specify the report family or category you want to associate with a particular service type using the Select Report Family or Category dialog box. The report family or category you choose will be available for all services associated with this service type.

To associate a report family or category with a service type

1. From the Reports and Graphs tab of the Properties dialog box, click Add to open the Select Report Family or Category dialog box.
2. From the list displayed, choose the family or category you want to associate with this service type.
3. Click OK to confirm your choices and close this dialog box.
4. If you selected a report category, the family and category appear in the Reports box of the Properties dialog box Reports and Graphs tab. If you selected a report family, only the family appears.
5. Click Apply to apply your changes without closing this dialog box.
6. Click OK to confirm your choices and close this dialog box.

Related Topics:
- Select graph family or category for service types
- Select tools for service types
- Select policy group for service types

Select graph family or category for service types

Specify the graph family or category you want to associate with a particular service type using the Select Graph Family or Category dialog box. The graph family or category you choose will be available for all services associated with this service type.

To associate a graph family or category with a service type

1. From the Reports and Graphs tab of the Properties dialog box, click Add to open the Select Graph Family or Category dialog box.
2. From the list displayed, choose the family or category you want to associate with this service type.
3. Click OK to confirm your choices and close this dialog box.
4. If you selected a graph category, the family and category appear in the Graphs box of the Properties dialog box Reports and Graphs tab. If you selected a graph family, only the family appears.
5. Click Apply to apply your changes without closing this dialog box.
6. Click OK to confirm your choices and close this dialog box.

Related Topics:
- Select report family or category for service types
- Select tools for service types
- Select policy group for service types

Select tools for service types

Specify the tools you want to associate with a particular service type using the Select Tools dialog box. The tools you choose will be available for all services associated with this service type.

To associate tools with a service type

1. From the Tools tab of the Properties dialog box, click Add to open the Select Tools dialog box.
2. From the list of tools displayed, choose those you want to associate with this service type. You can select multiple tools.
3. Click OK to confirm your choices and close this dialog box.
4. The tools you selected appear in the Tools box of the Properties dialog box Tools tab.
5. To remove a tool from the list, select the tool and click Remove .
6. Click Apply to apply your changes without closing this dialog box.
7. Click OK to confirm your choices and close this dialog box.

Related Topics:
- Select report family or category
- Select graph family or category
- Select policy group

Select policy group for service types

Specify the policy groups you want to associate with a particular service type using the Select Policy Groups dialog box. The policy groups you choose are deployed when services of this type are created.

To associate policy groups with a service type

1. From the Deployment tab of the Properties dialog box, click Add to open the Select Policy Groups dialog box.
2. From the list of policy groups displayed, choose those you want to associate with this service type. You can select multiple policy groups.
3. Click OK to confirm your choices and close this dialog box.
4. The policy groups you selected appear in the Policy Groups box of the Properties dialog box Deployment tab.
5. Click Apply to apply your changes without closing this dialog box.
6. Click OK to confirm your choices and close this dialog box.

Related Topics:

- Select report family or category for service types
- Select graph family or category for service types
- Select tools for service types

Custom icons for service types

If you prefer not to use the default icon provided for a service type, you can use your own custom icon.

To use a custom icon

1. Create your icon in two sizes. You must have a 32 x 32 pixel and a 16 x 16 pixel version of the image in the same `.ico` file.
2. Manually place this `.ico` file on every installed console. The location is:

   `%OvDataDir%\conf\OvOW\en\icons\Service`
3. If the console is already open, you must close and reopen the console to display the new icon. If the console was closed when you created the icon, the new icon displays immediately when you open the console.

Related Topics:

- Configure general information for application services

Configuring service auto-discovery

Service Discovery policies automatically populate the service map in the HP Operations Manager for Windows console, based on discovery rules executed within the managed environment. Discovered attributes may be hardware resources, operating system attributes, applications, and other information that can be retrieved from a managed object.

In HPOM for Windows, a service tree and rule set is defined by a management module. A service auto-discovery policy references a management module and includes user-defined parameters and an execution schedule, which can be configured. Predefined management modules are provided either as part of the HPOM installation or are available with Smart Plug-ins.

Service auto-discovery policies allow you to:

- Configure a predefined management module discovery
- Schedule the execution of the discovery by the management module
- Define additional parameters if required

You must deploy a service auto-discovery policy to a managed node before it can be executed.

To configure service auto-discovery

1. In the console tree, select Policy Management → Policies grouped by type → Service Auto-Discovery .
2. Right-click to open the shortcut menu. Select New → Policy to open the Service Auto-discovery editor. The Discover tab displays by default.
   a. In the Discover tab, specify the management module and service type definition you want to configure.
   b. Use the Schedule tab to specify the time, date, and frequency of discovery.
   c. Check the status line at the bottom of the policy editor for messages about your policy configuration.
   d. When configuration is complete, click Save and Close to save the changes to the policy and close this policy editor.

For more information about auto-discovery, see the help topics Configure Deployment for Service Types and Node Group Deployment Properties .

Related Topics:

- About Management Modules

- Configure Discovery
- Configure Schedule

About management modules

Management modules are predefined sets of discovery rules and actions that allow distributed discovery. An executed management module, using a service auto-discovery policy , automatically creates instances of service type definitions in the service map. An instance of a service type is an actual service visible in the service map.

A management module is stored on the HP Operations management server. By itself it is not deployed or active. To be executed, any management module has to be referenced and started by a service auto-discovery policy.

HPOM does not allow you to create custom management modules or to define new service types. Using the Service Type editor, you can make changes to certain attributes of a given service type. For example, you might assign graphs and reports and define auto-deployment attributes. However, you cannot add a new service type to existing management modules.

Related Topics:

- Using a Service Auto-discovery Policy
- Configuring Service Auto-discovery

Using a service auto-discovery policy

Service auto-discovery policies are supplied by Smart Plug-ins (SPIs) to discover services in your managed environment and display them on a service map.

Before you can execute a service auto-discovery policy, you must deploy it to a managed node, either manually or through auto-deployment. See the specific SPI documentation for details on auto-deployment, which varies somewhat from SPI to SPI.

Usually there is no need to modify the SPI auto-discovery policy. However, certain SPIs may require that you configure this policy by adding parameter data such as a user name or password to allow access and discovery of specific applications on a managed node.

Undesired services in the service map that were discovered can be manually removed. These services will not be rediscovered unless a change in the environment is detected. However, you can start commands that force agents to resend service discovery data. When the management server receives the service discovery data, it recreates the service map, including any services that you previously removed. You can also use these commands for troubleshooting if services fail to appear on the management server. You can start these commands on nodes using the following tools:

- Tools → HP OpenView Tools → Resend service discovery data (Windows agents)
- Tools → HP OpenView Tools → Resend service discovery data (UNIX agents)

To disable automatic service discovery, remove service auto-discovery policies from the Policy Groups that are configured for auto-deployment.

Related Topics:

- Configure deployment for service types
- Node group Deployment properties
- Applying tools to managed nodes, services, and messages
- ovagtrep

Configure discovery

Use the Discover tab in the Service Auto-Discovery policy editor to specify the management module that contains the service type definition that you want to configure.

1. In the console tree, select Policy Management → Policies grouped by type → Service Auto-Discovery .
2. Right-click to open the shortcut menu. Select New → Policy to open the Service Auto-discovery editor. The Discover tab displays by default.
3. From the Management Modules list, select a module name. Service type definitions associated with the selected module display in the Service Type Definitions box.
4. Select the service type definition you want to configure. Use the Up and Down arrows to navigate through the listed service type definitions. The arrows expand the definitions that can be edited and skip over those that contain no editable data.
5. Any editable parameters associated with the selected service type definition display in the User Editable Parameters box.
6. Select a parameter and type in the field beside it to change the parameter.

To continue to configure service auto-discovery, select the Schedule tab.

Configure schedule

Use the Schedule tab of the Service Auto-Discovery policy editor to specify a discovery schedule for the current policy. You can specify:

- Frequency (hourly, daily, weekly, monthly, once, or specially)
- Time of day (hour, minute)
- Day of the week

The contents of the Schedule tab will change with your selections. For example, if you select the Daily option, the Time Reoccurrence dialog box becomes available, allowing you to set the exact time that auto-discovery should occur and the hourly intervals (if any) at which it should recur. In all cases, your choices are summarized in the Schedule Summary at the bottom of the screen. Options include:

Option	Parameters
Schedule Task	Choose every hour, daily, weekly, monthly, once, or specially. ■ Every hour: Specify the minute within the hour that the task should begin. ■ Daily: Specify the specific time of day the task should run and the hours it should recur, if any. ■ Weekly: Specify the day of the week, the specific time of day that the task should run, and the hours it should recur, if any. ■ Monthly: Specify the day of the month, the specific time of day that the task should run, and the hours it should recur, if any. ■ Once: Specify a particular day and date and the day and date the task should recur, if any. ■ Specially: This option provides the most choice. You can specify months, days of the month, weeks, days of the week, and hours the task should recur, if any. By default, the command runs at the top of every hour. To change the default, you can set the day of the week, the day of the month, and the month: • To set the parameter: Click in the appropriate day, date, or month interval to select it or drag the cursor over multiple intervals and click Set . A blue bar indicates the selection has been set. • To delete the setting: Right-click in the timeline or interval and click Clear or select Clear All from the shortcut menu.

Configure time reoccurrence

Use the Time Reoccurrence dialog box to specify any multiple intervals during which auto-discovery should run. You can configure the task to run around the clock or at particular specified hours. You can specify multiple intervals for tasks that are scheduled to run Daily, Weekly, Monthly, and Once.

To set multiple time intervals

1. Click Times in the Schedule tab of the Service Auto-Discovery dialog box to open the Time Reoccurrence dialog box. By default, all the intervals are selected, which means that the command runs every hour.
2. To specify the minute of the specified hour when the command should be run, click in one of the intervals in the On specified minute display. You can select only one minute during the hour. A blue bar indicates your choice.
3. Select Clear all to clear the On specified hours default settings.
4. To set one or more hours for the task to be run, click in one or more hour intervals or drag the cursor over multiple intervals. A blue bar indicates your choice. Click Set all to select all the hours.
5. To delete one or more hourly settings, click in the interval or timeline. Click Clear All to delete all the settings.
6. Click Cancel to close this dialog box without saving your changes.
7. Click OK to save your changes and close this dialog box.

Configuring user roles

As an administrator, you can configure an operator's view of the environment to focus on specific assigned tasks and responsibilities. By defining roles for specified users, you control the operator's view of your enterprise and the range of activities which that user has permission to perform. By assigning users to well-defined, specific roles, you can distribute monitoring and maintenance tasks across a group of individuals with their own particular areas of expertise and experience and customize each operator's console view.

User roles can include both administrative and operator tasks and allow operators to focus on their primary tasks without the distraction of information that is not relevant to their assigned responsibilities. Only those items for which the operator has permissions are visible in that operator's console view.

You can configure roles in two ways in HPOM:

- Preconfigured user roles: HPOM provides a number of preconfigured user roles which you can use as they are or edit to change tasks and permissions allowed for specified users. The list of preconfigured roles appears when you open the User Roles configuration editor. To edit, click Properties and edit the information.
- Customized user roles: To create your own customized user roles, open the User Roles configuration editor and click New to open the New User Role dialog box. Specify the user or users for this role and define the tasks and permissions associated with it, then save your changes.

To specify permission for:

- Policies and Packages: Specify which users can read, deploy, edit, or delete policies and packages. You might give one user role permission to view policies and packages and give another user role permission to edit and delete policies or packages. You can specify any combination of allowable actions.
- Services, Nodes, and Tools: Specify which services, nodes, and tools will be available to specific user roles. Only those services, nodes, and tools will be visible in the console for users assigned to that user role. You can specify any combination of services, nodes, and tools.
- Messages: Specify which message groups will be available to operators. This setting controls which messages are visible and which actions operators can perform on a selected message. You can also specify whether users can view messages that are not in assigned message groups. Message operations for which you can set permissions include:
  - View
  - Own/Disown
  - Acknowledge/Unacknowledge
  - Change severity
  - Change text

- Assign
- Relaunch automatic command
- Launch operator-initiated command

- Users: Select users or groups of users from the HP-OVE-OPERATORS local group to add to this role. You can also delete users from this user role.

Related Topics:

- Assigning user roles
- Configure a new user role

Assign user roles

User roles can contain both administrative and operator tasks, but they are assigned to operators only. An operator may be assigned to more than one user role. Administrators can access all operator and administrative functions.

Before operators can be assigned a user role, they must belong to the user group HP-OVE-OPERATORS. If a user is assigned to the HP-OVE-ADMINS groups, that user has unrestricted administrative access.

NOTE:
As an administrator, you can remove yourself from any assigned user roles. This allows access to all operational views and tasks.

Operators who belong to the HP-OVE-OPERATORS group but have not been assigned any user roles can access all service, node, and tool instances and any associated messages. However, you can change this behavior by setting the value Specify operator lockdown in the User Roles namespace in the Server Configuration dialog box. When this value is set to true, an operator who is not assigned to any user role cannot view or do anything.

Operators cannot access to any administrative tasks, such as service, node, tool, user role, or policy configuration and deployment. Administrative tasks are explained in detail in the "Administering your Environment" section of the help.

To assign an operator to a user role, use the User Roles configuration editor .

Configure a new user role or group

Use the User Roles configuration editor to define the services, nodes, tools, and messages you want to assign to a particular user role. You then assign a user, several users, or Windows group to this role. The settings you specify for the user role determine what the assigned users can see in their console views and which tools are available to them.

Operators assigned to a role are able to view and operate on the instances of services, nodes, and tools you assign to their roles. For example, all service components and dependencies below the Exchange service that you select can be seen in the console Scope pane and map. All messages pertaining to those services appear in the message browser.

If you modify a user role or assign or delete an operator while that operator is working in the console, your changes do not take effect until the operator's console has been closed and restarted.

To configure a new user role

1. From the Scope pane, select a service, node, or tool.
2. Right-click to open the shortcut menu.
3. Select Configure → User Roles to open the User Roles dialog box.
4. Click New to open the New User Role dialog box.

   Alternatively, select an existing user role and click Duplicate to create a copy for you to edit.
5. Use the tabs to configure:
   - General information about the user role, such as name and description.
   - Services that will be visible to operators assigned to this user role.
   - Nodes that will be visible to operators assigned to this user role and can be acted upon.
   - Tools that will be visible to operators assigned to this user role.
   - Messages defined for message groups for this user role and permissions allowed for these messages.
   - Policies defined for this user role, permissions specified for these categories, and policies and packages that contain these categories.
   - Users , Windows groups, or Active Directory groups to be added or deleted from this user role.
6. Click Apply to see the effects of your changes.
7. Click OK to confirm your changes and close this dialog box.

8. Click Close to close the User Roles dialog box and configuration editor.

Configure general information for user roles

Each user role you create must be identified by specifying properties for that user role. Use the General tab in the New User Role dialog box to name and describe this user role.

To configure general information for user roles

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the User Role Properties dialog box. The General tab displays by default.
3. In the Role Name box, type a unique name for this user role. This information is required.
4. In the Description box, type an optional description of the user role you are creating.
5. Specify the reports and graphs viewing rights for this role by selecting Reports can be viewed , Graphs can be viewed , or both.
6. Select Change unplanned outage state to allow the user to change the unplanned outage state of a node or service.
7. Select Create message filters to allow the user to create personal message filters.
8. Click Apply to apply your changes.
9. Click OK to confirm your choices and close this dialog box.
10. Select the Services tab to continue configuring this user role.

Configure services for user roles

Use the Services tab in the New User Role dialog box to specify the services you want to associate with this user role. Operators assigned to this user role will be able to view only those services you specify here.

Assigning different operators to a different service or group of services has several benefits:

- Focuses the operator view on a particular area of responsibility
- Restricts access to information that is not needed by that operator

To configure services for user roles

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box. The General tab displays by default.
3. Click the Services tab.
4. From the tree view of services, select one or more services that you want to associate with this user role. When you select a check box, the selection automatically includes *all* the service's superordinates and subordinates. You can then clear check boxes for individual subordinates if necessary. You cannot, however, select a service without its superordinates.

   NOTE:
   Operators cannot view dependent services in the map unless you include those services in their user role.

5. Click Apply to apply your changes.
6. Click the Nodes tab to continue configuring this user role.

Configure nodes for user roles

Use the Nodes tab in the New User Role dialog box to specify the managed node groups you want to associate with this user role. Operators assigned to this user role can view and operate on only those managed nodes contained in the node group that you specify here.

When you assign responsibilities to operators for a specific group of nodes, your operators can develop in-depth knowledge of a specialized area. For example, you might designate one user role to manage Windows nodes and another to manage UNIX nodes, or assign nodes hosting various database servers to different user roles.

To configure nodes for user roles

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box. The General tab displays by default.
3. Select the Nodes tab.
4. From the tree view of nodes, select one or more node groups that you want to associate with this user role. When you select a check box, everything below that entry is also selected. The selection includes the nodes as well as the node groups of the selected node group. It is not possible to select a parent node group without the node groups it contains.
5. Click Apply to apply your changes.
6. Click OK to confirm your choices and close this dialog box.
7. Select the Tools tab to continue configuring this user role.

Configure tools for user roles

Use the Tools tab in the New User Role dialog box to specify the tools you want to associate with this user role. Operators assigned to this user role will be able to apply only those tools you specify here.

Any tool contained in the tool group that you assign to a role can be launched from the scope pane, map, or message browser. Tools associated with a service or node, but not found in an operator's assigned role are disabled.

You cannot launch actions on services not assigned to this user role. If Service A and Service B are defined in a role, and Service A is dependent on Service B, and Service B is dependent on Service C, the actions contained in the assigned tools can be launched on Services A and B but not on Service C (the service on which A and B depend.)

To configure tools for user roles

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box. The General tab displays by default.
3. Select the Tools tab.
4. From the tree view of tools, select one or more tool groups that you want to associate with this user role. When you select a check box, everything below that entry is also selected. The selection includes the tools as well as the tool groups of the selected parent tool group. It is not possible to select a parent tool group without the tool groups it contains.
5. Click Apply to apply your changes.
6. Click OK to confirm your choices and close this dialog box.
7. Select the Messages tab to continue configuring this user role.

Configure messages for user roles

Use the Messages tab in the New User Role dialog box to specify the message groups you want to associate with this user role. Operators you assign to this role can view and perform permitted operations on messages that contain the message groups you assign to this role.

Operators can view messages from message groups not assigned to their user role, but cannot perform message operations on them. You can restrict an operator's view to just those message groups specified for this user role if you prefer.

Different user roles can permit access to the same message group but have different message operation permissions assigned to the message group. For example, one role could allow an operator to own, disown, and launch operator-initiated actions for all messages for a particular message group, but not allow the operator to acknowledge, change severity, or launch automatic actions again. Another operator could be assigned a different user role that permits all message operations for the same message group. If an operator is in two message groups, with two different permissions, the user is given the permissions and not restricted.

A default message group will be assigned to each role and cannot be removed from the list of message groups in the Messages tab. This default group allows you to grant permission to perform operations on messages that do not belong to any of the message groups assigned to this user role.

To configure messages for user roles

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box. The General tab displays by default.
3. Select the Messages tab.
4. Type the name of the message group you are creating in the Enter message group box or select from the list, which shows the available values for this message group. Click Add to add the message group to the Message Groups assigned this Role box. Message filters in the console and web console limit the message group length to 32 characters (the same length limit of the field in messages).
5. To set permissions for this message group, select the message group name. The Remove and Permissions buttons become available.
6. Click Permissions to open the Permitted Operations for Msg Group dialog box. The title of the dialog box reflects the name of the message group you selected.
7. Specify the permissions you want to associate with this message group and click OK to confirm your choices and close the Permitted Operations for Msg Group dialog box.

8. Click Apply to apply your changes.
9. Select the Policies tab to continue configuring this user role.

Set permitted operations for message groups

When you configure a message group for a user role, you must specify which operations your operators are permitted to perform on messages that belong to this group. You can specify one or more of these operations for this message group:

- View
- Own
- Disown
- Acknowledge
- Unacknowledge
- Change Severity
- Change Text
- Assign
- Launch operator-initiated command
- Relaunch automatic command

To specify permitted operations for this message group

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box.
3. Select the Messages tab.
4. Enter the message group name or select an available message group from the list and click Add to add the name to the Message Groups assigned this Role box.
5. Select the message group name and click Permissions to open the Permitted Operations for Msg Group dialog box.
6. Select the operations you want to associate with this message group.
7. Click OK to confirm your choices, close this dialog box, and return to the New User Role dialog box. Available operations are indicated by abbreviations across the top of the Message Groups assigned this Role box. The operations you associated with this message group are indicated by Xs beneath the abbreviations.
8. Click Apply to apply your changes.
9. Click OK to confirm your choices and close this dialog box.

Related Topics:

- Configure messages for user roles

Configure policies for user roles

Use the Policies tab in the New User Role dialog box to specify the permitted operations you want to associate with the policies and packages containing categories assigned to this role. A category is set in the policy editor for each particular policy type. Operators you assign to this role can view specified policies and perform permitted operations on them, such as configuring or modifying the policies.

A default policy category will be assigned to each role and cannot be removed from the list of policy categories in the Policies tab. This default group allows you to grant permission to perform administrative tasks on policies that do not contain any of the categories assigned to this user role. Default rights apply unless different rights have been assigned to a particular object.

In addition, you can also assign general administrative rights to a user role. These are independent of policy categories. Think of these as global rights that you can set for a particular user role.

You can use these permissions to provide levels of security by assigning different rights to different operators so that powerful features are not controlled by one user.

To configure policies for user roles

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box. The General tab displays by default.
3. Select the Policies tab.
4. Type or select the name of the policy category you are assigning to this user role in the Enter Category box.
5. Click Add to add the policy category to the Categories assigned to this Role box.
6. To set permissions for this policy category, select the policy category name. The Remove and Permissions buttons become available.
7. Click Permissions to open the Policy Permissions for Category dialog box. The title of the dialog box reflects the name of the policy category you selected.
8. Specify the permissions you want to associate with this policy category and click OK to confirm your choices, close the Policy Permissions for Category dialog box, and return to the Policies tab in the New User Role dialog box.

   Available categories are indicated by abbreviations across the top of the Categories assigned to this Role box. The permissions you associated with this policy category are indicated by Xs beneath the abbreviations.
9. Specify any general administrative rights you want to assign to this user role:

- View Policy Management: You must select this check box in order for operators to be able to view the policy management selections in the scope pane. Unless you set this check box, operators will see nothing relating to policy management in the console.
- Administer Policies and Packages: The operator can configure the Policy Management server. This includes operations such as add/remove packages/policy types, deployment of packages and instrumentation, reinstall all, and uninstall all.
- Ignore Policy Owner: The operator can deploy policies even if a version of these policies has already been deployed by another management server.
- Policy Group Handling: The operator can create, rename, delete, copy, and move policy groups. The operator can also assign and unassign policies to and from groups.
- Job Handling: The operator can start and suspend deployment jobs.

10. Click Apply to apply your changes.
11. Select the Users tab to continue configuring this user role.

Related Topics:

- Set permissions for policy categories
- User roles and levels of security

Set permissions for policy categories

When you configure a policy category for a user role, you must specify which operations your operators are permitted to perform on policies that belong to this category. You can specify one or more of these operations for this policy category:

- Read: The operator can view the policy stream, but cannot modify or create policies.
- Deploy: The operator can deploy and undeploy policies and policy groups.
- Edit: The operator can create and modify policies.
- Delete: The operator can delete policy versions.

To specify permitted operations for this policy category

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box.
3. Select the Policies tab.
4. Enter the policy category name and click Add to add the name to the Enter category box.
5. Select the policy category name and click Permissions to open the Policy Permissions for Category dialog box.
6. Select the permissions you want to associate with this policy category by selecting the check box for the item.
7. Click OK to confirm your choices, close this dialog box, and return to the New User Role dialog box Policies tab.

You can also specify general administrative rights for user roles which are independent of policy categories. See Configure policies for user roles for details.

Related Topics:

- Configure policies for user roles

Configure users and groups for user roles

Use the Users tab in the New User Role dialog box to specify the users and groups that you want to associate with this user role, and to remove users and groups from this role. Operators assigned to this user role can view only those services, nodes, tools, messages, and policies you configure and can perform only the operations permitted for this role.

To configure users and groups for a user role

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box. The General tab displays by default.
3. Select the Users tab.
4. Click Add to open the Select Users or Groups dialog box, in which you associate users and groups with the user role you are configuring.
5. In the Select Users or Groups dialog box, use the Add button to add users or groups to this role. Use the Members button to add members of groups to this role.
6. Complete your settings in the Select Users or Groups dialog box and click OK to close the dialog box. The users or groups you have added appear in the list of users in the New User Role dialog box.
7. Click Apply to apply your changes.
8. Click OK to confirm your choices and close the New User Role dialog box.
9. The name of the new user role you have created appears in the User Roles dialog in the Name box.

Select users or groups for user roles

After you have defined a user role by specifying the services, nodes, tools, message groups, and policy categories that you want to associate with this user role, you then assign users and groups to the role. Operators belonging to the assigned group see the services, nodes, tools, messages, and policies that you specified and can carry out the tasks you have given them permission to perform.

The only users and groups displayed in this dialog box are those found in the HP-OVE-OPERATORS group on the management server. Users and groups must be added to the HP-OVE-OPERATORS group before they are available here.

A group can be a local Windows group account or Windows domain group account. When you change the membership in the group, the user role is automatically updated to reflect this change.

To select users or groups for user roles

1. Open the User Roles dialog box if it is not already open.
2. Click New to open the New User Role dialog box.
3. Select the Users tab.
4. Click Add to open the Select Users or Groups dialog box, in which you associate users or groups with the user role you are configuring.
5. The names of the users and groups belonging to this domain display in the Name box:
   - To add individual users or groups, select the name of a user or group and click Add . The selected name is added to the list at the bottom of the dialog box.
   - To add members of groups, select the name of a group and click Members .

     The Select Users dialog box opens, which lists all the members of the group you selected. Select users from the list and click Add to add these users to the list at the bottom of the Select Users or Groups dialog box.
6. Click OK to confirm your choices and close the Select Users or Groups dialog box. The users or groups you have added appear in the list of users in the Users tab of the New User Role dialog box.
7. Click Apply to apply your changes.
8. Click OK to confirm your choices and close the New User Role dialog box.
9. The name of the new user role you have created appears in the User Roles dialog box in the Name box.

Select users from groups

You can select individual members of groups to add to a user role.

To select users from a group

1. Select a group from the Name list in the Select Users or Groups dialog box.
2. Click Members to open the Select Users dialog box and display a list of the group members.
3. Select one or more members of the group as users in the user role you are configuring.
4. Click Add to add the selected names to the list of selected users in the Select Users or Groups dialog box.

Specify operator lockdown

Operators who belong to the HP-OVE-OPERATORS group but have not been assigned any user roles can access all service, node, and tool instances and any associated messages. You can block these operators access to all services, node and tool instances and any associated messages displayed in the console.

To specify operator lockdown

1. In the console tree, right-click Operations Manager , and then click Configure → Server . The Server Configuration dialog box opens.
2. Click Namespaces , and then click User Roles .
3. Set the value Specify operator lockdown .

   When set to True , the console is locked. Members of the HP-OVE-OPERATORS group without assigned user roles will be unable to see any services, node, and tool instances and any associated messages in the console.

   When set to False , the lockdown is disabled. Operators without assigned user roles can still see everything in the console.
4. Click Apply .

User roles and levels of security

As the administrator who assigns user rights to various operators, you will want to spend some time determining a strategy for making these assignments. By carefully diversifying these rights among your operators, you can be sure that the power to make key changes is not concentrated in just a few individuals.

For example, you might give one user permission to edit policies and a different user permission to deploy policies. By distributing the editing and deployment rights between two users, you make it difficult for an unscrupulous user to create and deploy a harmful policy.

For even greater security, you could give user A edit rights, user B group rights, and user C deployment rights. This is more secure because anyone who has group rights can change a group so that it is an auto-deploy group, thus circumventing the lack of deployment permission.

Additionally, the operator lockdown feature allows you to prevent users from accessing any console functionality.

The table shows three user roles. Users 1 and 2 are assigned to role 1. User 3 is assigned to role 2, and user 4 is assigned to role 3.

- Role 1 users can:
  - Deploy policies of category a
  - Read, create, modify, and delete policies of category b
  - Read and deploy all other policies
  - View Policy Management
  - Perform Policy Group and Job handling
- The Role 2 user can:
  - Read
  - View Policy Management
- The Role 3 user cannot view policy management at all.

Role	User	Category	Category-based Rights	Global Rights
Role 1	User 1	a	Deploy	View Policy Management, Job Handling, Group Handling
	User 2	b	Edit, Delete, Read	View Policy Management, Job Handling, Group Handling
		Default	Read, Deploy	View Policy Management, Job Handling, Group Handling
Role 2	User 3	Default	Read	View Policy Management
Role 3	User 4	Default	/	/

NOTE:
The predefined Default denotes all categories for which no special rights have been defined.

Monitor policy types

HP Operations provides a number of policy types that allow you to configure what is monitored and what the response to a specific situation should be. This section of the help describes the policy types that monitor specific events, such as entries in a log file, WMI entries, threshold values, SNMP traps, and so on. The different event policy types have many things in common, and so are treated a group in the help.

Note that there are other policy types that are explained elsewhere in the help.

Choose a policy type

A policy is a set of configuration information that helps to automate network and system administration. Using HPOM, administrators can deploy policies on various computers to provide consistent, automated administration across a network.

Every policy belongs to one of several policy types . A policy type is a set of rules that defines what a policy can do. An understanding of the policy types helps you choose the right policy for the management problem that you want to solve. The currently available policy types can be grouped into the following categories:

- Monitor policy types
- Node configuration policy types
- Server configuration policy types

After you have decided what kind of policy you want to write, see Quick Start: How to create a policy to learn how to create and deploy the policy.

Monitor policy types

Logfile Entry policy type

This policy type monitors entries in a text or binary log file and responds when text that you choose appears in the log file. Choose this policy type if you want to monitor entries in a log file.

Measurement Threshold policy type

This policy type evaluates performance data and responds if the data does not remain within acceptable levels. Choose this policy type if you want to monitor parameters that are constantly changing, such as CPU load, disk space, number of running processes, and so on.

Open Message Interface policy type

This policy type monitors and responds to messages that are sent from the HP Operations process opcmsg . This process allows scripts or programs to generate HP Operations messages and send them directly to the management server . Use this policy if you want to filter these messages before they are sent to the management server.

Process-monitor policy type

This policy type monitors services and policies which are running on the managed node and sends a message when the state of the service or the process changes. You can define the status to monitor and the action to take should the status change. Choose this policy type if you want to monitor the status of Windows services or processes, which are running on the managed node.

SNMP Interceptor policy type

This policy type monitors SNMP events, and responds when a character pattern that you choose is found in an SNMP message. Choose this policy type if you want to monitor network components that send SNMP messages

Windows Event Log policy type
: This policy type monitors entries in a Windows event log and responds when a character pattern that you choose appears in the event log. Choose this policy type if want to monitor entries in a Windows event log.

Windows Management Interface policy type
: This policy type monitors the properties of WMI classes and instances, and responds when a property matches a value you select, or when an instance you select is created. Choose this policy type if you want to monitor objects that provide WMI information.

Windows Service Monitor policy type
: This policy type monitors the Windows services and processes which are running on the managed node and sends a message when the state of the service or the process changes. You can define the status to monitor and the action to take should the status change. Choose this policy type if you want to monitor the status of Windows services or processes, which are running on the managed node.

Node configuration policy types

ConfigFile policy type
: This policy type provides a consistent way to configure managed node instrumentation. It is used by Smart Plug-ins (SPIs) to enhance the management capabilities for specific enterprise applications like SAP and Microsoft Exchange. ConfigFile policies indirectly perform monitoring through another instrumentation component.

Flexible Management policy type
: This policy type is used to configure agent-based flexible management scenarios on the managed node. You can configure the node to send messages to different management servers based on the type of problem, or send messages to different servers based on the time of day.

NOTE:
You can deploy only one policy of this type to a node .

Node Info policy type
: This policy type allows you to configure some aspects of agent behavior, for example, buffer sizes, IP addresses, and port numbers for client-server communication.

Scheduled Task policy type
: This policy type allows you to schedule commands to run on managed nodes, and will send a message to the management server to indicate the success or failure of the command. Use this policy if you want to run commands on one or more managed nodes—either once or according to a specific schedule.

Service Auto-Discovery policy type
: Discovery policies are supplied by Smart Plug-ins (SPIs) to discover services in your managed

environment and display them on a service map. Usually there is no need to modify the SPI auto-discovery policy. However, certain SPIs may require that you configure this policy by adding parameter data such as a user name or password to allow access and discovery of specific applications on a managed node.

Server configuration policy types

Remote Action Security policy type

By default, any node can send a message with a remote automatic action, and the management server runs that action on the remote node. However, you can configure a management server to allow or deny remote automatic action requests. You do this using a remote action security policy. Remote Action Security policies are deployed to the management server.

Server-based Flexible Management policy type

Server-based flexible management enables you to forward messages, message operations, and action responses from one management server to another. Server-based flexible management policies are deployed to the management server.

Server-based MSI policy type

The server-based Message Stream Interface (MSI) enables programs external to the HPOM message and action server to read and change incoming messages before they are stored in the HPOM database. Server-based MSI policies are deployed to the management server.

Related Topics:

- Quick start: how to create a policy

Create a new policy

You can create a new policy in **Agent policies grouped by type**, **Server policies grouped by type** or in **Policy groups**. Remember that if you create a new policy in one of your policy groups, the policy will also be visible under the appropriate policy type.

To create a new policy (View alternate method)

1. In the console tree, under **Agent policies grouped by type** or **Server policies grouped by type**, right-click the type of policy that you want to create.
2. Select **New** → **Policy**.
3. When the policy editor for that policy type appears, set up the policy.
4. Select **File** → **Save** to save the policy.

NOTE:
It is often easier to edit an existing policy and save it under a new name, than to create an entirely new policy.

Related Topics:
- Save policy as
- Quick start: How to create a policy

Policy objects and examples

The objects listed here are available for each policy and can be manipulated with Visual Basic Scripting Edition or with Perl. These policy objects can only be used in scripts that run within a policy. They cannot be used in standalone scripts that are executed from the command line.

CAUTION:
Policy scripts provide administrators with a powerful tool to evaluate and manipulate data. If, however, a script is incorrectly written, it could cause the agent to fail. Hewlett-Packard Company is not responsible for agent failures resulting from incorrectly written scripts.

Quick Navigation

Policy	Select a method
Source	Select a method
Session	Select a method
Rule	Select a method
ConsoleMessage	Select a method
ExecuteCommand	Select a method

The Policy object

This object is used to access the attributes of a policy.

Policy Method :	Source
Parameter :	*name* (The Short name indicated in the threshold source properties .)
Return Type :	VB Script: IDispatch object of type "Source" (This is the default method for the Policy object.) Perl: source object
VB Script Syntax :	`Policy.Source("`*name*` ")`
Perl Syntax :	`$Policy->Source("`*name*` ");`
Description :	Returns the source object for the defined source and metric. Measurement type sources must use a separate source for each metric.

NOTE:
To improve performance, assign the source object to a variable instead of using the Source method every time it is needed.

Policy Method :	Name
Parameter :	void
Return Type :	VB Script: BSTR Perl: string
VB Script Syntax :	`Policy.Name()`
Perl Syntax :	`$Policy->Name();`
Description :	Returns the name of the policy that started the script.

Policy Method :	CreateObject
Parameter :	*progID* (string of format: `[Vendor.]Component[.Version]`)
Return Type :	VB Script: IDispatch Perl: not applicable
VB Script Syntax :	`Policy.CreateObject("`*progID*` ")`
Perl Syntax :	not applicable
Description :	Creates a component instance of a COM object. Note that this method is valid only on Windows nodes, and cannot be used in a Perl script.

Policy Method :	SourceEx
Parameter :	*expression* (See Description, below, for valid expressions.)

Return Type :	VB Script: IDispatch object of type "Source" Perl: source object
VB Script Syntax :	`Policy.SourceEx("`*expression*` ")`
Perl Syntax :	`$Policy->SourceEx("`*expression*` ");`
Description :	Returns the source object instance of the source defined by the expression. This source object is identical to the object returned by the Policy.Source method, but because it does not have to be configured in the policy, it can be used for scheduled tasks, as well as for measurement threshold policies. The expression can have the following format depending on which component the performance metric will be collected from:

- `NTPERFMON\\Object\\Counter\\Instance`
  Access a perflib metric (not supported on UNIX nodes). Object, Counter, and Instance are strings as specified in the current monitor configuration for NT performance monitors. Example: `NTPERFMON\\Process\\Elapsed Time\\*`
- `SNMP\\object id[\\hostname]`
  Perform an SNMP get on the specified object id (OID). By default, the collection will be done on the managed node but can be elsewhere if the optional hostname is given. For SNMP, the method will have to wait until the value is returned which might take some time. Example: `SNMP\\.1.3.6.1.2.1.1.7.0\\onion.veg.com`
- `PROGRAM\\command[\\monname]`
  Run the specified command or script for gathering the monitored value. The command or script must at some point run the opcmon command to return the value associated with the monitor. If no monitor name is specified, then the default DynPROGRAM must be used. For example, to specify the monitor mymonname: opcmon `mymonname=value` ; to specify the default, opcmon `DynPROGRAM=value` . Examples:
  `PROGRAM\\opcmon DynPROGRAM=12`
  `PROGRAM\\opcmon testmon=25\\testmon`
- `EXTERNAL[\\monname]`
  Wait for a value returned by the execution of the opcmon command. This is similar to the PROGRAM expression but a command is not directly carried out. An external command previously triggered by the ExecuteCommand object must provide the monitor value. The default value is DynEXTERNAL (opcmon DynExternal=10) Examples:
  `EXTERNAL`
  `EXTERNAL\\testmon`
- `WBEM\\namespace\\class name\\property name`
  WMI interface (not supported on UNIX nodes). Get access to WBEM values. Namespace, class name and property name are strings as specified in the current monitor configuration for WBEM. Example:
  `WBEM\\ROOT\CIMV2\\Win32_PerfRawData_PerfDisk_LogicalDisk\\DiskReadBytesPersec`
- `CODA\\data source\\collection\\metric name`
  Query a metric from the embedded performance component. Data source, collection and metric name are strings as specified in the monitor configuration for the embedded

performance component. Currently if the data source is empty, the string Coda will be used. Example: `CODA\\\\CPU\\BYCPU_CPU_TOTAL_UTIL`

You can view a of list of available metrics in the *HP Performance Agent Dictionary of Operating System Performance Metrics* which is available at http://ovweb.external.hp.com/lpe/doc_serv/ . (Select the product Performance Agent , the required version, OS, and language.)

NOTE:
In Perl, the backslash character '\' is an escape code. A backslash is only introduced in a string when preceded by another backslash. Because of this, tokens in expressions need to be separated by quadruple backslashes '\\\\'. Example for Perl: `my $TestSource = $Policy->SourceEx("PROGRAM\\\\/tmp/script.sh\\\\testmon");`

Policy Method :	SourceExTimeout
Parameter :	*seconds* (integer)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`Policy.SourceExTimeout(`*seconds* `)`
Perl Syntax :	`$Policy->SourceExTimeout(`*seconds* `);`
Description :	Specifies the maximum amount of time, in seconds, the SourceEx and SourceCollection methods will wait before a value is returned. Default is 30 seconds.

Policy Method :	Execute
Parameter :	*command* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`Policy.Execute("`*command* `")`
Perl Syntax :	`$Policy->Execute("`*command* `");`
Description :	Run the specified command asynchronously. The command is executed in the context of agent security, so could be run as Local System or any other user-selected user to run the agent. The method will return immediately. See the ExecuteCommand method Command for more information about how to indicate commands.

Policy Method :	Output
Parameter :	*string*
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`Policy.Output("`*string*` ")`
Perl Syntax :	`$Policy->Output("`*string*` ");`
Description :	Appends the string to the annotation field of the message sent to the message browser in response to the success or failure of a scheduled task. This method is valid only for scheduled task policies.

Policy Method :	ExecuteEx
Parameter :	*command* (string)
Return Type :	VB Script: BSTR Perl: string
VB Script Syntax :	`Policy.ExecuteEx("`*command*` ")`
Perl Syntax :	`$Policy->ExecuteEx("`*command*` ");`
Description :	Run the specified command synchronously and wait for it to complete before returning the output of the command. The command is executed in the context of agent security, so could be run as Local System or any other user-selected user to run the agent. If the command is successful, STDOUT is returned. If the command is not successful (return value non-zero), the string "ERROR:\n" followed by STDERR will be returned. Note that you must either use complete paths or ensure that any needed path is included in the PATH variable. Example: dir_con = Policy.ExecuteEx ("cmd /c dir c:\")

Policy Method :	StoreCollection
Parameters :	▪ *expression* : (An embedded performance component metric in the format: `CODA\\data source\\collection\\metric name[\\category]`) ▪ *sourceobj* : (Any valid source object)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`Policy.StoreCollection("`*expression*`", `*sourceobj*` )`
Perl Syntax :	`$Policy->StoreCollection("`*expression*`", `*sourceobj*` );`
Category Type :	Describes available category types , such as DELTA, GAUGE, and COUNTER.
Description :	Stores the source object into the embedded performance component data source identified by the expression. Example: `Policy.StoreCollection "CODA\\DBSPI\\TABLE\\SPACE",Source`

Policy Method :	SourceCollection
Parameters :	▪ *expression* : An embedded performance component metric in the format: `CODA\\data source\\collection\\metric name` . ▪ *rangeofseconds* : The number of seconds for which metrics should be returned. ▪ *endtime* : End time for *rangeofseconds* . The format of time is of type DATE for VB Script or a string (format DD/MM/YYYY HH:MM:SS) for Perl. The date is optional.
Return Type :	VB Script: IDispatch object of type "Source" Perl: source object
VB Script Syntax :	`Policy.SourceCollection ("`*expression*`", `*rangeofseconds*` , `*endtime*` )`
Perl Syntax :	`$Policy->SourceCollection ("`*expression*`", `*rangeofseconds*` , `*endtime*` );`
Description :	Returns the source object containing all values collected by the specified embedded performance component metric. For each instance, all metrics collected between the expression "*endtime - rangeofseconds* " and "*rangeofseconds* " will be returned. If *endtime* is 0 (NULL for Perl) it is evaluated with the current time. Example: `Policy.SourceCollection ("CODA\\\\CPU\\BYCPU_CPU_TOTAL_UTIL",300,0)` The number of seconds specified should usually be less than 3600 (one hour), since retrieving a large number of values takes time and consumes resources.

The Source object

The source object is used to access the current values of the metrics. The source object instances can be created by any method that returns the source object.

Source Method :	Value
Parameter :	void
Return Type :	VB Script: variant (This is the default method for the Source object.) Perl: string
VB Script Syntax :	`Sourceobj.Value()`
Perl Syntax :	`$Sourceobj->Value();`
Description :	Current instance value if the option *Process each instance separately* is selected in the Processing options .

Source Method :	Name
Parameter :	void
Return Type :	VB Script: BSTR Perl: string
VB Script Syntax :	`Sourceobj.Name()`
Perl Syntax :	`$Sourceobj->Name();`
Description :	Returns the name of the current instance if option *Process each instance separately* is selected in the Processing options of the measurement threshold policy.

Source Method :	InstanceCount
Parameter :	void
Return Type :	VB Script: Int Perl: integer
VB Script Syntax :	`Sourceobj.InstanceCount()`
Perl Syntax :	`$Sourceobj->InstanceCount();`
Description :	Returns the number of instances that the source has.

Source Method :	`Count`
Parameter :	void
Return Type :	VB Script: Int Perl: integer
VB Script Syntax :	`Sourceobj.Count()`
Perl Syntax :	`$Sourceobj->Count();`
Description :	Same as InstanceCount. This parameter exits to provide compatibility with HP Operations ManageX .

Source Method :	`Item`
Parameter :	*index*
Return Type :	VB Script: IDispatch object of type "Source" Perl: source object
VB Script Syntax :	`Sourceobj.Item(`*index* `)`
Perl Syntax :	`$Sourceobj->Item(`*index* `);`
Description :	Access to the instance defined by the index. The index is a number from 0 to InstanceCount - 1. The returned source object can be extracted using the Value and Name methods. This parameter exits to provide compatibility with HP Operations ManageX .

Source Method :	`ValueOf`
Parameter :	*index* (integer)
Return Type :	VB Script: variant Perl: string
VB Script Syntax :	`Sourceobj.ValueOf(`*index* `)`
Perl Syntax :	`$Sourceobj->ValueOf(`*index* `);`
Description :	Direct access to the value of the instance defined by the index. This method is useful for looping over all instances, if the option *Process all instances once* is defined. The index is a number from 0 to InstanceCount - 1.

Source Method :	NameOf
Parameter :	*index* (integer)
Return Type :	VB Script: BSTR Perl: string
VB Script Syntax :	`Sourceobj.NameOf(`*index* `)`
Perl Syntax :	`$Sourceobj->NameOf(`*index* `);`
Description :	Direct access to the name of the instance defined by the index. The index is a number from 0 to InstanceCount - 1. This method is useful for looping over all instances, if the option *Process all instances once* is selected in the Processing options .

Source Method :	Top
Parameter :	*number*
Return Type :	VB Script: IDispatch object of type "Source" Perl: source object
VB Script Syntax :	`Sourceobj.Top(`*number* `)`
Perl Syntax :	`$Sourceobj->Top(`*number* `);`
Description :	Returns a new source object instance that contains only the instances with the <number> highest values. For example, if these three instances exist: `c: = 90%` ; `d = 80%` ; `e = 40%` then Sourceobj.Top(2) returns `c:` and `d:` .

Source Method :	Bottom
Parameter :	*number*
Return Type :	VB Script: IDispatch object of type "Source" Perl: source object
VB Script Syntax :	`Sourceobj.Bottom(`*number* `)`
Perl Syntax :	`$Sourceobj->Bottom(`*number* `);`
Description :	Returns a new source object instance that contains only the instances with the <number> lowest values. For example, if these three instances exist: `c: = 90%` ; `d = 80%` ; `e = 40%` then Sourceobj.Bottom(2) will return `d:` and `e:` .

Source Method :	Exclude
Parameter :	*namepattern* , *valuepattern*
Return Type :	VB Script: IDispatch object of type "Source" Perl: source object
VB Script Syntax :	`Sourceobj.Exclude("`*namepattern*`", "`*valuepattern*`")`
Perl Syntax :	`$Sourceobj->Exclude("`*namepattern*`", "`*valuepattern*`");`
Description :	Returns a new source object instance excluding values specified by the patterns. You can specify two parameters, one for the name of the variable (type, object, and instance) and one for the value. Specify NULL if no matching is required for one argument. Patterns should be valid HP Operations pattern-matching expressions .

Source Method :	Include
Parameter :	*namepattern* , *valuepattern*
Return Type :	VB Script: IDispatch object of type "Source" Perl: source object
VB Script Syntax :	`Sourceobj.Include("`*namepattern*`", "`*valuepattern*`")`
Perl Syntax :	`$Sourceobj->Include("`*namepattern*`", "`*valuepattern*`");`
Description :	Returns a new source object instance including only values specified by the patterns. You can specify two parameters, one for the name of the variable (type, object, and instance) and one for the value. Specify NULL if no matching is required for one argument. Patterns should be valid HP Operations pattern-matching expressions .

Source Method :	Time
Parameter :	void
Return Type :	VB Script: DATE Perl: string (format: `DD/MM/YYYY HH:MM:SS`)
VB Script Syntax :	`Sourceobj.Time()`
Perl Syntax :	`$Sourceobj->Time();`
Description :	Returns the time when the expression was evaluated.

Source Method :	`TimeOf`
Parameter :	index (integer)
Return Type :	VB Script: DATE Perl: string (format: `DD/MM/YYYY HH:MM:SS`)
VB Script Syntax :	`Source.TimeOf(`*index* `)`
Perl Syntax :	`$Sourceobj->TimeOf(`*index* `);`
Description :	Returns the time when the expression was evaluated for a specific instance. The index is a number from 0 to InstanceCount - 1.

Source Method :	`Add`
Parameter :	*instancename* , *value*
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`Sourceobj.Add "`*instancename* `:",`*value*
Perl Syntax :	`$Sourceobj->Add("`*instancename* `:",`*value* `);`
Category Type :	Describes available category types , such as DELTA, GAUGE, and COUNTER.
Description :	Adds the instance name to the source object and sets the value. If this instance is already part of the source object, the new instance will not be added and the value will be replaced. This method can be used on a newly created object or an object retrieved from any method returning a source object. This method is used to store data into the embedded performance component.

VB Script example:

```
set Sourceobj = Policy.CreateObject("Ito.OvEpScriptMetric")
Sourceobj.Add "a:",10
Sourceobj.Add "b:",25
Policy.StoreCollection "CODA\\floppy\\disk\\space\\\\gauge ",Sourceobj
```

Perl example:

```
my $Sourceobj = new Source;
$Sourceobj->Add("a:",10);
$Sourceobj->Add("b:",25);
$Policy->StoreCollection("CODA\\\\floppy\\\\disk\\\\space\\\\gauge
",$Sourceobj);
```

Source Method :	DataAvailable
Parameter :	void
Return Type :	VB Script: Boolean Perl: integer
VB Script Syntax :	`Sourceobj.DataAvailable`
Perl Syntax :	`$Sourceobj->Sourceobj.DataAvailable;`
Description :	Returns TRUE if the source object contains any value, otherwise, returns FALSE.

Source Method :	ValueOfInstance
Parameter :	*instancename*
Return Type :	VB Script: variant Perl: string
VB Script Syntax :	`Sourceobj.ValueOfInstance("`*instancename*` ")`
Perl Syntax :	`$Sourceobj->ValueOfInstance("`*instancename*` ");`
Description :	Direct access to the value of the instance defined by the instance name.

The Session object

The Session object can be used to store data and to access it later within the script running at a different interval. The session object can also be used to transfer data from the script to the message attributes by using the variable <$SESSION(KEY)>. The Session object is unique for each policy.

Session Method :	IsPresent
Parameter :	*key*
Return Type :	VB Script: Boolean Perl: integer
VB Script Syntax :	`Session.IsPresent("`*key*` ")`
Perl Syntax :	`$Session->IsPresent("`*key*` ");`
Description :	Returns TRUE if a value for *key* exists. Returns FALSE if no value for *key* exists. Keys are set with the Session.Value method.

Session Method :	Remove
Parameter :	*key*
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`Session.Remove("`*key*` ")`
Perl Syntax :	`$Session->Remove("`*key*` ");`
Description :	Removes the key specified from the session object.

Session Method :	RemoveAll
Parameter :	void
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`Session.RemoveAll()`
Perl Syntax :	`$Session->RemoveAll();`
Description :	Removes all keys from the session object.

Session Method :	Value
Parameter :	*key* *value* (for Perl only)
Return Type :	VB Script: variant (This is the default method for the Session object.) Perl: string

VB Script Syntax :	for put: `Session.Value("`*key*` ")=value`
	for get: `value=Session.Value("`*key*` ")`

Perl Syntax :	for put: `$Session->Value("`*key*` ","`*value*` ");`
	for get: `Value = $Session->Value("`*key*` ");`

Description :	Gets or puts a value for the defined key.

The Rule object:

The Rule object is used to indicate to the policy whether a threshold has been crossed or not. TRUE = threshold crossed, FALSE = threshold not crossed.

Rule Method :	Status
Parameter :	void
Return Type :	VB Script: Boolean Perl: integer

VB Script Syntax :	for put: `Rule.Status = `*`boolvalue`*
	for get: boolvalue = `Rule.Status`

Perl Syntax :	for put: `$Rule.Status(`*`boolvalue`*` );`
	for get: `boolvalue = $Rule.Status();`

Description :	Puts or gets the value for threshold status. For scheduled task policies, FALSE indicates that the scheduled task failed.

The ConsoleMessage object:

The ConsoleMessage object provides a method for sending messages directly to the message browser. Messages sent in this way will not be intercepted by an open message interface policy, but instead will be sent directly to the management server (message will go to MSI, if configured). The specified message will be sent to the message agent. Multiple uses of the Send method are supported. The same script can then send multiple messages to the HP Operations Manager console depending on which problem it detects.

NOTE:
HP Operations policy variables cannot be used by the ConsoleMessage object.

ConsoleMessage Method :	Application
Parameter :	*application* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.Application = "`*`application`*` "`
Perl Syntax :	`$ConsoleMessage->Application("`*`application`*` ");`
Description :	This optional method sets the content of Application in the general message properties of the message sent to the browser.

ConsoleMessage Method :	Object
Parameter :	*object* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.Object = "`*object*` "`
Perl Syntax :	`$ConsoleMessage->Object("`*object*` ");`
Description :	This optional method sets the content of Object in the general message properties of the message sent to the browser.

ConsoleMessage Method :	MsgText
Parameter :	*msgtext* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.MsgText = "`*msgtext*` "`
Perl Syntax :	`$ConsoleMessage->MsgText("`*msgtext*` ");`
Description :	This method sets the message text for the message that is sent.

ConsoleMessage Method :	Severity
Parameter :	*severity* (valid strings are: Unknown|Normal|Warning|Minor|Major|Critical)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.Severity = "`*severity*` "`
Perl Syntax :	`$ConsoleMessage->Severity("`*severity*` ");`
Description :	Sets the severity of the message that is sent. If not specifically set with this method, the default is Normal. If an invalid string is supplied, severity Unknown will be used.

ConsoleMessage Method :	MsgGrp
Parameter :	*messagegroup* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.MsgGrp = "`*messagegroup*` "`
Perl Syntax :	`$ConsoleMessage->MsgGrp("`*messagegroup*` ");`
Description :	Sets the value for the Message Group in general message properties of the message sent to the browser. If this method does not supply a value, Misc is used.

ConsoleMessage Method :	Node
Parameter :	*nodename* (IP address or fully qualified hostname)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.Node = "`*nodename*` "`
Perl Syntax :	`$ConsoleMessage->Node("`*nodename*` ");`
Description :	Sets the value for Primary Node Name that will be displayed in the general message properties of the message sent to the browser. IP addresses and fully qualified hostnames are valid. If this method does not supply a value, the hostname of the managed node is used by default.

ConsoleMessage Method :	ServiceId
Parameter :	*serviceid* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.ServiceId = "`*serviceid*` "`
Perl Syntax :	`$ConsoleMessage->ServiceId("`*serviceid*` ");`
Description :	This optional method sets the Service ID for the message.

ConsoleMessage Method :	MessageType
Parameter :	*messagetype* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.MessageType = "`*messagetype* `"`
Perl Syntax :	`$ConsoleMessage->MessageType("`*messagetype* `");`
Description :	This optional method sets the value for the message type field of the general message properties of the message sent to the browser.

ConsoleMessage Method :	MessageKey
Parameter :	*messagekey* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.MessageKey = "`*messagekey* `"`
Perl Syntax :	`$ConsoleMessage->MessageKey("`*messagekey* `");`
Description :	This optional methods sets a key for message correlation.

ConsoleMessage Method :	AcknowledgeMessageKey
Parameter :	*messagekey* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.AcknowledgeMessageKey = "`*messagekey* `"`
Perl Syntax :	`$ConsoleMessage->AcknowledgeMessageKey("`*messagekey* `");`
Description :	This optional method sets the message key to indicate which messages are automatically acknowledged in the browser. See Configure message correlation for more information about acknowledging messages with message keys.

ConsoleMessage Method :	`TroubleTicket`
Parameter :	*Booleanvalue*
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.TroubleTicket =` *Booleanvalue*
Perl Syntax :	`$ConsoleMessage->TroubleTicket(`*Booleanvalue* `);`
Description :	This optional method specifies if the message is to be sent to a trouble ticket interface. Default is FALSE.

ConsoleMessage Method :	`Notification`
Parameter :	*Booleanvalue*
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.Notification =` *Booleanvalue*
Perl Syntax :	`$ConsoleMessage->Notification(`*Booleanvalue* `);`
Description :	This optional method specifies if the message is sent to the notification mechanism. Default is FALSE.

ConsoleMessage Method :	`AgentMSI`
Parameter :	*type* (valid strings are: copy|divert|none)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.AgentMSI = "`*type* `"`
Perl Syntax :	`$ConsoleMessage->AgentMSI("`*type* `");`
Description :	This optional method specifies if the message is to be sent through the message stream interface on the agent. Default (or if string misspelled) is none. See Send messages to the message stream interface for more information.

ConsoleMessage Method :	ServerMSI
Parameter :	*type* (valid strings are: copy|divert|none)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.ServerMSI = "`*type*` "`
Perl Syntax :	`$ConsoleMessage->ServerMSI("`*type*` ");`
Description :	This optional method specifies if message is sent through the message stream interface on the server. Default (or if string misspelled) is none. See Send messages to the message stream interface for more information.

ConsoleMessage Method :	Send
Parameter :	void
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ConsoleMessage.Send()`
Perl Syntax :	`$ConsoleMessage->Send();`
Description :	This method sends the message to the management server. The MsgText method must set the message text before using this method. Multiple uses of the Send method are supported. HP Operations variables will not be expanded.

The ExecuteCommand object

Object used for requesting a command to be run. It starts a command to be run by the HP Operations agent.

ExecuteCommand Method :	Command
Parameter :	*command* (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ExecuteCommand.Command = "`*command*` "`
Perl Syntax :	`$ExecuteCommand->Command("`*command*` ");`
Description :	This mandatory method is the name of the command to run with all necessary parameters. NOTE: For scripts that will run on Windows nodes, internal commands such as Copy, Rename, and DIR use a command interpreter that must be started before the command can be run. For commands of this type, the command must be preceded with `cmd /k` , followed by any other parameters required.

ExecuteCommand Method :	KillonTimeout
Parameter :	*seconds* (integer)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ExecuteCommand.KillonTimeout = `*seconds*` ;`
Perl Syntax :	`$ExecuteCommand->KillonTimeout(`*seconds*` );`
Description :	This method sets the maximum time, in seconds, that the command will run. The default is unlimited. Valid only with the StartEx method.

ExecuteCommand Method :	UserName
Parameter :	username (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ExecuteCommand.UserName = "`*username*` "`
Perl Syntax :	`$ExecuteCommand->UserName("`*username*` ");`
Description :	User name under which the command should be run. Optional, default is $AGENT_USER.

ExecuteCommand Method :	Password
Parameter :	password (string)
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ExecuteCommand.Password = "`*password*` "`
Perl Syntax :	`$ExecuteCommand->Password("`*password*` ");`
Description :	Password for accessing the specified user account. To prevent the password from being visible in the script, use the following instructions: 1. Open a command prompt. 2. Change directory to the agent install directory. By default, this directory is: \Program Files\HP\HP BTO Software\Installed Packages\{790C06B4-844E-11D2-972B-080009EF8C2A}\bin\OpC\install 3. Encrypt your password with the command: opcpwcrpt **yourpassword** 4. Use the output string as the password in your script. In some cases it is better not to supply a password. See Should I provide the password? in the scheduled task topic for more information.

ExecuteCommand Method :	Start
Parameter :	void
Return Type :	VB Script: void Perl: void
VB Script Syntax :	`ExecuteCommand.Start()`
Perl Syntax :	`$ExecuteCommand->Start();`
Description :	Run the command specified by ExecuteCommand.Command and return immediately the control to the script so the next lines can be processed right away.

ExecuteCommand Method :	StartEx
Parameter :	void
Return Type :	VB Script: BSTR Perl: String
VB Script Syntax :	`ExecuteCommand.StartEx`
Perl Syntax :	`$ExecuteCommand->StartEx();`
Description :	Run the command ExecuteCommand.Command and wait until it finishes. If the command is successful, STDOUT is returned. If the command is unsuccessful (return value non-zero) the string "ERROR:\n" followed by STDERR is returned. Commands can be run synchronously or asynchronously, as needed. Multiple uses of the Start method are supported. This way, the same script can trigger multiple external commands.

Related Topics:

- Writing scripts for measurement threshold policies
- Quick start: how to create a policy
- Set general threshold rule properties

Category Types

Category Types	Meaning
UNDEFINED	Ignored
NOTAPPLICABLE	Ignored
KEY	Columns that uniquely identify instances of an object.
ATTRIBUTE	Static definitions or values, such as the OS name, version, release, physical memory, and CPU clock speed.
DELTA	Show the activity during the last interval, such as intervalized counts,rates, and utilizations.
GAUGE	Numeric value that shows the current use or value at the time of the observation, such as the run queue, number of users, and files system space utilization.
COUNTER	Cumulative counts of activity, such as CPU times, physical IOs, paging, network packet counts, and interrupts.

Configure Sources

The source of an event policy is the specific thing that is being monitored for events. For example, the source for a logfile entry policy is a specific log file; the source of a WMI policy is a specific entry in the WMI database. This section of the help explains how to specify exactly what source a particular event policy should monitor.

Logfile Entry policy type

Related Topics:

- Quick start: how to create a policy

Select the logfile to monitor

This feature allows you to indicate the log file that the policy reads. Type the drive letter and the full path for the location of this log file on the appropriate managed node . You can use Windows environmental variables (for example winnt or clusterlog) to make your policies more flexible. The proper syntax for these variables is `<$variablename>` .

You can also call a script or command that returns the path and name of the log file you want to monitor. For example, type

```
<`command`>
```

where command is the name of a script that returns the path and name of the log file you want to monitor. The command can also return more than one log file path separated by spaces. HP Operations monitors each of the files using the same options and conditions as configured for this policy. This is very useful when you want to dynamically determine the log file path or monitor multiple instances of a log file.

CAUTION:
You must ensure that the log file can be processed by HP Operations. For example, log files that contain binary data cannot be read by the policy and may cause the policy to stop responding or even quit. If your log files do contain binary data, use log file preprocessing to preprocess your files.

To select the logfile to monitor

1. Right-click the policy and select All Tasks → Edit...
2. Select Source .
3. In the Log file path \ name box, type the name of the log file you want to monitor.

Related Topics:

- Set log file source properties
- Quick start: how to create a policy

Set the logfile polling interval

You can indicate how often the Logfile Entry policy should read the logfile. This period of time is the polling interval. The polling interval should be as large as possible, although this depends on the amount of new data written to the file and the read mode that you choose. As a minimum, set the interval for more than 30 seconds; usually 5 minutes is appropriate. Note, however, that a policy begins to evaluate data *after* the first polling interval passes. A shorter polling interval is better when you are testing a policy.

To set the logfile polling interval

1. Right-click the policy and select All Tasks → Edit...
2. Select Source.
3. Select a number for hours, minutes, and seconds.

Related Topics:

- Set log file source properties
- Quick start: how to create a policy

Set the read mode

The read mode of the log file or eventlog policy indicates whether the policy should process the entire log file or should only process new log file entries. The three available read modes are described in the table below. Note that every policy reads log files independently from any other policies. This means, for example, that if "Policy 1" with read mode Read from Begin (First Time) is enabled on a node where "Policy 2" with the same read mode already exists, "Policy 1" will still read the entire log file after it has been enabled.

Log file read modes

Mode: Description	Advantage / Disadvantage
Read from Last File Position where the file is a Windows EventLog : The policy reads only new—appended—entries written in the EventLog while the policy is enabled on the managed node . If the EventLog decreases in size between readings, then the entire EventLog is read. EventLog entries that are added to the EventLog when the policy is disabled are not processed by the policy. If the agent stops, all entries written to the monitored EventLog while the agent is not running will be processed after the agent restarts. Choose this option if you are concerned only with EventLog entries that occur when the policy is enabled.	Advantage: No chance of reading the same entry twice. (Unless the EventLog decreases in size because some entries were deleted.) Disadvantage: Entries written to the EventLog while the policy is disabled will not be processed by the policy.
Read from Last File Position where the file is a text log file : The policy reads only new—appended—entries written in the log file while the policy is enabled on the managed node . If the log file decreases in size between readings, then the entire log file is read. Log file entries that are added to the log file when the policy is disabled are not processed by the policy. Choose this option if you are concerned only with log file entries that occur when the policy is enabled.	Advantage: No chance of reading the same entry twice. (Unless the log file decreases in size because some entries were deleted.) Disadvantage: Entries written to the log file while the policy is disabled or the agent is not running will not be processed by the policy.
Read from Begin (First Time) where the file is a text log file or a Windows EventLog : The policy reads the complete log file each time the policy is enabled or the agent restarts on the managed node. This ensures that all entries in the log file are compared with the rules in the policy. Each successive time that the policy reads the log file, only new (appended) entries in the log file are processed.	Advantage: Every existing and future entry in the log file will be processed by the policy. Disadvantage: Duplicate entries can occur if an enabled policy is disabled and

Choose this option if you want to ensure that every existing and future entry in the log file will be processed by the policy while it is enabled.	reenabled, or if the agent stops and restarts.
Read from Begin (Always) : Where the file is a text log file : (This mode is not available for the Windows Event Log policy type.) The policy reads the complete log file every time it detects that the log file has changed. The policy scans the log file at the specified polling interval. If no change is detected, the log file is not processed. Any logfile entries overwritten while the agent is not running or the policy is disabled will not be evaluated by the policy. Choose this option if you are monitoring a log file that is overwritten, rather than appended.	Advantage: Ensures that log files that are overwritten are correctly processed. Disadvantage: Only valid for log files that are overwritten, rather than appended.

To set the read mode

1. Right-click the policy and select All Tasks → Edit...
2. Select Source .
3. Select a read mode.
4. You can also indicate whether a message should be sent to the message browser and whether the policy should close the log file after reading.

Related Topics:

- Set log file source properties
- Quick start: how to create a policy

Set log file source properties

The Source tab of the logfile policy tab allows you to indicate which log file the policy reads, and to set some characteristics about how the policy should read the log file. You can specify:

- preprocessing commands
- the polling interval
- the character set of the log file
- the read mode that the policy should use

To access this tab

1. Right-click the policy and select All Tasks → Edit...
2. Select Source .

Related Topics:

- Quick start: how to create a policy

Log file preprocessing

Use this option if the original log file needs to be reformatted before monitoring begins. If you specify a command or program in this box, it is carried out before the log file is monitored. This command or program can produce a file which the policy uses instead of the original log file.

NOTE
Log files which contain binary data cannot be processed by the Logfile Entry policy, and must be preprocessed using this option.

Process a log file before monitoring:

1. Right-click the policy and select All Tasks → Edit...
2. Select Source.
3. Select Preprocessing . Type the file or command to be carried out and the file to be read.

Related Topics:

- Set log file source properties
- Quick start: how to create a policy

Select the logfile character set

Indicate the name of the character set used by the log file that you are monitoring. It is important to choose the correct character set. If the character set that the policy is expecting does not match the character set in the log file, pattern matching may not work, and message text can have incorrect characters or be truncated in the message browser . If you are unsure of which character set is used by the logfile that you want to monitor, consult the documentation of the program that writes the logfile.

NOTE:
The character set of the eventlog must be convertible to the agent node character set. For example, if agent character set is iso88591 (English) then, ACP 1252, ACSII, ISO 8859-1, OEMCP 850, OEMCP 437, ROMAN 8, or EBCDIC may be used. If the agent character set is sjis (Japanese), then ACP 932, ACSII, or EUC may be used. Japanese, and sjis may be used. If the agent character set is iso88595 (Cyrillic), then iso88595, ASCII, ACP 1251, or OEMCP 866 may be used.

The character sets supported by Windows and HP-UX nodes are:

ACP 1250	Central European
ACP 1251	Cyrillic
ACP 1252	Western European
ACP 932	Includes all characters defined in the shift-JIS code. This character set is supported by the Japanese versions of Microsoft Windows NT and Microsoft Windows 95/98.
ACSII	English (American Standard Code for Information Interchange)
BIG-5 Taiwanese	Taiwanese
EBCDIC	(Extended Binary-Coded Decimal Interchange Code) Generally used only on large IBM computers.
EUC Japanese	(Extended UNIX Code) Japanese
EUC Korean	(Extended UNIX Code) Korean
EUC Taiwanese	(Extended UNIX Code) Taiwanese
GB-2312-80 Chinese	Chinese
ISO 8859-1	Most West European languages, including French, Spanish, Catalan, Basque, Portuguese, Italian, Albanian, Rhaeto-Romanic, Dutch, German, Danish, Swedish, Norwegian, Finnish, Faeroese, Icelandic, Irish, Scottish, and English. Also Afrikaans, Swahili.

ISO 8859-15	Latin alphabet
ISO 8859-2	Central and Eastern European languages, including Czech, Hungarian, Polish, Romanian, Croatian, Slovak, Slovenian, Sorbian.
ISO 8859-5	Languages that use Cyrillic characters, including Bulgarian, Belorussian, Macedonian, Russian, Serbian and Ukrainian.
ISO 8859-6	Arabic
ISO 8859-7	Greek
ISO 8859-8	Hebrew and Yiddish
ISO 8859-9	Same as ISO 8859-1, but with Turkish, instead of Icelandic.
OEMCP 437	U.S. English
OEMCP 737	Greek (formerly 437G)
OEMCP 775	Baltic
OEMCP 850	All the characters used by most European, North American, and South American languages
OEMCP 852	Slavic (Latin II)
OEMCP 857	IBM Turkish
OEMCP 860	Portuguese
OEMCP 861	Icelandic
OEMCP 862	Hebrew
OEMCP 863	Canadian-French
OEMCP 864	Arabic
OEMCP 865	Nordic
OEMCP 866	Russian
OEMCP 869	IBM Modern Greek
ROMAN 8	European characters
SHIFT-JIS	Microsoft's standard encoding for Japanese.
UCS-2	This codeset is intended to express all characters in the world in a united character set.
UTF-8	(Unicode Transformation Format-8) This codeset is intended to express all characters in the world in a united character set.

To select the logfile character set

1. Right-click the policy and select Edit .

2. Select Source .
3. Use the Logfile Character set pull-down menu to indicate the character set of the log file.

Related Topics:

- Set log file source properties
- Quick start: how to create a policy

Measurement Threshold policy type

Related Topics:

- Quick start: how to create a policy

Set threshold source properties

The Threshold Source properties tab allows you to indicate the source that you want a Measurement Threshold policy to monitor. This source provides the data that is compared against the rules that you write for the policy.

- The Short name and Description are labels that you choose to help you recognize the value or metric for a threshold source. These labels are visible in the source tab and are helpful if you write a policy with multiple sources. When using a script to determine the threshold level, these names are used in the script to identify the sources.
- Source Type : Embedded Performance Component
- Polling Interval : Indicate how often the policy should check the source for new information. This period of time is the polling interval. To increase performance, the polling interval should be as large as possible, while still being frequent enough to monitor data at the rate that it is expected to change. Note that a policy begins to evaluate data *after* the first polling interval passes. A shorter polling interval is better when you are testing a policy.
- Add Source : If you want the policy to monitor more than one threshold source, you can click this button to add sources. Policies with multiple sources require you to write scripts to evaluate the threshold levels. Note that switching from single to multiple sources automatically converts the rules to VB Script .

  CAUTION:
  Ensure that the scripting language that you choose will run on the operating system where you intend to use the policies.

To access this tab

1. Right-click the policy and select Edit .
2. Select Source .

Related Topics:

- Quick start: how to create a policy
- Writing Scripts for Measurement Threshold Policies
- Agentless monitoring

Writing scripts for measurement threshold policies

You need to use a script to determine the threshold for your measurement threshold policy if the source that you choose delivers something other than a number or a Boolean value or if you want to evaluate multiple sources. A script makes it possible for you to perform your own calculations and decide if the threshold has been crossed. Here is how a script works together with a measurement threshold policy:

1. First, you create a policy that monitors one or more threshold sources . You assign a Short Name to each source in the policy.
2. If the source has multiple instances (for example, multiple logical disks), then use the measurement threshold processing options to indicate if the script should process only one instance at a time, or should process all instances at once.
3. Next, you create a threshold rule that uses VB Script or Perl to determine the threshold limit. The script should use the short names and the policy object model to access the value for each source, and should perform some calculation to determine if a threshold has been crossed. The script should set the Rule Object to TRUE if threshold has been crossed or FALSE if it has not been crossed.
4. When the policy is deployed, the script will evaluate the sources and sets the rule object to TRUE or FALSE after each polling interval. If rule object is set to TRUE, the policy will carry out the Start, Continue, or End Actions depending on how long the threshold has been crossed. You can also use the script to send messages or execute commands directly if you require more flexibility than the Start, Continue, and End Actions provide.

NOTE:
The agent runs as a service that has no standard input, standard output, or standard error streams. Therefore, the predefined file handles STDIN, STDOUT, and STDERR are not available for Perl scripts in measurement threshold policies. It is also not possible to open file handles that use command pipes or capture the standard output from commands within backticks (`).

Related Topics:

- Policy objects and examples

Configure Start/Continue/End Actions

Measurement threshold policies provide the ability to perform actions at different times, depending on how long the threshold is crossed. You can specify Start actions , Continue actions , and End actions .

- Start actions are carried out the first time that the threshold is crossed.
- Continue actions are carried out at each subsequent polling interval if the reset value is not reached.
- End actions are carried out after the threshold crosses the reset value.

If two or more rules in a policy measure the same performance value but with different thresholds, then the following rules determine which actions are carried out:

- The actions of only one rule will be carried out per polling interval.
- Start actions are carried out the first time that the value exceeds a threshold at a polling interval. (If the value exceeds the threshold of several rules, the actions of the rule with the most severe value are carried out.)
- Continue actions are carried out at each subsequent polling interval if the reset value is not reached, but only if the start actions for that rule were carried out.
- End actions are carried out after the threshold crosses the reset value, only if the start actions for that rule were carried out. If the value drops below two thresholds within one polling interval, the end actions of the lowest rule that performed start actions are carried out.

For each set of actions, you can Configure message attributes , Add operator-initiated commands or automatic commands

Set the Measurement Threshold general rule properties:

1. Right-click the policy and select All Tasks → Edit...
2. Select Rules .
3. Select the rule to which you want to set or modify the start, continue, or end actions.
4. Select Modify.
5. Select Start actions , Continue actions , or End actions .

Related Topics:

- Quick start: how to create a policy
- Set threshold source properties

Performance monitor examples

The Performance Monitor can be used as the source for a threshold alarm. Here are some examples of frequently used threshold monitors. For a complete listing and description of all default object counters, see the Microsoft Windows documentation.

To monitor

the percentage of free disk space on a C drive on SCSI port 0,

select these options:

Object:	LogicalDisk
Counter:	% Free Space
Instance:	0/C:''

Additional configuration:

- If the counter has a percent sign (%), it can be omitted if you want to receive the raw value instead of a percent
- For instances which have parent instances, a question mark (?) can be used as a wildcard to match any parent instance. For example: `?/C:` matches `0/C` and `1/C`

Related Topics:

- Threshold source properties
- Quick start: how to create a policy

Set measurement threshold processing options

NOTE:
This feature is only available for the Measurement Threshold policy type .

You can choose how HP Operations processes multiple instances of the value being measured. For example, if a policy monitors disk space, then each disk in the managed node is one instance, and you can choose whether to treat each disk separately or all disks as a whole.

- Process each instance separately : Select this option if you want each instance to be processed by the policy separately. For example, if the policy monitors each CPU in a multiple CPU server, and the activity of all CPUs exceeds the threshold, a message will be generated for each CPU.
- Process all instances once : This option can only be used if the threshold rules use the output of a script as the threshold (instead of minimum or maximum). Select this option if the script evaluates all instances and delivers one value to be tested by the policy. (Make sure that the scripting language that you choose is supported on the platform where you plan to distribute your policy.)
- Select Show only newest message in active browser if you want to ensure that only the most current status of a threshold is shown in the active message browser . The values that measurement threshold policies monitor can change rapidly. A condition that produces an error message might only exist for a short time. In order to prevent your message browser from filling up with threshold messages that might not be current, you can use Show only newest message in active browser . This feature causes a measurement threshold policy message to acknowledge all messages in the message browser which were created by the same policy, and which have the same node and instance. (Note that although a message cannot acknowledge itself, you can send an End Action message directly to the acknowledged messages browser.)

To access this tab:

1. Right-click a measurement threshold policy and select Edit .
2. Select Options .

Related Topics:

- Set threshold source properties
- Quick start: how to create a policy

Performance metrics

The HP Operations agent embedded performance component (EPC; also known as coda) collects performance counter and instance data. You can use these metrics in defining event/action thresholds that generate alarms in real time based on availability, response time, and throughput measurements.

The following types of metrics are collected:

- Basic (golden) metrics

  These are approximately 30 metrics that are collected for all supported platforms. They can be used to answer most of your questions about a system's global configuration, CPU, disk, swap, and memory usage and have been chosen to offer the best information for the widest number of platforms.

- Additional metrics

  The data collection component also provides you with additional performance metrics on each of the supported platforms. Although these metrics vary by platform, they are available on most platforms and are generally useful for drill down and diagnosis on a particular system. On HP-UX, for example, there are approximately 120 of these additional metrics.

The collection interval is five minutes. All metrics, including golden metrics and the additional metrics, are collected. The data is kept in the data store for up to five weeks, at which time a week's worth of data is rolled out.

You can view a of list of available metrics in the *HP Performance Agent Dictionary of Operating System Performance Metrics* which is available at http://ovweb.external.hp.com/lpe/doc_serv/ . (Select the product Performance Agent , the required version, OS, and language.)

Related Topics:

- Set threshold source properties
- Policy objects and examples

Editing parameters in the Threshold Policy Type editor

HPOM for Windows can use VBScript or Perl scripts in measurement threshold policies to do more complicated calculations to evaluate a threshold or to add additional functionality. By configuring parameters in scripts that can easily be changed later, you can customize the variables in the script without the need to change the script itself. The script parameters are changed on the Script Parameters tab and the editor changes the script.

Parameter Definitions

Parameters are defined during policy development. The policy developer decides which script variables a customer can customize and that should be displayed on the Script Parameters tab. The parameters are defined in the script with fixed syntax. As shown in the following example, the character # (comment in Perl) is used to define a comment. For VBScript, you must use the ' character.

The parameter definition block has to be at the beginning of the first script of a policy. The parameter definition block starts with:

```
#PARAMETERS START
```

The parameter definition block ends with:

```
#PARAMETERS END
```

The script parameter definition block is changed when you change script parameters on the Script Parameters tab. Therefore, it should only contain script parameter definitions. All other comments or script code in the script parameter definition block will be overwritten. Each parameter is defined with at least the following information:

```
#PARAMETER <name> [STRING|INT|DOUBLE] DEFAULT <default value>
```
For example:

```
#PARAMETER CriticalThreshold INT DEFAULT 95
```

If the parameter is defined as STRING value, you must use " at the beginning and end of the value.

For example:

```
#PARAMETER CriticalThresholdMsg STRING DEFAULT "Critical Message"
```

The name, data type, and the default value of a script parameter definition are required.

When you save the script with the parameter definition, the editor parses the script parameter block and inserts the Perl or VBScript variables you have defined.

For example:

```
#PARAMETERS START
#PARAMETER CriticalThreshold INT DEFAULT 95
#PARAMETERS END
```

and saves the script. The editor creates the following script parameter block for a Perl script:

```
#PARAMETERS START
#PARAMETER CriticalThreshold INT DEFAULT 95
my $CriticalThreshold;
$CriticalThreshold = 95;
#PARAMETERS END
```

The VBScript is as follows:

```
'PARAMETERS START
'PARAMETER CriticalThreshold INT DEFAULT 95
DIM CriticalThreshold
CriticalThreshold = 95
'PARAMETERS END
```

In most cases, the complicated script policies contain only one script that contains all logic. But threshold policies can be much more complex, with multiple instance filter rules or threshold levels with threshold and reset scripts. The parameter definition block is evaluated only from the first script.

- In a policy with one or multiple threshold levels, the script parameter definition block must be in the threshold script of the first threshold level.
- In a policy with instance filter rules, the script parameter definition block must be in the condition script of the first instance filter rule.

If the order of threshold levels or instance filter rules is changed or if a policy is changed to use instance filter rules, the editor moves an already existing script parameter block automatically to the correct script. However, it is possible to define that the script parameters are also stored as SESSION parameters, as explained in the help topic SESSION parameters .

Create a measurement threshold policy with parameters

HPOM for Windows uses VBScript or Perl scripts in measurement threshold policies to do more complicated calculations to evaluate a threshold or to add additional functionality. By configuring parameters in scripts that can easily be changed later, you can customize the variables in the script without the need to change the script itself. The script parameters are changed on the Script Parameters tab; the Measurement Threshold policy editor changes the script.

Parameter Definitions

Parameters are defined during policy development. Use the Measurement Threshold policy editor to create a policy containing parameters that adapt the policy to your particular needs.

To create a measurement threshold policy with parameters:

1. Select Policy Management → Policies grouped by type .
2. Right-click Measurement Threshold to open the context menu.
3. Select New → Policy to open the Measurement Threshold policy editor with the Source , Threshold levels , and Options tabs visible. The Script_Parameters tab is not visible at this time.
4. Enter your preferences on the Source tab. For details, see the help topic Set threshold source properties.
5. Select the Threshold levels tab. In the list, select VB Script or Perl script. The Script Parameters tab opens without any parameters displayed.
6. Click New to open the New threshold level dialog box with the General , Start Actions , Continue Actions , and End Actions tabs displayed. Use the General tab of this dialog box to define the threshold limit script.
7. Add the #PARAMETERS block with script parameter definitions into the Threshold limit script and click OK to confirm your action and close the dialog box.

   Because this is the first threshold level, the editor parses the VBScript or Perl script and finds the #PARAMETERS block with parameters, causing the defined parameters in the Script Parameters tab to display. If the #PARAMETERS block does not exist in the script, the Parameters tab will be empty. You can view the defined parameters list in the Script_Parameters tab.

8. To edit a parameter in this list, follow the instructions in the help topic Edit an existing measurement threshold policy with parameters .
9. Click Save , then Close to name and save the policy and close the policy editor.

Edit an existing measurement threshold policy with parameters

By editing the parameters in scripts, you can change functionality without having to change the script itself.

To edit an existing measurement threshold policy with parameters:

1. Select Policy Management → Policies grouped by type .
2. Select Measurement Threshold in the console tree to open the list of available measurement threshold policies in the details pane.
3. From the list, double-click the name of the policy you want to edit. The Measurement Threshold policy editor opens and displays the selected policy.
4. Select the Script Parameters tab.
5. Select the script parameter you want to change and click Edit to open the parameter properties dialog box for the selected parameter. Use this dialog box to modify the parameter's value.
6. Click OK to save your script parameter changes and close the dialog box. The changed script parameter value is updated in the script parameter definition block.
7. Click Save , then Close to save the policy and close the policy editor.

SESSION parameters

Complicated threshold policies can contain multiple threshold levels or instance filter rule and conditions with threshold and reset scripts, but the parameter definition block is evaluated only from the first script.

However, if you define that the parameters are also stored as SESSION variables, you can use the SESSION variable to access the parameters in all scripts of the policy.

To define that a parameter should be stored in a SESSION variable:

Use the following syntax to save a parameter as a SESSION variable:

#PARAMETER <name> [STRING|INT|DOUBLE] DEFAULT <value> [SESSION ["<session name>"]]

If the SESSION keyword is used without a session name, the parameter name will be used also for the session variable. If the SESSION keyword is used, the created Perl code for the example above would be as shown:

```
#PARAMETERS START
#PARAMETER CriticalThreshold INT DEFAULT 95 SESSION
my $CriticalThreshold;
$CriticalThreshold = 95;
$Session->Value('CriticalThreshold', $CriticalThreshold);
#PARAMETERS END
```

The VBScript for the same situation is shown below:

```
'PARAMETERS START
'PARAMETER CriticalThreshold INT DEFAULT 95 SESSION
DIM CriticalThreshold
CriticalThreshold = 95
Session("CriticalThreshold") = CriticalThreshold
'PARAMETERS END
```

In other threshold level or instance rule scripts, it is now possible to use the SESSION variable to access the defined script parameter value. In a Perl script this would look like the following:

```
if ( $src->Value() > $Session->Value('CriticalThreshold') )
{
  $Rule->Status(TRUE);
}
The VBScript for the same situation is shown below:
```

```
If Src.Value > Session.Value("CriticalThreshold") Then
  Rule.status = True
End If
```

The Parameter definition keywords help topic contains additional keywords.

Parameter definition keywords

Use the keywords shown in the table as needed for your parameter definitions.

Keyword	Definition	Syntax
CAPTION	A more descriptive text that appears on the windows to change the parameter value.	CAPTION "<text>"
DEFAULT	This keyword defines the default value for the parameter. It is required and must be defined by the policy developer.	DEFAULT <default>
VALUE	This keyword should not be used during parameter definition. It will be created automatically at the time a user changes a parameter value; it is necessary for the editor to know the changed value. If the changed value is available, the generated code will always contain this value and not the DEFAULT value.	VALUE <value>
MIN, MAX	these keywords are only valid for parameters of type INT or DOUBLE. If a user changes a numeric parameter for that a MIN or MAX value or both is defined, the editor does a checking whether the new value is within the defined range.	MIN <value> MAX <value>
SESSION	This keyword defines that the parameter value should also be saved in a session variable. For details, see the topic SESSION Parameters .	SESSION "<session name>"

The complete syntax for a parameter is shown below:

```
#PARAMETER <name>STRING|INT|DOUBLE DEFAULT <value> [VALUE
<value>] [SESSION ["<session name>"]] [CAPTION "<caption string>"] [MIN
<numeric value>] [MAX <numeric value>]
```

The parameter definition must always start with #PARAMETER <name>. The other keywords can be in any order.

Example

```
#PARAMETER CriticalThreshold INT DEFAULT 95 SESSION "CritThres" MIN 0 MAX 100
```

Scheduled task overview

This policy type allows you to schedule commands to run on managed nodes, and will send a message to the management server to indicate the success or failure of the command. Use this policy if you want to run commands on one or more managed nodes—either once or according to a specific schedule.

Create a Scheduled Task Policy:

1. Right-click the scheduled task policy type and select New → Policy
2. Select Task and indicate the command that should be run and any messages that you want to receive.
3. Select Schedule and indicate when the command should be run.
4. Save the policy.

Related Topics:

- Quick start: how to create a policy

Scheduled task

In the Task tab, you specify a command or script that you want to run on the managed node, as well as any messages that you want to receive.

- Task type: choose Command
- Command : Type the command that you want to run on the managed node

  If the command is not in the action, command, monitor or category directory on the managed node, or in the default path for the user account under which the command will be run, then the path to the command must be included. (The action, command, monitor and category directories are located under `Program Files\HP\HP BTO Software\Installed Packages\{790C06B4-844E-11D2-972B-080009EF8C2A}\bin\OpC\vpwin\`)
- Execute Under agent account ($AGENT_USER) : This is the default choice. It runs the command under the same account as the agent is running, which is Local System by default. If selected, the Execute as user: options are not available.
- Execute as user : To specify a password for this user, select the Specify password check box and type the user name under which the command should be run. The user must exist and have permission to run the command on the managed node.

  NOTE:
  With agent versions lower than 8.53, if you specify a non-existent user, the command runs under the same account as the agent. With agent version 8.53 and higher, if you specify a non-existent user, the command fails to run.

  If you type the password, then a standard Windows *logon as user* is performed before executing the command. If no password is entered, then an HPOM *switch user* is performed before executing the command.

  Should I provide the password or not?
- Type the VB script in the window. Refer to Policy objects and examples for information about writing scripts for HP Operations.
- Type the Perl script in the window. Refer to Policy objects and examples for information about writing scripts for HP Operations.

  NOTE:
  The agent runs as a service that has no standard input, standard output, or standard error. Therefore,

the predefined file handles STDIN, STDOUT, and STDERR are not available for Perl scripts in scheduled task policies. It is also not possible to open file handles that use command pipes or capture the standard output from commands within backticks (`).

- Decide if you want to receive any notification messages, and make the appropriate selection. Design the start, success, or failure Message that you want to receive.
- When Append output of command as annotation to success / failure message is selected, an annotation is added to the message when the command has completed. The annotation contains the start time, output, exit value, and finish time of the command. If a command fails, an annotation is provided even if this item is not selected.

To access this tab

1. Right-click the Scheduled Task icon in the console tree.
2. Select New → Policy .
3. Select the command tab.

Related Topics:

- Scheduled task policy type
- Quick start: how to create a policy
- Choose schedule options
- Change password or user name for multiple scheduled tasks policies

Choose schedule options

The following schedule options are available for the scheduled task policy .

- When Every minute is selected, the command will be run at 60 second intervals.
- When Every hour is selected, the command will be run at 60 minute intervals. You can also indicate how many minutes after the hour the command should run. For example, if you select 34, the command will run at 1:34, 2:34 and so on.
- When Daily is selected, the command will every day at the time you indicate. You can schedule the command to run at multiple times each day by selecting Multiple Times and clicking the Times button.
- When Weekly is selected, the command will be run one day per week at the time you indicate. You can schedule the command to run at multiple times on this day by selecting Multiple Times and clicking the Times button.
- When Monthly is selected, the command will be run one day per month at the time you indicate. You can schedule the command to run at multiple times on this day by selecting Multiple Times and clicking the Times button.
- When Once is selected, the command will be run on one specific day at the time you indicate. You can schedule the command to run at multiple times on this day by selecting Multiple Times and clicking the Times button.
- When Specially is selected, you can indicate specific days and times when the command should be run. You select specific days of the week, specific days of the month, and specific months. This allows you to specify odd schedules such as, "On Monday when it falls on the 2nd of the month." You can also indicate that the command should only be run during a specific year.
- When Once per interval is selected, the command will be run once each time the interval that you indicate passes.

Related Topics:

- Quick start: how to create a policy

Set time reoccurrence

In the time reoccurrence window, you can specify specific hours and minutes when a command should be run. By default, none are selected, meaning that the command will never run. click an interval to select it, or drag the pointer over multiple intervals. To access the Time Reoccurrence window:

1. Right-click a Scheduled Task policy in the console tree.
2. Select All Tasks → Edit .
3. Select the Schedule tab.
4. Click the Times... button.

Related Topics:

- Scheduled Task policy type
- Quick start: how to create a policy

Windows Event Log policy type

Related Topics:

- Quick start: how to create a policy

Select the event log to monitor

Windows produces several event logs. You can choose which event log you want a policy to monitor. If you want to monitor more than one event log, you need more than one policy.

Select the Event Log to Monitor

1. Right-click the event log policy and select All Tasks → Edit...
2. Select Source .
3. Type the event log name, or select it from the pull-down menu.

Related Topics:

- Set Windows event log source
- Quick start: how to create a policy

Set windows event log source

This tab allows you to indicate which event log the policy reads and where the policy should begin to read the event log . You can also choose to receive a message if the event log is missing .

1. Right-click a Windows Event Log policy and select All Tasks → Edit...
2. Select Source .

Related Topics:

- Quick start: how to create a policy

Receive a message if the event log is missing

If you write a policy that monitors entries in a Windows event log, you might want to receive a message if for some reason the log file is missing.

Receive a Message if the Event Log is Missing

1. Right-click the policy and select All Tasks → Edit...
2. Select Source.
3. Select Send message if event log does not exist.

Related Topics:

- Quick start: how to create a policy

Windows Management Interface policy type

Related Topics:

- Quick start: how to create a policy

Select WMI source

The WMI source tab allows you to choose the instance or event that you want the WMI policy to monitor.

- Object path

  The object path defines the WMI object that you want to monitor.

  How do I know what WMI object I should monitor?

  - Node: The node that hosts the WMI database that you want to monitor. This can be an agentless node.
    If you do not specify a node, HPOM monitors the WMI database of the node that has this policy deployed. Use the browse button to select HPOM nodes or type a node name into the box. See Identify the originating node for information about adding agentless nodes to HPOM.
  - WMI Namespace: The namespace that contains the data that you want to manage.
  - Object type: Choose Event or Instance . Note that if you used the browse button to fill in the Object path fields, the Object type will probably be correctly set. If, however, the class is not correctly located in the class hierarchy, the setting might be wrong.
    What are WMI events and instances?
  - Event/Instance class name : Type the class that contains the event or instance that you want to monitor. (A class is a collection of data properties that is defined for information that will be stored in the WMI repository.)
  - Connect as non-agent user: If selected, the agent accesses the node's WMI database using the following account information. This account must exist on the agentless node and must have local administrator privileges. If not selected, the agent account is used.
  - User name: Type the user name of the account that the agent will use to connect to the WMI database.
  - Login password: Type the password of the connecting account.

- Type of query

  The type of query depends on the object type that you are monitoring. If you are monitoring an event for which a provider is defined, then you do not need to enter any information here. If you are monitoring an intrinsic event for which no provider is defined, then you need to specify a polling interval. If you are monitoring an Instance, then you need to provide the following information:

  - Select Query instances of class if you want to match specific values contained within the class. You must indicate the Polling interval to indicate the frequency with which the Windows Management Interface policy checks the instances you selected.

- Select Query the intrinsic event for these instances if you want to check for the creation, modification or deletion of the instance, the class that contains the instance, or the namespace that contains the instance. If there is no provider for the event, you must also set the Polling interval to indicate the frequency with which the Windows Management Interface policy will check the object you selected. (This results a WBEM Query Language within clause.) How do I check for a provider?

- View global WQL filter... A global filter can be described as a rule . It is a test that is applied to the instance or event before the policy begins to evaluate it. A global filter can improve performance, because events or instances that do not get through the filter are not evaluated by the policy. (The global filter is a WBEM Query Language where clause.)

  Sample global filters

  The syntax of a global filter has three parts:

  PROPERTY OPERATOR VALUE

  for example: `_PATH = "C:/program files"`

  If the global filter filters intrinsic events, the syntax is somewhat different:

  TargetInstance.*PROPERTY OPERATOR VALUE*
  or
  TargetClass.*PROPERTY OPERATOR VALUE*
  or
  TargetNamespace.*PROPERTY OPERATOR VALUE*

  for example,

  ```
  TargetInstance.InteractWithDeskTop = 1
  TargetNamespace.name = "CIMV2"
  ```

To access this tab

1. Right-click the Windows Management Interface policy and select Edit .
2. Select Source .

Related Topics:

- Quick start: how to create a policy
- Agentless monitoring
- For more information about WMI, download the Microsoft Windows Management Instrumentation (WMI) SDK from the Microsoft Developers Network (msdn.microsoft.com)

WMI instance browser

The WMI instance browser is a tool provided by Microsoft that allows you to view the data in each instance of a WMI class. This information is often helpful because it allows you to see what text actually exists in each property. You need to know this information to design match conditions for WMI policies.

To access the WMI instance browser

1. Open a WMI policy
2. Select the Rules tab
3. Select New , (or modify an existing policy)
4. Select Launch instance browser...

Related Topics:

- Quick start: how to create a policy
- Select WMI source

WMI class browser

The WMI class browser allows you to inspect all WMI classes installed on any Windows computer that is accessible to the management server . To use the WMI class browser:

1. Click the browse button ( ) from the WMI policy source tab .
2. When the connect to namespace window appears, click the browse for namespace button ( )
3. In the Machine Name text box, type the name of the computer where you want to inspect WMI classes, and click Connect . The WMI root namespace of that computer appears in the window.
4. Click the root namespace to expand the folder, select one of the available namespaces (for example Hewlett-Packard\OpenView\console), and select OK . The class browser appears, populated with the classes present in the namespace that you selected.
5. Browse or search for classes in this namespace. Click the check box before one class name to select it, and select OK. The class and namespace are inserted into the appropriate Object path boxes.

Related Topics:

- Quick start: how to create a policy
- Select WMI source

Process Monitor policy type

This policy type monitors the processes that are running on the managed node and sends a message when the status of the process changes (for example, from running to stopped).

You can define the status to monitor and the action to take if the status changes. Choose this policy type if you want to monitor the status of processes running on the managed node.

Related Topics:

- Quick start: how to create a policy
- Add a process to monitor
- Set up a process monitor policy
- Process monitor properties

Set up a process monitor policy

If you are setting up a policy to monitor processes for the first time, the initial dialog box is empty. You will need to specify which processes you want to monitor, how you want the monitoring to take place, and how often the policy should check the monitored processes.

The following actions are available to you:

- **Browse**

  Opens the **Add new process** dialog box and by default displays a list of Windows processes running on the localhost.

  NOTE:
  It is not possible to browse processes on UNIX and Linux nodes using the Browse button.

  Use this option if you are not sure of the details of the Windows processes that you want to monitor (for example, the process names or the name of the node where the processes are running). The list does not show stopped or disabled processes.
- **New**

  Opens the **Process Properties** dialog box and allows you to add a new process monitor. Use this option if you *do* already know all the details of the processes that you want to monitor but *cannot* access the node where the processes are running.
- **Modify**

  Opens the **Process Properties** dialog box and allows you to check or modify an existing process monitor.
- **Copy**

  Opens the **Process Properties** dialog box and allows you to copy an existing process monitor. Use this option if you want to re-use conditions or actions, which you have already defined in an existing monitor for a process.
- **Delete**

  Deletes selected processes: you will be prompted for confirmation of the requested action.
- **Defaults**

  Opens the **Defaults** dialog box where you can define the default actions to perform if a threshold is met or a condition matched.

To set up a process monitor policy

1. In the console tree, under **Policy Management** → **Agent policies grouped by type** , right-click **Service/Process Monitor** and then click **New** → **Policy** .

   The policy editor opens.

2. From the Monitoring box, select **Processes** .

3. Click **New** .

   The Process Properties dialog box opens.

   NOTE:
   It is not possible to browse processes on UNIX and Linux nodes using the Browse button.

4. Specify the process that you want to monitor.

   See Process monitor properties for information about how to specify process details (process name, parameters, number of processes, and actions).

5. Click **OK** to return to the main policy editor window.

6. Click **Save and Close** .

   The Save As dialog box opens. Specify a **Name** and **Description** , and then click **OK** .

Related Topics:

- Quick start: how to create a policy
- Add a process to monitor
- Process monitor properties
- Process monitor default actions
- Save a policy

Add a process to monitor

Use the Add New Process dialog box to specify which processes you want to monitor. If you know the name of the node where the process you want to monitor is running, you can type it directly into the Node box. If you do not know the name of the node, click the [...] button to browse a list of nodes.

NOTE
It is not possible to browse processes on UNIX and Linux nodes using the Browse button.

After you have selected the node, click Refresh to display a list of the Windows processes running on the selected node. The list shows only *running* processes; the list does not display processes that are either disabled or stopped.

NOTE
If you use the browse option, the policy editor attempts to connect to the node as the Windows user, who started the console. If this user is not allowed to access process information on the specified node, HPOM displays an error message and prompts you to type a user name and password to log on to the node.

To add a new process to monitor

1. Select Processes in the Monitoring drop-down menu in the Policy dialog.
2. If you know the name of the process you want to add:
   a. Click New .
   b. Type the details of the process to monitor.
   c. Choose whether you want to use default actions with the new process or define your own custom actions.
   d. Click OK .
3. If you do not know the name of the process:
   a. Click Browse to select from a list of available processes on a node.
   b. Type the name of the node in the Node box or click the [...] button to browse a list of nodes. The list of nodes is retrieved from the HP Operations database and does not indicate that a connection to the individual nodes in the list has been successful.
   c. Click Refresh to display a list of processes on the specified node.

NOTE

If the policy editor cannot connect to the selected node to retrieve a list of processes, it displays an error message or asks you for more information.

d. Select the process that you want to monitor and click OK if you do not want to add any more Windows processes, or click Apply if you want to continue adding processes to the process monitor policy.

e. Select a process from the list and click Edit if you want to inspect or change the default actions associated with the selected process monitor.

4. Save the new or modified process monitor policy.

Related Topics:

- Quick start: how to create a policy
- Process monitor properties
- Process monitor default actions

Copy a process monitor

Use the Copy button to copy an existing process so that you can then modify it. For example, if you want to re-use customized actions that you defined for an existing monitor or if you want to monitor two instances of a process on a node, but with different parameters:

- `svchost -k rpcss`
- `svchost -k netsvcs`

NOTE
The policy editor compares the string you type in the Process box with the list of existing process names defined in the policy. It does not allow duplicate names or recognize the following special characters; \ (backslash), " (quotation mark), : (colon), & (ampersand), < > (angle brackets), ? (question mark), | (pipe), and spaces ( ).

To copy a process monitor

1. Select the process monitor that you want to copy in the list of existing processes in the process monitor policy editor.
2. Click Copy .
3. Type the details of the process to monitor.
4. Click OK.

   The copy appears in the list of process monitors in the process monitor policy editor.

Related Topics:

- Quick start: how to create a policy
- Process monitor default actions

Modify a Process Monitor

Use the Modify button if you want to change or update an existing process monitor (for example, to change parameters or threshold conditions or to specify your own, customized actions to associate with messages). The policy editor displays the Process Properties dialog box, which you can use to change details or settings as required.

NOTE
Remember to redeploy the modified policy after changing the contents of the policy.

To modify a process monitor

1. Select the process monitor that you want to modify in the list of existing processes in the process monitor policy editor.
2. Click Edit .
3. Modify the details of the process to monitor.
4. Click OK .

Related Topics:

- Quick start: how to create a policy
- Process monitor default actions
- Process monitor properties

Delete a process monitor

Use the Delete button to specify which processes you want to remove from the list of processes in the policy editor.

NOTE
Remember to redeploy the modified policy if you change its contents (for example, if you remove any processes from the list of processes that are monitored by the policy). Otherwise, you will continue to receive messages associated with process monitors you thought you had removed.

To Delete a process monitor

1. Select the process monitor that you want to remove from the list of processes in the process monitor policy editor.
2. Click Delete .
3. Click OK to confirm deletion of the selected process monitor.

   The selected process monitor is removed from the list of process monitors in the process monitor policy editor.
4. Save the policy.

Related Topics:

- Quick start: how to create a policy
- Save a policy

Process monitor properties

Click the New button in the Service/Process Policy Editor to display the New Process dialog box. Use this dialog box to specify details of the processes that you want to monitor with the process monitor policy. Use this option if you know the details of the process that you want to monitor but cannot access the node where the process is running.

Provide the following information:

- Process :

  The name of the process that you want to monitor. The name you enter here is checked against the list of existing processes defined for this policy.

  For Windows nodes, the string you enter here must match the name of the process as it is known to Windows, including the file extension, for example: "notepad.exe". Duplicates are not allowed.

  For UNIX or Linux nodes, specify *only* the name of the executable file for the process that you want to monitor. Do not include the path.

  You can monitor multiple instances of a process by using parameters to differentiate between the instances (for example, `svchost.exe -k rpcss` and `svchost.exe -k netsvcs`). For more information, see Parameters below.

- Parameters :

  Define the strings or parameters that you need to match. If you use this option, the parameters you specify are used to identify the running process. Standard HPOM pattern matching is used to evaluate the contents of this box, which for Windows managed nodes are not case sensitive. Note that:

  - If the Parameters box is empty, the policy editor matches only processes running without parameters.
  - If the Parameters box contains a string with no pattern-matching characters, the policy editor matches only processes with the defined string.
  - If the Parameters box contains pattern-matching characters, the policy editor matches all process parameters with the string defined (for example, <*> matches *all* parameters, and <*>abc<*> matches all parameters containing the string "abc").

- Number of Processes :

  Use the pull-down menu to specify the number of monitored processes. You can specify an exact number with the "equals" operator (==), or use the less-than (<) or more-than (>) operators to define a range (for example, >=1).

NOTE

The value you enter here defines the state which the policy *expects* to find and considers correct. The policy sends a message to the console only if the state it finds is *not* the expected one. For example, use >= 1 (greater than or equal to one) to check that one or more instances of the notepad.exe process are running. If the policy discovers that 0 (zero) instances of notepad.exe are running, it sends a message to the console.

- Actions :

  Choose whether you want to use the default actions defined for the process, or specify your own, customized actions:

  - Default Actions :

    Click Defaults in the Policy dialog box to specify that you want the process monitor to use the default actions defined for the policy.

  - Custom Actions :

    Click Custom defined if you want to define custom actions for a process monitor. If you check this option, you will have to use the options in the Start actions , Continue actions and End actions tabs to configure the actions that you want to occur when the policy finds a match. The actions tabs you use here are the standard HPOM tabs for defining a message and automatic or operator-initiated action.

Related Topics:

- Quick start: how to create a policy
- Add a process to monitor
- Edit a process monitor
- Process monitor default actions

Process Monitor: Default Actions

You can define a "Start" action, a "Continue" action, and an "End" action. A start action is triggered when a threshold is reached or a policy rule matched. Continue actions are carried out at each subsequent polling interval if the reset value is not reached. End actions are carried out after the threshold crosses the reset value.

NOTE:
Actions attached to process monitor policies can make use of session variables, which are illustrated in the examples below and described in more detail in Policy variables .

The default actions which are defined for the process monitor policies are:

- Default *Start* Action
  - Send a message to the active message browser: YES
  - Message severity: CRITICAL
  - Message key:

    `<$NAME>:<$MSG_NODE_NAME>:<$MSG_OBJECT>:START`
  - Message Acknowledgement with message key:

    `<$NAME>:<$MSG_NODE_NAME>:<$MSG_OBJECT>:<*>`
  - Message text:

    ```
    <$SESSION(PROCESSNBRAVAILABLE)> processes "<$SESSION(PROCESSNAME)>" with parameter
    "<$SESSION(PROCESSPARAMETERS)>" are running. Expected: <$SESSION(PROCESSMODE)>
    <$SESSION(PROCESSNBREXPECTED)> process.
    ```

    Example: `0 processes "notepad.exe" with parameter "<*>abc<*>" are running. Expected: 1 process.`
- Default *Continue* Action

  No "continue" action is defined: do NOT start any "continue" action
- Default *End* Action
  - Send a message to the active message browser:YES
  - Message severity: NORMAL
  - Message key: `<$NAME>:<$MSG_NODE_NAME>:<$MSG_OBJECT>:END`
  - Message Acknowledge with message key:

`<$NAME>:<$MSG_NODE_NAME>:<$MSG_OBJECT>:<*>`

- Message text:

  `<$SESSION(PROCESSNBRAVAILABLE)> process "<$SESSION(PROCESSNAME)>" with parameter "<$SESSION(PROCESSPARAMETERS)>" is running.`

  Example: `1 processes "notepad.exe" with parameter "<*>abc<*>" is running.`

To set or change the default process monitor actions

1. Create a new (or edit an existing) process monitor policy.
2. Make sure that you select Processes in the Monitoring drop-down menu in the Policy dialog box.
3. Click Defaults .
4. Type the details of default actions you want to occur when the defined policy threshold is matched. For example, a policy typically sends a message when a threshold is reached (and even reset) and starts an action to solve the problem to which the message relates.
5. Click OK .

Related Topics:

- Quick start: how to create a policy
- Start/continue/end actions

Windows Services Monitor policy type

This policy type monitors the Windows services, which are running on the managed node and sends a message when the status of the monitored services changes (for example, from running to stopped or disabled). You can define the status to monitor and the action to take if the status changes. Choose this policy type if you want to monitor the status of Windows services, which are running on the managed node.

Related Topics:

- Quick start: how to create a policy
- Add a Windows service to monitor
- Set up a Windows service monitor policy
- Windows service monitor properties

Set up a Windows service monitor policy

If you are setting up a policy to monitor Windows services for the first time, the initial dialog box is empty. You will need to specify which Windows services that you want to monitor, how you want the monitoring to take place, and, by means of the global polling interval, how often the policy should check the monitored services.

The following actions are available to you:

- Browse

  Opens the Add New Service dialog box and by default displays a list of Windows services for the localhost. Use this option if you are not sure of the details of the Windows services that you want to monitor (for example, the service names or the name of the managed node where the services are running).
- New

  Opens the Services Properties dialog box and allows you to add a new Windows service to the policy. Use this option if you already know all the details of the Windows services that you want to monitor but *cannot* the node where the Windows services are running.
- Modify

  Opens the Services Properties dialog box and allows you to check or modify an existing monitor for a Windows service.
- Copy

  Opens the Services Properties dialog box and allows you to copy an existing monitor for a Windows service. Use this option if you want to re-use conditions or actions, which you have already defined in an existing monitor for a Windows service.
- Delete

  Deletes the selected Windows services from the policy after a prompt for confirmation of the requested action.
- Defaults

  Opens the Defaults dialog box, where you can define the default actions to perform when a message is generated as a result of a broken threshold or a matched condition.

To set up a Windows service monitor policy

1. In the console, browse to the Service / Process Monitoring policy type:

   Policy Management → Policies grouped by type → Process / Service Monitoring .
2. Right-click the policy group and select New → Policy .

   The policy editor opens.
3. Use the buttons and pull-down menus to specify the Windows services that you want to monitor.

4. Save the policy.

Related Topics:

- Quick start: how to create a policy
- Add a service to monitor
- Windows service monitor properties
- Service monitor default actions

Add a Windows service to monitor

Use the Add New Services dialog box to specify which Windows services you want to monitor with the policy you are creating. If you know the name of the node where the Windows service you want to monitor is running, you can type it directly into the Node box. If you do not know the name of the node, click the [...] button to browse a list of known nodes. After you have selected the node, click Refresh to display a list of the Windows services running on the selected node.

NOTE
The policy editor attempts to connect to the node you type into the Node box as the Windows user who started the console. If this user is not allowed to access service information on the specified node, an error message appears.

To add a new Windows service to monitor

1. Select Services in the Monitoring drop-down menu in the Policy dialog.
2. If you know the name of the Windows service you want to add:
   a. Click New .
   b. Type the details of the Windows service to monitor.
   c. Click OK .
3. If you do not know the name of the Windows service:
   a. Click Browse to select from a list of available Windows services on the specified node.
   b. Type the name of the node in the Node box or click the [...] button to browse a list of nodes. The list of nodes is retrieved from the HP Operations database and does not indicate that a connection to the individual nodes in the list has been successful.
   c. Click Refresh .

      NOTE
      If the policy editor cannot connect to the selected node to retrieve a list of Windows services, it displays an error message or asks you for more information.

   d. Select the Windows service that you want to monitor.
   e. Click OK if you do not want to add any more Windows services, or click Apply if you want to continue adding Windows services to the monitor policy.

Related Topics:

- Quick start: how to create a policy
- Windows service monitor properties
- Service monitor default actions

Copy a Windows service monitor

Use the Copy button to copy an existing Windows service monitor so that you can then modify it (for example, if you want to re-use customized actions that you have defined for an existing Windows service monitor).

NOTE
The policy editor compares the string that you type in the Service Name box with the list of existing service names defined in the policy. It does not allow duplicate names or recognize the following special characters; \ (backslash), " (quotation mark), : (colon), & (ampersand), < > (angle brackets), ? (question mark), | (pipe), and spaces ( ).

To copy a Windows service monitor

1. Select the service that you want to copy in the list of existing Windows services displayed in the policy editor.
2. Click Copy .
3. Type the details of the new Windows service to monitor. By default, the policy editor inserts the word "Copy" in front of the service name.
4. Click OK .

   The new copy appears in the list of Windows-service monitors in the policy editor.

Related Topics:

- Quick start: how to create a policy

Modify a Windows service monitor

Use the Modify button to specify which Windows services you want to change or update (for example, to change parameters or threshold conditions or to specify your own, customized actions to associate with messages). The policy editor displays the Service Properties window, which you can use to change details or settings as required.

To modify a Windows service monitor

1. Select Processes in the Monitoring drop-down menu in the Policy dialog.
2. If you know the name of the process you want to add:
   a. Click New .
   b. Type the details of the process to monitor.
   c. Click OK .
3. If you do not know the name of the process:
   a. Click Browse to select from a list of available processes on the specified node.
   b. Type the name of the node in the Node box.
   c. Click Refresh .
   d. Select the process that you want to monitor.
   e. Click OK if you do not want to add any more Windows processes, or click Apply if you want to continue adding processes to the monitor policy.

Related Topics:

- Quick start: how to create a policy

Delete a Windows service monitor

Use the Delete button to specify which Windows service or services you want to remove from the list of Windows services in the selected policy.

NOTE
Remember to redeploy the modified policy if you change its contents (for example, if you remove any processes from the list of processes that are monitored by the policy). Otherwise, you will continue to receive messages associated with process monitors you thought you had removed.

To Delete a Windows service monitor

1. Select the Windows service monitor that you want to remove from the list of services in the policy editor.
2. Click Delete .
3. Click OK to confirm deletion of the selected Windows service monitor.

   The selected Windows service monitor is removed from the list of Windows services in the policy editor.
4. Save the policy.

Related Topics:

- Quick start: how to create a policy

Windows service monitor properties

Click the New button in the Service/Process Policy Editor to display the New Service dialog box, which you use to enter details of the services that you want to monitor with the Windows service policy. Use this option if know the details of the Windows service that you want to monitor but cannot access the node where the services are running.

Provide the following details:

- Service Name :

  Type the *real* name of the Windows service that you want to monitor. Duplicates or empty strings are not allowed. The name you enter here is checked against the list of existing Windows services that have already been defined for this policy.

  The policy editor does not try to establish whether the Windows service you specify exists (for example, because you have not typed the service name correctly). Select the Send message if Service does not exist option to ensure that HPOM informs you if the Windows service you specify here is *not* present when you deploy the policy to the node.

- Display Name :

  The Display Name is used in the policy editor for information purposes only. It is not used to identify the Windows service. By default, the Display Name is the name that Windows shows in the Name column of Services dialog box.

- Monitoring :

  Select the state that you want to monitor for the selected Windows service. For example, the default monitoring status "Running" checks whether the selected Windows service is running. Other states include "Disabled" and "Stopped". If the policy detects a change in state for the selected Windows service, it starts the actions defined in Actions, below.

- Actions :

  Choose whether you want to use the default actions defined for the Windows service, or specify your own, customized actions:

  - Default Actions :

    Click Defaults in the Policy dialog to specify that you want the Windows-service monitor to use the default actions defined for the policy.

  - Custom Actions :

    Click Custom defined if you want to define custom actions for a Windows-service monitor. If you check this option, you must use the options in the Start actions , Continue actions and End actions tabs to configure the actions that you want to occur when the policy finds a match. The actions tabs you use here are the standard HPOM tabs for defining a message and automatic or operator-initiated action.

To add a new Windows service to monitor

1. Select Services in the Monitoring drop-down menu in the Policy dialog.
2. If you know the name of the Windows service that you want to add:
   a. Click New .
   b. Type the details of the Windows service to monitor.
   c. Click OK .
3. If you do not know the name of the Windows service:
   a. Click Browse to select from a list of available services on the specified node.
   b. Type the name of the node in the Node box.
   c. Click Refresh .
   d. Select the Windows service that you want to monitor.
   e. Click OK if you do not want to add any more Windows services, or click Apply if you want to continue adding Windows services to the monitor policy.

Related Topics:

- Quick start: how to create a policy
- Windows service monitor default actions

Windows service monitor default actions

You define a "Start" action, a "Continue" action, and an "End" action. A start action is triggered when a threshold is reached or a policy rule matched. Continue actions are carried out at each subsequent polling interval if the reset value is not reached. End actions are carried out after the threshold crosses the reset value.

NOTE:
Actions attached to Windows service monitor policies can make use of session variables, which are illustrated in the examples below and described in more detail in Policy variables .

The default actions which are defined for the Windows service monitor policies are:

- Default *Start* Action
  - Send a message to the active message browser: YES
  - Message severity: CRITICAL
  - Message key:

    `<$NAME>:<$MSG_NODE_NAME>:<$MSG_OBJECT>:START`
  - Acknowledge message with message key:

    `<$NAME>:<$MSG_NODE_NAME>:<$MSG_OBJECT>:<*>`
  - Message text:

    `<$SESSION(SERVICEDISPLAYNAME)> is not <$SESSION(SERVICEMONITORSTATE)>. Current state is <$SESSION(SERVICECURRENTSTATE)>.`

    For example: `Service Telnet is not running. Current state is stopped.`
  - Operator-initiated action:

    <$SESSION(SERVICEACTION)>
- Default *Continue* Action

  No "continue" action is defined: do *not* start any "continue" action
- Default *End* Action
  - Send a message to the active message browser: YES
  - Message severity: NORMAL
  - Message key: `<$NAME>:<$MSG_NODE_NAME>:<$MSG_OBJECT>:END`
  - Acknowledge message with message key:

`<$NAME>:<$MSG_NODE_NAME>:<$MSG_OBJECT>:<*>`

- Message text:
  `<$SESSION(SERVICEDISPLAYNAME)> is <$SESSION(SERVICEMONITORSTATE)>.`

  For example: `Service Telnet is running.`

To set or change the default Windows service monitor actions

1. Create a new (or edit an existing) Windows service monitor policy.
2. Ensure that you select Services in the Monitoring drop-down menu in the Policy editor.
3. Click Defaults... .
4. Type the details of default actions that you want to occur when the defined policy threshold is matched. For example, a policy typically sends a message when a threshold is reached (and even reset) and starts an action to solve the problem to which the message relates.
5. Click OK .

Related Topics:

- Quick start: how to create a policy
- Policy variables

Agentless monitoring

Agentless monitoring is defined as monitoring an object without installing any additional software on it. A system that is managed in an agentless fashion with HPOM is called a monitored node. With an agentless solution, the proxy node receives data which is collected using remote calls to the monitored systems. See Deployment models for more information about managing proxy nodes and agentless nodes.

If the full capabilities of the HP Operations agent are not needed, for example, if the depth of information, command, and control requirements are limited, agentless monitoring can provide adequate monitoring while decreasing the effort for deployment, maintenance, and upgrade tasks associated with an agent-based solution.

Agentless monitoring with HPOM requires the help of an agent (*proxy agent*) installed on one managed node. This managed node acts as a *proxy system* to remotely access monitored nodes. Even though the policies for agentless monitoring belong to the monitored node, they are installed on the proxy system.

You can monitor the following event sources without an agent:

- SNMP (Simple Network Management Protocol)
- WMI (Windows Management Instrumentation)

NOTE:
There are further remote agentless monitoring possibilities. If you can remotely access and retrieve data from your systems using technologies such as `SSH` or `remsh`, you can feed this data into HPOM using the `opcmon` and `opcmsg` utilities.

Agentless monitoring with integrated products

One of the benefits of the HP BTO Software portfolio is the ability to integrate multiple products to solve more complex IT problems and for more flexible solution implementations. Remote monitoring of the IT infrastructure is a good example of how multiple HP BTO Software products can be used either individually to meet specific needs, or can be combined to provide a broader remote monitoring solution.

The following products integrate with HPOM to enable remote or agentless monitoring of network elements, systems, applications, and services:

- HP Network Node Manager

  HP NNM is primarily a remote-monitoring solution, in that it uses existing SNMP agents for network element discovery and status, without requiring additional code to be installed on the managed object.

  HPOM and HP NNM integration components and documentation are included with HPOM.

- HP SiteScope / HP Internet Services

  HP SiteScope and HP Internet Services are primarily remote-monitoring solutions. However, unlike HP NNM, they use a probe-based approach to query monitored nodes. HP Internet Services has been superseded by HP SiteScope.

  HPOM and HP Internet Services, as well as HPOM and HP SiteScope integration components and documentation are included with HP Internet Services.

Related Topics:

- Deployment models
- Identify the originating node
- Monitor SNMP devices
- Monitor WMI information
- HP NNM Adapter
- HP SiteScope Adapter
- License HP Operations Manager for Windows

Deployment models

There is no fixed upper limit for the number of nodes you can monitor within an HPOM environment. Scalability of agent-based or agentless monitoring in your environment primarily depends on the resources on the management server system, the managed nodes, and the network.

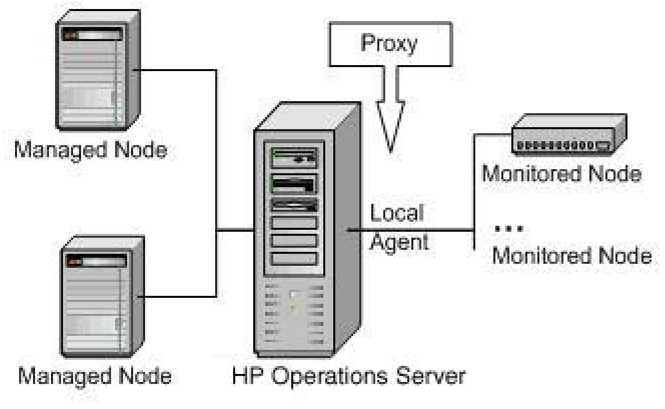
The effect agentless monitoring has on the CPU and memory utilization on the proxy system is dependent on the number of nodes, the number of metrics that are gathered and the frequency of gathering that is used. The network connection and bandwidth between the proxy and the monitored nodes can also influence the time needed for gathering the data.

Therefore, we recommend that you monitor resource utilization on the proxy system to avoid problems with other applications running on the proxy. If monitoring significantly affects other applications, reduce the gathering frequency, choose another proxy system with more resources or split the work among several proxy systems, sharing the monitoring of nodes and metrics, as described in Proxy systems in large environments

The location and arrangement of proxy systems in agentless monitoring can be divided into the following scenarios:

- HPOM management server system as the proxy system
- Independent proxy system
- Proxy systems in large environments

HPOM management server system as the proxy system



The most simple arrangement is to use the HP Operations agent running on the HPOM management server system as a proxy agent for remote monitoring, that is the HPOM management server system also acts as a proxy system.

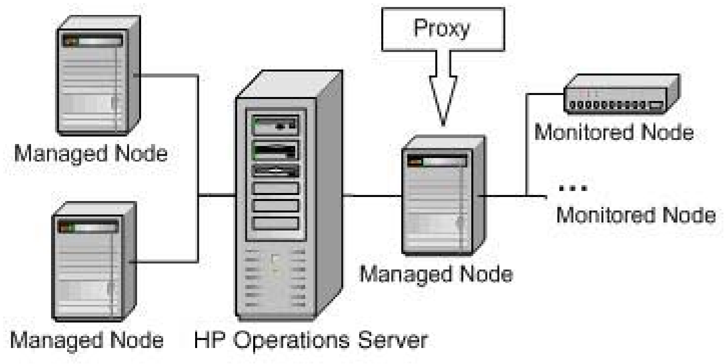
NOTE:

If you plan to install additional HP BTO Software products, such as HP NNM, HP Service Desk, HP Performance Manager, HP Reporter, or HP Internet Services on the system hosting the HPOM management server and proxy agent, make sure that the available system resources are sufficient for all installed products. Resource consuming processes reduce the performance of remote monitoring as well as the performance of the HPOM management server and other installed software.

The following recommendations apply:

- The maximum number of agentless monitored nodes should not exceed ten. This is not a hard-coded limit, but a recommendation to help ensure that resource utilization related to agentless monitoring does not result in a significant deterioration of the HPOM management server performance.
- The HPOM management server system has sufficient resources to deal with the additional work load from agent-based monitoring in addition to its regular tasks.

Independent proxy system

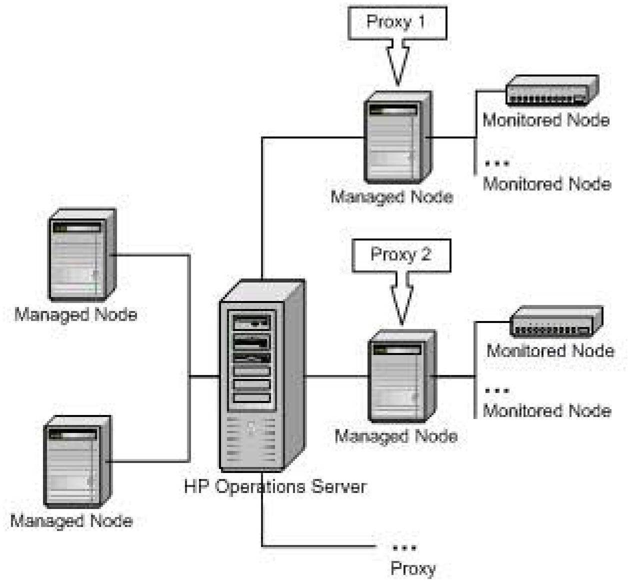


Using an independent proxy system is recommended when the HPOM management server system does not have sufficient resources to handle the proxy tasks without a noticeable impact on the performance of the HPOM management server. In this case, select a suitable managed node to play the role of the proxy system.

The following recommendations apply:

- The selected managed node has sufficient resources to deal with the additional proxy work load.

Proxy systems in large environments



Whenever the number of agentless nodes being monitored is a strain on a single proxy system, we recommend using additional proxy systems.

For each proxy system, the following recommendations apply:

- The selected managed node has sufficient resources to deal with the additional proxy work load.

NOTE:
Any combination of the possible proxy system configurations is acceptable, that is, you could have one proxy system being the HPOM management server system, and one or more proxy systems being any other suitable managed node.

Related Topics:

- Agentless monitoring
- Identify the originating node
- Monitor SNMP devices
- Monitor WMI information
- License HP Operations Manager for Windows

Identify the originating node

In agentless monitoring, events on the monitored nodes are detected and processed by policies on the proxy system. Based on the deployed policies, HPOM messages are generated and forwarded to the HPOM management server by the proxy agent.

It is important that whenever a message is generated, the HPOM management server can identify the node generating that message. Otherwise the message would be filtered out and would not have any influence on the status of an HPOM service. Therefore, it is very important that the incoming message belongs to one of the following:

- Proxy system (required)

  See Configure the proxy system .

- Monitored node (recommended)

  See Configure monitored nodes .

Configure the proxy system

The system used as the proxy system for agentless monitoring must be configured as a managed node in the HPOM management server.

If you only configure the proxy system as a managed node (and not the agentless nodes as shown in the next section), then you have to make sure that all messages that are created by agentless monitoring specify the proxy system as the originating node. This is required because all messages have to be associated with a managed node. If policies set the node property to another node, and this node is not configured as managed node in HPOM, then the corresponding message is discarded.

In addition, you can put the originating node name in the message, for example in the message text or the object box, so that you do not loose this information. This setup can be used if it is not necessary to distinguish between several monitored nodes, and if you do not use the monitored node names in HPOM service definitions. The proxy system is treated as the owner of the problem and messages are shown as messages belonging to the proxy system itself.

Configure monitored nodes

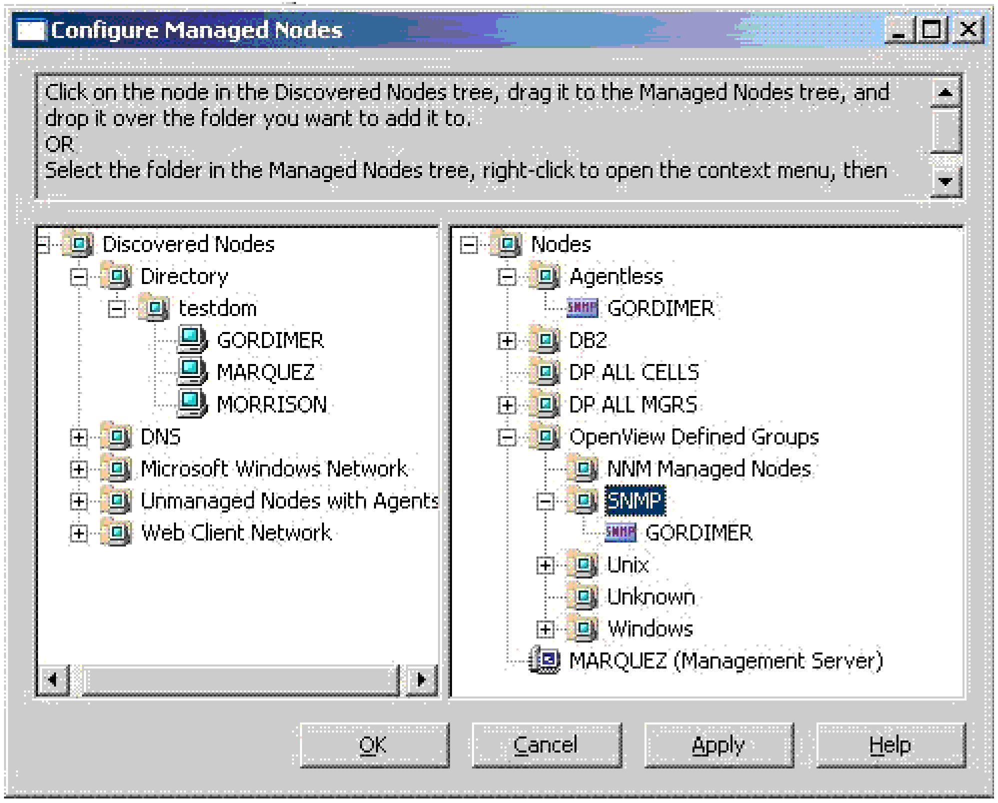
Besides the proxy server, which must be configured as a managed node in HPOM and has an HP Operations agent running on it, it is also recommended to also configure the monitored nodes in HPOM.

The advantage of configuring the monitored nodes in HPOM is that policies can set the node name to the originating node and messages will then show up as messages of the originating node in the message browser. It also allows you to create and use services that are hosted-on monitored nodes. You can configure agentless nodes in the following ways:

- Integrate discovered agentless nodes
- Add agentless nodes individually
- Configure external nodes

Integrate discovered agentless nodes

In most cases, the nodes that you should monitor agentlessly are already discovered by HP BTO Software integrated products such as HP NNM. You can normally find theses nodes within the Discovered Nodes groups, which are accessible from the Configure Managed Nodes dialog box:


Configure Managed Nodes
Click on the node in the Discovered Nodes tree, drag it to the Managed Nodes tree, and drop it over the folder you want to add it to.
OR
Select the folder in the Managed Nodes tree, right-click to open the context menu, then
Discovered Nodes
Directory
testdom
GORDIMER
MARQUEZ
MORRISON
DNS
Microsoft Windows Network
Unmanaged Nodes with Agents
Web Client Network
Nodes
Agentless
GORDIMER
DB2
DP ALL CELLS
DP ALL MGRS
OpenView Defined Groups
NNM Managed Nodes
SNMP
GORDIMER
Unix
Unknown
Windows
MARQUEZ (Management Server)
OK
Cancel
Apply
Help

They can be drag-and-dropped directly into the managed node list from the Discovered Nodes group.

Add agentless nodes individually

It is also possible to add a monitored node by right-clicking any group in the Nodes list to display the shortcut menu and select New Node from that menu. To manage devices such as printers, routers, computers with unsupported operating systems, and computers without the need for full HP Operations agent monitoring, you must ensure that they have an IP address or a primary node name.

The monitored nodes should be set up as follows:

- System type: Other
- Operating System: SNMP (version v1 or v2) or Unknown

  These two types of operating system start these nodes for HPOM management without an HP Operations agent installation.

  NOTE:
  For Microsoft Windows agentless nodes, avoid selecting any Windows operating system name. If a Windows operating system name is selected, automatic deployment is enabled by default, and an HP Operations agent is installed on the monitored node and it becomes a standard managed node.

Configure external nodes

It is also possible to add a monitored node by right-clicking any group in the Nodes list to display the shortcut menu and select New External Node from that menu. Specify a pattern that matches the fully qualified domain names (FQDN), IP addresses, or node names of the agentless nodes. The advantage of setting up agentless nodes as external nodes is that one external node can represent multiple agentless nodes. This means that fewer nodes must be configured.

Related Topics:

- Monitor SNMP devices
- Monitor WMI information
- Agentless monitoring
- Deployment models
- License HP Operations Manager for Windows

Monitor SNMP devices

HPOM is able to receive SNMP events generated by SNMP agents or applications and it can query MIB data using SNMP GET requests. The system can receive events using SNMP interceptor policies and monitor MIB variables using Measurement threshold policies.

The HP NNM integration for HPOM is an example of agentless monitoring with SNMP. It includes SNMP policies that allow HPOM to receive HP NNM SNMP events. The HP NNM server itself serves as a proxy for agentless monitoring. The HP NNM Adapter installation configures the HP NNM server as a managed node in HPOM and automatically deploys SNMP interceptor policies to the HP NNM server. This is the first step in enabling agentless monitoring of the HP NNM monitored nodes. The next step requires the HPOM administrator to configure the nodes in HPOM using the node configuration editor as described in Configure monitored nodes individually . There are also many predefined services and tools for use with HP NNM monitored nodes.

The monitored nodes are discovered in a separate HP NNM discovered nodes group and can be manually added to HPOM. With the HP NNM Adapter, monitored nodes are not configured with the type SNMP, but with their discovered operating system type. Therefore, make sure that you change the operating system type if you do not intend to install an agent on these nodes.

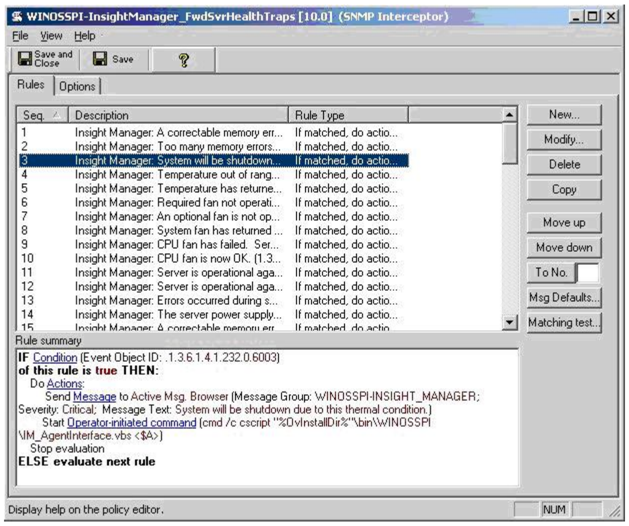
NOTE:
The HP NNM integration is an example of a solution where the HP NNM server acts as proxy, generating messages for all nodes managed by HP NNM. These monitored nodes must be configured in HPOM. Otherwise, HP NNM-related messages of those nodes will not be received by the management server.

SNMP event monitoring

The SNMP interceptor policy is a remote monitoring policy, as the SNMP interceptor is able to receive events from any system in the environment and not just the system where the SNMP interceptor is running. SNMP policies are used, for example, for the HP NNM integration, for Novell monitoring, or with HP Systems Insight Manager.

The SNMP interceptor policy type monitors SNMP events, and allows filtering based on originating node, event object ID or when a character pattern that you choose is found in an SNMP event.

The following example shows how one SNMP interceptor policy is able to monitor large volumes of SNMP events after establishing a small number of rules:



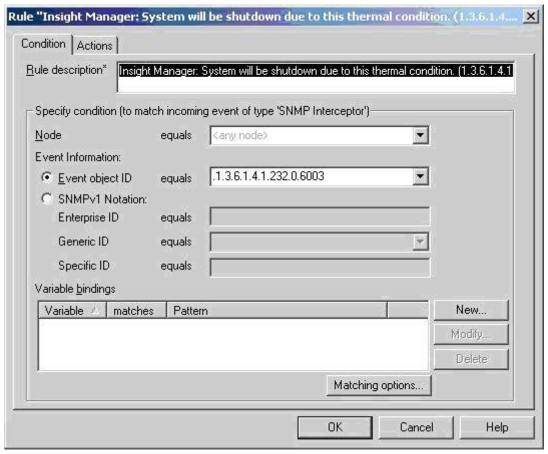
Seq.	Description	Rule Type
1	Insight Manager: A correctable memory err...	If matched, do actio...
2	Insight Manager: Too many memory errors...	If matched, do actio...
3	Insight Manager: System will be shutdown...	If matched, do actio...
4	Insight Manager: Temperature out of rang...	If matched, do actio...
5	Insight Manager: Temperature has returne...	If matched, do actio...
6	Insight Manager: Required fan not operati...	If matched, do actio...
7	Insight Manager: An optional fan is not op...	If matched, do actio...
8	Insight Manager: System fan has returned ...	If matched, do actio...
9	Insight Manager: CPU fan has failed. Ser...	If matched, do actio...
10	Insight Manager: CPU fan is now OK. (1.3...	If matched, do actio...
11	Insight Manager: Server is operational aga...	If matched, do actio...
12	Insight Manager: Server is operational aga...	If matched, do actio...
13	Insight Manager: Errors occurred during s...	If matched, do actio...
14	Insight Manager: The server power supply...	If matched, do actio...
15	Insight Manager: A correctable memory err	If matched, do actio

Rule summary

```
IF Condition (Event Object ID: .1.3.6.1.4.1.232.0.6003)
of this rule is true THEN:
   Do Actions:
       Send Message to Active Msg. Browser (Message Group: WINOSSPI-INSIGHT_MANAGER;
Severity: Critical;  Message Text: System will be shutdown due to this thermal condition.)
      Start Operator-initiated command (cmd /c cscript "%OvInstallDir%"\bin\WINOSSPI
\IM_AgentInterface.vbs <$A>)
   Stop evaluation
ELSE evaluate next rule
```

Display help on the policy editor.

Each rule looks for a certain event object ID:


Rule "Insight Manager: System will be shutdown due to this thermal condition. (1.3.6.1.4....
Condition
Actions
New...
Modify...
Delete
Matching options...
OK
Cancel
Help

Rule description*: Insight Manager: System will be shutdown due to this thermal condition. (1.3.6.1.4.1

Specify condition (to match incoming event of type 'SNMP Interceptor')

Node	equals	<any node>
Event Information:		
(●) Event object ID	equals	.1.3.6.1.4.1.232.0.6003
( ) SNMPv1 Notation:		
Enterprise ID	equals	
Generic ID	equals	
Specific ID	equals	

Variable bindings

Variable	matches	Pattern

NOTE:

It is recommended that a specific SNMP interceptor policy is only deployed to one proxy system.

Node in rule condition

If you leave the Node box blank, that is <any node> in the Rule Condition dialog box, SNMP events of all nodes are matched. If you want to limit the number of monitored nodes, you can specify any of the following:

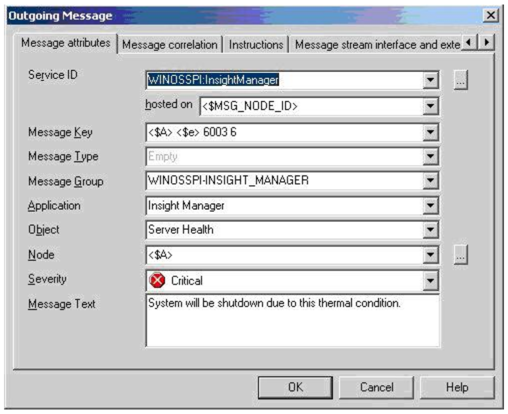
- Monitored node name
- Multiple node names with the OR operator |

- Multiple node names using variables, for example, `<$OPC_MGMTSV>`

Node in outgoing message

It is also recommended to insert SNMP variables into HPOM messages that show the originating node, for example, `<$A>` is the variable containing the hostname of the originating host in the following example for the HP Systems Insight Manager. The variables differ from application to application, so carefully check the SNMP event and variable definitions.

The following example shows how the variable `<$A>` is used to set the node property of the outgoing message. This assumes that the originating node is configured as a managed node. See Configure monitored nodes individually for details.

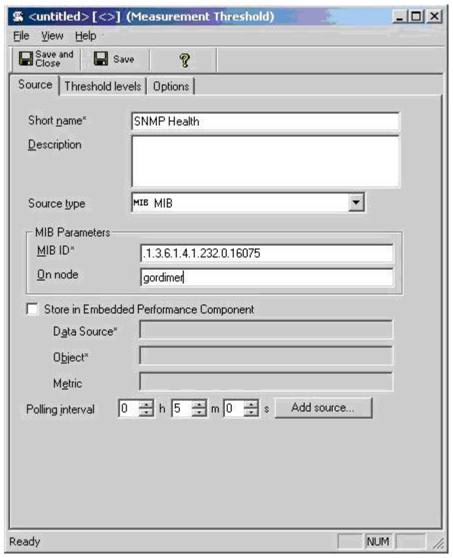


Service ID	WINOSSPI:InsightManager
hosted on	<$MSG_NODE_ID>
Message Key	<$A> <$e> 6003 6
Message Type	Empty
Message Group	WINOSSPI-INSIGHT_MANAGER
Application	Insight Manager
Object	Server Health
Node	<$A>
Severity	Critical
Message Text	System will be shutdown due to this thermal condition.

SNMP MIB Monitoring

The Measurement Threshold policy type includes the source type MIB , which allows thresholding on values for a specified MIB on a selected node.

If source type MIB is selected, an SNMP GET is performed on the specified object ID (OID). By default, the collection is made on the local managed node but can be made remotely by specifying the optional hostname. The following example shows how the MIB with the OID `.1.3.6.1.4.1.232.0.16075` is monitored on node `gordimer` :



Short name* SNMP Health

Description

Source type MIB

MIB Parameters

MIB ID* .1.3.6.1.4.1.232.0.16075

On node gordimer

☐ Store in Embedded Performance Component

Data Source*

Object*

Metric

Polling interval 0 h 5 m 0 s

Related Topics:

- Monitor WMI information
- Agentless monitoring
- Deployment models
- Identify the originating node
- Quick start: how to create a policy
- License HP Operations Manager for Windows

Monitor WMI information

Windows Management Instrumentation (WMI) is a component of Microsoft Windows operating systems and can be used to manage systems without installing an HP Operations agent. It provides standardized means for managing computer systems in an enterprise environment locally and remotely. WMI enables monitoring and controlling of managed objects, which complements system management with HPOM for Windows.

The WMI Interceptor and the Measurement Threshold policy type (source type WMI) can both access remote systems. For each policy type, you can specify the remote node as well as the account that the policy uses to access the WMI database of the remote node.

The remote node must be configured as follows:

- DCOM enabled

  DCOM must be enabled on the remote node.
- Local administrator privileges

  The specified user must be a member of the local Administrators group on the remote note.

Related Topics:

- Agentless monitoring
- Deployment models
- Identify the originating node
- Monitor SNMP devices
- Select WMI source
- Set threshold source properties
- Quick start: how to create a policy
- License HP Operations Manager for Windows

Configure Rules

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and an action . The condition is the part of an event policy that describes the type of event in the source that will trigger an action and the action is the response that the policy should take if an event that matches the condition occurs. A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

Set policy rule properties

With this tab you create , modify and delete the individual rules that make up the policy . You also indicate the order in which the rules are evaluated, and can view some of the rules properties. The order in which the rules are evaluated can have a big influence on the type and number of messages that the policy generates.

The Rules tab also provides a textual summary of each rule . You can use this summary as a navigational aid—click any underlined text in the rule to jump to the window that allows you to set that value.

CAUTION:
The advanced threshold policy rules tab provides the ability to specify whether the rules define a maximum or minimum threshold, or use scripts. A Measurement Threshold policy can only contain one of these three types of rules. Note that a conversion between threshold types is not always possible:

- changing between minimum and maximum: rules are not deleted
- changing from minimum or maximum to VisualBasic or Perl: the rules are converted to script
- changing from VisualBasic or Perl to minimum or maximum: rules are deleted
- changing between VisualBasic and Perl: no conversion occurs, you must rewrite the script

Set Policy Rule Properties

1. Right-click the policy and select All Tasks → Edit...
2. Select Rules .

Related Topics:

- Quick start: how to create a policy

Change the rule order

The order in which rules are evaluated has a large effect on the type of messages you receive. It also affects the speed with which messages are sent and the amount of processor time that is required by the policy .

For example, you might have a policy that monitors CPU activity, containing these two rules:

1.
   ```
   If usage is greater than 80%,
   send a warning message and stop processing rules.
   ```
2.
   ```
   If usage is greater than 95%,
   send a critical message and stop processing rules.
   ```

If the rules were evaluated in the order shown, disk usage of 99% would only produce a warning message. If the order were reversed, however, a critical message would be sent. You could solve the problem by making the rules more specific, so that the order was not important:

1.
   ```
   If usage is between 80% and 94%,
   send a warning message and stop processing rules.
   ```
2.
   ```
   If usage is greater than 95%,
   send a critical message and stop processing rules.
   ```

In the example above, disk usage of 99% produces a critical message regardless of which rule is evaluated first. However, if the rules are evaluated in the order shown, disk usage of 99% is evaluated by two rules. If the order were reversed, it would be evaluated only by the first rule, thereby sending the message more quickly and reducing processing time on the managed node .

To change the rule order

1. Right-click the policy and select All Tasks → Edit...
2. Select a rule you want to move, and select Move Up or Move Down .

Related Topics:

- Quick start: how to create a policy

Create a rule

1. Right-click the policy and select All Tasks → Edit...
2. Select Rule.
3. Select New.
4. Select the conditions and actions for the rule.

Related Topics:

- Event log rule conditions properties
- Logfile rule conditions
- Open Message Interface policy rule conditions properties
- SNMP rule conditions properties
- Windows measurement instrumentation rule conditions properties
- WMI rule conditions properties
- General threshold rule properties
- Set message defaults for all new rules
- Quick start: how to create a policy

Delete a rule

1. Right-click the policy and then click All Tasks → Edit...
2. Click the rule that you want to delete.
3. Click Delete.

Related Topics:

- Quick start: how to create a policy

Modify a rule

1. Right-click the policy and select All Tasks → Edit...
2. Select the rule you want to modify, and select Modify .
3. Change the conditions and actions for the rule.

Related Topics:

- Set message defaults for all new rules
- Quick start: how to create a policy

Configure conditions

Conditions are the part of a rule that define what kind of event will trigger an action . For example, In a Windows Event Log policy, a condition could be the following string in the Description text box: The `<1*>: disk is at or near capacity` . This condition would match this entry in the event log for any disk on the managed node. Another example is a specific SNMP trap, or the creation of a specific WMI intrinsic event. This section of the help explains how to define conditions for event policy types.

Set log file rule conditions

This tab lets you specify a string that the policy searches for in the log file that the policy monitors. If the policy finds a match, the actions associated with this rule are carried out.

In the Log file line box, type in the pattern that you want the policy to compare with the source that it is evaluating. When this pattern is matched, the actions are carried out.

If you only want to match the log file from a specific node, type the FQDN , the primary node name, or the IP address in the Node box. Give multiple entries with the OR operator (for example, celery.veg.com|broccoli.veg.com) or leave the field blank for all nodes. You can also use the variable <$OPC_MGMTSV>.

You can use message text pattern conventions to specify the matching pattern. For more information about pattern matching, see Pattern-matching and variables

Parts of the matched string can be built into the message displayed in the message browser by defining variables. For more information about variables see, Variables and parameters . You can use the Launch Log file button to browse to a log file and then open it to inspect or copy values.

To set log file rule conditions

1. Right-click the policy and select All Tasks → Edit...
2. Select Rules .
3. Select the rule for which you want to modify the conditions.
4. Select Modify.
5. Select Conditions and type the conditions for this rule.

Related Topics:

- Set log file source properties
- Quick start: how to create a policy

Threshold rule condition (instance filter)

Instance filters provide a way for the measurement threshold policy to apply different sets of threshold levels to different instances of the object being monitored. For example, a threshold policy that monitors disk usage will apply the same threshold to all disks, but if you specify instance filters, you can specify one set of threshold levels for disk C:, another set for disk D: and so on.

Instance filters can be used with policies that evaluate the threshold based on MIN, MAX or scripts. Instance filters are not available for threshold policies based on the source MIB.

To create instance filters:

1. Provide a description of the rule (for example, *matches the C drive*).
2. In the Object name text box, type a pattern matching string that will match the instance (or instances) for which you want to write specific rules.
3. Select the Actions tab and create the General Threshold Rule Properties for this instance.
4. Repeat for each instance.

NOTE:
If you don't know what the instance names are, you can use this trick to find them. Write a policy with a very low threshold (one that will be broken immediately), and deploy it onto a node . The message that you receive will show the name of the instance in the object field of the message properties.

Related Topics:

- Quick start: how to create a policy
- Set threshold source properties

Set general threshold rule properties

NOTE:
The general rule properties tab is only valid for the Measurement Threshold Policy.

In this tab you set the threshold level for the rule, a minimum time for which the threshold must be broken to produce a message , and a reset level.

- Threshold level description : This is a name you give to the rule to help you identify it. This name is visible in the rules list.
- Threshold Limit : If you are setting a Minimum or Maximum rule, set the value that triggers a message if met or crossed.

  Use the following syntax guidelines when specifying the threshold:

Sequence of Digits:	May include a decimal separator. (The character used as the separator is determined by the operating system language.) For example: 0.5, 100.1
Sign (optional):	Plus sign (+). For example: +50 Minus sign (-). For example: -730
Exponent (optional):	Exponent character: e or E. For example: 15e2, 7E4 Exponent sign. For example: 8e+2, 4E-2 One or more decimal digits. For example: 25.88e4

  TIP:
  If you set a minimum or maximum threshold limit, you can override it for individual nodes if necessary.

  If you are setting a Visual Basic© Scripting Edition or Perl rule, you must write a script that evaluates the data you are monitoring and sets the Rule Object to either TRUE or FALSE. See Policy objects and examples for information about writing threshold scripts.

- Short-term peaks Since it may not be reasonable to create a message when a threshold is exceeded only for a short time, HP Operations allows you to define a minimum time period over which the monitored value must exceed the threshold before generating a message. For a message to be sent, the value must be greater than the threshold each time the value is measured during a duration that you select. If the duration is set to 0 or the box is left empty, an alarm is generated as soon as HP Operations detects that the threshold has been equaled or crossed.
- Reset The reset value is a limit below which the monitored value must drop (or exceed, for minimum

thresholds) to return the status of the monitored object to normal. After the status of a monitored object returns to normal, a new start message can be issued if the monitored value again crosses the threshold value. You can either use the same value as the threshold limit, or specify a different reset value.

Set the Measurement Threshold general rule properties:

1. Right-click the policy and select All Tasks → Edit...
2. Select Threshold levels .
3. Select the rule to which you want to modify the conditions.
4. Select Modify.
5. Select General .

Related Topics:

- Set threshold source properties
- Quick start: how to create a policy

Set event log rule conditions properties

This tab lets you specify an entry in the Windows event log that causes the rule to carry out the actions that are associated with the rule. If the policy finds an event log entry that matches what you specify here, the actions that you specify in the action tab are carried out.

- Rule Description : Type a description that will help you remember what this rule does. This description is visible in the rules list.
- Computer , Source , Category , Type , Event ID , Format , Description : In these text boxes, indicate the content of these event log fields that you want this rule to match. Note that you can start the event viewer with a button at the bottom of the window if you want to copy data from the event log.

  - Node names

    If you only want to match event log entries from a specific node, type the FQDN , the primary node name, or the IP address in the Computer field. Give multiple entries with the OR operator (for example, celery.veg.com|broccoli.veg.com) or leave the field blank for all nodes. You can also use the variable <$OPC_MGMTSV>.

  - Pattern matching

    Pattern matching may be used in the Description field. The match patterns may not contain newline characters. If you need to match a multi-line pattern, use the special character <*> to match any carriage return/linefeed characters.

    You can also use pattern matching in the Source field, but you must first enable this on the nodes that you want to use it on.

  - Policies generated from existing HP Operations Manager for UNIX templates

    Policies generated from existing HP Operations Manager for UNIX templates have a combined Event ID and Description text box. You can convert to two text box format by selecting Use combined format for event ID and description and then saving the policy and re-opening. Note that you must re-type the information in the two new boxes.

Parts of the matched string can be built into the message displayed in the message browser by defining variables. For more information about variables see, Pattern-matching and variables

Set Event Log Rule Conditions

1. Right-click the policy and select All Tasks → Edit...
2. Select Rules .
3. Select the rule to which you want to modify the conditions.
4. Select Modify.

5. Select Condition and choose the conditions for this rule.

Related Topics:

- Set Windows event log source
- Quick start: how to create a policy

Set WMI rule conditions

With this tab you specify the conditions for a Windows Management Interface Policy rule. Conditions for these rules are sets of WMI event or instance properties, along with values that these properties must have in order for a match to be successful.

- Property name : Select the property that you want the rule to inspect. (You can also type the name of the property if you know it. Note that properties must begin with a letter.)
- Property type : Indicate the property of the selected class that you want to inspect. If the property is an array (for example, as in _DERIVATION), you should indicate whether the value must be present in all elements, in one element, or in one specific element.

  In the case that the property is a reference to another class, and you want to reference a property of this subclass, use the format `subClass.PropertyName` .
- Operator : Select the comparison operator that you want to use.
- Select value or property : Indicate whether you want to type the value to be compared, or whether you want to use another property as the comparison value, then either type the value, or select the property.
- Specific value to compare :Type the value (or property) that you want to compare. This is the value or property that will be compared—using the comparison operator you selected—against the property selected under Property name . (You can also type the name of the property if you know it. Note that properties must begin with a letter.)

Set WMI Rule Conditions:

1. Right-click the Windows Management Interface policy and select Edit .
2. Select Rules .
3. Select New .
4. Select Add .

Related Topics:

- Quick start: how to create a policy
- Select WMI source

Set open message interface policy rule conditions

HP Operations can integrate messages generated by its own message command, opcmsg . If the Open Message Interface policy is not installed on the computer where opcmsg generates messages, all messages generated by opcmsg are sent directly to the management server . The HP Operations Open Message Interface policy filters messages into HP Operations by defining match conditions for these messages. If a message matches, the policy sends a message to the management server. All other messages are suppressed.

Rule Description : Type a description that will help you remember what this rule does. This description is visible in the rules list.

Node : Type the FQDN , the primary node name, or the IP address if you only want to match messages generated on a specific node. Give multiple entries with the OR operator (for example: `kohlrabi.veg.com|beet.veg.com`), or leave blank for all nodes. You can also use variable `<$OPC_MGMTSV>` .

Message Group : Type the message group that you want to assign to messages generated by this rule. Give a message group or leave blank for all message groups. A message can belong to only one message group. You can use multiple message groups to set the filter in the policy's rule condition, to select the messages that will arrive.

Application : Type the name of the application that generated the message.

Object : Type the name of the object that generated the message.

NOTE:
Although the term *application* generally refers to a general program name and *object* generally refers to a process or sub-program, you should use these values to assist your own organizational scheme.

Severity : Select the severities that the message should have.

Message Text : Type in the pattern that you want the policy to compare with the message text in the source message that it is evaluating.

To set open message interface policy rule conditions

1. Right-click the Open Message Interface policy and select All Tasks → Edit...
2. Select Rules .
3. Select the rule to which you want to modify the conditions.
4. Select Modify.

5. Select Condition and choose the conditions for this rule.

You can use message text pattern conventions to specify the matching pattern. For more information about pattern matching see, Pattern-matching and variables .

Related Topics:

- Quick start: how to create a policy

Set SNMP rule conditions properties

With this tab, you set the match conditions for an SNMP interceptor rule .

- Rule Description : Type a description that will help you remember what this rule does. This description is visible in the rules list.
- Node : If you only want to match SNMP events from a specific node, type the FQDN , the primary node name, or the IP address. Give multiple entries with the OR operator (for example, `celery.veg.com|broccoli.veg.com`), or leave blank for all nodes. You can also use variable `<$OPC_MGMTSV>` .
- Event Information
  Select this option to change the way in which the event information appears. The option you choose determines what fields are visible in the window and what information you have to supply. If you want to match a specific event, select Event Object ID . If you want to match a range of events or specify a specific event in SNMPv1 Notation , select the SNMPv1Notation option.
  - Event object ID

    Type the complete Event Object Identifier for the SNMP event that you want to match.

    For example: .1.2.6.1.4.1.11.2.17.1.0.40000001
  - SNMPv1 Notation

    You can type the complete event object ID in SNMPv1 format or you can specify only part of the identifier. For example, by specifying only the Enterprise ID, you can match all events with a specific Enterprise ID.
  - Enterprise ID

    Type in the enterprise ID for incoming SNMP traps to be compared with this condition. The enterprise ID is a vendor-specific identifier for the trap. Standard HPOM pattern-matching syntax may not be used in this field; however, it is possible to match a range of objects by entering only a prefix. For instance, the pattern:

    .1.3.6.1.4.1.11.2.17

    would match:

    .1.3.6.1.4.1.11.2.17.1

    .1.2.6.1.4.1.11.2.17.2

    and so on.
  - Generic ID

    From the list, select the appropriate Generic Trap ID. Possible values are:

- (0) ColdStart
- (1)WarmStart
- (2) LinkDown
- (3) LinkUp
- (4) Authentification
- (5) EgpNeighborLoss
- (6) EnterpriseSpecific
- (7) don't care

If you select (6) EnterpriseSpecific , you can type in the specific trap ID. Select don't care to intercept any kind of trap.

- Specific ID

  Type in the specific trap ID if you have selected (6)EnterpriseSpecific in Generic Trap. Enterprise-specific SNMP traps can be implemented by vendors on their specific network devices. The specific trap ID is used to identify the source of the trap.

NOTE:
The SNMP syntax used by the editor requires that the trap string being with a point. If you forget to add the point at the beginning of the string, the program will add it for you when you save the policy.

- SNMP V2 Traps

  Follow these steps to specify that the Windows agent can receive SNMPV2 traps:

  a. Disable the Windows SNMP Service/SNMP Trap Service.

  b. Add the following line to opcinfo:

     SNMP_SESSION_MODE NNM_LIBS

  c. Restart the agent.

- Variable Bindings

  Select the variable bindings you want the policy to monitor, and write one or more match patterns for each binding. Standard HP Operations pattern-matching rules can be used when matching variable bindings.

  $1 represents the first variable binding in the event, $2 the second variable, and so on. Use the matching options button to set the case sensitivity and field separators for all variable bindings.

Set SNMP Rule Conditions Properties:

1. Right-click the policy and select All Tasks → Edit...
2. Select Rules .

3. Select the rule for which you want to modify the conditions.
4. Select Modify.
5. Select Condition and type the conditions for this rule.

Related Topics:

- SNMP Variables
- Quick start: how to create a policy

Manage Intelligent Devices

Using SNMP , HP Operations can manage intelligent SNMP devices where policies cannot be installed (for example, printers, routers, computers with unsupported operating systems). Using the procedure below, such devices can be integrated into and managed from HP Operations.

To manage intelligent devices

1. Ensure that the device you want to manage has an IP address and is SNMP-enabled. Only devices that meet these conditions can be managed in this way.
2. Add the device to the list of managed nodes . In the System tab, set Operating System to SNMP , and Version to v1 .
3. If you will be monitoring traps, (and not MIB data), configure the device to send the SNMP traps to the managed node where you intend to distribute the policy that will monitor the traps.
4. Write an SNMP policy that with rules that match the SNMP events that you are looking for (or a threshold policy to monitor MIB variables), with appropriate actions. The messages can be configured to regard the source of the SNMP messages as a service, so that the device can be added to the service hierarchy .
5. Distribute the policy to the node to which the intelligent device is sending the SNMP traps (or to the node that will access the MIB data from the intelligent device).

Configure actions

Actions are the part of a rule that define what the policy should do if it detects an event that matches (or does not match) the rule's condition . The action can consist of one or more of the following:

- Send a message to the active message browser

  This is the most common action. A message notifies the operator that an event has occurred, can supply the operator with instructions for responding to the event and can provide a link to a command that the operator can run. You can also configure messages acknowledge or suppress duplicates .

- Send a message to the acknowledged messages browser

  You may want to send a message to the management server for tracking reasons, but don't need to have the user take any action or acknowledge the message. In this case, you can send the message directly to the acknowledged messages browser.

- Automatically run a command

  Certain events could represent conditions that can always be helped by running a certain command. If you know that a command should always be run in response to an event, you can configure an automatic command .

- Nothing

  In some circumstances, you may want a rule to stop the policy from evaluating the event (to improve performance) without providing the operator with any notification.

NOTE:
The maximum supported string length for an operator-initiated or automatic command is 2048 characters. Be particularly careful to stay within this limit when using variables substitution within the action call.

Configure policy actions

In the actions tab, you indicate what the policy should do after evaluating a particular rule . The policy can send a message to the management server , start a command, prepare a command for the operator to start, or any combination or none of these actions.

NOTE:
In all cases, if a rule evaluates as true, no more rules are processed. It is important to pay attention to the rule order .

The selections that you make in this tab determine the rule type for this rule. The three types are:

- If matched, do actions and stop.
- If matched, stop.
- If not matched, stop.

Select what the policy should do according to how this rule evaluates. Then, as required, set up the Message and the Automatic command or Operator-initiated command .

To access this tab:

1. Right-click the policy and select All Tasks → Edit... .
2. Select Rules .
3. Select the rule for which you want to set up or modify actions.
4. Select Modify... .
5. Select Actions .

Related Topics:

- Change rule order
- Create a rule
- Unmatched events
- Quick start: how to create a policy

Configure message attributes

This tab allows you to set the message attributes for a specific message (or for the message defaults). These attributes are visible in the message browser and help the operator to organize and evaluate the messages. Note that not all message attributes can be set for the default message and that some attributes are not supported by some policy types .

- Service ID : A service ID is a unique identifier for a service in your service hierarchy. When a message with a service ID is sent to the management server, the severity of that message will be included in the status calculation for the service that has the matching service ID.

  Type the ID of the service that you want to associate with this message. (The service IDs are assigned to services in the Service Editor.) To find and insert a service ID, you can click the browse button (...), browse to the service to which this message will belong, and select OK . This action will also fill in the Hosted on text box (if the service is not a virtual service). Generally, it is best to use this method to fill in these two text boxes. Changing the node name in the Hosted on text box is only recommended if you are writing a policy that will be used for more than one service hierarchy , when you could use a variable instead of a node name.

- Message Key : A message key is an identifier that you can assign to messages so that other processes can identify them. Note that the message key is different from the message ID: while every message has a unique message ID, all messages produced by a particular rule will have the same message key. The same message key can also be used by more than one policy or rule.

  Since you assign the message keys, you know in advance what key a particular message will have, and can set up other processes to look for this key. If you use messages keys, it is important to develop naming conventions that allow you to specifically identify an event or sets of events.

  You can incorporate HP Operations variables into the message key. For example, if you want to ensure that messages generated on one computer have a different key from messages generated on another computer, you can include the variable <$MSG_NODE_NAME> as a part of the message key. A useful message key could look like this:

  ```
  <$NAME><$MSG_NODE_NAME><$MSG_OBJECT>:START<$THRESHOLD>
  ```

- Message Type : Use this box if you want to provide another organizational category for this message, for example, if you want to indicate two types of messages that belong to the same message group.

- Message Group : Type the message group that you want to assign to messages generated by this rule . Message filters in the console and web console limit the message group length to 32 characters (the same length limit of the field in messages).

  Every message that the management server receives is assigned to a message group. Message groups organize messages belonging to the same policy or having some logical connection, for example, messages from backup and output tasks.

- Application :Type the name of the application which generated the message.
- Object :Type the name of the object which generated the message.

  NOTE:
  Although, *application* generally refers to a general program name and *object* generally refers to a process or sub-program, you should use these values to assist your own organizational scheme.

- Node : Type the name of the node to which the message will be assigned. You can also use the default `<$MSG_NODE> (Local Node)` , or use the variable `<$OPC_MGMTSV>` for the management server node. For agentless monitoring, type the name of the originating node.
- Severity : Choose the severity that you want the message to have. The severity indicates to the operator the importance of the event which triggers this message.
- Message Text : Type the text that the message should have. You can use HP Operations variables to construct messages that are specific to the event that causes the message. See HP Operations Policy Variables and User-defined variables .

  You can add hyperlinks to URLs in your message or instruction text. Insert your cursor in the text of an active message on the Text tab and type the required URL. Apply and save your changes. This feature is useful for linking to external sites of all types, such as support sites, documentation repositories, troubleshooting information, and similar sites.

To configure message attributes:

1. Right-click the policy and select All Tasks → Edit... .
2. Select Rules .
3. Select the rule for which you want to change the message attributes.
4. Select Modify .
5. Select Actions .
6. Select Message .
7. Select Message attributes .

Related Topics:

- Quick start: how to create a policy
- Set message defaults
- Agentless monitoring

Configure message correlation

Message correlation helps to prevent your message browser from becoming cluttered by messages that describe the same problem. When message correlation is enabled, you can set the type of duplicate message suppression and define the method used to suppress duplicate messages. Note that for Measurement Threshold and Scheduled Task policies , only the first item (Acknowledge messages with message key) is available.

- Acknowledge messages with message key : If messages with the message key that you type here exist in the active messages browser when this message is received by the management server, these messages are automatically acknowledged. You can use the pattern-matching and variables to match multiple message keys. For example, consider the following pattern:

  `<$MSG_SEV>:<$MSG_NODE_NAME>:<_><5*>`
  This pattern is evaluated by first replacing the variables with the value that they resolve to, for example:
  `critical:cabbage.veg.com:<_><5*>`
  This pattern is then compared—using pattern matching rules—against the message keys for all messages in the active message browser. The pattern above would match the following message keys:

  ```
  critical:cabbage.veg.com:    12345
  critical:cabbage.veg.com:  TEST1
  ```

  When writing patterns for this box, note the following:

  - Although user-defined variables can be included in this box, these variables can only be expanded after the policy is deployed and therefore are not included in the syntax check that is performed when the policy is saved. Because of this, it is important to ensure that any user-defined variables in this box are correctly used.
  - Pattern-matching test has limited usage because if the 'Patterns to test' contains variable the replacement of those cannot be simulated.

  NOTE:
  For threshold policies, you can specify automatic acknowledgement of messages related to the same policy by using Automatic escalation of threshold levels, set in the Options tab.

Suppress messages which are: This setting lets you choose what kind of messages are considered to be identical.

- Generated by same rule
  Select this option to suppress messages that match the message text pattern specified for the selected rule. This is a more general setting for the suppression of duplicate messages. For example, a logfile

entry policy might contain a rule with this match pattern: `Error Message<# >` The logfile lines `Error Message10` and `Error Message20` are not identical, but would both match this rule.

- Generated by the same input event. Select this option to suppress messages that were sent in response to two separate events that are identical except for the date and time that the event was generated (for example, identical entries in a log file).
- Identical relative to their attributes
  Select this option to suppress either messages that have the same message key or (if no message key is present) messages that have identical message attributes (except for the date and time that the message was generated).

Suppression Method

For message correlation, you can define one of three correlation methods:

- Time interval: This correlation method lets you define an interval during which duplicate events will be ignored. For more information, see Message suppression simulation for an interactive demonstration of these concepts, or read this detailed example.
- Counter: This correlation method counts the number of matching events and sends a message only after the number of matching events equals the counter threshold. The counter can also be reset to zero after a time period that you specify. For more information, see Message suppression simulation for an interactive demonstration of these concepts, or read this detailed example.
- Time interval/Counter If you use the Time interval and Counter together, events are evaluated first by the timer. If an event passes the timer, it is then evaluated by the counter, which either suppresses it or sends a message to the management server .

To configure message correlation

1. Right-click the policy and select All Tasks → Edit... .
2. Select Rules .
3. Select the rule for which you want to correlate messages.
4. Select Modify .
5. Select Actions .
6. Click Message .
7. Select Message correlation .

Related Topics:

- Message suppression simulation
- Quick start: how to create a policy

Add custom message attributes

Custom message attributes (CMAs) are additional attributes that contain any information that is meaningful to you. For example, you might add a company name, contact information, or a city location to a message. You can have more than one CMA attached to a single message.

This attribute information displays in the message browser in a column you have previously created to contain it.

To add custom message attributes

1. Right-click the policy and select All Tasks → Edit...
2. Select Rules .
3. Select the rule to which you want to add custom message attributes.
4. Select Modify .
5. Select Actions .
6. Select Message .
7. Select CMAs .
8. Click Add . The opens.
9. Type a name for the new custom message attribute in the Name box, or select an existing custom message attribute name in the list.
10. Type the custom message attribute to associate with this name in the Value box.
11. Click OK . The custom message attribute appears in the CMAs tab.

Related Topics:

- Quick start: how to create a policy
- Change message browser column display options

Write instructions to accompany a message

Messages generated by a policy can include instructions that explain what to do when the message is generated. This instruction text can often help an operator to solve a problem when a particular type of message is received. The operator can view the instructions included with a message by viewing the message details in the message browser . You can define default instructions for all rules in a policy. You can also override the default with different instructions for any rule .

To write instructions to accompany a message

1. Right-click the policy and select All Tasks → Edit... .
2. Select Rules .
3. Select the rule to which you want to add instructions.
4. Select Modify .
5. Select Actions .
6. Select Message .
7. Select Instructions .
8. Select Instruction Text and type the instructions that you want to accompany the message.

NOTE:
You can add hyperlinks to URLs in your message or instruction text. Insert your cursor in the text of an active message on the Text tab and type the required URL. Apply and save your changes. This feature is useful for linking to external sites of all types, such as support sites, documentation repositories, troubleshooting information, and similar sites.

Related Topics:

- Quick start: how to create a policy

Message stream interface and external services

This tab lets you configure the interface between HP Operations messages and external programs. Here you can set up the message stream interface and external services for a message.

NOTE:
This tab contains functions that provide compatibility with the HP Operations Manager for UNIX management server . They cannot be used with an HP Operations management server. For more information, consult the HP Operations Manager for UNIX documentation.

To access this tab

1. Right-click the policy and select All Tasks → Edit...
2. Select Rules .
3. Select the rule for which you want to configure message stream interface and external services options.
4. Select Modify .
5. Select Actions .
6. Select Message .
7. Select Message stream interface and external services .

Related Topics:

- Send messages to the message stream interface
- Forward messages to external services
- Quick start: how to create a policy

Send messages to the message stream interface

The message stream interface allows external applications to interact with the internal message flow of HP Operations. The external application can be a read-write applications, for example, a message processing program that can read HP Operations messages , modify attributes, and generate new messages for retransmission to the management server . The application could also read messages, or send its own messages.

When you enable the message stream interface, you can also allow external applications using the interface to set up automatic or operator-initiated commands.

Select Agent message stream interface to allow messages to be directed to the message stream interface on the managed node. When switched on, you can choose between the following options:

- Divert a message to the message stream interface instead of to the message manager when a message is requested by an external application.
- Send the message directly to the message browser , and a copy of the message to the message stream interface.

Select Server message stream interface to allow messages to be directed to the message stream interface on the management server. When switched on, you can choose between the following options:

- Divert a message to the message stream interface instead of to the message manager when a message is requested by an external application.
- Send the message directly to the message browser, and a copy of the message to the message stream interface.

Select Immediate local automatic commands if you want to allow local automatic actions at the managed node for messages that can be diverted to the message stream interface.

You can use this function when local automatic commands are configured for messages that can be output to the message stream interface. For example, if a message starts an automatic command at the managed node and is then diverted to the message stream interface, it may be discarded by an event correlation engine connected to the message stream interface. In this case, a response from the local action would never be seen by an operator as the corresponding message does not exist in the HP Operations database.

If local automatic commands are not allowed, the automatic command is triggered by the management server as soon as the message is received, if the message is not discarded on its way through the message stream interface.

You can only select this function when output to the message stream interface is enabled and set to "Divert Messages".

Related Topics:

- Quick start: how to create a policy

Forward messages to external services

HP Operations Manager can forward messages to external services. If **Forward to trouble ticket** is selected, HP Operations Manager forwards message parameters to a predefined external trouble ticket system when the message is received by the management server . HP Operations Manager does not provide trouble ticket services, but supports an interface to export event-specific details to an external trouble ticket service.

If **Forward to notification service** is selected, the message will trigger a configured notification service such as a beeper or paging service.

NOTE:
These functions are only available on UNIX management servers. To use this functionality, you must forward the message to an HP Operations Manager for UNIX management server. Refer to the HP Operations Manager for UNIX documentation for more information about this feature.

Related Topics:

- Quick start: how to create a policy
- Server-based flexible management

Add operator-initiated commands to a policy rule

For every rule , you can configure an operator-initiated command to be attached to the message that the rule sends to the message browser . This command can be started by the operator from the message browser. The command might be a script that requires operator input to solve the problem, or instructions that appear in the web browser on the management console.

You can set the following properties:

- Graph (available only for the Measurement Threshold policy type). Select Graph if the source of your policy is a metric in the Embedded Performance Component and you want the operator-initiated command to show a graph of that metric's value over time. See Set threshold source properties for more information. Use Date range to indicate how much historical data the graph should present. For example, if you select 1 Hour , the graph shows the metric's value over the hour before the threshold was crossed.
- Graph Template (available only for the Measurement Threshold policy type). By selecting Graph Template, you can choose a predefined HP Performance Manager graph that you want the operator-initiated command to show. When an operator starts this command, HP Performance Manager generates a graph using data from the affected node and the selected graph template.
  - Graph template name : Click Browse and select the name of the graph template that you want to provide to the operator.
  - Date Range : Select the span of time for which data should be displayed in the graph.
  - Filter on Instance : If want the graph to only show data from a specific instance, enter the instance name here.

  NOTE:
  Graphs are available only if HP Performance Manager integration is correctly configured on the management server. For more details, see HP Performance Manager Integration .
- In the Command text box, you specify the command to run when the command is started for this message. The command runs on the node specified in the Node box. If the command contains spaces, enclose it in quotation marks. Commands that are internal to the Windows command shell (for example echo or `move`), must be preceded by `cmd /c` . See the Windows help for more information about `cmd` .
- In the Node text box, you specify the name of the node on which the command will be started. You can also use the variables `<$OPC_MGMTSV>` , `<$MSG_NODE_NAME>` , `<$OPC_GUI_CLIENT>` , or `<$OPC_GUI_CLIENT_WEB>` , to configure reusable policies for replicated sites. See policy variables for more information. Note that the options below cannot be used when the variables `<$OPC_GUI_CLIENT>` , or `<$OPC_GUI_CLIENT_WEB>` are selected.
- When Append output of command as annotation to the message is selected, an annotation is added to the message when the command completes. The annotation contains the start time, output, exit value,

and finish time of the command. If a command fails, an annotation is provided even if this item is not selected.

- When Acknowledge the message when command is successful is selected, the message is automatically acknowledged (that is, moved to the acknowledged messages browser) if the command is successful.

To set up an operator-initiated command

1. Right-click the policy and select All Tasks → Edit...
2. Select Rules .
3. Select the rule to which you want to add a command.
4. Select Modify.
5. Select Actions.
6. Select Operator-initiated command and set up the command.

Related Topics:

- Automatic command
- Quick start: how to create a policy

Add automatic commands to policy rules

For every rule , you can configure an automatic command to be run when the rule is matched. For example, you could configure a log file Entry policy to automatically delete the contents of `C:\Temp` when the System event log reports "`The C: disk is at or near capacity.` "

For every automatic command, you can set the following properties:

- In the Command text box, you specify the command and parameters to run when the command is started for this message . The command runs on the node you specify in the Node box. If the command contains spaces, enclose it in quotation marks. Commands that are internal to the Windows command shell (for example echo or move), must be preceded by `cmd /c` . See the Windows help for more information about cmd .
- In the Node text box, you specify the name of the node on which the command will be started. You can also use the variables `<$OPC_MGMTSV>` or `<$MSG_NODE_NAME>` , to configure reusable policies for replicated sites. See policy variables for more information.
- When Append output of command as annotation to the message is selected, an annotation is added to the message when the command completes. The annotation contains the start time, output, exit value, and finish time of the command. If a command fails, an annotation is provided even if this item is not selected.
- When Acknowledge the message when command is successful is selected, the message is automatically acknowledged (that is, moved to the acknowledged message browser) if the command is successful.

HP Operations allows you to define how the management server is informed about the progress of local automatic commands.

- Send the message immediately: Send a message to the management server as soon as a local automatic command starts on the managed node . This is the default setting.
- You can also choose to wait until the local command completes , and then send the message to the management server based on whether the command was successful or failed.

  Other options can help to reduce the amount of unnecessary network traffic to the management server. For example, if a local automatic command solves the problem that generated the message, you may choose not to inform the management server.

To set up an automatic command

1. Right-click the policy and select All Tasks → Edit...

2. Select Rules .
3. Select the rule to which you want to add a command.
4. Select Modify.
5. Select Actions.
6. Select Automatic Command .

NOTE:
If you want to prevent automatic commands from being run on a node other than the node where the message originates, add the following registry key to the registry:

HKLM\SOFTWARE\Hewlett-Packard\OVEnterprise\Management server\MsgActSrv:

Name: DISABLE_ALL_REMOTE_ACTIONS
Type: REG_SZ
Value: TRUE

If you set this registry key, note that you may need to reconfigure existing policies (for example, policies supplied by Smart Plug-ins) to function properly with this restriction.

Related Topics:

- Operator-initiated command .
- Quick start: how to create a policy

Set message defaults

The outgoing message defaults dialog box allows you to indicate default settings for all messages in a policy . The default settings include:

- Message attributes
- Message correlation
- Instructions
- Message Stream Interface

These defaults only affect new rules and can be changed in individual rules after they are created. If a rule contains empty message attributes, the agent will use these defaults for the new message.

To set message defaults

1. Right-click the policy, and then click All Tasks → Edit...
2. Click Rules .
3. Click Defaults... .

Related Topics:

- Quick start: how to create a policy

Configure Policy Options

The options tab allows you to establish several policy behaviors. From this tab you can:

- Define which events are locally logged on the managed node from which they originated.
- Configure the policy to send a message to the management server when an event does not match any rule in the policy.
- Define the default pattern matching options that the policy uses when evaluating text.
- For the Measurement Threshold policy you can also decide on processing options for rules and thresholds, and can enable automatic escalation .

Related Topics:

- Quick start: how to create a policy

Log local events on the managed node

HP Operations Manager allows you to define which events, if any, are logged on the managed node from which they originated. These events are logged on the local managed node in the log file: <*data_dir* >`\log\OpC\opcmsglg` .

Three logging options are available.

- Log local events that match a rule trigger a message . This selection logs any events in the message source that match the policy rules.
- Log local events that match a rule and are ignored . This selection logs any events in the message source that are suppressed (that is, they do not cause a message to be sent to the HP Operations management server).
- Log local events that don't match any rule . This selection logs any events that don't match any of the rules in the policy.

To log local events on the managed node

1. Right-click the policy and select Edit .
2. Select Options .
3. In the Log local events box, choose which types of events should be logged on the local managed node.

Related Topics:

- Quick start: how to create a policy

Capture unmatched events

To ensure that unexpected events that might be important do not go unreported, you can configure a policy to send a message to the management server when an event does not match any rule in the policy.

To choose how unmatched events are handled

1. Right-click the policy and select Edit .
2. Select the Options tab.
3. In the Unmatched events group box, choose whether unmatched events are sent to the message browser , are sent to the acknowledged messages browser , or are ignored .

   NOTE:
   Each policy (except for Open Message Interface and SNMP Interceptor policies) that sends unmatched events to the management server creates a message with the default values of the policy. If several policies forward unmatched messages to the management server you could receive multiple messages from a single event.
   For Open Message Interface and SNMP Interceptor policies, the behavior is somewhat different; unmatched events create a message only if they are unmatched in all policies on the node , and only one message will be sent.

Related Topics:

- Quick start: how to create a policy

Set field separators

In this box, you can indicate which characters should be considered to be field separators. Field separators are used in the message text pattern as separator characters for the rule condition. You can define up to seven separators, including these special characters:

- \n new line (NL)
- \t horizontal tab (HT)
- \v vertical tab (VT)
- \b backspace (BS)
- \r carriage return (CR)
- \f form feed (FF)
- \a alert (BEL)
- \\ backslash (\)

For example, if you wanted a backslash, an asterisk, and the letter A to define the fields in the message text, you would type \\*A (with no spaces separating the characters).

If you leave this box empty, the default separators (a blank and the tab character) are used by default.

Set field separators

1. Right-click the policy and select All Tasks → Edit...
2. Select Options .
3. Type the character that you want to use in the Field Separators box.
4. The setting applies to all new rules in a policy. Click Apply to all to apply it to all existing rules in a policy.

NOTE:

You can indicate the separator characters for individual rules in a policy by selecting the > button in the Condition tab of a rule.

Related Topics:

- Variables and pattern matching
- Quick start: how to create a policy

Set case sensitivity

You can choose whether the case (uppercase/lowercase) of a text string is considered when the message text of a rule is compared with the destination text in the message source. When switched on, a match only occurs if the use of uppercase and lowercase letters is exactly the same in both the message source and the message text string.

To set case sensitivity

1. Right-click the policy and select All Tasks → Edit...
2. Select Options .
3. Select or clear Case sensitive check to indicate the behavior you want.
4. The setting applies to all new rules in a policy. Click Apply to all to apply it to all existing rules in a policy.

NOTE:

You can set case sensitivity for individual rules in a policy by selecting the > button in the Condition tab of a rule.

Related Topics:

- Quick start: how to create a policy

Variables and pattern matching

To make your policies as flexible as possible, you can use HP Operations pattern-matching syntax, and a variety of variables. The pattern-matching syntax makes is possible to write rule conditions that match strings very specifically, while the variables can be used to compose messages, and can be passed as parameters to commands. They can also be passed to external applications through the instruction text interface.

Click the links below to learn more about the pattern-matching syntax and variables.

Variables

NOTE:
HP Operations variables are reserved words; they must not be used for any other purpose, such as creating user-defined variables. User-defined variables can be created in those windows used to define HP Operations messages.

- Policy Variables
  Learn about the variables that represent various HP Operations elements, for example, managed nodes, the management server , users, and so on. Learn about what the variables return, and where they are valid.
- SNMP Variables
  Learn about the SNMP variables that can be used within HP Operations or passed to external programs.
- Environment variables
  Learn how to use environment variables in scripts, tools, and commands that run on a Windows node.

Pattern matching

- Details of pattern-matching expressions
  Learn how to construct match patterns for your policy rules.
- User-defined Variables
  Learn how to assign parts of matched text to variables that can be used with commands, instructions, or to reword the message .
- Pattern matching for variables
  Learn how to test a string or variable against a pattern, and define an output string that is conditional on the result.
- Set field separators

Learn how to change the character that HP Operations uses to delimit fields in strings for pattern matching.

- Set case sensitivity
  Learn how to set the case sensitivity for a rule.
- Numeric Range Operators
  Learn how to write match patterns that match a range of numerical values.
- Examples of pattern matching in rule conditions
  Examine examples of pattern matching.
- Test pattern matching
  Learn how to use the HP Operations interface to test the patterns that you write, before deploying your policies.

Related Topics:

- Quick start: how to create a policy

Policy variables

The variables listed below can be used in most event policy editor text entry boxes (exceptions are noted). The variables can be used within HP Operations, or passed to external programs. Each variable is shown with the required syntax.

NOTE:
It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces.

`<$FULLNAME>`
: Returns the name of the policy and the source (concatenated with -). As defined for templates, this would return the same as <$NAME> This variable, along with <$NAME> and <$FULLNAME>, can be used in the field Program Name as follows so that when you are renaming a policy, you would not need to modify the Program name field.
mymonitorscript.bat <$NAME>-<SRCNAME>
mymonitorscript.bat <$FULLNAME>

`<$MSG_APPL>`
: Returns the name of the application associated with the event that caused the message. Only events from the Open Message Interface (parameter: application) or the Windows Event Log (parameter: source) will set this variable. Sample output: `/usr/bin/su(1) Switch User`

`<$MSG_GRP>`
: Returns the default message group of the message. Only events from the Open Message Interface (parameter: message_group) will set this variable. Sample output: `Security`

`<$MSG_ID>`
: Returns the unique identity number of the message, as generated by the message agent. Note that identity numbers are not generated for suppressed messages. Sample output: `6e998f80-a06b-71d0-012e-0f887a7c0000`

`<$MSG_NODE>`
: Returns the IP address of the managed node on which the message originates. Sample output: `14.136.122.123`

`<$MSG_NODE_ID>`
: Returns the GUID that the management server assigned to the node on which the message originates. Because this value is only known by the management server, this variable cannot be resolved on the managed node. This variable is valid for the service_id message attribute and in the Command box for an automatic or operator-initiated command. Sample output: `{6e998f80-a06b-71d0-012e-0f887a7c0000}`

`<$MSG_NODE_NAME>`
: Returns the name of the managed node on which the message originates. This variable is not fixed,

however, and can be changed by a policy on a per-message basis. For example, if the policy is intercepting SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated. If the policy is monitoring a logfile on a network share where applications on several nodes write messages, you could extract the name of the node from the error message, save it in a user-defined variable, and assign it to MSG_NODE_NAME.

`<$MGMTSV_KNOWN_MSG_NODE_NAME>`

Returns the name of the managed node on which the message originates. The management server resolves this variable to the node's hostname. This variable may be different to `<$MSG_NODE_NAME>` , which is the hostname that the agent resolves.

You can use `<$MGMTSV_KNOWN_MSG_NODE_NAME>` in the following message attributes:

- Service ID
- Message Key
- Message Type
- Message Group
- Application
- Object
- Message Text
- CMA Name
- CMA Value
- Automatic Command
- Automatic Command Node
- Operator-initiated Command
- Operator-initiated Command Node

This variable is useful in environments where management servers and agents resolve different hostnames for the same the node (for example, NAT environments).

`<$MSG_OBJECT>`

Delivers the name of the object associated with the event. Only events from the Open Message Interface (parameter: msg_object) and Windows Event Log (parameter: category) will set this variable.

`<$MSG_SEV>`

Returns the default value for the severity of the event. Only events from the Open Message Interface (default is "Normal') and the Windows Event Log (parameter: converted) will set this variable. Note that the following severity conversions are performed when this variable is set by the Windows Event Log: information=Normal, warning=Warning, error=critical, success audit=Normal, failure audit=Critical, default=unknown). Sample output: `Normal`

`<$MSG_TEXT>`

Returns the full text of the message. For the Open Message Interface, this value is the msg_text parameter. For the Windows Event Log this value is the event ID and description. In general, there are default texts for all editors derived from incoming event properties (this is shown in the message text field of outgoing message properties). Sample output: `SU 03/19 16:13 + ttyp7 bill-root`

`<$MSG_TYPE>`

Delivers the name set for Message Type .

`<$NAME>`

Returns the name of the measurement threshold policy or the scheduled task policy that sent the message. Sample output: `cpu_util`

This variable, along with <$FULLNAME> and <$SRCNAME>, can be used in the field Program Name as follows so that when you are renaming a policy, you would not need to modify the Program name field.
mymonitorscript.bat <$NAME>-<SRCNAME>
mymonitorscript.bat <$FULLNAME>

`<$OPC_GUI_CLIENT>`

Returns the hostname of the client where the HP Operations GUI is currently running. This variable is valid in the Node box for an operator-initiated command and for message attributes.

`<$OPC_GUI_CLIENT_WEB>`

Returns the hostname and default web browser of the client where the HP Operations GUI is currently running. This can be used with an operator-initiated command to load a web page in the default browser on the HP Operations GUI client. This variable is valid in the node field for an operator-initiated command and for message attributes.

`<$OPC_MGMTSV>`

Returns the name of the current HP Operations management server . This variable is valid in the Command text box and in the Node text box for an automatic or operator-initiated command. This variable is only resolved on the management server. Sample output: `zucchini.veg.com`

`<$OPTION(N)>`

Returns the value of an optional variable that is set by opcmsg or opcmon (for example, $OPTION(A) $OPTION(B), and so on.).

`<$SRCNAME>`

Returns the name of the source for a policy. For a template, this would return an empty string. This variable, along with <$NAME> and <$FULLNAME>, can be used in the field Program Name as follows so that when you are renaming a policy, you would not need to modify the Program name field.
mymonitorscript.bat <$NAME>-<SRCNAME>
mymonitorscript.bat <$FULLNAME>

Additional information about variables <$NAME>, <$FULLNAME>, and <$SRCNAME>

The execute command made by the monitor agent now includes additional processing that allows special opc defined variables for the policy/monitor/source name to be resolved.

<$NAME> currently exists and evaluates only to the name of the template. To remain consistent, the variable <$FULLNAME> is used for the complete policy-source name which would generally be used in the newer Policy definitions.

The " <$>"character combination can be suppressed with the "\" escape character. If the " <$>" character combination is found, but the variable is unknown, or no closing bracket (">") is found, then no substitution is performed. Parsing for escape characters would be limited to the characters directly before a known variable, as shown in examples 3, 4 and 5).

This feature will be available by default on all agent platforms.

It would be possible to disable this for a node by setting the OPCINFO/NODEINFO key:

OPC_MON_DISABLE_PROG_VARS TRUE

Examples:

```
Policy: SNMP-service-Win2k with source name service
Example:     1
Definition:  opcservice  SNMP <$NAME>-<$SRCNAME>
Resolved:    opcservice  SNMP SNMP-service-Win2k-service
Notes:       Create required name for opcmon from both the policy
             name and source name variables

Example:     2
Definition:  opcservice  SNMP <$FULLNAME>
Resolved:    opcservice  SNMP SNMP-service-Win2k-service
Notes:       Resolves to the combined policy and source name.

Example:     3
Definition:  opcservice  SNMP \<$FULLNAME>
Resolved:    opcservice  SNMP  <$FULLNAME>
Notes:       Single escape character, therefore the variable is ignored

Example:     4
Definition:  opcservice  SNMP \\<$FULLNAME>
Resolved:    opcservice  SNMP  \SNMP-service-Win2k-service
Notes:       Double escape character,
             resolved to single and variables resolved.
```

The following variables are valid only in measurement threshold policies:

`<$THRESHOLD>`

Returns value for the threshold limit set in General Threshold Rule Properties . If the threshold is determined with a script, the name of the scripting language is returned, for example, `VBScript`
Sample output: `95.00`

`<$VALUE>`

Returns the value measured by a Measurement Threshold policy. Sample output: `100.00`

`<$VALAVG>`

Returns the average value of all messages reported by the Measurement Threshold policy. Sample output: `100.00`

`<$VALCNT>`

Returns the number of times that the threshold monitor has delivered a message to the browser. Sample output: `1`

`<$MSG_TIME_CREATED>`

Returns the time the message was created on the managed node in seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time. Sample output: `950008585`

`<$INSTANCE>`

Returns the name of the current instance Sample output: `C;`

`<$SESSION(key)>`

Returns the value of a key stored in the Session object by using the Value method.

The following variable is valid only in messages sent from Windows Management Interface policies

`<$WBEM:WMI class property>`

(for example, `&lt$WBEM:TimeCreated>` Sample output: `19991130105330.000000+060`)

The following variables are valid only in messages sent from Scheduled Task policies:

`<$PROG>`

Returns the name of the program executed by the Scheduled Task policy Sample output:`check_for_upgrade.bat`

`<$USER>`

Returns the name of the user under which the scheduled task was executed. Sample output:`administrator`

The following variables are valid only in messages sent from Logfile Entry policies

`<$LOGFILE>`

Returns the name of the logfile that contains the event which caused the message. Sample output:`program_log.txt`

`<$LOGPATH>`

Returns the name and path of the logfile that contains the event which caused the message. Sample output:`C:\temp\mylogfile\program_log.txt`

The following variables are valid only in messages sent from Process-monitor policies

The following session variables are set automatically and can be used to define actions in the format <$SESSION(session variable)>:

`<PROCESSNAME>`
: Defines the name used to access the process on the Managed Node

`<PROCESSPARAMETERS>`
: Defines the parameter pattern used to access the process on the Managed Node

`<PROCESSNBREXPECTED>`
: Defines the number of monitored processes

`<PROCESSNBRAVAILABLE>`
: Defines the number of available processes matching the process name and parameter pattern

`<PROCESSMODE>`
: Defines the string used to build the message text. It depends on the monitor you specify, for example:

- MIN

  PROCESSMODE is: ">= "
- MAX

  PROCESSMODE is: "<= "
- EQUAL

  PROCESSMODE is: " " (empty string)

The following variables are valid only in messages sent from Windows Services-monitor policies

The following session variables are set automatically and can be used in the actions in the format <$SESSION (session variable)>:

`<SERVICENAME>`
: Defines the name used to access the Windows service on the Managed Node

`<$SERVICEDISPLAYNAME>`
: Defines the display name of the Windows service. This value is retrieved on the specified Managed Node and can be displayed in the local language of the Managed Node.

`<$SERVICEMONITORSTATE>`
: Defines the state of the Windows service to monitor, for example; "running", "stopped", or "disabled". If an agent catalog is available in the local language set on the Managed Node, this is the localized

text for the monitor state. If no agent catalog is available in the local language of the Managed Node, English text is used to display the monitor state.

`<$SERVICECURRENTSTATE>`

Defines the current state of the Windows service being monitored, for example; "running", "stopped", or "disabled". If an agent catalog is available in the local language set on the Managed Node, this is the localized text for the monitor state. If no agent catalog is available in the local language of the Managed Node, English text is used to display the monitor state.

`<SERVICEACTION>`

Defines the string used to build the message text. It depends on the monitor mode you define:

- Monitor state "running"

  net start /Y <*service_name* >

- Monitor state "stopped"

  net stop /Y <*service_name* >

- Monitor state "disabled"

  empty

Related Topics:

- Pattern-matching and variables
- Quick start: how to create a policy

SNMP variables

The variables listed below can be used in most SNMP Interceptor text entry boxes (exceptions are noted). The variables can be used within HP Operations Manager, or passed to external programs. Each variable is shown with the required syntax, but note that it is also often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces.

`<$#>`

Returns the number of variables in an enterprise-specific SNMP event (generic event 6 Enterprise specific ID). Sample output: 2

`<$*>`

Returns all variables assigned to the event up to the possible fifteen. Sample output: `[1] .1.1 (OctetString): arg1 [2] .1.2 (OctetString): turnip.veg.com`

`<$@>`

Returns the time the event was received as the number of seconds since Jan 1, 1970 using the *time_t* representation. Sample output: 859479898

`<$1>`

Returns one or more of the fifteen possible event parameters that are part of an SNMP event. (`<$1>` returns the first variable, `<$2>` returns the second variable, and so on.)

`<$\>1>`

Returns all attributes greater than *n* as *value* strings, useful for printing a variable number of arguments. `<$\>0>` is equivalent to `$*` without sequence numbers, names, or types. Sample output: `bokchoy.veg.com`

`<$\>+1>`

Returns all attributes greater than *n* as *name:value* string. Sample output: `.1.2: asparagus.veg.com`

`<$+2>`

Returns the nth variable binding as *name:value* . (Note: not valid in the command box.) Sample output: `.1.2: artichoke.veg.com`

`<$\>-n >`

Returns all attributes greater than *n* as *[seq] name (type): value* strings. Sample output: `[2] .1.2 (OctetString): cauliflower.veg.com`

`<$-2>`

Returns the nth variable binding as *[seq] name-type:value* . (Note: not valid in Command Box.) Sample output: `[2] .1.2 (OctetString): brusselsprouts.veg.com`

`<$A>`

Returns the node which produced the event. Sample output: `eggplant.veg.com`

`<$C>`

Returns the community of the event. Sample output: `public`

`<$c>`

Returns the event's category. Sample output: `SNMP`

`<$E>`

Returns the enterprise ID of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1`

`<$e>`

Returns the enterprise object ID. Sample output: `.1.3.6.1.4.1.11.2.17.1`

`<$F>`

Returns the textual name of the remote postmaster daemon's computer if the event was forwarded. Sample output: `cress.veg.com`

`<$G>`

Returns the generic event ID. Sample output: `6`

`<$MSG_OBJECT>`

Returns the name of the object associated with the event. This is set in the Message Defaults section of the policy editor.

`<$N>`

Returns the event name (textual alias) of the event format specification used to format the event, as defined in the Event Configurator. Sample output: `OV_Node_Down`

`<$O>`

Returns the name (object identifier) of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1.0.58916865`

`<$o>`

Returns the numeric object identifier of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1.0.58916865`

`<$R>`

Returns the true source of the event. This value is inferred through the transport mechanism which delivered the event. Sample output: `carrot.veg.com`

`<$r>`

Returns the implied source of the event. This may not be the true source of the event if the true source is proxying for another source, such as when a monitoring application running locally is reporting information about a remote node. Sample output: `rutabaga.veg.com`

`<$S>`

Returns the specific event ID. Sample output: `5891686`

`<$s>`

Returns the event's severity. Sample output: `Normal`

`<$T>`

Returns the event time stamp. Sample output: `0`

`<$V>`

Returns the event type, based on the transport from which the event was received. Currently supported types are SNMPv1, SNMPv2, CMIP, GENERIC, and SNMPv2INFORM. Sample output: `SNMPv1`

`<$X>`

Returns the time the event was received using the local time representation. Sample output: `17:24:58`

`<$x>`

Returns the date the event was received using the local date representation. Sample output: `03/27/97`

Related Topics:

- Pattern-matching and variables
- Quick start: how to create a policy

Environment variables

On nodes that have a Windows operating system, the agent installation creates several environment variables. These are user environment variables for the account that runs the agent, and therefore are not visible to other users. You can use these in scripts, tools, and commands that run on the node .

	HTTPS agents	DCE agents
HPOM agent installation directory	`%OvAgentDir%`	
HP BTO Software installation directory	`%OvInstallDir%`	
HP BTO Software data directory	`%OvDataDir%`	
Directory that contains the agent's Perl interpreter	`%OvPerlADir%`	N/A
Location of the agent's Perl interpreter	N/A	`%OvPerlBin%`

Pattern-matching details

HP Operations provides a powerful pattern-matching language that reduces the number of conditions you must use. Selected, dynamic parts of text-based events can be extracted, assigned to variables, and used as parameters to build message text or to set other attributes. These parameters can also be used for automatic and operator-initiated commands. HP Operations provides a pattern-matching language that allows you to very accurately specify the character string that you want a rule to match.

NOTE:
In text boxes where pattern-matching expressions are allowed you can right-click for a shortcut menu with pattern-matching values that can be selected and inserted into the text box.

- Matching special characters
- Matching characters at the beginning or end of a line
- Matching multiple characters
- Matching two or more different expressions
- Matching text that does not contain an expression
- The Mask (\ ) Operator
- Bracket ([and]) Expressions

- Matching special characters

  Ordinary characters are expressions which represent themselves. Any character of the supported character set may be used. However, if any of the following special characters are used they must be prefaced with a backslash (\) that masks their usual function.

  ```
  \ [ ] < > | ^ $
  ```

  If ^ and $ are not used as anchoring characters, that is, not as first or last characters, they are considered ordinary characters and do not need to be masked.

- Matching characters at the beginning or end of a line

  If the caret (^) is used as the first character of the pattern, only expressions discovered at the beginning of lines are matched. For example, "^ab" matches the string "ab" in the line "abcde", but not in the line "xabcde".

  If the dollar sign is used as the last character of a pattern, only expressions at the end of lines are matched. For example, "de$" matches "de" in the line "abcde", but not in the line "abcdex".

- Matching multiple characters

  Patterns used to match strings consisting of an arbitrary number of characters require one or more of the following expressions:

  - <*> matches any string of zero or more arbitrary characters (including separators)

- <n*> matches a string of *n* arbitrary characters (including separators)
- <#> matches a sequence of one or more digits
- <n#> matches a number composed of *n* digits
- <_> matches a sequence of one or more field separators
- <n_> matches a string of *n* separators
- <@> matches any string that contains no separator characters, in other words, a sequence of one or more non-separators; this can be used for matching words

Separator characters are configurable for each pattern. By default, separators are the space and the tab characters.

- Matching two or more different expressions

Two expressions separated by the special character vertical bar (|) matches a string that is matched by either expression. For example, the pattern:

```
[ab|c]d
```

matches the string "abd" and the string "cd".

- Matching text that does not contain an expression

The NOT operator (!) must be used with delimiting square brackets, for example:

```
<![WARNING]>
```

The pattern above matches all text which does not contain the string "WARNING".

The NOT operator may also be used with complex subpatterns:

```
SU <*> + <@.tty> <![root|[user[1|2]]].from>-<*ot>
```

The above pattern makes it possible to generate a "switch user" message for anyone who is not user1, user2 or root. Therefore the following would be matched:

```
SU 03/25 08:14 + ttyp2 user11-root
```

However, this line would not be matched, because it contains an entry concerning "user2":

```
SU 03/25 08:14 + ttyp2 user2-root
```

Notice that if the subpattern including the not operator does not find a match, the not operator behaves like a <*>: it matches zero or more arbitrary characters. For this reason, the HP Operations pattern-matching expression: <![1|2|3]> matches any character or any number of characters, except 1, 2, or 3.

- The Mask (\ ) Operator

The backslash (\ ) is used to mask the special meaning of the characters:

```
[ ] < > | ^ $
```

A special character preceded by \ results in an expression that matches the special character itself.

Notice that because ^ and $ only have special meaning when placed at the beginning and end of a pattern respectively, you do not need to mask them when they are used within the pattern (in other words, not at beginning or end).

The only exception to this rule is the tab character, which is specified by entering "\t" into the pattern string.

- Bracket ([and]) Expressions

  The brackets ([and]) are used as delimiters to group expressions. To increase performance, brackets should be avoided wherever they are unnecessary. In the pattern:

  ```
  ab[cd[ef]gh]
  ```

  all brackets are unnecessary--"abcdefgh" is equivalent.

  Bracketed expressions are used frequently with the OR operator , the NOT operator and when using subpatterns to assign strings to variables.

Related Topics:

- Examples of pattern matching in rule conditions
- Variables and parameters
- Quick start: how to create a policy

User-defined variables in patterns

Any matched string can be assigned to a variable, which can be used to compose messages or used as a parameter for action calls. To define a parameter, add ". parametername " before the closing bracket. The pattern:

```
^errno: <#.number> - <*.error_text>
```

matches a message such as:

```
errno: 125 - device does not exist
```

and assigns "125" to number and "device does not exist" to error_text .

When using these variables, the syntax is <*variable_name*> (for example, <number>).

Rules by which HP Operations assigns strings to variables

In matching the pattern <*.var1 ><*.var2 > against the string "abcdef", it is not immediately clear which substring of the input string will be assigned to each variable. For example, it is possible to assign an empty string to var1 and the whole input string to var2 , as well as assigning "a" to var1 and "bcdef" to var2 , and so forth.

HP Operations' pattern matching algorithm always scans both the input line and the pattern definition (including alternative expressions) from left to right. <*> expressions are assigned as few characters as possible. <#>, <@>, <S > expressions are assigned as many characters as possible. Therefore, var1 will be assigned an empty string in the example above.

To match an input string such as:

```
this is error 100: big bug
```

use a pattern such as:

```
error<#.errnumber>:<*.errtext>
```

In which:

- "100" is assigned to errnumber
- "big bug" is assigned to errtext

For performance and pattern readability purposes, you can specify a delimiting substring between two expressions. In the above example, ":" is used to delimit <#> and <*>.

Matching <@.word ><#.num > against "abc123" assigns "abc12" to word and "3" to num , as digits are permitted for both <#> and <@>, and the left expression takes as many characters as possible.

Patterns without expression anchoring can match any substring within the input line. Therefore, patterns such as:

```
this is number<#.num>
```

are treated in the same way as:

```
<*>this is number<#.num><*>
```

Using subpatterns to assign strings to variables

In addition to being able to use a single operator, such as * or # , to assign a string to a variable, you can also build up a complex subpattern composed of a number of operators, according to the following pattern: <[*subpattern*].var>

For instance: <[<@>file.tmp].fname>

In the example above, the period (.) between "file" and "tmp" matches a similar dot character, while the dot between "]" and "fname " is necessary syntax. This pattern would match a string such as "Logfile.tmp" and assigns the complete string to fname .

Other examples of subpatterns are:

- <[Error|Warning].sev >
- <[Error[<#.n><*.msg>]].complete >

In the first example above, any line with either the word "Error" or the word "Warning" is assigned to the variable, sev . In the second example, any line containing the word "Error" has the error number assigned to the variable, n , and any further text assigned to msg . Finally, both number and text are assigned to complete .

Related Topics:

- Pattern-matching in rule conditions
- Examples of pattern matching in rule conditions
- Quick start: how to create a policy

Pattern matching for variables

HPOM enables you to test a string or variable against a pattern, and define an output string that is conditional on the result. You can do this using **`$MATCH`** , which has the following syntax:

```
<$MATCH(string, pattern, true, [false])>
```

Specify the parameters as follows:

`string`
: Specify a literal string (for example, `TEST STRING`) or an HPOM variable (for example `<$LOGPATH>`).

`pattern`
: Specify a pattern, using HPOM pattern matching syntax. You can create user-defined variables in the pattern to use in the parameters `true` and `false` . The pattern is case sensitive.

`true`
: Specify a string to return if the string and pattern match. You can specify a literal string, or a user-defined variable, or an HPOM variable.

`false`
: *Optional.* Specify a string to return if the string and pattern do not match. You can specify a literal string, or a user-defined variable, or an HPOM variable.

Separate each parameter with a comma (,). To specify a comma within a parameter, you must precede it with two backslashes (\\).

You can use `$MATCH` within your policies in the following message attributes:

- Service ID
- Message type
- Message group
- Application
- Object
- Message text
- Automatic command
- Custom message attribute

NOTE:
You can use $MATCH only once in each message attribute. You cannot use $MATCH recursively.

Example

A logfile entry policy can monitor a number of log files. The name of path of the log file is available in the HPOM variable `<$LOGPATH>` . If part of the log file path corresponds to an application name, you can use $MATCH to set the application message attribute as follows:

```
<$MATCH(<$LOGPATH>,<[@].application>.log, <application>, Unknown)>
```

Related Topics:

- HP Operations policy variables
- Details of pattern-matching expressions
- User-defined variables

Numeric range operators

HPOM provides six numeric range operators that can be used in pattern matching. The operators are used in this way:

Operator name	Syntax	Example/Explanation
Less than	<[*pattern*] -lt *n* >	`<[<#>] -lt 5>` matches every number less than 5
Less than or equal to	<[*pattern*] -le *n* >	`<[<#>] -le 5>` matches 5 and every number less than 5
Greater than	<[*pattern*] -gt *n* >	`<[<#>] -gt 5>` matches every number greater than 5
Greater than or equal to	<[*pattern*] -ge *n* >	`<[<#>] -ge 5>` matches 5 and every number greater than 5
Equal to	<[*pattern*] -eq *n* >	`<[<#>] -eq 5>` matches 5 or 5.0
Not equal to	<[*pattern*] -ne *n* >	`<[<#>] -ne 5>` matches every number but 5 and 5.0
The operators can also be combined to produce matches according to ranges of numbers:		
Matches numbers that belong to the interval, excluding the limits	< *n* -lt [*pattern*] -lt *n* >	`<5 -lt [<#>] -lt 10>` matches every number between 5 and 10 (but not 5 or 10)
Matches numbers that belong to the interval, including the limits	< *n* -le [*pattern*] -le *n* >	`<5 -le [<#>] -le 10>` matches every number between 5 and 10 (including 5 and 10)
Matches numbers that do not belong to the interval, excluding the limits	< *n* -gt [*pattern*] -gt *n* >	`<10 -gt [<#>] -gt 5>` matches every number between 5 and 10 (but not 5 or 10)

Matches numbers that do not belong to the interval, including the limits	< *n* -ge [*pattern*] -ge *n* >	`<10 -ge [<#>] -ge 5>` matches every number between 5 and 10 (including 5 and 10)

Any time you are working with message text pattern-matching, you can use the left and right mouse buttons to insert expression symbols, as follows:

1. Mark the text you want to replace with an expression with the left mouse button.
2. Press the right mouse button for a list of replacement symbol choices.
3. Select the symbol from the list.

Note that these expressions do nothing unless you replace *pattern* with a match pattern and *n* with a number to compare against the value that the match pattern returns.

Related Topics:

- Quick start: how to create a policy
- Pattern-matching and variables

Examples of pattern matching in rule conditions

The following examples show some of the many ways in which the HP Operations pattern-matching language can be used.

- `Error`

  Recognizes any message containing the keyword `Error` at any place in the message. (It is case sensitive by default.)

- `panic`

  Matches all messages containing `panic` , `Panic` , `PANIC` at any place in the text, when case sensitive mode is switched off.

- `logon |logoff`

  Uses the OR `operator` to recognize any message containing the keyword `logon` or `logoff` .

- `^getty:<*.msg> errno<*><#.errnum>$`

  Recognizes any message such as: `getty: cannot open ttyxx errno : 6` -Or- `getty: can't open ttyop3; errno 16`

  In the example `getty: cannot open ttyxx errno : 6` , the string "cannot open ttyxx" is assigned to the variable `msg` . The digit 6 is assigned to the variable `errnum` . Note that the dollar sign ($) is used as an anchoring symbol to specify that the digit 6 will only be matched if it is at the end of the line.

- `^errno[ |=]<#.errnum> <*.errtext>`

  Matches messages such as: `errno 6 - no such device or address` -Or- `errno=12 not enough core` .

  Note the space before the OR `operator` . The expression in square brackets matches either this blank space, or the "equals" sign. The space between `<#.errnum >` and `<*.errtext >` is used as a delimiter. Although not strictly required for assignments to the variables shown here, this space serves to increase performance.

- `^hugo:<*>:<*.uid>:`

  Matches any `/etc/passwd` entry for user hugo and returns the user ID to variable `uid` . Notice that ":" in the middle of the pattern is used to delimit the string passed to `uid` from the preceding string. The colon ":" at the end of the pattern is used to delimit the string passed to `uid` from the succeeding group ID in the input pattern. Here, the colon is necessary not only as a speed enhancement, but also as a means of logical separation between strings.

- `^Warning:<*.text>on node<@.node>$`

  Matches any message such as: `Warning: too many users on node hpbbx` and assigns `too many users` to `text` , and hpbbx to `node` .

- `<<#> -le 45>`

This pattern matches all strings containing a number which is less than or equal to 45. For example, the message: *ATTENTION: Error 40 has occurred* would be matched.

Note that the number 45 in the pattern is a true numeric value and not a string. Numbers higher than 45, for instance, "4545" will not be matched even if they contain the combination, "45".

- `<15 -lt <2#> -le 87>`

  This pattern matches any message in which the first two digits of a number are within the range 16-87. For instance, the message: *Error Message 3299* would be matched. The string: *Error Message 9932* would not be matched.

- `^ERROR_<[<#.err>] -le 57>`

  This pattern matches any text starting with the string "ERROR_" immediately followed by a number less than, or equal to, 57.

  For example, the message: *ERROR_34: processing stopped* would be matched and the string 34 would be assigned to the variable, *err* .

- `<120 -gt [<#>1] -gt 20>`

  Matches all numbers between 21 and 119 which have 1 as their last digit. For instance, messages containing the following numbers would be matched: 21, 31, 41... 101... 111 and so on.

- `Temperature <*> <@.`*plant* `>: <<#> -gt 100> F$`

  This pattern matches strings such as: "Actual Temperature in Building A: 128 F". The letter "A" would be assigned to the variable, *plant* .

- `Error <<#> -eq 1004>`

  This pattern matches any message containing the string "Error" followed by a space and the sequence of digits, "1004".

  For example, *Warning: Error 1004 has occurred* would be matched by this pattern. However, *Error 10041* would not be matched by this pattern.

- `WARNING <<#> -ne 107>`

  This pattern matches any message containing the string "WARNING" followed by a space and any sequence of one or more digits, except "107". For example, the message: *Application Enterprise (94/12/45 14:03): WARNING 3877* would be matched.

Related Topics:

- Pattern-matching and variables

Test pattern matching

HP Operations provides the ability to test the match patterns that you write for your policies. You can test either individual patterns, or all the patterns in one policy. It is a good idea to test the pattern matching in your rules , to make sure that the policy actually produces the kinds of messages that you want to receive.

1. To test all rules, select the Rules tab in the policy editor and select Matching test..

   To test one rule, right-click in the text box that contains the pattern you want to test, and select Matching test...

   The match patterns from all rules in the policy are visible in the upper window, titled Patterns to test .

2. Click Copy from file... , navigate to a file that contains text against which you want to test your patterns, and click OK . The file will be visible in the Lines to match dialog. If you don't want to load an entire file, you can select Add Line... and type a few lines that you can use for the test. You can edit the lines in the window if you want to change the test.

3. Click Test patterns and all the patterns will be tested against the lines to match. The Status column indicates if the one of the patterns matched each line, and the Rule No. column indicates which rule or rules matched.

   If you did not get the results that you expected, you can double-click any pattern or line to match, modify it, and try again.

   NOTE:
   Remember that the rule order is important. If rule number 1 matches a line, that line will not be tested against any other rule.

4. If you want to use the changes you have made, select Apply changes .

Related Topics:

- Quick start: how to create a policy
- Pattern-matching and variables

Command-line Programs

HP Operations provides several command-line programs that allow you to interact directly with the agent, instead of using a policy.

See the help topics in Administering your Environment → HPOM Application Integration Guide → Command Line Utilities → Agent Command Line Utilities or Server Command Line Utilities for complete details.

Sending Messages through Email

This method for sending messages through email relies on the WMI instance class OV_Message that HP Operations Manager for Windows creates whenever a message is received by the management server. A WMI policy monitors the namespace for these instances. When an instance is created that matches criteria that you specify in the policy, an automatic command runs an executable which extracts the message information from WMI, formats it according to your specifications, and then sends it to an email address that you specify in the policy.

NOTE:
If message counters are enabled, only one message (the original) is sent by email. The duplicates to the original message are not sent by email. When the original message is acknowledged and a duplicate to the acknowledged message arrives, a new email will be created.

To receive messages through email:

1. Modify the automatic command
   a. Start the HP Operations Manager for Windows Management Console.
   b. Locate the policy *Samples/Send Email/FwdMsgAsEMail*
   c. Double-click the policy to open it.
   d. Click the *Rules* tab and use the *Automatic Command* link to access the following command:
      ```
      cscript.exe "%OvInstallDir%\bin\FwdMsgAsEMail.vbs" -MsgId
      <$WBEM:TargetInstance.Id> -to "Recipient@RecipientDomain.com" -from
      "Sender@SenderDomain.com" -mailsrv "fqdn of SMTP server" -name "HP Operations
      Manager"
      ```
      Perform the following changes:
      i. Replace `Recipient@RecipientDomain.com` with the email address you want to send the email to (for example Joe@mailserver.de). The email client contacts the RecipientDomain server by way of SMTP.
      ii. Replace `Sender@SenderDomain` with the email address you want to use to send the email from.
      iii. Replace "fqdn of SMTP server" with the fully qualified domain name of the SMTP server you want to use to send the emails.
2. Indicate which messages should be sent. You must now modify the rule with conditions that match

specific properties of the instance class OV_Message. For example, to forward every message with severity Critical, add a rule that looks like this:

Property of	`TargetInstance`
Property name	`Severity`
Operator	`== equal`
Select value or property	`value`
Specific value to compare	32

If you want to match more than one type of message, copy the rule and modify the conditions. You can also modify the email recipient if different messages should be mailed to different email addresses.

CAUTION:
The policy sends a message to the message browser to verify that email was sent. It is important that the match criteria that you define for each rule does not also match this verification message. If it does, then a loop condition will occur.

3. Save the policy and deploy to the management server .

Reference Information

FwdMsgAsEMail.vbs

This script retrieves important information message and node properties from WMI and calls `OvEpMail.exe` with the corresponding parameters to send an email.

Usage

```
FwdMsgAsEMail.vbs -from <Sender@SenderDomain.com name> -name <display name>
                  -to <Recipient@RecipientDomain.com>
                  mailsrv <fqdn of SMTP server> -MsgId <Message ID>
                  -subject <subject text> optional
```

-from:	`<Sender@SenderDomain.com>` is the senders email address
-name:	`<display name>` is the name that will be displayed as originator
-to:	`<Recipient@RecipientDomain.com>` is the recipients email address
-mailsrv:	`<fgdn of SMTP server>` is the fully qualified domain name (for example, `mail.mydomain.com`) of the SMTP server to be used for sending the email. address
-MsgId:	`<Message ID>` is an ID that identifies the message. It is usually passed through the HP Operations Manager for Windows variable `<$WBEM:TargetInstance.Id>` It is used to retrieve the message data from WMI.
-subject:	`<subject text>` specifies the text within the subject field of the email.

The script *FwdMsgAsHtmlEMail* is also available. Both scripts share the same parameters, but produce somewhat different output.

FwdMsgAsEMail.vbs takes the message text and passes it to OvEpMail with the option `-format convert` . This produces an email consisting of the preformatted message text.

FwdMsgAsHtmlEMail takes more properties of the message from WMI and passes them to OvEpMail with the `-format HTML` . This creates a formatted HTML page.

OvEpMail.exe

OvEpMail.exe is a command line mail client.

Usage

```
OvEpMail.exe -to <Recipient@RecipientDomain.com>[;<Recipient2@Recipient@Domain.xxx;<...>
             -from <Sender@SenderDomain.com name> [-name <display name>]
             -mailsrv <fqdn of SMTP server> -format <plain|HTML|convert>
             -body <body text> [-subject <subject text>]
```

-to:	`<Recipient@RecipientDomain.xxx>[;<Recipient2@Recipient2Domain.xxx>` ; <...>] is one or more recipient(s) email address. (Entering more than one email address separated by ";" is optional.)
-from:	`<Sender@SenderDomain.com>` is the sender's email address
-mailsrv:	`<fqdn of SMTP server>` is the fully qualified domain name (such as "mail.mydomain.com") of the SMTP server to be used for sending the emails.
-name:	`<display name>` is the name that will be displayed as originator
-format:	The format parameter takes the three values `plain` , `HTML` or `convert` plain: The body text of the email is a text that consists of ASCII characters. It will be sent as an email in text format. HTML: The body text is an entire HTML page. Every character is an ASCII character. This page will be sent as an email in HTML format. convert: With this option the body text may contain any character. The text will be sent as a preformatted HTML email. That means, there will be a replacement of non ASCII characters and HTML special characters. They will be replaced in the HTML style, with &#`<integer value of the character>` . This option enables you to send many kinds of texts, for example, HTML listings, texts including Chinese characters, and so on.

`-body`:	`<body text>` is the message you want to send.
`-subject`:	`<subject text>` specifies the text within the subject field of the email.

Node Info Policy Type

The node info policy type provides a way to modify configuration information on a managed node . It is primarily a tool for configuring the agent to communicate through a firewall, and a troubleshooting tool to be used when working with a Hewlett-Packard consultant.

A node info policy writes values in the nodeinfo file. This file is created automatically when HP Operations installs an agent on a node. Deploying a node info policy to a node will cause the values in the node info policy to be written to the end of the node info file. Removing the policy deletes the values. If values are defined twice in this file, the value that is defined last (from top to bottom) is the value that is used.

NOTE:
If you want to set some of these values and want to ensure that they are never changed by a node info policy, you can write most of them in the opcinfo file, as well.(The only exceptions are parameters that set values relating to HTTP communication. These values may only be set in the nodeinfo file.) Information in the opcinfo file takes precedence over the nodeinfo file.

The node info policy type provides a simple editor where parameters can be typed. Each parameter consists of a name and a string value. Only ASCII characters are permitted. The string value may not contain new line characters. Example:

```
OPC_MGMT_SERVER endive.veg.com
```

The name starts at the beginning of the line and ends at the first white space (space or tab). The value starts after the white spaces and ends at the end of the line. Parameter can be disabled by inserting a number sign (#) at begin of the name.

It is important to be somewhat cautious when writing and deploying node info policies. If a parameter is defined in more than one policy, the value in the policy that was deployed last is the value that will be used. However, when a node info policy is removed from the node, the value for the parameter is not rolled back to previous value but rather is set to the default value. This can make it difficult to know what the configuration state on the node actually is. To prevent confusion, it is recommended either to deploy only one node info policy per node, or to create one node info policy for each specific parameter or unique group of parameters that you want set.

Related Topics:

- Node Info policy keywords

Node Info Policy Parameters

CAUTION:
The node info policy is used primarily for troubleshooting and support purposes. The parameters documented here are a small subset that may help you to address some issues. Be aware, however, that you can substantially reduce the performance of the agent if you use these parameters incorrectly. In most cases, you will use a node info policy only if you are working with Hewlett-Packard to resolve a problem or to achieve a management scenario that requires advanced configuration techniques. Hewlett-Packard consultants may direct you to use parameters that are not listed in this documentation.

OPC_RESOLVE_IP

Description :	Specifies the IP-address of the managed node 's primary manager. This parameter can be used if name resolution is not working correctly in your network environment. Note that any changes dictated by an agent-based flexible management policy will override the value set here.
Type :	string, a.b.c.d (for example, 15.136.120.1)
Default :	(not set)

OPC_COND_FIELD_ICASE

Description :	Toggles the case-sensitivity of policy conditions that match the object, application, or message group fields. TRUE = case-insensitive. FALSE = case-sensitive.
Type :	Boolean
Default :	TRUE

OPC_INT_MSG_FLT

Description :	If TRUE, agent-internal messages (mainly HPOM-internal status- and error-messages) are passed to the HP Operations agent and can be filtered through opcmsg policies. This allows you to add your own actions, instruction text and so on. Information on how to configure filtering of server-internal messages (mainly agent health checks) can be found in the topic Agent Health Check
Type :	Boolean
Default :	FALSE

OPC_LE_CLOSE_MSG_DLL

Description :	The default value is TRUE. This causes the Windows message DLL for EventLog messages to be closed after every read. This might increase CPU usage of the log file encapsulator process but it prevents the agent from locking DLLs, allowing software to be updated while the HP Operations agent runs.
Type :	Boolean
Default :	TRUE

OPC_AGENT_LOG_SIZE

Description :	Specifies the maximum size for the agent log files (opcerror, opcerro1, opcerro2, opcerro3) in increments of 1/10 KB (default 10000 * 1/10 KB = 1000 KB = 1 MB). When the current log file reaches 1/4 of that maximum size (default 1/4 * 1000 KB = 250 KB), it is moved to the next name (opcerror → opcerro1 → opcerro2 → opcerro3 → deleted) and a new opcerror log file is created. This parameter is valid for the DCE agent only.
Type :	int
Default :	10000

OPC_BUFLIMIT_ENABLE

Description :	Enable or disable checking of buffer-file limit on agent. Checks applied on msgagtdf-file. If TRUE the file will not grow unchecked and fill the disk if the management server becomes temporarily unavailable. The message agent counts the number of discarded messages, started actions, and message operations like acknowledge requests, and forwards them when the server becomes available again. This parameter is valid for the DCE agent only.
Type :	Boolean
Default :	FALSE

OPC_BUFLIMIT_SIZE

Description :	If buffer-file limitation is set on agent, this value describes the limit for msgagtdf-file in kilobytes. This parameter is valid for the DCE agent only.
Type :	int, kilobytes
Default :	10000

OPC_BUFLIMIT_SEVERITY

Description :	Allows you to define a severity that overrides the buffer file limit, and allows messages of that severity, or higher, to be added to the buffer file. This parameter is valid for the DCE agent only.
Type :	string : normal, warning, minor, major, critical
Default :	major

OPC_NAMESRV_CACHE_SIZE

Description :	HP Operations agent processes use a name-resolution cache in the trap interceptor process to improve performance. If the cache is full, least frequently used entries are replaced by new ones. If a node is the SNMP target for over 100 nodes, it is useful to enlarge the cache.
Type :	int
Default :	100

OPC_NAMESRV_DISABLE_CACHE

Description :	Enable or disable the HPOM name-service cache. This can be useful if the node names in your environment change frequently.
Type :	Boolean
Default :	FALSE

OPC_MSI_CREATE_NEW_MSGID

Description :	Control the how message-IDs are created when messages are sent to the message stream interface (MSI). 1 = Create a new message ID each time a message attribute is changed or the copy-operator is called. 2 = Set no new message ID when attributes are changed if this message was sent to only one instance. (The message must be 'diverted' for that, not 'copied' so that the HP Operations server or other MSI API-users has also a copy of it.) If you apply the API copy-operator to a message, the copy is no longer 'diverted' and later attribute changes lead to a new message ID. Note that the message → original message-ID attribute which is accessible for API-users contains the original message ID, if it was changed (otherwise it contains a null-id). 3 = Same as 2, except that the copy-operator immediately creates a new message ID for the copy. 4 = Message IDs are not modified at all. The API-user is responsible for it.
Type :	int, 1 <= n <= 4
Default :	2

PROXY

Description :	Sets the proxy for any HP BTO Software HTTP clients running on the computer. Clients can be Reporter or Performance Manager (running on the management server) or the Service Discovery agent (running on a managed node). The format is `PROXY port +(a)-(b); proxy2:port2 +(c)-(d);` and so on. The variables a , b , c and d are comma separated lists of hostnames, networks, and IP addresses that apply to the proxy. Multiple proxies may be defined for one PROXY key. '-' before the list indicates that those entities do not use this proxy, '+' before the list indicates that those entities do use this proxy. The first matching proxy is used. Example: `PROXY web-proxy:8088-(*.veg.com)+(*.lettuce.veg.com)` Meaning: the proxy 'web-proxy' will be used with port 8088 for every server (*) except hosts that match `*.veg.com` , for example, `www.veg.com` .

	The exception is hostnames that match `*.lettuce.hp.com` . For example, `romaine.lettuce.veg.com` the proxy server will be used. Refer to the *HP Operations Manager for Windows Firewall Configuration* white paper for more information.
Type :	string
Default :	not set

CLIENT_BIND_ADDR(*app_name*)

Description :	Sets the address for the specified application's HTTP client. Valid application names are `com.hp.openview.CodaClient` (on the management server) and `com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML` (on the managed node). Example: CLIENT_BIND_ADDR(com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML) 65.114.4.69 Refer to the *HP Operations Manager for Windows Firewall Configuration* white paper for more information.
Type :	string
Default :	not set

CLIENT_PORT(*app_name*)

Description :	Sets the port number for the specified application' HTTP client. Valid application names are `com.hp.openview.CodaClient` (on the management server) and `com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML` (on the managed node). Example: `CLIENT_PORT(com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML) 8003` Refer to the *HP Operations Manager for Windows Firewall Configuration* white paper for more information.
Type :	string
Default :	not set

SERVER_BIND_ADDR(*app_name*)

Description :	Sets the address for the specified application's OpenView HTTP server. Valid application names are `com.hp.openview.Coda` (on the managed node) and `com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML` (on the management server). Example: `SERVER_BIND_ADDR(com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML) 65.114.4.69` Refer to the *HP Operations Manager for Windows Firewall Configuration* white paper for more information.
Type :	string
Default :	not set

SERVER_PORT(*app_name*)

Description :	Sets the port number for the specified application's OpenView HTTP server. Valid application names are com.hp.openview.Coda (on the managed node) and com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML (on the management server). Example: `SERVER_PORT(com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML) 8001` Refer to the *HP Operations Manager for Windows Firewall Configuration* white paper (available from your Hewlett-Packard representative) for more information.
Type :	string

Default :	`SERVER_PORT(com.hp.openview.Coda) 381`
	`SERVER_PORT(com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML) 6602`

OPC_IP_ADDRESS

Description :	When a node has several IP addresses, this parameter configures the agent to always use a specific IP address. This parameter is valid for the DCE agent only.
Type :	string (for example, 192.168.1.1)
Default :	not set (the agent uses the first IP address it finds)

OPC_ALTERNATIVE_AGENT_IDS

Description :	Sets alternative agentIDs for a particular node. Usually used in cluster environments. Example: `OPC_ALTERNATIVE_AGENT_IDS agentid1,agentid2` See Enable policy switching on DCE agents for more information.
Type :	string
Default :	Not set

SNMP_SESSION_MODE

Description :	Determines how the HP Operations agent intercepts SNMP events. To intercept SNMP V2 events on Windows managed nodes, disable the standard Windows SNMP service, which does not support SNMP V2 events. Set the keyword SNMP_ SESSION_MODE to NNM_LIBS in the node's opcinfo file. The keyword supports the following values: TRAPD: Use the Operations event interceptor (opctrapi). NO_TRAPD: Use direct port access mode. TRY_BOTH: Try both, the trap daemon and direct port access mode. NNM_LIBS: Use the OpenView NNM libraries (shipped with the Operations agent) to receive events. The opcinfo file can be found in these locations: AIX: /usr/lpp/OV/OpC/install/opcinfo UNIX: /opt/OV/bin/OpC/install/opcinfo Windows: <drive>: \usr\OV\bin\OpC\install\opcinfo
Type :	int
Default :	Not set

Related Topics:

- Node Info Policy Type

ConfigFile policy type

ConfigFile policies are used by HP Operations Smart Plug-ins (SPIs) to enhance the management capabilities for specific enterprise applications like SAP and Microsoft Exchange. These applications may require advanced monitoring and management capabilities that are not available through the standard set of HPOM policy types.

Therefore many SPIs include complex instrumentation that must be configured after deployment to nodes. ConfigFile policies perform this configuration task. They consist of configuration files that contain a set of rules or instructions for the SPI instrumentation. You can deploy ConfigFile policies like all other policy types.

NOTE:
ConfigFile policies for DCE agents cannot be enabled and disabled. This is only possible for HTTPS agents.

Related Topics:

- ConfigFile policy General tab
- ConfigFile policy Data tab
- Instrumentation
- Policy deployment
- opcdeploy

ConfigFile policy General tab

When a SPI is installed, one or more so-called ConfigFile varieties are also installed. ConfigFile varieties consist of the application, subgroup, and file name attributes. These three attributes determine the path and file name of the configuration file that is associated with the policy.

When a policy is newly created, the fields in the General tab can be changed. When an existing policy is edited, the fields cannot be changed.

- Application: Specifies the name of the managed application. This usually equals the name of the SPI itself, for example `dbspi` or `sapspi` .
- Sub-Group: Additional grouping mechanism that helps the SPI to manage configuration files by grouping them according to custom categories. For example, `sapspi` uses the subgroup attribute to differentiate between `global` and `local` scope of the configuration. `dbspi` has one subgroup for every supported database vendor.
- File name: Specifies the file name of the configuration file. For example, `sapspi` 's monitor instrumentation configuration files have `r3monxxx.cfg` names, where xxx is the abbreviation of the particular monitor.

NOTE:
Some SPIs may allow you to specify your own configuration file in the File name field. If this is the case, make sure to use alpha-numeric characters only in the file name. Special characters are not allowed.

Related Topics:

- ConfigFile policy Data tab
- ConfigFile policy type
- Instrumentation
- Policy deployment
- opcdeploy

ConfigFile policy Data tab

Use the Data tab to modify the configuration file that is written to the node when the corresponding ConfigFile policy is deployed. The ConfigFile text appears in the ConfigFile Content edit field.

Syntax and keywords

The syntax and keywords used in configuration files is determined by the SPIs and described in the SPI documentation.

The following generic keywords can be used at the beginning of all configuration files to notify external applications, for example HP Performance Agent Software, when a ConfigFile policy is added to or removed from a directory that is of particular interest to that application.

Keyword	Description

`#$Installcommand=<command>`
`#$Deinstallcommand=<command>`
	<command> contains the command to be run, including all required parameters. If necessary, use quotation marks to handle all platforms.

`#$Commandtype=<value>`	<value> specifies the type of command to be used: 1 Executable (default) If you do not specify the command type, the ConfigFile policy assumes that the command is an executable. 2 VBScript or shell script You do not need to add a .vbs or .sh extension to the command. HPOM automatically appends the appropriate extension so that a single policy can be run on both Windows and UNIX nodes. 3 Perl script

NOTE:
Although the ConfigFile policy editor supports non-ASCII characters, you will receive an error message when you deploy such policies to DCE agents. Only HTTPS agents accept ConfigFile policies with non-ASCII characters.

Template files

SPIs can install one template file for each ConfigFile variety. If a template is present, the Load Template and Save as Template buttons are available.

To load a template

1. Click Load Template to load a template into the edit field.

   If the edit field already contains text, you are asked if you want to replace the data with the template or cancel. Click Yes to replace the data in the edit field with the template information. You can modify the existing template and save your changes.
2. Click Save as Template to save the data to the template file. You are asked if you want to replace the existing template.
3. Click OK to replace the original template with the modified template.

Syntax validation

SPIs can install a validation mechanism that lets you verify the text in the ConfigFile Content field to ensure the syntax used is correct. Click Check Syntax to start the validation tool.

Help information

SPIs install help information with each ConfigFile variety. The help topic usually provides information about configuration options and syntax. Click Help on ConfigFile to view the help topic.

TIP:
You can split the text area vertically and horizontally by dragging the split controls. (The split controls are at the top of the vertical scroll bar and at the left of the horizontal scroll bar.)

Related Topics:

- ConfigFile policy General tab
- ConfigFile policy type
- Instrumentation
- Policy deployment
- opcdeploy

Policy Management and Deployment

Policies are collections of configuration information used to control the agent on a managed node. Using HPOM, administrators can deploy policies on various computers to provide consistent, automated administration across a network.

Policies fall into two broad classifications: monitor policies and configuration policies . With monitor policies, you decide what kinds of events to monitor, how often to monitor, what to look for in the events, and what to do if certain events are detected. With configuration policies, you can change settings that determine which management server the agent reports to, how large buffer files should be, which proxy should be used for communication through a firewall, and so on.

This portion of the help explains the organizational tools that HPOM provides to help you manage the policies that you create, and explains how to deploy the policies to a managed node .

NOTE:
You must have the appropriate user rights to work with policies. See Configure policies for user roles for more information.

The policy management and deployment functionality runs as service (OvPmad), which you can stop and start by means of the standard Windows services manager. This is particularly advantageous if the management server runs in a high-availability environment, where the cluster software needs to be able to stop and start services on demand in the event of a system fail over.

It is also essential to be able to stop and start HPOM-related Windows services manually when you enable or disable the security-audit feature on the management server, for example if you want to audit which policies have been renamed, copied, or modified and, in addition, which policies or packages have been deployed to the managed nodes, when, and by whom.

HPOM provides a number of command line utilities you can use with the policy editor. See the help section Command-Line Utilities for details.

Related Topics:

- Choose a policy type
- Policy development
- Deploy a policy or policy group
- Audit policy management and deployment
- Enable and disable security audits

Policy properties

Every policy has a set of policy properties . These properties include:

Policy Related Information:

Policy Name
: The name under which the policy was saved.

Version
: The version that was assigned to this version of the policy when it was saved.

Version ID
: The GUID that was assigned to this version of the policy when it was saved. Each version of a policy has a unique ID.

Policy ID
: The GUID that was assigned to the policy when it was created. Different policies have unique Policy IDs, while different *versions* of one policy have the same Policy ID.

Last Modification
: The date that this version of the policy was saved.

Last Modified By
: The domain and user name active when this version of the policy was saved.

Category
: A comma-separated list of strings. You can use categories to associate policies with instrumentation and user roles.

Description
: A description of the policy. This is the only policy property that can be changed.

Policy Type Related Information:

Name
: The name of the policy type to which this policy belongs.

Version
: The version of the policy type.

Version ID
: The GUID assigned to this version of the policy type.

Policy Group Related Information:

Policy Groups

A list of policy groups that contain this policy.

Only show for current version

Enables you to restrict the list of policy groups to show only the groups that contain this particular version of the policy.

Policy Deployment Related Information:

Deployed on nodes

A list of nodes to which this policy is currently deployed.

Only show for current version

Enables you to restrict the list of nodes to show only the nodes that have this particular version of the policy.

Related Topics:

View policy properties

View policy properties

Policy properties include the version, modification date, description, and so on. You can also view a list of policy groups that contain this policy, and a list of nodes to which this policy is currently deployed.

1. Right-click the policy in the details pane.
2. Select **Properties** .

NOTE:
Policy properties can only be accessed from policies displayed in the policy management portion of the console tree. Properties are not available from the policy inventory view.

Related Topics:

- Policy properties

Policy versioning

When a policy is newly created, or is saved under a new name, that policy is saved as version 1.0 . Each time a policy is saved under the same name, a new version number is suggested by the policy editor , with the number to the right of the period automatically incremented (1.1, 1.2, and so on). You can change this number before saving. The number to the left of the period can also be changed by the user, but is never automatically incremented. The largest number available is *9999.9999* .

The policy with the largest version number is considered to be the newest version of the policy. Other factors, such as creation date, are not considered. You cannot save a policy with a version number that already exists for that policy name.

NOTE:
The numbers to the right and left of the period are both regarded as integers, and any leading zeros are ignored. This means that 0.10 is a more recent version than 0.9, and that 0.01 is identical to 0.1.

Because each saved version of a policy has a different version number, you can access any previously saved version of a policy. These older versions can be edited or deployed just as any other policy.

CAUTION:
If you change the number to the left of the period when saving a new version of a policy, for example, 4.5. to 5.0, the policy editor prompts you to confirm this choice. This prompt is designed to avoid conflicts with version numbers in future releases of HPOM. To avoid your changes being overwritten by future installations, try to use *only* the numbers to the left of the period to distinguish policy versions. For example, change 4.5 to 4.6.

Related Topics:

- Save policy as
- View policy properties
- Add a policy to a policy group

Policy view filter

The policy view filter lets you determine what kinds of policies are visible in the details pane when you select a policy type in the console tree. You can choose to see only the latest versions of each policy, or all versions. You can also filter on the deployment state, or on whether the policy is assigned to a policy group .

Selection criteria:

- Latest version of all policies . This selection will show only the highest version number of each policy.
- All versions of the policy below . This selection allows you show all versions of one specific policy that you choose.
- node deployment state allows you to indicate which policies should be shown, based on whether they are deployed on some managed node . You can choose between these three options:
  - deployed
  - not deployed
  - doesn't matter
- group assignment state allows you to indicate which policies should be shown, based on whether they are assigned to any policy group under *Policy groups* . You can choose between these three options:
  - assigned
  - not assigned
  - doesn't matter

To access this window:

1. In the console tree, under Agent policies grouped by type or Server policies grouped by type , right-click a policy type, for which you want to filter the policy view.
2. Select Set Filter...

Related Topics:

- View installed policies
- View installed packages and subpackages

Policy groups

Policy groups are sets of policies that share some common attribute or logical connection. They enable you to more easily work with multiple policies simultaneously. For example, you can deploy all the policies in a group to managed nodes together.

HPOM and some Smart Plug-ins create policy groups automatically, which you can see in the console tree under Policy management . You can also create your own policy groups, and change the groups that policies belong to.

Related Topics:

- View policies by group
- Find policies in policy groups
- Create new policy group
- Delete policy group
- Add a policy to a policy group
- Remove policy from policy group
- Deploy a policy or policy group
- Update to latest

View policies by group

You can view policies by selecting a policy group in the console tree. The policies in this group are visible in the details pane.

Related Topics:

- View installed policies
- View policy properties

Find policies in policy groups

You can search the policy groups to find a specific policy within a group. This helps you to locate policies more quickly (for example to remove them from the group or update them).

To find policies in policy groups

1. In the console tree, right-click Policy Management , and then click All Tasks → Find Policy... . The Find Policy dialog box opens.
2. Type a search string in either or both Name and Description .
3. *Optional.* Select the Case sensitive check box if you want your search results to match exactly the characters that you typed in Name and Description box.
4. Click OK . The Find Policy Results dialog box opens, which shows the policies that match your criteria.
5. Click a policy in the list, and then click Select . The policy group is selected in the console tree, and the policy is selected in the details pane.

Related Topics:

- Policy properties

Create new policy group

You can create policy groups . under the Policy groups icon in the console tree.

1. Select Policy groups in the console tree. (To create a nested policy group, select an existing policy group.)
2. Right-click the selected group.
3. Select New → Policy Group .

   A policy group with the name New Group is created.
4. Type a meaningful name for the new policy group.

Related Topics:

- Add a policy to a policy group
- Delete policy group

Delete policy group

You can delete any policy group under Policy groups . If the policy group that you delete contains policies or other policy groups, they are also deleted. Note that the policies are not deleted from the management server and can be accessed again under Agent policies grouped by type or Server policies grouped by type .

To delete a policy group

1. Right-click the policy group in the console tree and select Delete .

Related Topics:
- Edit an old version of a policy
- Delete a policy from the management server
- Remove policy from node

Add a policy to a policy group

1. Right-click the policy type or policy group that contains the policy you want to add to a group.
2. Right-click the policy and select **Copy** .
3. Right-click the policy group to which you want to add the policy and select **Paste** .

NOTE:
You can also use drag-and-drop to add policies to policy groups. Policies dragged from a policy type will be copied. Policies dragged from a policy group will be moved— if the shift key is pressed while dragging, they will be copied.

Related Topics:

- Assign the latest policy version to a group
- Remove policy from policy group

Remove policy from policy group

You can remove a version of a policy from one of your policy groups . The policy will not be removed from other groups, and will not be deleted from the management server .

To remove a policy from a policy group

1. Select the policy group from which you want to delete the policy.
2. Right-click the policy you want to delete from the group and select Delete .

Related Topics:

- Delete a policy from the management server

Assign the latest policy version to a group

By default, a policy group contains the specific policy version that you add to the group, even if newer versions of the policy exist. You can change this behavior so that the policy group always contains the latest version of the policy.

However, if you do not always want the policy group to contain the latest version, you can force an update by right-clicking the policy and choosing All Tasks → Update to latest .

To assign the latest policy version to a group

1. Open the policy group that contains the policy.
2. Right-click the policy, and then click Properties .
3. Select or clear Always Use Latest Policy Version .
4. Click OK .

Related Topics:

- Update to latest

Update to latest

To ensure that all the policies in a policy group contain the most current versions:

1. Right-click the policy group.
2. Select All Tasks → Update to latest .

You can also use this procedure on a specific policy, instead of on a policy group. The policy will be updated to the latest version in the policy group where you accessed it.

NOTE:
Remember that the most current version of a policy is the policy with the highest version number.

Related Topics:

- Assign the latest policy version to a group
- Policy versioning

Policy deployment

You can install policies on a management server as part of a Smart Plug-in, or develop them yourself. You can then use the policies to configure or monitor nodes by deploying the policies to the nodes.

For each node, the management server maintains a policy inventory, which enables you to see which policies you have already deployed to each node. This helps you to decide when to update, redeploy, or remove policies. You can also disable policies on individual nodes temporarily, and enable them again later.

When you deploy policies from the management server to nodes, the management server creates deployment jobs. You can follow the progress of these jobs to ensure that the deployment is successful.

By default, HPOM automatically deploys certain core policies to nodes. You can also automatically deploy additional groups of policies by associating policy groups with node groups and service types. If you prefer to manually deploy your policies, you can disable automatic deployment.

Related Topics:

- Deploy a policy or policy group
- Remove policy from node
- Delete policies manually from a node
- View installed policies
- Create a policy inventory report
- Synchronize policies and packages
- Reinstall policies
- Update policy on node
- Enable policy
- Disable policy
- Disable policy autodeployment

Deploy a policy or policy group

You can install policies on a management server as part of a Smart Plug-in, or develop them yourself. You can then use the policies to configure or monitor nodes by deploying them to the nodes. You can deploy individual policies or groups of policies to any number of nodes at the same time.

To deploy a policy or policy group

1. Open Policy management in the console tree and select the policies that you want to deploy:
   - To select one policy, click the policy in the details pane. You can also select multiple policies by pressing SHIFT or CTRL . If you cannot see the version of a policy that you want to deploy, change the policy view filter.
   - To select one policy group, click the policy group in the console tree or details pane. You can also select multiple policy groups by pressing SHIFT or CTRL .

   If you select a policy group that contains subgroups (a policy group tree), the management server deploys policies from the parent policy group first, and then deploys policies from the subgroups.

   If a policy group tree contains different versions of the same policy, the management server deploys only the latest policy.
2. Right-click your selection, and then click All Tasks → Deploy on... . The Deploy policies on... dialog appears.
3. Select the nodes that you want to deploy the policies to by selecting check boxes for the nodes and node groups in Managed nodes . If you selected only one policy, the following extra options appear, which enable you to automatically select nodes based on policy inventory:
   - Select all nodes on which the current version of the policy is deployed .
   - Select all nodes on which any version of the policy is deployed .

   If you need to modify the automatic selections, click Select nodes from the tree .
4. *Optional.* If you want to deploy the policies to nodes that already have a more recent version of the policy, clear deploy policy only if version is newer .
5. *Optional.* If you want to deploy the policies, but immediately disable them, select disable policy after deployment .
6. *Optional.* If another management server has already deployed a version of a policy to a node, that management server owns the policy on that node. If you want to redeploy the policy from this management server, and transfer the ownership, select the ignore policy owner check box. To do this, you must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node.
7. Click OK . The management server creates deployment jobs, which deploy the policies to the nodes. If

a policy requires a newer version of the agent than is currently installed on the node, the job also deploys the latest agent packages to the node.

TIP

You can also deploy policies and policy groups by drag and drop. Select policies or policy groups and then drag and drop them onto a node or node group in the console tree. This deploys the policies with the following settings:

- The management server does not deploy policies to nodes that have a newer version.
- The management server enables the policies that it deploys, unless an older version of the policy already exists on the node in the disabled state.
- The management server does not ignore policy ownership. If another management server already owns a policy on a node, the job that deploys the policy to that node fails. To transfer policy ownership, use the Deploy policies on... dialog and select the ignore policy owner check box.

Related Topics:

- Policy view filter
- Enable policy
- Deployment jobs

Remove policy from node

If you have previously deployed policies to nodes , but no longer require the monitoring or configuration that the policy provides, you can remove the policies from the nodes using the console.

To remove policies from nodes

1. Open Policy management in the console tree and select the policies that you want to remove:
   - To select one policy, click the policy in the details pane. You can also select multiple policies by pressing SHIFT or CTRL . If you cannot see the version of a policy that you want to remove, change the policy view filter.
   - To select one policy group, click the policy group in the console tree or details pane. You can also select multiple policy groups in the details pane by pressing SHIFT or CTRL .
2. Right-click your selection, and then click All Tasks →Uninstall from ... . The Uninstall policies on... dialog box opens. (If this menu item is not available, you selected a version of a policy that is not installed on any nodes.)
3. Select the nodes that you want to remove the policies from by selecting check boxes for the nodes and node groups in Managed nodes . If you selected only one policy, the following extra options appear, which enable you to automatically select nodes based on policy inventory:
   - Select all nodes on which the current version of the policy is deployed .
   - Select all nodes on which any version of the policy is deployed .

   If you need to modify the automatic selections, click Select nodes from the tree .
4. *Optional.* If you know that the job to remove the policy will fail (for example, because the node is unreachable), but you still want to remove the policy from the inventory on the management server , select the force policy removal check box.
5. *Optional.* If you want to remove the policies from nodes that have a different version of the policy than your selection, select remove all versions .

   This check box is automatically cleared if you select Select all nodes on which the current version of the policy is deployed and automatically selected if you select Select all nodes on which any version of the policy is deployed .
6. *Optional.* If another management server owns a policy on a node, this management server will not uninstall it by default. If you want to this management server to remove the policies that other management servers own, select the ignore policy owner check box. To do this, you must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node.
7. Click OK . The management server creates deployment jobs, which uninstall the policies.

TIP:

You can also remove a policy from a node in the policy inventory view:

1. In the console tree, right-click the node from which you want to remove a policy, and then click View → Policy Inventory .
2. In the details pane, right-click the policy that you want to remove, and then click All Tasks → Remove from node . The Confirm policy removal dialog appears.
3. *Optional.* Select the ignore policy owner check box if you want to remove a policy that is owned by a different management server.
4. *Optional.* If you know that the job to remove the policy will fail (for example, because the node is unreachable), but you still want to remove the policy from the inventory on the management server , select the force removal of policy check box.
5. Click Yes . The management server creates a deployment job, which uninstalls the policy.

Related Topics:

- Policy view filter
- Disable policy
- Delete a policy from the management server
- Remove package from node
- Deployment jobs

Delete policies manually from a node

Under certain circumstances, it is possible for the policy inventory on the management server to get out of synch with the actual managed nodes.

For example, when a SPI is uninstalled from the management server, a forced undeployment for all SPI policies is executed. That is, the SPI policies are removed from the policy inventory on the management server, even if the managed node is currently not running or just not reachable. After the node is up and running again, however, the policies (which where not removed) may still generate lots of messages.

These policies are no longer in the policy inventory on the server, it is not possible to remove them from the managed node using the console. If the node has the HTTPS agent, you can synchronize the inventory, and then remove the policy using the console. Alternatively, you can remove the policy manually. If the node has the DCE agent, you must remove the policies manually from the managed node.

The procedure to delete policies manually differs, depending on whether the node has a HTTPS agent or a DCE agent.

NOTE:
A forced policy undeployment can also be started using the PMAD API.

To delete policies manually from a node that has the HTTPS agent

1. Log in to the node as a user with administrative rights, and open a command or shell prompt.
2. On nodes that run a UNIX or Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - On HP-UX, Solaris, or Linux, type **`export PATH=/opt/OV/bin:$PATH`** and then press Enter .
   - On AIX, type **`export PATH=/usr/lpp/OV/bin:$PATH`** and then press Enter .
   - On Tru64, type **`export PATH=/usr/opt/OV/bin:$PATH`** and then press Enter .
3. Type `ovpolicy -list` , and then press Enter . A list of policies appears.
4. Use `ovpolicy -remove` to delete individual policies, or all policies.
   - To remove one policy, type `ovpolicy -remove -polname <`*`name`*` >` , and then press Enter .
   - To remove all policies, type `ovpolicy -remove -all` , and then press Enter .
5. *Optional.* In the console, right-click the node, and then click All Tasks→Synchronize policies . This updates the current policy inventory with up-to-date details of installed policies from the node.

TIP:

You can also remove a policy remotely from the management server by adding the `-host` option. For example: `ovpolicy -remove -all -host <`*`hostname`*` >`.

To delete policies manually from a node that has the DCE agent

1. Log in on the remote managed node.
2. Shut down the agent with the following command line:

   `opcagt -kill`
3. Delete all files in directory "%OvAgentDir%\conf\ConfigFile\policies" to remove all ConfigFile policies from the node. If this directory does not exist, no ConfigFile policies are currently deployed on the node.

   Delete all files in directory "%OvAgentDir%\conf\svcdisc\policies" to remove all Service Auto-Discovery policies from the node. If this directory does not exist, no Service Auto-Discovery policies are currently deployed on the node.

   Delete all files in directory "%OvAgentDir%\conf\nodeinfo\policies" to remove all Node Info policies from the node. If this directory does not exist, no Node Info policies are currently deployed on the node.

   Delete all files in directory "%OvAgentDir%\conf\mgrconf\policies" to remove all Flexible Management policies from the node. If this directory does not exist, no Flexible Management policies are currently deployed on the node.

   Delete all files in directory "%OvAgentDir%\conf\OpC\vpwin" to remove all other policies from the node.
4. Restart the agent with the following command line:

   `opcagt -start`
5. Open the console, connect to the server, and click the managed node in the console tree. Start the operation "Redeploy policies and instrumentation." Calling this operation ensures that the policy inventory on the server gets in synch with the managed node again.

Special care must be taken for the node info and flexible management policies which define the content of the following two configuration files on the managed node:

- %OvAgentDir%\conf\OpC\nodeinfo
- %OvAgentDir%\conf\OpC\mgrconf

If you manually delete policies of the Node Info or the Flexible Management policy type on the managed node, you must make sure that the content of the above two files is still valid before you restart the agent. If this is not the case, the agent may not work properly.

Related Topics:

- Synchronize policies and packages
- Remove policy from node

View installed policies

The management server keeps an inventory of the policies that are currently installed on the node . You can see which policies are installed on a node by viewing this policy inventory. If you have an environment in which multiple management servers manage the same nodes, you can see the which management server owns each policy.

Alternatively you can request a list of policies directly from the node. This does not check or update the policy inventory, which the management server stores in its database.

To view installed policies in the policy inventory

1. In the console tree, right-click the node for which you want to view the installed policies.
2. Click View → Policy Inventory . The policies installed on the node are visible in the details pane.

To request policy lists from nodes

1. In the console tree, click Tools → HP Operations Manager Tools .
2. In the details pane, double-click List policies installed on agent . A dialog appears that lists nodes and services.
3. Select Nodes and click Launch... . The Tool Status dialog appears and shows progress.
4. After an item in the Launched Tool list succeeds, click it. A list of policies for that node appears in Tool Output.

NOTE:
If the management server you are using is not a node's primary management server, the node must allow this management server to run actions for this tool to succeed. You can specify action allowed managers using flexible management policies.

Related Topics:

- Synchronize policies and packages
- Configure action-allowed and secondary managers
- Remove policy from node
- Disable policy
- Enable policy

Create a policy inventory report

With the tools for inventory reporting, you can view information about the policies and packages that are installed on every node managed by one management server .

To create and view a policy and package inventory report.

1. In the console tree, select Tools , Reporting , Inventory Reporting to display the reporting tools available:
   - Generate and view inventory report (creates and displays report in web browser)
   - Generate inventory report (creates report)
   - View inventory report (displays in the web browser the report previously generated)
2. Right-click the tool you prefer and select All Tasks →Launch Tool...
3. Wait for the xml report to be generated (this takes about 20 seconds, or longer, depending on the speed of your computer and the size of your managed environment). If you selected one of the view options, the report appears in the web browser. You can choose one of three views for the inventory data from the pull-down menu in the top-right corner of the browser.

For information about additional ovconfreporter options, type `ovconfreporter -?` at a command prompt.

Synchronize policies and packages

The management server keeps an inventory of the policies and packages that are currently installed on the node . It is possible for this inventory to become inaccurate in the following situations:

- After an administrator manually installs or removes the policies or packages on the node.
- After an administrator or program disables or enables a policy locally on a node.
- In an environment with multiple management servers, after a different management server changes the policies or packages on a node.
- After an administrator removes a policy or package using the force option, but the policy or package remains on the node.

You can update the inventory information on the management server by synchronizing. The management server replaces the current inventory with up-to-date details of installed policies and packages from the node.

If a node has a policy or package that does not exist on the management server, the management server adds a placeholder to the inventory. The management server does not receive the contents of the policy or package, so you cannot change it or deploy it to other nodes.

NOTE:
You can synchronize policies and packages on nodes that have the HTTPS agent. On nodes that have the DCE agent, you can only synchronize packages.

If a node has no agent (because, for example, an administrator removed it manually), it is not possible to synchronize policies or packages at all.

To synchronize policies and packages

1. In the console tree, right-click the node or node group that you want to synchronize.
2. Click one of the following menu items:
   - All Tasks→Synchronize inventory →Packages
   - All Tasks→Synchronize inventory →Policies
   - All Tasks→Synchronize inventory →Policies and packages

   The management server creates a deployment job to retrieve the inventory from each node.

Related Topics:

- View installed policies
- View installed packages and subpackages
- Configuring Agents
- Synchronize packages
- Deployment jobs

Reinstall policies

You can reinstall the same versions of all the policies that are currently deployed to selected nodes and node groups. In addition, this deploys the instrumentation required by the categories to which the policies belong. This option is useful to restore policies and instrumentation after you manually reinstall an agent on a node.

If you reinstall policies and instrumentation on a node, the deployment job first removes all existing policy-related instrumentation from the selected managed nodes. Any instrumentation previously that you deployed using the Deploy instrumentation dialog box is not redeployed unless it belongs to a category that is shared by one of the redeployed policies. The default instrumentation categories for the DCE agent (action, command, and monitor) are not usually linked to policies and, as a result, the reinstallation job does not remove or redeploy them. If you reinstall policies on managed nodes where only policies with no category exist, then the reinstallation job does not remove or redeploy any instrumentation.

NOTE:
If you reinstall policies, the management server does not reinstall the agent or any other packages.

To reinstall policies and instrumentation

1. In the console tree, select the nodes or node groups on which you want to reinstall policies and instrumentation.
2. Right-click your selection, and then click All Tasks → Reinstall/Update... . The Reinstall/Update Node dialog box opens. The reinstall option is already selected.
3. In the list under Scope , select Policies .
4. *Optional.* Select the Ignore missing policies/packages check box. If the node inventory contains policies that are not available on this management server, the management server creates a warning in the event log and the job succeeds. Otherwise, if you clear this check box, the job succeeds only if all the policies in the node inventory are available on this management server.
5. *Optional.* If another management server has already deployed a version of a policy to a node, that management server owns the policy on that node. If you want to redeploy the policy from this management server, and transfer the ownership, select the Ignore policy owner check box. You must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node. If you do not select this check box, the management server reinstalls only the policies that it owns.
6. Click OK . The management server creates a deployment job, which reinstalls the policies and instrumentation.
7. In the console tree, click Deployment jobs and use the details pane to verify that the deployment job has completed successfully.

NOTE:

Small deployment jobs sometimes complete so quickly that you do not get the chance to observe them. In addition, the list of deployment jobs which *remain* visible in the details pane for any length of time have typically failed, or are either pending or suspended.

Related Topics:

- Instrumentation
- Deploy a policy or policy group
- Reinstall all

Update policy on node

You can ensure that the latest version of all policies are deployed to a managed node , or update an individual policy on a node.

To update policies on a node

1. In the console tree, select the nodes or node groups for which you want to update policies.
2. Right-click your selection, and then click All Tasks → Reinstall / Update... . The Reinstall/Update Node dialog box opens.
3. Select Update .
4. In the list under Scope , select Policies .
5. *Optional.* Select the Ignore missing policies / packages check box. If the node inventory contains policies that are not available on this management server, the management server updates only the policies that are available. Otherwise, if you clear this check box, the job succeeds only if all the policies in the node inventory are available on this management server.
6. *Optional.* If another management server has already deployed a version of a policy to a node, that management server owns the policy on that node. If you want to update the policy from this management server, and transfer the ownership, select the Ignore policy owner check box. You must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node. If you do not select this check box, the management server updates only the policies that it owns.
7. *Optional.* If you want to deploy the policies to nodes that already have a more recent version of the policy, clear the Update only if version is newer check box. If you do this, the management server deploys the latest version of the policy that is available on the management server, even if the node already has the same version or a newer version.
8. Click OK . The management server creates deployment jobs, which update the policies.

To update an individual policy

1. In the console tree, click Policy management→Agent policies grouped by type . Click a policy type.
2. In the details pane, right-click the policy and select All Tasks →Deploy on... . The Deploy policies on... dialog box opens.
3. Select the nodes that you want to deploy the policies to by selecting check boxes for the nodes and node groups in Managed nodes . The following options appear, which enable you to automatically select nodes based on policy inventory:

- Select all nodes on which the current version of the policy is deployed .
- Select all nodes on which any version of the policy is deployed .

  If you need to modify the automatic selections, click Select nodes from the tree .

4. Click OK . The management server creates deployment jobs, which update the policies.

Related Topics:

- Policy versioning
- Deployment jobs

Enable policy

You can enable a policy that you disabled. If you enable a previously disabled policy, it begins to function again on the node where it is installed.

To enable a previously disabled policy

1. In the console tree, select the node where you want to enable a policy.
2. From the menu bar, select View → Policy Inventory .
3. In the details pane, right-click the installed policy that you want to enable.
4. Select All Tasks → Enable .
5. *Optional.* If another management server owns the policy on that node, a dialog box opens for confirmation. To enable the policy using this management server, click Yes . To do this, you must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node.

   If you click No, the management server still creates a job to enable the policy, but the job fails. You can restart the job with new options to ignore the policy owner, or cancel the job.

Related Topics:

- Disable policy
- Remove policy from node
- Configure action-allowed and secondary managers
- Restart job with new options
- Cancel job

Disable policy

If you disable a policy , the policy remains installed on the node , but does not function until it is enabled.

To disable a policy

1. In the console tree, select the node where you want to disable a policy.
2. From the menu bar, select View → Policy Inventory .
3. In the details pane, right-click the installed policy that you want to disable.
4. Select All Tasks → Disable .
5. *Optional.* If another management server owns the policy on that node, a dialog box opens for confirmation. To disable the policy using this management server, click Yes . To do this, you must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node.

   If you click No, the management server still creates a job to disable the policy, but the job fails. You can restart the job with new options to ignore the policy owner, or cancel the job.

Related Topics:
- Enable policy
- Remove policy from node
- Configure action-allowed and secondary managers
- Restart job with new options
- Cancel job

Disable policy autodeployment

By default, HPOM automatically deploys certain core policies to nodes. You can also automatically deploy additional groups of policies by associating policy groups with node groups and service types.

If you prefer to manually deploy your policies, you can disable automatic deployment for all nodes and services. Alternatively, you can disable autodeployment for individual nodes in the Network tab of the Node Properties dialog box.

To disable policy autodeployment for all nodes and services

1. In the console tree, right-click Operations Manager , and then click Configure Server . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Policy Management and Deployment . A list of values appears.
3. Set the value for Disable autodeployment for all nodes and services to True .
4. Click Apply , and then click OK to save your changes and close the dialog box.

Related Topics:

- Node group deployment properties
- Configure deployment for service types
- Configure network information for managed nodes

Policy development

Policy development refers to the process of writing and refining the policies, so that they meet the management needs of your organization. Administrators develop policies using an in-depth understanding of the capabilities of the available policy types, and of the management needs of their organization.

Develop policies carefully-- make small changes and additions and deploy the updated policy to only a few test computers. After testing the results, continue to update and check the policy in this manner until you have the results you want.

Related Topics:

- Quick start: how to create a policy

Maintain policies

HPOM provides two vehicles for maintaining and organizing policies , policy types and policy groups .

Policy type

A policy type is a set of configuration information that defines what a policy can manage. Every policy belongs to one policy type. In the console tree, Agent policies grouped by type contains the types of policies that you can deploy to nodes. Server policies grouped by type contains the types of policies that you can deploy to management servers. Note that you can filter the policies that are visible by adjusting the policy view filter .

Policy group

A policy group is a collection of any policies that you create. Any policy can be placed in a policy group, though two versions of the same policy cannot exist in the same group. You can create as many policy groups as you need. Policy groups can contain policies and other policy groups, though two policy groups cannot contain each other. One version of a policy can exist in more than one group.

Policy groups allow you organize your policies, but how you organize them is up to you. You might group policies by the kind of system that they manage (workstation or server), by operating system (Windows or UNIX), or by function (network management or service management).

If you edit a policy by clicking on its icon in a policy group, that group will contain the new version of the policy. Other policy groups may still contain the old version of the policy, although you can assign the latest policy version to a group , or use the update to latest command to quickly update the polices in other policy groups. (The policy type icon will display the new version if the new version has the newest version number.) Policies can be deleted from the system only if they are not currently deployed on any managed node .

NOTE:
You may not be able to edit some policies, because the policy data is not available on the management server . When you synchronize a node, the management server adds the node's policies to the inventory, even if the policy data does not exist on the management server. Policy data may be missing for one of the following reasons:

- Another management server deployed the policy, and there are different policies and policy versions available on that management server.
- The policy exists on a node, but was removed from the management server using the force option.

To modify a policy

1. Find the policy in the console tree under Agent policies grouped by type , Server policies grouped

by type , or Policy groups .

2. Right-click the policy and select All Tasks → Edit...
3. When the policy editor appears, make your changes.
4. When you're done editing the policy, select Save and Close .

Related Topics:

- Change policy version
- Create a new policy
- Delete a policy from the management server
- Edit an old version of a policy
- View installed policies

Edit an old version of a policy

1. Right-click the policy type group.
2. Select Set Filter... Select all versions of the policy below , and select the name of the policy where you want to edit an older version.
3. Select OK . All versions of that policy will appear in the details pane.
4. Right-click the version of the policy that you want to edit.
5. Select All Tasks → Edit .

Remember that when you save this policy, it will have a new version number.

Related Topics:

- Change policy version

Save a policy

Policies are saved from the policy editor . Saving a policy creates a new version ID to distinguish between different versions of the policy. The Policy ID and the policy name remain identical.

- Name : Type a name that will identify the policy. Note that spaces and special characters are not allowed in policy names.
- Version : The version number will automatically increment unless you specifically change it. For more information refer to help topic, Policy versioning .
- Description : Type a description of what the policy does. You might also add other notes, for example data sources that are used, dependent policies, and so on.
- Category :You can use categories to associate policies with instrumentation and with user roles. For more information, see Associate instrumentation with policies and Configure policies for user roles .

To save a policy

- From the File menu of the policy editor, select Save... .

NOTE:
When you click the Save and Close or the Save toolbar button, the policy version number is automatically incremented. Use the Save dialog box to change the version number before saving. (The Save dialog box opens when you click File → Save... .)

Related Topics:

- Save policy as

Save policy as

If you save a policy under a new name, you will have two policies with two versioning schemes. You must choose a name that does not already exist for a policy of that type.

Save a policy under a new name

1. Right-click the policy.
2. Select All Tasks → Edit... to open the policy editor.
3. Click File , and point to Save As...
4. Type a new name for the policy and select OK . Note that spaces and special characters are not allowed in policy names.

Related Topics:

- Save a policy

Change policy version

1. Right-click the policy.
2. Select All Tasks → Edit... to open the policy editor.
3. Click File , and then click Save As...
4. Change the version number.

Related Topics:

- Save policy as

Change a policy description

The properties of a policy include data such as the version ID, policy ID , category and description. Most of these properties are set when the policy is created or changed, but you can change category and description in the Policy Management Properties dialog. This change does not create a new policy version.

To change a policy description

1. In the console tree, open Policy management and click the policy group or policy type that the policy belongs to.
2. In the details pane, right-click the policy, and then click All tasks → Edit .
3. Click File → Properties .
4. Change the Description as appropriate.
5. Click OK .
6. Click File → Exit .

Related Topics:

- Change policy version
- Add categories to a policy
- Save a policy

Add categories to a policy

Categories are logical classifications of policies . You can use categories to associate policies with instrumentation and with user roles:

- You can create corresponding instrumentation categories to ensure that the management server deploys certain instrumentation along with the policy.
- You can configure user roles to control the activities that users can perform on policies that belong to a category.

To add categories to a policy

1. In the details pane, right-click a policy and then click Properties . The policy properties dialog box appears.
2. Type a category name in Category . To add multiple categories, separate each category with a comma.
3. Click OK .

Related Topics:

- Associate instrumentation with policies
- Configure policies for user roles

Delete a policy from the management server

You can delete policies from the server only if they are not installed on any managed node.

To delete a policy from the management server:

1. In the console tree, under Agent policies grouped by type or Server policies grouped by type , right-click the policy you want to delete.
2. Select All Tasks → Delete from server . Note that this menu item will only be available if the policy is not deployed on any managed node.

NOTE:
You can use Ctrl-click to select multiple policies to delete, but if any of the policies are installed on a managed node, the command cannot be carried out.

Related Topics:

- Remove policy from policy group
- View installed packages and subpackages
- Remove policy from node

Mass operations on policies

Some policy operations can be performed on many nodes at once, thus saving you time and preventing errors that can happen when performing repetitive actions. You can move, copy or delete multiple polices, as well as deploy multiple policies.

Related Topics:

- Deploy a policy or policy group
- Add a policy to a policy group

Import OVO for UNIX templates

The tool ImportPolicies imports OVO for UNIX templates into HPOM for Windows.

Usage

```
ImportPolicies /f <policy file>
              [/g <policy group>]
              [/r | /d]
              [/c ASCII|/c UTF8|/c ISO81|/c ISO82|/c ISO85|/c ISO815|/c ROMAN8|
               /c SJIS|/c EUCJP|/c GB2312|/c BIG5|/c EUCTW|/c EUCKR]
```

All imported policies are stored in subdirectories of the policy directory `Policy Management\Policy Groups\Imports From File`.

Policy File (/f)

The policy file contains one or more templates that will be imported as policies into HPOM for Windows. This file is created by the config download command (`opccfgdwn`) on an OVO for UNIX server. For every template contained in the file a new policy will be added.

NOTE:
Refer to the *OVO for UNIX Administrator's Reference* for information about downloading template files.

The table below shows the OVO for UNIX message source types and shows the policy type to which they are converted:

Message Source Type	Policy Type
Logfiles	Logfile Entry or Windows Event Log
SNMP Trap	SNMP Interceptor
Message Interface	Open Message Interface
Threshold Monitor	Measurement Threshold
Scheduled Action	Scheduled Task

Other OVO for UNIX message source types cannot be imported.

Policy Group (/g)

If this optional parameter is given, the policies of the file are stored in the policy group "...\Imports From File\<policy group>". If the given group doesn't exist it will be created.
If the /g option is omitted a new policy group named with the current date and time will be created and used.

Replace (/r) Flag

Use this option if you want to replace any existing policies within <policy group> that have the same name as any template that you are importing.

CAUTION:
The old policy is replaced and cannot be recovered.

If neither the /r nor /d flag is used, policies that have the same name as an existing policy contained in <policy group> will not be imported.

Duplicate (/d) Flag

Use this option if you want to import policies that have the same name as existing polices in the same <policy group> without overwriting them. The policies will have different GUIDs.
If neither the /r nor /d flags is used, policies to be imported that have the same name as a policy contained in <policy group> will not be imported.

Codeset (/c)

This parameter must be set if the policy file contains non-ASCII characters (for example, a download file from an English OVO for UNIX server is encoded in iso8859-1, a download file from a Japanese OVO for UNIX server is encoded in Shift-JIS). In this case the policies will be converted to the multi-byte Unicode encoding UTF8. (This is necessary because within HPOM for Windows all polices are stored in a Unicode encoding.)
If omitted, it is assumed that the given file (<policy file>) only contains ASCII data.
It can accept the following values: ASCII, UTF8, ISO81 (for ISO88591), ISO82 (for ISO88591), ISO85 (for ISO88595), ISO815 (for ISO885915), ROMAN8, SJIS, EUCJP, GB2312, BIG5, EUCTW, EUCKR, and UNICODE (for Windows Unicode).

Convert OVO for UNIX Operations templates to HPOM for Windows policies.

1. Download the templates from the OVO for UNIX management server using the command `opccfgdwn` (see OVO for UNIX documentation) to a config download package.
2. Copy the download package to the HPOM for Windows management server.
3. If any template names contain any special characters, open the config download package and remove them. Special characters are not allowed in policy file names.
4. Upload the templates to the HPOM for Windows management server, where they become policies. Use the command `ImportPolicies` , as described above.
5. Customize the policies as required

Instrumentation

Instrumentation refers to programs that are deployed to a managed node . The programs are scripts or executables that can be used by tools, automatic or operator-initiated commands, or policies. HPOM provides some instrumentation for UNIX nodes, and that some policies and tools will not function until you deploy instrumentation to those nodes.

If you develop policies that require custom instrumentation, you can associate the policies and instrumentation using categories. This ensures that the management server automatically deploys the instrumentation when it deploys the policy. You can also deploy categories of instrumentation independently from policies. In addition, you can redeploy policies and instrumentation that already exist on a node to ensure their integrity.

You store instrumentation in folders on the management server, ready for deployment to nodes. HPOM version 8.10 introduces a different structure for instrumentation folders. If you upgraded from OVO 7.50 you could still have your own custom instrumentation in folders that follow the deprecated structure. You may need to migrate this instrumentation to the new folder structure, particularly if you want to deploy it to nodes that have the HTTPS agent.

Related Topics:

- Associate instrumentation with policies
- Migrate existing instrumentation
- Deploy instrumentation
- Reinstall policies

Associate instrumentation with policies

If you develop policies that require custom instrumentation, you can associate the policies and instrumentation using categories. This ensures that the management server automatically deploys the instrumentation when it deploys the policy.

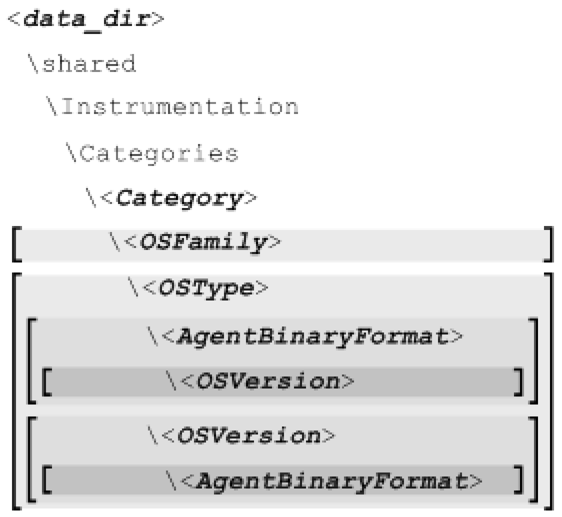
To associate instrumentation with a policy, you must first add a category to the policy. You then copy the instrumentation to a specific folder on the management server. All instrumentation is in the following folder on the management server:

`<`*`data_dir`*`>\shared\Instrumentation`

This folder contains the "Categories" folder, within which you create a new folder with the same name as the category that you added to the policy. You can copy your instrumentation directly into this category folder.

NOTE:
The name of the category folder that you create must have no more that 32 characters.

However, you may also need to deploy specific instrumentation to nodes that run on different platforms. To do this, you use a hierarchy of folders under the category folder. You then copy the platform specific instrumentation into these folders, use the same file name for each instance. The following figure shows the structure of the instrumentation folders. (Brackets show optional levels in the hierarchy.)



The instrumentation folder structure enables you to distinguish between nodes according to the following attributes:

OSFamily

The only supported OS family is UNIX. If you copy instrumentation to the UNIX folder, the management server deploys it to nodes that run any of the following operating systems: AIX, HP-UX, Linux, Solaris, Tru64.

OSType

Example names for OS type folders are AIX, HP-UX, Linux, OpenVMS, Solaris, Tru64, and Windows. You can create OS type folders directly under category folders, or also under an OS family folder.

AgentBinaryFormat

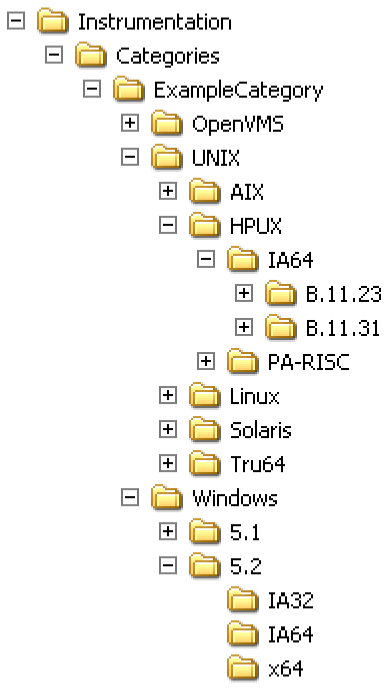
Example names for agent binary format folders are Alpha, IA32, IA64, PA-RISC, PowerPC, Sparc, x64. Within an OS type folder, you can create one folder for each agent binary format. The agent binary format folder can be either directly under the OS type folder or under an OS version folder. For example, you must not create two folders called `Instrumentation\ExampleCategory\Windows\IA64\5.2` and `Instrumentation\ExampleCategory\Windows\5.2\IA64` on the same management server, because `IA64` appears twice under `Windows` .

OSVersion

Supported names for OS version folders correspond to version numbers of operating systems on which the agent is currently supported. For each OS type, you can create only one OS version folder, but it can be either directly under the OS type folder or under an agent binary format folder.

For more information on supported operating systems and agent binary formats, see the support matrix at HP Software Support Online.

The following figure shows an example hierarchy of folders for a category called ExampleCategory.



You can copy instrumentation to any folder in the hierarchy, and you only have to create the folders that you need. A good strategy is to create the most general folders possible. For example, if you can develop platform independent instrumentation, which is compatible with all nodes, you can create just the category folder and copy the instrumentation into it. This means that you only have to maintain one instance of the instrumentation on your management server.

If you have instrumentation files with the same name more than once within a category, the management server always deploys the most platform specific instrumentation. For example, if you have two instances of the same instrumentation in different folders:

- `Instrumentation\Categories\ExampleCategory\Windows\5.2\IA64\ExampleInstrumentation.vbs`
  The management server deploys this instance if the node runs Windows 5.2 (Windows 2003) and has the IA64 architecture.
- `Instrumentation\Categories\ExampleCategory\Windows\ExampleInstrumentation.vbs`
  The management server deploys this instance to any other node than runs a Windows operating system.

NOTE:
Up to OVO version 7.50, the structure for instrumentation folders was different. Therefore, if the management server was upgraded from a previous version, you could have instrumentation folders that follow the deprecated structure. For more information, see Migrate existing instrumentation .

CAUTION:
The management server deploys all instrumentation to the same folder on managed nodes. Ensure that you do not inadvertently use duplicate file names for different instrumentation anywhere in the instrumentation folder hierarchy. (Only use duplicate file names for platform specific instances of the same instrumentation.)

Related Topics:

- Add categories to a policy

Migrate existing instrumentation

You store instrumentation in folders on the management server, ready for deployment to nodes. HPOM 8.00 introduced a different structure for instrumentation folders. However, after you upgrade from OVO 7.50 you could still have your own custom instrumentation in folders that follow the deprecated structure.

NOTE:
If you have not yet upgraded a Smart Plug-in (SPI), this could also have placed instrumentation in folders that follow the deprecated structure. However, when you upgrade the SPI it will automatically migrate its instrumentation to the new folder structure.

In OVO 7.50, the instrumentation folders have the following structure:
<*data_dir* >`\shared\Instrumentation\<OS Name>\<OS Version>\<Category>`

For example:
<*data_dir* >`\shared\Instrumentation\Windows XP\5.1\ExampleCategory`

In HPOM 8.10, the equivalent folder is:
<*data_dir* >`\shared\Instrumentation\Categories\ExampleCategory\Windows\5.1`

OVO 7.50 also creates the following default instrumentation categories:

- <*data_dir* >`\shared\Instrumentation\<OS Name>\<OS Version>\Action`
  may contain instrumentation for automatic- and operator-initiated commands, or scheduled task policies.
- <*data_dir* >`\shared\Instrumentation\<OS Name>\<OS Version>\Command`
  may contain scripts or programs used by tools.
- <*data_dir* >`\shared\Instrumentation\<OS Name>\<OS Version>\Monitor`
  may contain monitoring scripts or programs used by measurement threshold or log file policies.

In the HPOM 8.10 instrumentation folder structure, there is no equivalent of the Action, Command, and Monitor folders. You must create alternative categories.

HPOM 8.10 still supports the deprecated instrumentation folder structure for nodes that have a DCE agent:

- If you deploy a policy with a category that exists in only the OVO 7.50 instrumentation folder structure, the management server deploys the instrumentation from this folder to nodes that have the DCE agent.
- If you deploy a policy with a category that exists in both the OVO 7.50 and HPOM 8.10 instrumentation folder structures, the management server deploys the instrumentation from the OVO 7.50 folders to nodes that have DCE agents.
- If you deploy a policy with a category that exists in only the HPOM 8.10 instrumentation folder structure, the management server deploys the instrumentation from this folder to all nodes.

If you have your own custom instrumentation in the deprecated folder structure, consider migrating it to the new folder structure. This is essential if you want to deploy the instrumentation to nodes that have an HTTPS agent.

To migrate existing instrumentation, create folders according to new folder structure, and move the instrumentation files into them. For more information on the new instrumentation folder structure, see Associate instrumentation with policies .

Related Topics:

- Configuring Agents

Deploy instrumentation

You can associate instrumentation with policies using categories. This ensures that the management server automatically deploys the instrumentation when it deploys the policy. You can also explicitly deploy categories of instrumentation using the console.

The instrumentation is deployed to the following directories on the various agent platforms:

Platform	Agent	Instrumentation directory
Windows	HTTPS	*<data_dir>* `\bin\instrumentation`
Windows	DCE	*<install_dir>* `\Installed Packages\{790C06B4-844E-11D2-972B-080009EF8C2A}\bin\Instrumentation`
HP-UX	HTTPS/DCE	`/var/opt/OV/bin/instrumentation`
Solaris	HTTPS/DCE	`/var/opt/OV/bin/instrumentation`
AIX	HTTPS	`/var/opt/OV/bin/instrumentation`
Linux	HTTPS/DCE	`/var/opt/OV/bin/instrumentation`
Tru 64	HTTPS	`/usr/var/opt/OV/bin/instrumentation`
Tru 64	DCE	`/var/opt/OV/bin/instrumentation`

If you deploy the default instrumentation categories (Agent, Command, and Monitor) to DCE agents, the instrumentation is deployed to the following directories:

Platform	Instrumentation directory
Windows	*<install_dir>* `\Installed Packages\{790C06B4-844E-11D2-972B-080009EF8C2A}\bin\OpC\vpwin\<actions|cmds|monitor>`
HP-UX	`/var/opt/OV/bin/OpC/vpwin/<actions|cmds|monitor>`
Solaris	`/var/opt/OV/bin/OpC/vpwin/<actions|cmds|monitor>`
Linux	`/var/opt/OV/bin/OpC/vpwin/<actions|cmds|monitor>`
Tru 64	`/var/opt/OV/bin/instrumentation`

NOTE:
You can only deploy instrumentation to nodes on which the agent is installed. If the agent is not installed, running this command will deploy it.

To deploy instrumentation

1. Right-click the node or node group in the console tree.
2. Select All Tasks → Deploy Instrumentation .
3. In the Deploy Instrumentation dialog box, select the category of instrumentation that you want to deploy. Press CTRL-SHIFT to select multiple directories.

NOTE:
HPOM version 8.10 uses an new structure for instrumentation folders on the management server. However, after you upgrade from OVO 7.50 you could still have your own custom instrumentation in folders that follow the deprecated structure.

If any of the nodes that you select have the DCE agent, the Deploy Instrumentation dialog box enables you to select instrumentation categories from both folder structures. However, the management server does not deploy instrumentation from categories in the OVO 7.50 instrumentation folders to nodes that have an HTTPS agents. To deploy instrumentation to a node that has an HTTPS agent, the instrumentation must exist in an HPOM 8.10 instrumentation folder.

4. If you want to remove all instrumentation from the node before deploying the new instrumentation, select Remove all existing instrumentation before deploying new instrumentation .

CAUTION:
If you select any category other than a default category (Agent, Command, or Monitor), all instrumentation from all categories is removed from the node. This includes any categories of instrumentation that are associated with policies. Therefore, errors may occur if you remove instrumentation that a policy requires.

5. Click OK. This creates new deployment jobs, which deploy the instrumentation to the nodes.

NOTE:
If you select a single node, the Deploy Instrumentation dialog box shows only the categories that contain instrumentation for the node's platform.

If you select a node group that contains less than 20 nodes, the dialog box shows all the categories that contain instrumentation for any of the nodes' platforms.

If you select a node group that contains more than 20 nodes, the dialog box shows all categories, without checking whether the categories contain instrumentation for the nodes' platforms.

In all cases, the management server checks a node's platform before it deploys instrumentation. If a category that you select does not contain instrumentation for a node's platform, the management server does not instrumentation from that category to the node.

Related Topics:

- Associate instrumentation with policies
- Migrate existing instrumentation
- Add categories to a policy
- Deployment jobs
- Deploy a policy or policy group

Instrumentation with multiple management servers

You can configure an environment in which multiple management servers deploy policies to nodes that have the HTTPS agent. Every policy on a node is owned by the management server that deployed it. The same ownership concept does not exist for instrumentation.

Management servers all deploy instrumentation to the same folder on managed nodes. You must ensure that one management server does not make inappropriate changes to instrumentation that another management server deployed.

If you need to deploy the same instrumentation from multiple management servers, ensure that all management servers have the same version of the instrumentation. Otherwise, it is possible for one management server to overwrite instrumentation that already exists on a node. For example:

- If you deploy a policy that belongs to a category, the management server deploys the associated instrumentation for that category. If another management server deploys a different policy that belongs to the same category, this management server overwrites the existing instrumentation.
- If you deploy instrumentation explicitly using the console, this can overwrite instrumentation that another management server deployed.

You must also ensure that one management server does not remove instrumentation that is required by another management server's policies. For example :

- If you deploy instrumentation explicitly using the console and set the option to remove all existing instrumentation first, you remove instrumentation that other management servers deployed.
- If you redeploy all policies to a node without setting the ignore owner option, you could remove or replace instrumentation that other management servers deployed.

Related Topics:

- Scalable Architecture for Multiple Management Servers

Deployment packages

A deployment package usually contains agents, programs, or instrumentation that are ready for installation on nodes . A package can contain individual files and also subpackages, which are collections of files.

Deployment packages are provided by HPOM and Smart Plug-ins to enable you to manage various services and nodes. In some cases the management server automatically deploys packages when you discover nodes and deploy policies to them. You can also explicitly deploy and remove packages using the console.

For each node, the management server maintains a package inventory, which enables you to see which packages exist on each node. This helps you to decide when you need to redeploy or remove packages.

Related Topics:

- Deploy deployment package
- View details of available packages
- View installed packages and subpackages
- Synchronize packages
- Reinstall packages
- Reinstall all
- Update packages
- Update all
- Remove package from node
- Uninstall all

Deploy deployment package

A deployment package is a set of files, usually containing agents, programs, or instrumentation, that is ready for deployment .

To deploy a deployment package

1. Click Deployment packages in the console tree.
2. Select the packages you want to deploy. (For more details about deploying an agent package, see Install agents remotely .)
3. Right-click the selected packages and select All Tasks →Deploy on.. .
4. Select the managed nodes to which you want to deploy the packages.
5. Click OK .

You can also install deployment packages by dragging them onto a node or node group. You can watch the progress of the deployment by clicking Deployment jobs in the console tree. Only pending, active, failed, or suspended deployment jobs appear.

NOTE:
You can also install deployment packages using a drag-and-drop operation.

Related Topics:

- View installed packages and subpackages

View details of available packages

When you click Deployment packages in the console tree, the details pane shows a list of the packages that are available on the management server . If you need more information about the packages, you can view the deployment packages report.

To view details of available packages

1. In the console tree, right-click Deployment packages , and then click View → Package details . The Deployment Packages report appears.
2. *Optional.* In the report, select View including subpackages or View without subpackages . The report shows or hides details of subpackages accordingly.

Related Topics:

- View installed packages and subpackages

View installed packages and subpackages

You can see which packages and subpackages are installed on any node . Each package can contain individual files and also subpackages, which are collections of files. Packages and subpackages both have version numbers. A package version number corresponds to a major release of the software. Subpackage version numbers correspond to minor releases and patches. Therefore, if you want to check whether patches are installed on a node, check the version numbers in the subpackage inventory.

NOTE:
The subpackage inventory is only available for nodes that have the HTTPS agent.

To view installed packages

1. In the console tree, right-click the node where you want to check the installed packages.
2. Click View → Package Inventory . A list of installed packages appears in the details pane.

To view installed subpackages

1. In the console tree, right-click the node where you want to check the installed subpackages.
2. Click View → Subpackage Inventory . A list of installed subpackages appears in the details pane.

Related Topics:

- View installed policies

Synchronize packages

The procedure below will update the inventory of all packages deployed to managed nodes. The inventory is kept on the management server . Reinstall all if you want to reinstall the policies and packages on the node.

To update the package inventory on the management server

1. In the console tree, right-click the node or node group where you want to query the installed packages.
2. Select All Tasks → Synchronize inventory → Packages .

Related Topics:

- Deploy deployment package

Reinstall packages

You can reinstall all the packages that are currently deployed to selected nodes and node groups, including the agent package. This first uninstalls the packages before installing them again. However, you cannot reinstall the DCE agent package on UNIX and Linux nodes from the console. The agent must be removed and reinstalled manually.

To reinstall packages

1. In the console tree, select the nodes or node groups on which you want to reinstall packages.
2. Right-click your selection, and then click All Tasks → Reinstall/Update... . The Reinstall/Update Node dialog box opens. The reinstall option is already selected.
3. In the list under Scope , select Packages .
4. *Optional.* Select the Ignore missing policies/packages check box. This enables you to reinstall the packages that are available on this management server. If the node inventory contains packages that are not available on this management server, the management server creates a warning in the event log and the job succeeds.
5. Click OK . The management server creates a deployment job, which reinstalls the packages.

Related Topics:

- Deployment jobs

Reinstall all

You can reinstall all the packages and policies that are currently deployed to selected nodes and node groups. This overwrites the existing policies on the node. When using this command, note the following points:

- This command does not upgrade any policies that are installed on the node. Even if more current versions of a policy exist, the installed versions are redeployed according to the inventory. Refer to Update policy on node to deploy the most current policies.
- On UNIX and Linux nodes that have the DCE agent, this command only removes and redeploys policies—the agent must be removed and reinstalled manually. If the node does not host the newest compatible agent version, you will receive an error message to inform you. On other nodes, this command always deploys the newest agent version that is available on the management server .
- The default instrumentation categories for the DCE agent (action, command, and monitor) are not usually associated with policies and, as a result, are not removed or redeployed. If instrumentation still exists in the action, command, and monitor categories, you can redeploy it to nodes that have DCE agents using the Deploy instrumentation command.
- If a policy in the node's inventory has associated instrumentation, this command removes all instrumentation from the node. This includes any instrumentation that you deployed to the node manually. (It does not include instrumentation that belongs to the action, command, and monitor categories on the DCE agent.) The command redeploys only the instrumentation that is associated with the policies in the node's policy inventory.
- If none of the policies in the node's inventory have associated instrumentation, then no instrumentation is removed or redeployed.

To reinstall all policies and packages

1. In the console tree, select the nodes or node groups on which you want to reinstall all policies and packages.
2. Right-click your selection, and then click All Tasks → Reinstall/Update... . The Reinstall/Update Node dialog box opens. The reinstall option is already selected.
3. In the list under Scope , select All .
4. *Optional.* Select the Ignore missing policies/packages check box. If the node inventory contains policies and packages that are not available on this management server, the management server creates a warning in the event log and the job succeeds. Otherwise, if you clear this check box, the job succeeds only if all the policies and packages in the node inventory are available on this management server.
5. *Optional.* If another management server has already deployed a version of a policy to a node, that

management server owns the policy on that node. If you want to reinstall the policy from this management server, and transfer the ownership, select the Ignore policy owner check box. To do this, you must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node. If you do not select this check box, the management server reinstalls only the policies that it owns.

6. Click OK . The management server creates a deployment job, which reinstalls the policies, packages, and instrumentation.

You can follow the progress of the redeployment by clicking Deployment jobs in the console tree.

Related Topics:

- Instrumentation
- View job
- Deploy a policy or policy group

Update packages

You can update all the packages that are currently deployed to selected nodes and node groups, including the agent package. This upgrades the packages without removing the current version.

To update packages

1. In the console tree, select the nodes or node groups on which you want to update packages.
2. Right-click your selection, and then click All Tasks → Reinstall/Update... . The Reinstall/Update Node dialog box opens.
3. Select Update .
4. In the list under Scope , select Packages .
5. *Optional.* Select the Ignore missing policies/packages check box. If the node inventory contains packages that are not available on this management server, the management server removes them and does not redeploy them. Otherwise, if you clear this check box, the job succeeds only if all the packages in the node inventory are available on this management server.
6. *Optional.* If you want to deploy packages to nodes that already have a newer version of the package, clear the Update only if version is newer check box. If you do this, the management server deploys the latest version of the package that is available on the management server, even if the node already has the same version or a newer version.
7. Click OK . The management server creates a deployment job, which updates the packages.

Related Topics:

- Deployment jobs
- Migrate from DCE to HTTPS

Update all

You can update all the policies and packages that are currently deployed to selected nodes and node groups, including the agent package. This upgrades the packages without removing the current version and updates all the policies on the node to the lastest version.

To update all

1. In the console tree, select the nodes or node groups on which you want to update policies and packages.
2. Right-click your selection, and then click All Tasks → Reinstall/Update... . The Reinstall/Update Node dialog box opens.
3. Select Update .
4. In the list under Scope , select All .
5. *Optional.* Select the Ignore missing policies/packages check box. If the node inventory contains policies and packages that are not available on this management server, the management server updates only the policies and packages that are available. Otherwise, if you clear this check box, the job succeeds only if all the policies and packages in the node inventory are available on this management server.
6. *Optional.* If another management server has already deployed a version of a policy to a node, that management server owns the policy on that node. If you want to update the policy from this management server, and transfer the ownership, select the Ignore policy owner check box. You must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node. If you do not select this check box, the management server updates only the policies that it owns.
7. *Optional.* If you want to deploy policies and packages to nodes that already have a newer version of the policy or package, clear the Update only if version is newer check box. If you do this, the management server deploys the latest version of the policy or package that is available on the management server, even if the node already has the same version or a newer version.
8. Click OK . The management server creates a deployment job, which updates the policies and packages.

Related Topics:

- Deployment jobs
- Migrate from DCE to HTTPS

Remove package from node

If you have previously deployed packages to nodes , but no longer require them, you can remove the packages from the nodes using the console. You can only remove an agent package from a node after you remove any other packages on the node.

To remove packages from nodes:

1. Open Policy management→Deployment packages in the console tree and select the packages that you want to remove. You can select multiple packages by pressing SHIFT or CTRL .
2. Right-click your selection, and then click All Tasks →Uninstall from ... . The Uninstall package from... dialog box opens.
3. Select the nodes that you want to remove the packages from by selecting check boxes for the nodes and node groups in Managed nodes .
4. *Optional.* If you know that the job to remove the package will fail (for example, because the node is unreachable), but you still want to remove the package from the inventory on the management server , select the Force removal of package check box.
5. Click OK .
6. If you are removing an HTTPS agent package, the Node Credentials dialog box opens. Specify the credentials that the management server uses to uninstall the package. Click one of the following options:
   - PMAD user . The management server attempts to uninstall the agent as the user under which the policy management and deployment (PMAD) service runs (called HP-OVE-Deleg-User by default).

     You can only use this for nodes with a Windows operating system. The nodes can belong to the same domain as the management server, a trusted domain, or a workgroup.

     NOTE:
     The PMAD user does not by default have administrative access to the node. For more information, see Start Windows node security setup .

   - Impersonate user . The management server attempts to uninstall the agent using the credentials that you are currently logged in to Windows with.

     You cannot use impersonation for nodes in untrusted domains or workgroups.

     You can use impersonation only if the PMAD user is trusted for delegation in Active Directory, unless your console runs directly on the management server. For more details on delegation, refer to the Active Directory documentation that Microsoft provides.

- User/Password. Type the Username of a user who has permission to uninstall software on the node and their Password.

  For nodes with a UNIX or Linux operating system, this is the only command available. You can also use this command to uninstall the HTTPS agent from Windows nodes that belong to the same domain as the management server, a trusted domain, or a workgroup.

  Click OK. The management server uses the same credentials for each node.

7. The management server creates deployment jobs, which remove the packages. If a job fails, you can restart it with advanced options to change the credentials.

TIP:

You can also remove a package from a node in the package inventory view:

1. In the console tree, right-click the node from which you want to remove a package, and then click View → Package Inventory.
2. In the details pane, right-click the package that you want to remove, and then click All Tasks → Remove from node... . The Remove Package dialog box opens.
3. If you are removing an HTTPS agent package, specify the credentials that the management server uses to uninstall the package.
4. *Optional.* Select the Force removal of package(s) check box.

Related Topics:

- Deployment jobs
- Remove policy from node
- Restart job with new options

Uninstall all

If you have previously installed policies, instrumentation, and packages to nodes, you can remove them all using the console. This includes uninstalling the agent. You must specify the credentials credentials of a user who has administrative access to the node.

To uninstall all

1. In the console tree, right-click the node or node group, and then click Uninstall all... . The Uninstall All dialog box opens.
2. If you are removing an HTTPS agent package, specify the credentials that the management server uses to uninstall the agent package. Click one of the following options:
   - PMAD user . The management server attempts to uninstall the agent as the user under which the policy management and deployment (PMAD) service runs (called HP-OVE-Deleg-User by default).

     You can only use this for nodes with a Windows operating system. The nodes can belong to the same domain as the management server, a trusted domain, or a workgroup.

     NOTE:
     The PMAD user does not by default have administrative access to the node. For more information, see Start Windows node security setup .
   - Impersonate user . The management server attempts to uninstall the agent using the credentials that you are currently logged in to Windows with.

     You cannot use impersonation for nodes in untrusted domains or workgroups.

     You can use impersonation only if the PMAD user is trusted for delegation in Active Directory, unless your console runs directly on the management server. For more details on delegation, refer to the Active Directory documentation that Microsoft provides.
   - User / Password . Type the Username of a user who has permission to uninstall software on the node and their Password .

     For nodes with a UNIX or Linux operating system, this is the only command available . You can also use this command to uninstall the HTTPS agent from Windows nodes that belong to the same domain as the management server, a trusted domain, or a workgroup.
3. *Optional.* If you know that the uninstallation job will fail (for example, because the node is unreachable), but you still want to remove the policies and packages from the inventory on the management server , select the Force removal of packages and policies check box.
4. *Optional.* If another management server owns a policy on a node, this management server will not

uninstall it by default. If you want to this management server to remove the policies that other management servers own, select the Ignore Owner check box. To do this, you must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node.

5. Click OK . The management server creates a remove all job for each of the nodes that you selected. To view the progress of these jobs, click Deployment jobs in the console tree. If an agent installation job fails, right-click the job, and then click Error description .

Related Topics:

- Deployment jobs
- Remove package from node

Deployment jobs

Deployment refers to the process of transferring agents, policies , and other software from the management server to one or more managed node .

Whenever you deploy a package, a policy or policy group to (or delete a policy from) a managed node, a deployment job is created. You can view all pending jobs by clicking on the job icon in the console tree. When a job has completed, it is deleted from the details pane.

NOTE:
You must have the appropriate user rights to perform these operations. Refer to Configure policies for user roles for more information.

The deployment jobs help you to monitor the actions you have requested. If a job remains in the details pane longer than you expect, you should investigate the cause of the delay. If some problem has prevented the execution of the deployment job, it will be displayed in the details pane with an *error* status. Jobs that are waiting show the status suspended . After correcting the cause of the problem, you can restart the job by right-clicking on the node and selecting All Tasks → Restart all deployment jobs .

You can also restart a failed job without performing a prerequisite check on the target node. This feature allows you to force the deployment of components to managed nodes, even if one or more of the target nodes does not meet the standard requirements for a managed node.

If you create a job that deploys agent packages, you must provide the credentials of a user who has access to the node. Also, if you restart a failed or suspended job that deploys agent packages, you must provide the credentials again. You may also need to provide credentials to deploy other packages to nodes that have the DCE agent.

Related Topics:

- Deploy a policy or policy group
- Deploy deployment package
- Deploy instrumentation

View job

To view unfinished jobs:

Select the Deployment jobs icon in the console tree. The details pane displays all unfinished jobs. When a job completes, it is no longer visible.

Related Topics:

- Restart job
- Cancel job

Suspend job

You can suspend all pending deployment jobs.

To suspend a specific job.

1. Click the Deployment jobs icon in the console tree.
2. Right-click the job you want to suspend.
3. Select Suspend job .

To suspend all jobs for all managed nodes.

1. Right-click the Deployment jobs icon in the console tree.
2. Select All Tasks → Suspend all jobs .

Related Topics:

- Restart job
- Cancel job

Restart job

If a deployment job is not successfully carried out due to a network problem, you can start the job again.

NOTE:
If the job deploys an agent, the management server attempts to deploy the agent using the credentials that you are currently logged in with. You must have permission to install software on the node. If the node has a UNIX or Linux operating system, or you want to use different credentials, restart the job with new options instead.

To restart a specific job:

1. Click the **Deployment Jobs** icon in the console tree.
2. Right-click the job you want to restart.
3. Click **Restart job** .

To restart all pending jobs for one managed node:

1. Right-click the managed node icon in the console tree.
2. Click **All Tasks** → **Restart all deployment jobs**

To restart all jobs for all managed nodes:

1. Right-click the **Deployment Jobs** icon in the console tree
2. Click **All Tasks** → **Restart all jobs** .

Related Topics:

- Restart job with new options
- Cancel job
- Suspend job

Restart job with new options

If a deployment job fails, you can try to correct the error by restarting the job with new options. The options available vary, depending on the type of job you are restarting.

To restart a deployment job with new options:

1. In the console tree, click **Deployment jobs** . A list of deployment jobs appears.
2. Right-click the job you want to restart, and then click All Tasks → **Restart job with new options** . The Advanced deployment options dialog appears.
3. *Optional* . If the job installs or uninstalls packages, you can specify the credentials that the management server uses to access the node. This is necessary if the original credentials were invalid. It is also necessary if the job fails for another reason, and you do not want to restart it using impersonation. Click one of the following options:
   - **PMAD user** . The management server attempts to access the node as the user under which the policy management and deployment (PMAD) service runs (called HP-OVE-Deleg-User by default).

     You can only use this for nodes with a Windows operating system. The nodes can belong to the same domain as the management server, a trusted domain, or a workgroup.

     NOTE:
     The PMAD user does not by default have administrative access to the node. For more information, see Start Windows node security setup .

   - **Impersonate user** . The management server attempts to access the node using the credentials that you are currently logged in to Windows with.

     You cannot use impersonation for nodes in untrusted domains or workgroups.

     You can use impersonation only if the PMAD user is trusted for delegation in Active Directory, unless your console runs directly on the management server. For more details on delegation, refer to the Active Directory documentation that Microsoft provides.

   - **User/Password** . Type the **Username** of a user who has permission to install software on the node and their **Password** .

     For nodes with a UNIX or Linux operating system, this is the only command available . You can also use this command to install the HTTPS agent to Windows nodes that belong to the same domain as the management server, a trusted domain, or a workgroup.
4. *Optional* . Select the **Skip node prerequisites check** check box. By default, the management server checks that each node meets the requirements for the deployment and installation of the agent. This

check is designed to reduce deployment errors and assist administrators in the performance of regular maintenance tasks.

You can restart a failed job without performing a prerequisite check on the managed node. This enables you to deploy the agent, even if the node does not meet the standard requirements, or a known problem is likely to prevent a deployment from completing successfully.

CAUTION:
If the node prerequisite check fails, it usually indicates that there is a problem with the managed node, and it is not recommended to continue with the deployment of the agent. Only use this option if you know exactly why the prerequisites check failed on the managed node and are sure that it is safe to install the agent anyway.

5. *Optional.* Select the Allow older packages check box. This check box enables you to deploy the latest available version of a package if the version on the node is newer than the version available on a management server.
6. *Optional.* Select the Update to latest policy/package version check box. This check box enables you to deploy the latest available versions of policies and packages instead of the redeploying the versions currently on the node.
7. *Optional.* Select the Ignore errors check box. This check box enables you to force a job that removes policies or packages to complete, even if errors occur. For example, if you try to remove a package from a node, but the node is unreachable, the management server removes the package from the node's package inventory anyway.
8. *Optional.* Select the Ignore package/policy version check box. This check box enables you to deploy a policy or a package even if a newer version already exists on the node.
9. *Optional.* Select the Enable policy check box. This enables the policy after the management server deploys it to the node.
10. *Optional.* Select the Disable policy check box. This disables the policy after the management server deploys it to the node.
11. *Optional.* Select the Ignore owner of policies on the node check box. If another management server has already deployed a version of a policy to a node, that management server owns the policy on that node. This check box enables you to deploy the policy from this management server, and transfer the ownership. To do this, you must have the user right to ignore policy ownership. The management server that you are connected to must be a primary or secondary manager of the node.
12. Click OK . The management server restarts the job with the new options.

Related Topics:

- Check prerequisites for managed nodes
- Restart job
- Cancel job
- Suspend job

Cancel job

You can cancel any pending deployment job.

To cancel a specific job.

1. Click the Deployment jobs icon in the console tree
2. Right-click the job you want to cancel.
3. Select Cancel Job .

To cancel all pending jobs for one managed node.

1. Right-click the managed node icon in the console tree
2. Select All Tasks → Cancel all deployment jobs

To cancel all jobs for all managed nodes.

1. Right-click the Deployment Jobs icon in the console tree
2. Select All Tasks → Cancel all jobs .

Related Topics:

- Suspend job
- Restart job

Change maximum parallel jobs

By default, the management server starts a maximum of five parallel deployment jobs. In some cases, you may want to increase or reduce this number:

- If the management server is too weak (for example, slow processor, not enough RAM), then too many parallel jobs can produce an unacceptable load on the server. In this case, reduce the maximum number of jobs.
- If the management server is very powerful and you want to reduce the time required for mass operations such as policy deployment, then increase the maximum number of jobs.

To change maximum parallel jobs

1. In the console tree, right-click Operations Manager , and then click Configure→ Server... . The Server Configuration dialog box appears.
2. Click Namespaces , and then click Policy Management and Deployment . A list of values appears.
3. Change the value of Number of concurrent deployment threads . The value should be between 5 and 10 in most cases.

Related Topics:

- Change server configuration values

Scalable Architecture for Multiple Management Servers

The HP Operations management server is available in two versions: a Windows version and a UNIX version (running on HP-UX or Solaris). Both versions of management servers can work together to manage the same nodes in your environment. The key features of interoperability are described below. For further details, see the Related Topics references.

Agent-based flexible management

Agent-based flexible management allows you to configure managed nodes to send messages to different management servers, based on time and message attributes. This is not about forwarding messages from one management server to another, but about specifying which messages from a managed node should be sent to a specified management server. Additional configuration provided by agent-based flexible management includes specifying which management server is allowed to execute actions on this managed node and which management server can become the primary management server of this managed node.

NOTE:
If your HPOM for Windows installation is based on the managed node limited management server license, the setup of integrated, multiple management servers is not supported.

Server-based flexible management

Server-based flexible management uses the same message forwarding and synchronizing techniques used in HPOM for UNIX. It allows forwarding messages directly from one management server to other management servers, including HPOM for UNIX management servers. Forwarded messages are kept synchronized between the management servers; with a correct setup a message is exactly in the same state on each management server to which it was forwarded. HPOM for Windows forwards messages, based on message attributes, to one or more management servers based on rules that you set up.

Configuration data exchange between management servers

HP Operations management servers allow exchanging configuration information between management servers. This is useful if you want to centrally develop policy and other configuration information and then deploy this configuration to multiple management servers.

Configuration synchronization is very helpful for forwarding and synchronizing messages between management servers. You can easily synchronize node configuration and instruction text configuration between the forwarding management servers, to set up a working message forwarding environment.

Related Topics:
- Agent-based flexible management
- Server-based flexible management
- Exchange configuration data between management servers

Scalability Scenarios

The scalable architecture of HPOM enables one or more management systems to be combined into a single, powerful management solution that meets the requirements of your organizational structure.

Agent-based flexible management enables you to configure managed nodes to send messages to management servers other than the primary management server. You can configure nodes to redirect messages based on the message attributes and the time.

Server-based flexible management enables you to configure management servers to forward messages to other management servers. Management servers can also forward message operations, so that the status of each forwarded message is up-to-date on all management servers.

You can combine agent-based and server-based flexible management to suit your organization's requirements. The related topics give examples of scalability scenarios.

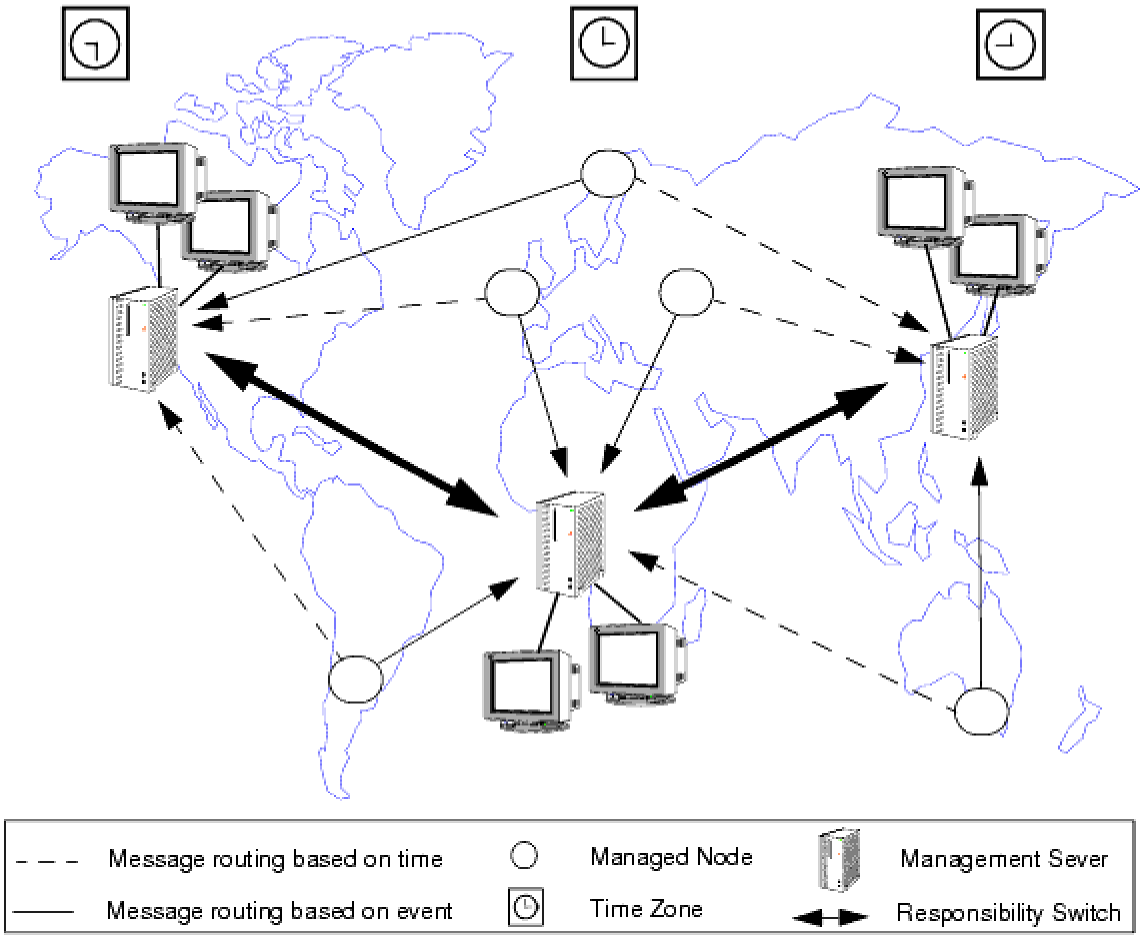
Related Topics:

- Follow-the-sun control
- Competence centers

Follow-the-sun control

If your distributed operations take place over several time zones (see the figure below), you can use HP Operations Manager for Windows to rotate management responsibilities by implementing follow-the-sun control. Depending on the time of day, managed nodes report to different management servers. The same capability enables you to set up specific management servers for weekend or holiday operations.

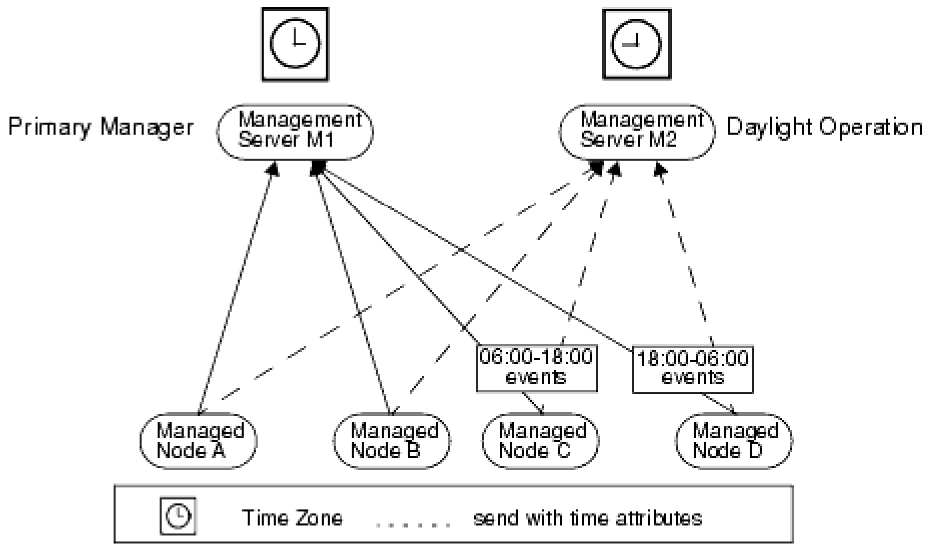
Worldwide Management Domain



The follow-the-sun concept is based on the idea of sending messages to different management servers, according to predefined time attributes. HP Operations Manager for Windows enables you to configure managed nodes to send messages to different management servers according to rules defined in a time template .

For example, the figure below shows how you can configure an agent so that all messages generated between 06:00 and 18:00 are sent to management server M1 from managed nodes C and D. Messages generated between 18:00 and 06:00 are sent to management server M2. With follow-the-sun functionality, you can control your entire environment throughout the day by assigning daylight operating shifts to the corresponding regional areas.

Using Time or Message Attributes to Forward Messages



For example, if your enterprise has implemented a 24-hour support desk at a central location, you can send messages from the regional nodes directly to the central management server during regional department off-hours. Implementing follow-the-sun policies requires the addition of two entries in the agent-based flexible management policy.

These two entries might take the following form:

```
CONDITION TIME 6am-6pm SEND TO $OPC_PRIMARY_MGR
CONDITION TIME 6pm-6am SEND TO Central Management Server
```

The follow-the-sun concept is not restricted to rules based on the time of day. You can also configure the sending of messages to different management servers based on the day of the week, a specific date or dates, or frequency. For details, see Time templates .

Related Topics:

- Syntax for agent-based flexible management policies

Competence centers

If you operate in a large enterprise with multiple management servers distributed over a wide area, specialist knowledge relating to a specific subject is not always available locally. For example, your organization might have a center responsible for all operating system-related problems. In addition, another center of expertise may be responsible for a database, which is used company-wide.

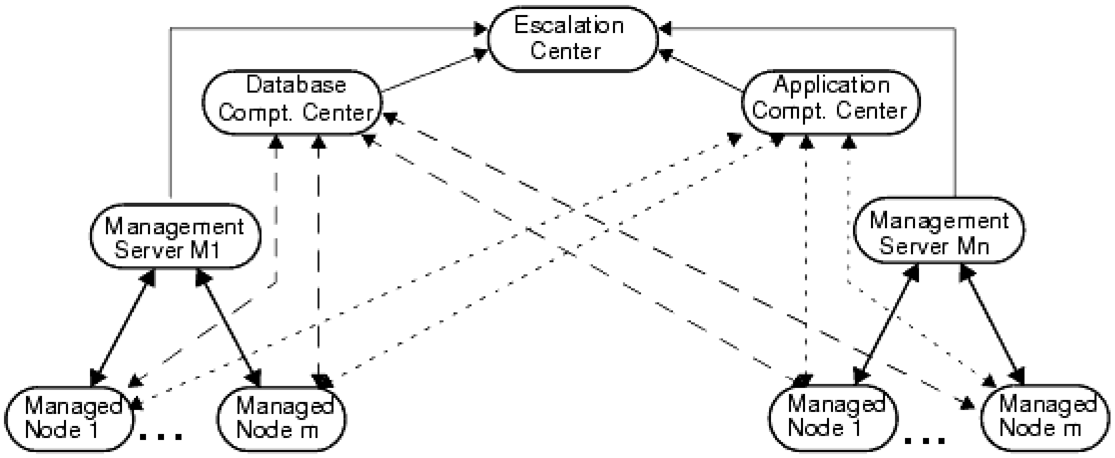
A competence center hierarchy distributes responsibility for managed nodes. Regional management servers are not solely responsible for managed nodes. Instead, messages about specific subjects go to a competence center management server, where expertise exists to solve similar problems for all managed nodes.

You can configure competence centers using agent-based or server-based flexible management, or both.

Competence centers with agent-based flexible management

Agent-based flexible management enables you to configure managed nodes to communicate directly with servers other than the primary management server. You can configure your managed nodes to communicate with the management servers of your choice anywhere in your network.

In the figure below all managed nodes send all database messages to the database competence center management server.

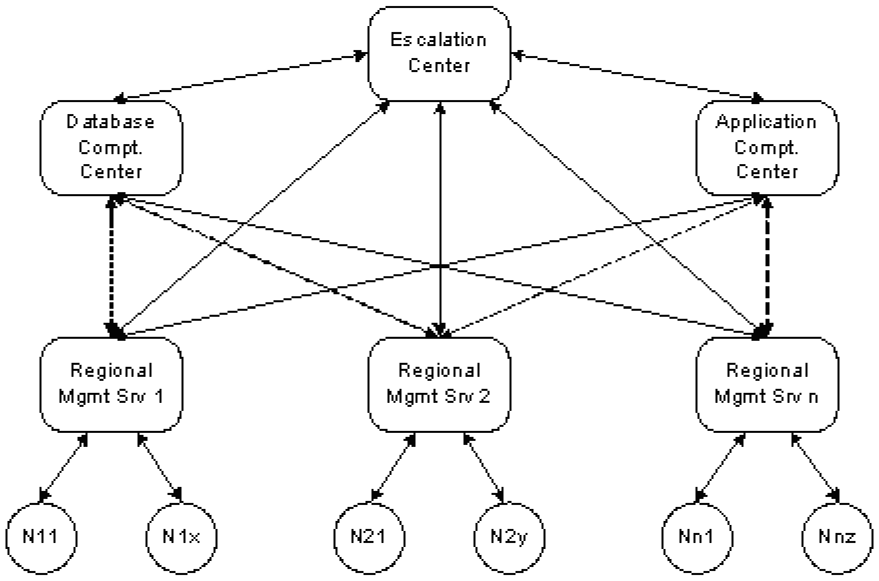


The advantage of configuring competence centers with agent-based flexible management is that you can also include follow-the-sun capabilities by adding time conditions to the relevant competence center conditions.

Competence centers with server-based flexible management

Server-based flexible management enables you to configure regional management servers to forward some messages to other servers in your network. You can configure your regional management servers to forward messages to the management servers of your choice anywhere in your network, based on message attributes such as originating node or application.

In the figure below all regional management servers forward all database messages to the database competence center management server.



Message operations (for example acknowledge, own, severity change) are synchronized between regional management servers, competence center management servers, and the central management server. This way the message states are always kept synchronized across the whole enterprise environment.

The advantages of configuring competence centers with server-based flexible management are:

- You do not need to configure the nodes, because the target for the messages is decided on the regional management servers.
- The messages forwarded to competence center management servers are also visible on (and synchronized with) the regional management servers, so you have the full context of messages on the regional management servers.

Related Topics:

- Agent-based flexible management
- Server-based flexible management

Exchange configuration data between management servers

In an environment with multiple management servers, you must configure each node on every management server that may receive the node's messages. Management servers discard messages if they originate from unknown nodes. This applies for both agent-based and server-based flexible management.

HPOM provides commands that enable you to exchange node configuration data between management servers. In addition, you can also optionally exchange the following types of configuration data:

- Policies
- Node inventories
- Instruction text
- User roles
- Tools
- Services
- Instrumentation

NOTE:
There are some restrictions on the data that you can exchange between HPOM for Windows and HPOM for UNIX. For more details, see Configuration interoperability .

To exchange configuration data, you use command line tools to first export the data to a file on one management server. You then copy the file to other management servers and import the data. There are different command line tools for working with HPOM for Windows and HPOM for UNIX management servers. The table below summarizes which command line tools you must use.

		Windows to Windows	Windows to UNIX	UNIX to Windows
Nodes	Export	`ovpmutil`	`ovowconfigexchange`	`opccfgdwn`
	Import	`ovpmutil`	`opccfgupld`	`ovowconfigexchange`
Policies	Export	`ovpmutil`	`ovpmutil`	`opccfgdwn`
	Import	`ovpmutil`	`opctempl`	`ImportPolicies`
Node inventories	Export	`ovpmutil`	Not supported	Not supported
	Import	`ovpmutil`		
Instruction text	Export	Only possible by exchanging policies	`ovowconfigexchange`	Only possible by exchanging policies
	Import		`opctempl`	
User roles	Export	`ovpmutil`	Not supported	Not supported
	Import	`ovpmutil`		
Tools	Export	`ovpmutil`	Not supported	Not supported
	Import	`ovpmutil`		
Services	Export	`ovpmutil`	`ovpmutil`	Not supported
	Import	`ovpmutil`	`opccfgupld`	
Instrumentation	Export	Copy from file system	Copy from file system	Copy from file system
	Import			

The command-line tools provide flexible options for data exchange. Therefore, you should plan how to create, maintain, and distribute configuration data between multiple management servers. For example, you could consider the following scenarios:

- Exchange all data so that every management server has the same configuration.
- Exchange subsets of data so that each management server has a specialized configuration.
- Perform all configuration tasks centrally on one management server, and then distribute the data to other management servers.
- Perform configuration tasks on all management servers, and exchange the data as necessary.

Related Topics:

- ovpmutil
- ovowconfigexchange
- ImportPolicies

Exchange node configurations

In an environment with multiple management servers, you must configure each node on every management server that may receive the node's messages. Management servers discard messages if they originate from unknown nodes. This applies for both agent-based and server-based flexible management.

In addition, before you can deploy a flexible management policy from a management server, you must configure nodes on that management server to represent all the other management servers that the flexible management policy mentions.

HPOM provides commands that enable you to exchange node configuration data between management servers. To speed up the distribution of node configurations, it may be useful to first upload all node configurations to one management server. You can then download the node configurations to all other management servers.

To exchange node configurations between HPOM for Windows management servers

1. In the console tree, right-click the node group that you want to export, and then click Properties .
2. Select the Unique ID , right-click it, and then click Copy . HP defined node groups have readable IDs, for example Root_Nodes. Other node groups have numerical IDs, for example {89BE12EB-5FB7-4CA5-ABAC-D540B63CFEE3}. The braces are part of the ID.
3. Open a command prompt and type the command:

   ovpmutil CFG NDS DNL <*filename.mof* > /p <*unique ID* >

   For example, to export all node configurations in the Windows node group to the file `c:\temp\windows.mof` , type:

   ```
   ovpmutil CFG NDS DNL c:\temp\windows.mof /p Root_Windows
   ```
4. *Optional.* Add either of the following optional parameters to the command:

   `/configuredonly`
   Export only nodes that have the operating system defined.

   `/externalonly`
   Export only external nodes.
5. Press Enter . The command generates a file with the name you specify.
6. Copy the file to your other HPOM for Windows management server.
7. Open a command prompt and type the command:

ovpmutil CFG NDS UPL <*filename.mof*> /noautodeploy

The `/noautodeploy` option clears the Enable Auto Deployment property for every imported node, regardless of the setting on the original management server. To import the setting from the original management server, omit this option. If automatic deployment is selected for any node, the management server begins to deploy policies to the node immediately.

8. *Optional.* In the console tree, right-click the imported node group and then click All Tasks → Synchronize inventory → Policies and packages . This updates the inventory to reflect policies and packages that already exist on the node. Alternatively, you can exchange node inventory data from the original management server.

To export node configurations for HPOM for UNIX management servers

1. Open a command prompt and type the command:

   ovowconfigexchange -ent NODES -dnl <*folder*> -src_path "<*path to node group*>"

   For example, to export all node configurations in the Windows node group to the folder `c:\temp\windows-nodes` , type:

   ```
   ovowconfigexchange -ent NODES -dnl c:\temp\windows_nodes -src_path "\HP Defined
   Groups\Windows"
   ```

2. *Optional.* Add either of the following optional parameters to the command:

   `-no_ip_detect`
   : Disable DNS lookups of IP addresses for nodes that have the FQDN specified.

   `-dest_codeset` <*code set*>
   : Export the files using a specific code set. The default is ASCII. The code set must match the code set that the HPOM for UNIX database uses.

3. Press Enter . The command generates several files in the folder you specify.
4. Copy the folder and its contents to your HPOM for UNIX management server. You can import the node configurations with the command `opccfgupld` . For more details, see the HPOM for UNIX documentation.

To import node configurations from HPOM for UNIX management servers

1. You can export node configurations on HPOM for UNIX management servers using the HPOM for UNIX user interface or with the command `opccfgdwn` . For more details, see the HPOM for UNIX documentation.
2. HPOM for UNIX exports configuration data to a target directory that you specify. The target directory

contains a hierarchy of subdirectories. Copy the entire directory, including all subdirectories and files, to a folder on your HPOM for Windows management server.

3. *Optional.* Disable automatic deployment on the management server. Otherwise, when you import the node configurations, the management server begins to deploy policies to the nodes immediately. For more details, see Disable policy autodeployment .

4. On the HPOM for Windows management server, open a command prompt and type the command:

   ovowconfigexchange -ent NODES -upl <*folder* > -dest_path "<*path to node group* >" -src_codeset <*code set* >

   -upl
   : Specify the folder that contains the configuration data from the HPOM for UNIX management server.

   -dest_path
   : Specify an existing node group.

   -src_codeset
   : Specify the code set that the HPOM for UNIX database uses.

   For example, to import node configurations in the SJIS code set from the folder c:\temp\UNIX_nodes\ into a node group called HPOM for UNIX Nodes, type:

   ```
   ovowconfigexchange -ent NODES -upl c:\temp\UNIX_nodes\ -dest_path "\HPOM for UNIX
   Nodes" -src_codeset SJIS
   ```

5. *Optional.* In the console tree, right-click the imported node group and then click All Tasks → Synchronize inventory → Policies and packages . This creates jobs to retrieve the inventory of policies and packages that already exist on the node.

Related Topics:

- ovpmutil
- Synchronize policies and packages
- Exchange node inventories
- ovowconfigexchange

Exchange node inventories

The management server keeps an inventory of the policies and packages that are currently installed on the node. HPOM provides commands that enable you to exchange node inventory data between HPOM for Windows management servers. Before you exchange node inventories, you should consider exchanging node configurations and policies. When you import inventory data on a management server, the management server ignores and associations between unknown nodes, policies, and packages.

Node inventory exchange is an alternative to inventory synchronization, which creates jobs to retrieve inventory data from each node over the network.

To exchange node inventories

1. In the console tree, right-click the node group that you want to export node inventories for, and then click Properties .
2. Select the Unique ID , right-click it, and then click Copy . HP defined node groups have readable IDs, for example Root_Nodes. Other node groups have numerical IDs, for example {89BE12EB-5FB7-4CA5-ABAC-D540B63CFEE3}. The braces are part of the ID.
3. Open a command prompt and type the command:

   ovpmutil cfg ppn dnl <*filename.xml* > /p <*unique ID* >

   Press Enter . The command generates a file with the name you specify. For example, to export node inventories for a node group with the ID {89BE12EB-5FB7-4CA5-ABAC-D540B63CFEE3} to the file inventory.xml in the current folder, type:

   ```
   ovpmutil cfg ppn dnl inventory.xml /p {89BE12EB-5FB7-4CA5-ABAC-D540B63CFEE3}
   ```
4. Copy the file to your other HPOM for Windows management server.
5. Open a command prompt and type the command:

   ovpmutil cfg ppn upl <*filename.xml* >

   For example to import node inventories from the file inventory.xml in the current folder, type:

   ```
   ovpmutil cfg ppn upl inventory.xml
   ```

Related Topics:

- Synchronize policies and packages
- ovpmutil

Exchange instruction texts

Messages often have useful instruction texts that give in-depth information and helpful advice about the problem that caused the message. These instruction texts can be very lengthy. To reduce bandwidth consumption, instruction texts are not sent within messages, but are stored on the management server as part of the policy that generates the message. The message itself contains a reference to the policy, so the management server can therefore retrieve the instruction text for the message. However, if a management server receives a message that refers to a policy that is not available on that management server, the management server cannot retrieve the instruction text.

To ensure that instruction texts are available on all management servers, exchange policies (and all policy versions) that contain the instruction texts to be copied. To exchange policies between two HPOM for Windows management servers, you can use the command line tool `ovpmutil` as follows:

1. Export all policies and all policy versions on the source management server using this command:
   ```
   ovpmutil CFG POL DNL all_policies.mof /a
   ```
2. Copy the `all_policies.mof` to all other management servers and import it using this command:
   ```
   ovpmutil CFG POL UPL all_policies.mof
   ```

You can also export instruction texts without the policies for later importing to an HPOM for UNIX management server. Export instruction texts using this command:

```
ovowconfigexchange -ent INSTR_TXT -dnl <folder >
```

This especially useful if your HPOM for UNIX management server receives messages that reference policies of a type that HPOM for UNIX does not support.

Related Topics:

- ovpmutil
- ovowconfigexchange

Configuration interoperability

Moving configuration information between version HP Operations Manager (HPOM) 7.x or 8.x management servers on UNIX or Windows can be expected to function with the exceptions noted below:

- Templates/Policies
  Special characters and spaces are allowed in template names, but not in policy names. Rename templates that contain such characters before attempting to upload them to an HPOM for Windows management server.
  - Log file template / Log file Entry policy: Supported.
    HPOM for UNIX log file templates with a Windows Event Log source will be converted to Windows Event Log policies.
  - Windows Event Log policy: Supported.
    HPOM for Windows policies of this type will be converted to log file templates with source 'Event Log'.
  - opcmsg template / Open Message Interface policy: Supported.
    No restrictions.
  - SNMP trap template/ SNMP Interceptor policy: Supported.
    No restrictions.
  - Scheduled action template / Scheduled task policy: Supported if no scripting is used.
    HPOM for Windows 7.x policies can only be imported into HPOM for UNIX if they do not use scripting.
  - Threshold Monitor template / Measurement Threshold policy: Conversion is possible from HPOM for UNIX 7.x and 8.x to HPOM for Windows 8.x.
  - Agent-based flexible management template/flexible management policy: Cannot be downloaded or uploaded with command-line tools. Use the editors to cut and paste instead. Escalation, message forwarding, and service outage are supported only in HPOM for UNIX environments.
  - You cannot exchange the following types of HPOM for UNIX template, because there are no equivalent policy types in HPOM for Windows:
    - Event correlation circuit (EC) templates
    - MPE/iX console (Console) templates
  - You cannot exchange the following types of HPOM for Windows policies, because there are no equivalent template types in HPOM for UNIX:
    - Windows Management Interface (WMI) policies
    - Service auto-discovery policies
    - Node info policies
- Nodes
  Use `ovowconfigexchange` to exchange configuration data (some specific subset) between HPOM for

Windows and HPOM for UNIX servers in either direction. Use the tool `ovpmutil` to exchange configuration data between two HP Operations management servers. Nodes with a type that is not supported with HPOM for Windows cannot be uploaded (for example, Node for External Events, MPE node).

- User Roles
  Only possible between two HPOM for UNIX management servers or two HPOM for Windows management servers. Exchange of user roles is not possible between HPOM for Windows and HPOM for UNIX.
- Tools/Applications
  Only possible between two HPOM for UNIX management servers or two HPOM for Windows management servers. Exchange of tools and applications is not possible between HPOM for Windows and HPOM for UNIX.
- Services
  Only possible for two HPOM for UNIX management servers or two HPOM for Windows management servers, but not from HPOM for UNIX to HPOM for Windows. From HPOM for Windows 8.x to HPOM for UNIX, with known restrictions.
  - Reports and graphics associated with services on Windows are ignored when uploaded into UNIX.
  - GUIDs of hosting nodes are replaced by fully qualified host names.
  - Some fields are known on UNIX but not on Windows (depth, background, title, service attributes, icons) and are not filled.
  - Operator assignment must be done manually after uploading a Windows service configuration into UNIX.
  - The tool-user field might not be set on Windows, but is a required field on UNIX. Set this before exporting, or edit the exported XML file.
  - The tool-password field on Windows is ignored. (An equivalent field does not exist in HPOM for UNIX.)
- Instrumentation
  Although instrumentation, (the contents of the action, monitor, and command directories) can be copied between servers, the UNIX management server does not allow subdirectories. This means categories cannot be established to automatically deploy instrumentation along with a particular policy.

Homogeneous configuration synchronization

In a complex environment consisting of multiple management servers, you might synchronize the configuration information between multiple management servers or distribute it from a central management server. Synchronizing or distributing configurations to different management servers involves three major steps:

- Select parts of configuration to download

  Use the ovpmutil tool to select which parts of the configuration should be downloaded. This information is stored in files.

  Examples

  a. Download all nodes in the node group *Windows* to the file *windows.mof* file in the current directory:

     Ovpmutil needs the Unique ID of the node group. You can copy the Unique ID from the properties page of the node group. In this example, the *Windows* node group has the Unique ID *{CA017CBA-5B0E-11D5-A4F2-00108335D390}.*
     ```
     ovpmutil CFG NDS DNL windows.mof /p OV_NodeGroup.Name=\"{CA017CBA-5B0E-11D5-A4F2-
     00108335D390}\"
     ```

  b. Download all services in the node group *Exchange 55* to the file *exchange.mof* file in the current directory: Ovpmutil needs the Unique ID of the service. You can copy the Unique ID from the properties page of the service. In this example, the *Exchange 55* service has the Unique ID *Exchange55.*
     ```
     ovpmutil CFG SVC DNL exchange.mof /p OV_Service.Name=\"Exchange55\"
     ```

  c. Download all policies in the policy group *Samples* to the file *policies.mof* file in the current directory: Ovpmutil needs the complete path of to the policy group, as shown in the console tree, starting under Policy groups.
     ```
     ovpmutil CFG POL DNL exchange.mof /p "\Samples\"
     ```

- Distribute Downloaded Files

  Distribute the downloaded files to another management server using a tool of your choice (for example, ftp).

- Upload Files

  The administrator at the receiving management server uploads the files using the ovpmutil tool.

  Examples

  a. Upload all nodes saved to the *windows.mof* file in the current directory:
     ```
     ovpmutil CFG NDS UPL windows.mof
     ```

  b. Upload all services saved to the *exchange.mof* file in the current directory:
     ```
     ovpmutil CFG SVC UPL exchange.mof
     ```

c. Upload all policies saved to the exchange.mof file in the current directory:

   ```
   ovpmutil CFG POL UPL exchange.mof
   ```

CAUTION:
Before you perform an upload on the target server, be sure that none of the nodes or services that you are uploading already exist on that server with different unique IDs (for example, because you manually set up this node or service on that server).

If the nodes or services already exist with different unique IDs, the upload will create duplicate entries, which can cause problems when forwarded messages arrive because they are tied to the Unique service ID known on the source management server.

Heterogeneous configuration synchronization

Configuration information, including policies or templates and services, can be moved from an HPOM for Windows management server to an HPOM for UNIX management server and vice versa. The instructions below explain the general procedure. Refer to the documentation of both products for the syntax of the download and upload tools.

NOTE:
Some types of policies and templates cannot be used on other platforms. See Configuration interoperability for more details.

To move node configuration from HPOM for UNIX to HPOM for Windows

1. Download the HPOM for UNIX templates to a download config package with the command opccfgdwn . (See the HPOM for UNIX Administrators Reference for more information).
2. Copy the config download package to the HPOM for Windows management server.
3. Upload the config download package with the command-line tool ovowconfigexchange .

To move all other template information from HPOM for UNIX to HPOM for Windows

1. Download the HPOM for UNIX templates to a download config package with the command opccfgdwn . (See the HPOM for UNIX Administrators Reference for more information.)
2. Copy the config download package to the HPOM for Windows management server.
3. Upload the config download package with the command line tool ImportPolicies
4. Customize the policies as required and deploy to the appropriate nodes.

To move node configuration and instruction texts from HPOM for Windows to HPOM for UNIX

This procedure applies only to node configuration and instruction texts.

1. Export the HPOM for Windows configuration to HPOM for UNIX templates using the HPOM for Windows command ovowconfigexchange .
2. Copy the files from the HPOM for Windows management server to the HPOM for UNIX management server using a method of your choice.

3. Upload the templates to HPOM for UNIX using `opccfgupld` or `opctempl` . (See the HPOM for UNIX Administrators Reference for more information.)
4. Additional steps may be necessary. See the ovowconfigexchange help topic.

To move all other configuration information from HPOM for Windows to HPOM for UNIX

1. Export the HPOM for Windows configuration to HPOM for UNIX templates using the HPOM for Windows command ovpmutil . Do not mix policies with other configuration information. If you want to move templates and other configuration information, create two structured storage files.
2. For downloaded policies, convert the downloaded structured storage file to an ASCII file with the command `ovpmutil PCV/x` .
3. Copy the file from the HPOM for Windows management server to the HPOM for UNIX management server using the method you prefer.

   NOTE:
   In the Japanese environment, convert the file from uti-8 to Shift-JIS..

4. Upload the policy information to the HPOM for UNIX management server database with the command line tool `opctempl` . An XML service file can be used without further conversion.
5. If you uploaded templates, customize them as required and deploy to the appropriate nodes.

Using the ovowconfigexchange command

With the release of HPOM for Windows 7.50, you can use the ovowconfigexchange tool to exchange a subset of configuration data between cross-platform HP Operations management servers (to support message forwarding), in most cases between HPOM for Windows and HPOM for UNIX servers in either direction.

This tool operates only on the HPOM for Windows platform. Its counterparts on HPOM for UNIX are `opccfgupld/opccfgdwn` and `opctempl` tools. It can upload data from HPOM for UNIX or download local configuration to files, formatted for upload to HPOM for UNIX. Files must be manually copied between the systems. See ovowconfigexchange for further details.

Switch the primary management server

By default, nodes send messages to the management server that you used to install the agent. This is called the primary management server. If you want to switch the primary management server after you install the agent, you launch a tool on the new management server. After you do this, the nodes send their messages to the new primary management server.

NOTE:
It is also possible to configure a different primary management server in the agent installation defaults, but you must do this before you install the agent.

You may want to prepare to switch primary management servers in a backup scenario. For example, you could configure a second management server to monitor the health of a primary management server. If the second management server detects a system failure at the primary management server, an administrator could then switch primary management responsibility to the second management server. The second management server would assume control of all the managed nodes previously managed by the failed management server.

Before you can switch primary management servers, you must configure the following prerequisites:

- Exchange trusted certificates between the management servers, and update the trusted certificates on the nodes. (See Configure trusted certificates for multiple management servers .)
- Exchange node configuration between the management servers. The new primary management server must exists as a node on the current primary management server. The nodes for which you want to switch the management server must exist on both management servers. (See Exchange node configurations .)
- Configure the new primary management server as a secondary management server of the nodes. You do this by deploying a flexible management policy to the nodes from the current primary management server. (See Configure action-allowed and secondary managers .)

In addition, if the primary management server is currently an HPOM for UNIX management server, you must consider the following prerequisites:

- If the HPOM for UNIX management server deployed the agent package to the node, the agent package version must be 8.51 or higher. (You cannot switch the primary management server if the HPOM for UNIX management server deployed a lower version of the agent package. This restriction includes all DCE agents.)
- You cannot manage some HPOM for UNIX templates from an HPOM for Windows management server. Similarly, you cannot manage some HPOM for Windows policies from an HPOM for UNIX management server. Consider removing or disabling these types of templates and policies before you switch the primary management server. (See Configuration interoperability .)

To switch the primary management server

1. Open a console connected to the new primary management server.
2. In the console tree, expand Tools → HP Operations Manager Tools .
3. In the details pane, double-click Switch Management Server . The Parameters page opens.
4. Select the nodes that you want to switch to this management server. Click Next . The Login page opens.
5. *Optional* . Type a User Name and Password . The user must be a member of the HPOM administrators group. If you leave the User Name and Password blank, the tool starts with the credentials that you are currently logged in to Windows with.
6. Click Launch... . The Tool Status dialog opens, which updates when the tool succeeds or fails.

TIP:
You can also use the opcragt command-line tool to switch the primary management server on all selected nodes. For more details, see opcragt .

Related Topics:

- Configure HTTPS agent installation defaults
- Configure DCE agent installation defaults

Agent-based flexible management

Agent-based flexible management enables you to configure managed nodes to send messages to different management servers based on time and message attributes. This enables you to manage your worldwide network more effectively across time zones (for example, by using follow-the-sun control). It also enable you to increase efficiency (for example, by creating competence centers).

Key features of agent-based flexible management are summarized below.

- You can configure agents to communicate with both HPOM for Windows and HPOM for UNIX management servers, regardless of the platform from which the agent was installed. Agent-based flexible management allows you to manage your environment in several important ways:
    - Configure agents to send messages to different management servers based on criteria in the message. See Message target rules for more information.
    - Configure agents to allow actions from several management servers. See Configure action-allowed and secondary managers for more information.
    - Switch primary management server function to another server to manage an expanding network environment, and reduce primary server bottlenecks. See Switch the primary management server for more information.
- You can deploy policies or templates to any agent (regardless of which management server installed the agent) from the management server configured to be the agent's primary management server. For nodes that have an HTTPS agent, you can also deploy policies from secondary management servers.
- Even though some policies or templates are specific to a particular management server, the messages that result from the policies or templates can be sent to any management server.

Related Topics:
- Follow-the-sun control
- Competence centers
- Create an agent-based flexible management policy

Create an agent-based flexible management policy

An agent-based flexible management policy, enables you to configure the following:

- Action-allowed and secondary management servers that define which management servers can run actions on the managed node.
- Date-and-time rules that define when the managed node sends messages to which management server.
- Message-attribute rules that define when the managed node sends messages to which management server.

If you want the configuration to apply to all nodes in a given environment, you would develop one policy for all nodes. If you want varying configuration on different nodes, you would develop one policy for each configuration type.

To create an agent-based flexible management policy

1. In the console tree, under Policy management → Agent policies grouped by type , right-click Flexible Management and then click New→Policy . The flexible management policy editor appears.

   Alternatively, create a copy of the sample policies, which are provided in Policy groups→Samples→ Agent-based Flexible Management .
2. In the General tab, type the agent-based flexible management policy. The following topics provide more information on the policy syntax:

   - Syntax for agent-based flexible management policies
   - Keywords for flexible management policies
   - Time templates
   - Message target rules
   - Configure action-allowed and secondary managers

   TIP:
   You can split the text area vertically and horizontally by dragging the split controls. (The split controls are at the top of the vertical scroll bar and at the left of the horizontal scroll bar.)
3. *Optional.* To check the policy's syntax, click Check Syntax . A message appears, which gives details of any errors.
4. Save the policy, and then deploy it to the nodes that you want to configure.

Related Topics:

- Save a policy
- Deploy a policy or policy group

Syntax for agent-based flexible management policies

You can use the syntax described in the following sections as a basis for configuring agent-based flexible management features.

Special Characters in agent-based Flexible Management Templates:

The syntax examples below use the following special characters:

Symbol	Description
e	Empty string. Note that "e" is used only in the examples. In the actual template, an empty string should, in fact, be used.
#	Comment. If you want to include a comment in a template, include a number sign (#) before every line of the comment. Every character in the line is treated as part of the comment. Example: # This is a comment
\	Escape character. If you want to use quotation marks in a syntax string, escape the quotation marks with a backslash (\). Example: \"quotation\"

Syntax for Responsible Management Server Configuration policies

Use the following syntax for responsible management server configuration policies:

```
respmgrconfigs ::= <respmgrconfigs> RESPMGRCONFIG DESCRIPTION  <string> <respmgrconds> | e

respmgrconds   ::= SECONDARYMANAGERS <secondmgrs> ACTIONALLOWMANAGERS <actallowmgrs>
                   [MSGTARGETRULES <msgtargetrules>]

secondmgrs     ::= <secondmgrs> SECONDARYMANAGER NODE <node>  [DESCRIPTION <string>] | e

actallowmgrs   ::= <actallowmgrs> ACTIONALLOWMANGER NODE <node> [DESCRIPTION <string>] | e

msgtargetrules ::= <msgtargetrules> MSGTARGETRULE DESCRIPTION <string> <msgtargetrule> | e

msgtargetrule  ::= MSGTARGETRULECONDS <mtrconditions> MSGTARGETMANAGERS <msgtargetmgrs>
                   | MSGTARGETRULECONDS <mtrconditions> MSGTARGETMANAGERS <msgtargetmgrs> ACKNO
```

```
mtrconditions  ::= <mtrconditions> MSGTARGETRULECOND DESCRIPTION <string> <mtrcond> | e

mtrcond        ::= <mtrcond> SEVERITY <severity> |
                   <mtrcond> NODE <nodelist> |
                   <mtrcond> APPLICATION <string> |
                   <mtrcond> MSGGRP <string> |
                   <mtrcond> OBJECT <string> |
                   <mtrcond> MSGTYPE <string> |
                   <mtrcond> TEXT <string> |
                   <mtrcond> SERVICE_NAME <string> |
                   <mtrcond> MSGCONDTYPE <msgcondtype> | e

severity      ::= Unknown | Normal | Warning | Critical |
                  Minor | Major

msgcondtype   ::= Match | Suppress

nodelist      ::= <node> | <nodelist> <node>

node          ::= IP <ipaddress> | IP <ipaddress> <string>

string        ::= "any alphanumeric string"

ipaddress     ::= <digits>.<digits>.<digits>.<digits>
```

Syntax for Time Templates

Use the following syntax for time templates:

```
timetmpls     ::= <timetmpls> TIMETEMPLATE <string>
                  DESCRIPTION
                  <string> <conditions> | e
conditions    ::= TIMETMPLCONDS <timetmplconds> | e
timetmplconds ::= <timetmplconds> TIMETMPLCOND <timetmplcond>
timetmplcond  ::= [TIMECONDTYPE <timecondtype>] [TIME FROM
                  <time> TO <time>] [WEEKDAY <weekday>]
                  [DATE <exact_date>] | e
timecondtype  ::= Match | Suppress
time          ::= <hh>:<mm>
weekday       ::= ON <day> | FROM <day> TO <day>
exact_date    ::= ON <date> | FROM <date> TO <date>
day           ::= Monday | Tuesday | Wednesday | Thursday
                 | Friday | Saturday | Sunday
date          ::= <mm>/<dd>/<yyyy> |<mm>/<dd>/*
```

NOTE:
The time template is compared with the creation time of the message on the managed node . Message creation time is always defined in GMT.

Syntax for Management Responsibility Switching

Use the following syntax for templates that switch management server responsibility:

```
configfile := [TIMETEMPLATES <timetmpls>] RESPMGRCONFIGS
              <respmgrconfigs>
```

Syntax for Message Target Rules

Use the following syntax for templates that define message target rules:

```
msgtargetmgrs ::= <msgtargetmgrs> MSGTARGETMANAGER
                  TIMETEMPLATE <string> OPCMGR <node> |
                  <msgtargetmgrs> MSGTARGETMANAGER
                  TIMETEMPLATE <string> OPCMGR <node>
                  MSGCONTROLLINGMGR | <msgtargetmgrs>
                  MSGTARGETMANAGER TIMETEMPLATE <string>
                  OPCMGR <node> NOTIFYMGR | e
```

NOTE:
You can replace the <string> variable with $OPC_ALWAYS to specify that the time condition is always true. To specify that the current primary management server is always used as the message target server, replace the <node> variable with $OPC_PRIMARY_MGR. Note also that pattern matching is only available in <string>.

Keywords for flexible management policies

Keyword	Definition
RESPMGRCONFIG	Responsible manager configuration.
DESCRIPTION	Short description of the manager.
SECONDARYMANAGERS	Secondary managers of an agent. Each of these management servers have permission to take over responsibility and become the primary manager for an agent. ■ SECONDARYMANAGER: Name of the secondary manager. ■ NODE <node>: Node name of the secondary manager. ■ DESCRIPTION: Description of the secondary manager.
ACTIONALLOWMANAGERS	Management servers that are allowed to execute actions on the managed node. The action response (for example, command broadcast) is sent to this manager. Only the primary manager can configure action-allowed managers for an agent. ■ ACTIONALLOWMANAGER: Name of the manager allowed to execute actions on the managed node. ■ NODE: Node name of the action-allowed manager. You can use the variable $OPC_PRIMARY_MGR to specify that this node name is always the node name of the primary manager. ■ DESCRIPTION: Short description of the action-allowed manager.
MSGTARGETRULES	Message target rules. ■ MSGTARGETRULE: Rule to configure the message target conditions and the message target manager. ■ DESCRIPTION: Description of the message target rule.
MSGTARGETMANAGERS	Message target managers. Management server to which the agents send messages, as well as the action responses to those messages. The result of a message is sent to only one management server. The keyword is also used to escalate messages from one management server to another. ■ MSGTARGETMANAGER: Message target manager. Management server to which you forward a message. Always specify the IP address of the target management server as 0.0.0.0. The real IP address is then resolved by the domain name server (DNS). ■ TIMETEMPLATE: Time template. Name of the time template corresponding to the target manager. If the time condition is always true, you can use the variable $OPC_ALWAYS. If you use this keyword, message transfers to the target manager will not depend on the time.

- OPCMGR: Node name of the target manager. You can use the keyword $OPC_PRIMARY_MGR to indicate that this will always be the primary manager.
- MSGCONTROLLINGMGR: Message-controlling manager. Enables message target manager to switch control of a message.
- NOTIFYMGR: Notify manager. Enables the message target manager to notify itself. This attribute is set by default if no attribute is defined for the message target manager.
- ACKNONLOCALMGR: Enables a message rule to force a direct acknowledgment of a notification message on a source management server.

MSGTARGETRULECONDS Message target rule conditions.

- MSGTARGETRULECOND: Condition that tells the agent to which management server to send specific messages. Messages are sent based on message attributes or time. The message agent evaluates the message target conditions by reading the file mgrconf. If the mgrconf file does not exist, the messages are sent to the management server name stored in the primmgr file. If the primmgr file does not exist, messages are sent according to instructions set using the ovconfchg command-line tool.
- DESCRIPTION: Description of the message target rule condition.
- SEVERITY: Severity level of the message. Can be Unknown, Normal, Warning, Minor, Major, Critical.
- NODE <node>: One or more node names, separated by spaces. You can specify a node in different ways (for example, NODE IP 0.0.0.0 hpbbn). If the node is defined using the format IP <ipaddress> or IP <ipaddress> <string>, you should use the IP address "0.0.0.0." The real IP address is then resolved by the domain name server (DNS).
- APPLICATION: Application name.
- MSGGRP: Message group name.
- OBJECT: Object name.
- MSGTYPE: Description of the message type.
- MSGCONDTYPE: Message condition type:
  - Match Condition is true if the specified attributes are matched.
  - Suppress Condition is true if the specified attributes are not matched.

TEXT A string containing all or part of the message text. Pattern-matching may be used.

- SERVICE_NAME: A string containing the unique identifier of the service. Pattern-matching may be used.

- MSGOPERATION: Message operation:
  - Suppress
  - Log-only
  - Inservice

Time templates

A time template is a set of conditions (or rules) that tells the agent to which management server and at what time a given managed node should send specific messages. You create time conditions and save them in time templates. You can combine simple rules to set up more complex constructions (for example, "on Monday, Wednesday and Thursday from 10 am to 11:35 am from January to March"). Time conditions are defined using the 24-hour clock notation (for example, for 1:00 p.m., you would enter "13:00").

Setting Time Intervals

You can set several different time intervals as follows:

- No Time

  If you specify no particular time, day of the week, or year, HP Operations Manager for Windows assumes you want the condition to be true from 00:00 to 24:00 every day of the year, every year. If you specify a condition, HP Operations Manager for Windows assumes the condition should apply continually for the time and day specified.

  For example, specifying "Tuesdays" triggers a condition every Tuesday from 00:00 to 24:00 throughout the year, every year.

- Span of Time

  Specify a time range (for example, "from 7:00 to 17:00").

- Wildcard (*) Date or Period

  Use wildcards (*) in dates or periods of time (for example, to set a condition for January 31 every year, you would enter "1/31/*").

Configuring Time-indifferent Templates

HP Operations Manager for Windows requires that you set up a time template for the message target rules even if your scheduled action is time-indifferent. HP Operations Manager for Windows provides the variable `$OPC_ALWAYS` to configure time-indifferent templates.

Related Topics:

- Time template examples and keywords

Time template examples and keywords

Time templates are a part of the agent-based flexible management policy syntax. They allow you to configure the agent to send messages to different management servers based on the time and day.

A time template consists of the following:

- Template name
- Time conditions

Each time condition defines a specific time period. This time period contains definitions of the time, day, date, or any combination of the three. The local time zone is always used to evaluate the template.

NOTE:
When specifying a time, use 24-hour clock notation. For example, for "1:00 p.m." type 13:00.

Examples of Time Templates

The following examples show various ways to specify time formats in the time templates:

- No Time

  If you do not specify a particular time, day of the week, or year, then the condition will be true for 24 hours, from 00:00 to 24:00 every day of the year.

  You have to set up a time template for the message target rules even if the scheduled action does not depend on time. You can use the variable $OPC_ALWAYS to configure time templates when the condition is always true.
- Specific Dates or Dates

  If you specify a condition, the conditions exist continually for the day or date specified:

  - Day
    If you specify only Tuesday, the condition will evaluate as true every Tuesday from 00:00 to 23:59 throughout the year, every year. Use the syntax:
    WEEKDAY ON Tuesday
  - Date
    Specifying January 1 and nothing else will match a condition every January 1st of every year. Use the syntax:
    DATE ON 01/01/*
- Time Periods

  You can set time periods:

- Time
  To set a time period from 7:00 to 17:00, use the syntax:
  TIME FROM 7:00 TO 17:00
- Day
  To set a time period from Monday to Friday, use the syntax:
  WEEKDAY FROM Monday TO Friday
- Date
  To set a time period from the year 1995 to 2000, use the syntax:
  DATE FROM 01/01/1995 TO 12/31/1999
- Date and Time
  To set a time on December 31 1998, from 23:00 to 23:59, use the syntax:
  TIME FROM 23:00 TO 23:59 DATE ON 12/31/1998

  If you include the day of the week (for example, Monday April 1, 1997), the day and date will be cross-checked to make sure that they match the calendar.

- Wildcards (*)

  You can set dates or periods using a wildcard character (*):

  - Specific Dates
    To set a condition for December 1st every year, use the syntax:
    DATE ON 12/01/*
  - Time Periods
    To set a condition from August 6th to September 10th every year, use the syntax:
    DATE FROM 08/06/* TO 09/10/*

Keywords for Time Templates

To define the various elements required in an agent-based flexible management configuration, the following keywords and definitions are used:

TIMETEMPLATE <string>
Template name is contained in <string>.
DESCRIPTION
Short description of the time template.
TIMETMPLCONDS
TIMETMPLCOND
TIMECONDTYPE
Condition defining a single time interval. Several time conditions together make up a time period. A time condition allows you to use combinations of day, date, and time to define a time period.
At least one of the following parts must be used for the definition:

- Match
- Suppress
  If the current time is within the defined time period, match is true and suppress is false.
  TIME FROM <time> TO <time>

Specifies a time period. Set the variable <time> using the format:

<HH>:<MM>

The FROM <time> variable must be before the TO <time> variable (for example, FROM 18:00 TO 24:00 or FROM 0:00 TO 6:00).

WEEKDAY

You can specify every day of the week: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday:

- ON <day>

  Day of the week (for example, ON Sunday).

  FROM <day> TO <day>

  Time period (for example, FROM Monday TO Wednesday).

  DATE

  Date must have one of the following formats:

  <MM>/<DD>/<YYYY>

  <MM>/<DD>/<YY>

  <MM>/<DD>/*

  Invalid time periods are not recognized. For example, 10/35/* is not recognized as an invalid date.

  You specify the date as follows:

  ON <date>

  FROM <date>

  TO <date>

Related Topics:

- Syntax for agent-based flexible management policies

Message target rules

You can use a list of message target rules to determine to which management server a message should be sent to.

Parts of a Message Target Rule

A message target rule consists of three parts:

- Message attribute rule
- Time template
- Defined management server

Example of a Message Target Rule for Printing Group

A message target rule for a printing group would have the following conceptual structure:

message group = "printing"

current time fits *time template 2 ....(message) --> mgr 2*

current time fits *time template 1 ....(message) --> mgr 1*

current time fits *time template 3 ....(message) --> mgr 3*

In this example, HP Operations forwards all messages with the message group "printing" that meet the time conditions in template 1 to the management server 1. All messages that meet the time conditions in template 2 will be forwarded to management server 2. Time template 3 functions the same.

Example of a Message Target Rule for a Database Group

A message target rule for a database group would have the following conceptual structure:

message group = "database"

current time fits *time template 1 ....(message) --> mgr 2*

current time fits *time template 2 ....(message) --> mgr 3*

current time fits *time template 3 ....(message) --> mgr 1*

In this example, HP Operations forwards all messages with the message group "database" that meet the time conditions in template 1 to the management server 2. All messages that meet the time conditions in template 2 are sent to the management server 3. And so on.

Configure action-allowed and secondary managers

By default, only a node's primary management server can start actions on the node. To enable other management servers to start actions on a node, you must specify action-allowed management servers in a flexible management policy and deploy it to the node. This is important if you forward messages that have automatic and operator-initiated actions to other management servers.

The primary management server is initially set during the agent installation. To enable other management servers to become a node's primary management server, you can specify secondary management servers in the same policy. If a node has the HTTPS agent, the secondary management servers can also deploy policies and packages to the node, without first becoming the primary management server.

A flexible management policy that configures action-allowed and secondary managers must contain the following statements:

```
RESPMGRCONFIGS
     RESPMGRCONFIG DESCRIPTION "Policy description"
     SECONDARYMANAGERS
     ACTIONALLOWMANAGERS
```

You can add to this minimal policy as many secondary managers and action-allowed managers as you need. You can specify either the IP address or host name of each management server. To specify only the host name, use the IP address 0.0.0.0. For example, the following policy specifies manager1.example.com and manager2.example.com as secondary and action-allowed managers. It also specifies that the management server with IP address 192.168.1.3 is an action-allowed manager.

```
RESPMGRCONFIGS
     RESPMGRCONFIG DESCRIPTION "Enable manager1, manager2, and 192.168.1.3"
     SECONDARYMANAGERS
           SECONDARYMANAGER NODE IP 0.0.0.0 "manager1.example.com"
           SECONDARYMANAGER NODE IP 0.0.0.0 "manager2.example.com"
     ACTIONALLOWMANAGERS
           ACTIONALLOWMANAGER NODE IP 0.0.0.0 "manager1.example.com"
           ACTIONALLOWMANAGER NODE IP 0.0.0.0 "manager1.example.com"
           ACTIONALLOWMANAGER NODE IP 192.168.1.3
```

If you use agent-based flexible management, you can add the secondary and action-allowed managers to an existing flexible management policy.

To configure action-allowed and secondary managers

1. In the console tree, open Policy management → Agent policies grouped by type . Right-click Flexible Management , and then click New → Policy . The flexible management policy editor appears.
2. In the General tab, type a flexible management policy that specifies secondary and action-allowed managers.
3. *Optional.* Click Check Syntax . A message appears, which gives details of any errors. Correct any syntax errors.
4. Save the policy, and then deploy it to the nodes that you want to configure.

Alternatively, you can edit or copy one of the sample flexible management policies, which are available in Policy management → Policy groups → Samples → Agent-based Flexible Management .

Related Topics:

- Syntax for agent-based flexible management policies
- Save a policy
- Deploy a policy or policy group

Configure DCE agents to communicate with HP Operations Manager for UNIX

The instructions below explain how to configure an DCE agent that was deployed by HP Operations Manager for Windows, so that it sends messages to, and carries out actions from, an HP Operations Manager for UNIX management server . (HP Operations Manager for UNIX will not be able to deploy policies to this node.) Depending on the agent-based flexible management template that is used, all messages, or a subset of messages can be sent to the HP Operations Manager for Windows management server. In this example, messages are sent to different management servers based on their message group. (For more information about agent-based flexible management configuration, see the HP Operations Manager for UNIX documentation.)

NOTE:
Beginning with the 7.20 release of HP Operations, the DCE agent package is installed by default as a Local System account. The HP ITO account and opc_op account that were used in previous versions of the agent are no longer created. If, in a previous installation you configured tools or policies to run as opc_op user, these tools and policies are not automatically mapped to the Local System account. You must either create the opc_op account manually or reconfigure such tools and policies to specify another user, such as Local System. (See Agent Users and Change the password for multiple tools .

To configure DCE agents to communicate the HPOM for UNIX

1. Prepare the HP Operations Manager for UNIX server:
   a. Set up the node in the HP Operations Manager for UNIX interface as 'controlled' (see the HP Operations Manager for UNIX documentation for details).
   b. Update the HP Operations Manager for UNIX configuration and start heartbeat polling for this node manually, using the following commands:
      i. `/opt/OV/bin/OpC/opcsw -installed <node>`
      ii. `/opt/OV/bin/OpC/opchbp -start <node>`
   c. As user root, copy the template below to the working directory:

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

# This template sets the following configuration:
#
#   - send messages with message group OpC to Unix management server
#   - send messages with message group VPW to Windows management server
#   - allow both servers to run actions on the node
#
```

```
TIMETEMPLATES
# none

RESPMGRCONFIGS
 RESPMGRCONFIG
 DESCRIPTION "responsible mgrs for messages and agents"
  SECONDARYMANAGERS
       SECONDARYMANAGER
     NODE IP 0.0.0.0 "unix.bbn.hp.com"
     DESCRIPTION "HP Operations Manager for UNIX management server"
       SECONDARYMANAGER
     NODE IP 0.0.0.0 "windows.bbn.hp.com"
     DESCRIPTION "HP Operations Manager for Windows management server"
            ACTIONALLOWMANAGERS
            ACTIONALLOWMANAGER
     NODE IP 0.0.0.0 "unix.bbn.hp.com"
     DESCRIPTION "HP Operations Manager for UNIX management server"
       ACTIONALLOWMANAGER
     NODE IP 0.0.0.0 "windows.bbn.hp.com"
     DESCRIPTION "HP Operations Manager for Windows management server"
  MSGTARGETRULES
       MSGTARGETRULE
       DESCRIPTION "Unix responsibility"
            MSGTARGETRULECONDS
                             MSGTARGETRULECOND
                             DESCRIPTION "Unix messages"
                                MSGGRP "OpC"
       MSGTARGETMANAGERS
         MSGTARGETMANAGER
                                TIMETEMPLATE "$OPC_ALWAYS"
    OPCMGR IP 0.0.0.0 "unix.bbn.hp.com"
       MSGTARGETRULE
       DESCRIPTION "Windows responsibility"
     MSGTARGETRULECONDS
                             MSGTARGETRULECOND
                             DESCRIPTION "Windows messages"
                                MSGGRP "VPW"
     MSGTARGETMANAGERS
        MSGTARGETMANAGER
                                TIMETEMPLATE "$OPC_ALWAYS"
    OPCMGR IP 0.0.0.0 "windows.bbn.hp.com"
       MSGTARGETRULE
              DESCRIPTION "Rest of News"
      MSGTARGETRULECONDS
      MSGTARGETMANAGERS
         MSGTARGETMANAGER
```

```
                                TIMETEMPLATE "$OPC_ALWAYS"
    OPCMGR IP 0.0.0.0 "$OPC_PRIMARY_MGR"
```

Rename the file either allnodes , if the file applies to all nodes, or the IP address of an individual managed node in Hex notation, generated using the command `/opt/OV/bin/OpC/install/opc_ip_addr` (use `opc_ip_addr -h` for more information).

d. Modify the file with the names of the HP Operations Manager for UNIX and HP Operations Manager for Windows management servers.

e. Run the HP Operations Manager for UNIX template validation tool opcmomchk(1) on the finished configuration file to ensure that your changes are correct:

   ```
   /opt/OV/bin/OpC/opcmomchk <file_name>
   ```

   See the man page opcmomchk(1) for more information.

f. As user root, copy the validated file to the configuration directory:

   ```
   cp <file_name> /etc/opt/OV/share/conf/OpC/mgmt_sv/respmgrs/
   ```

   NOTE:
   If several, but not all, managed nodes have the same configuration, you can apply a symbolic link to the Hex file name of the related managed node. In addition, the same configuration directory can contain both an allnodes file and files for specific managed nodes. If configuration files for specific nodes are present, they are used in preference to the allnodes file for those nodes.

   HP Operations Manager for UNIX distributes the management responsibility configuration file from this directory as part of the standard template distribution process.

2. Change the management server that is responsible for the agent:

   a. From the HP Operations Manager for UNIX management server, open a terminal window for the node.

   b. Execute `/var/opt/OV/conf/OpC/mgmt_sv.sh` (on a UNIX node) or `Program Files\HP\HP BTO Software\Installed Packages\{790C06B4-844E-11D2-972B-080009EF8C2A}\bin\OpC\install\mgmt_sv.vbs` (on a Windows node) to make the HP Operations Manager for UNIX management server responsible for this node. The script has no parameters, it will ask for the complete name of the HP Operations Manager for UNIX management server.

3. Distribute the agent-based flexible management template(s) to the appropriate managed nodes:

   a. On the HP Operations Manager for UNIX server, select the nodes in the Node Bank window or other submap, then select Actions:Agents → Install/Upgrade SW & Config...

   b. Select the Templates check box in the Install/Update Software and Configuration window.

   c. Click OK .

Or use the command line:
`opcragt -distrib -templates -force` *<name of HP Operations Manager for Windows managed node>*

4. To switch agent back to HP Operations Manager for Windows:

   Execute above script again, now with the name of the HP Operations Manager for Windows management server.

NOTE:
To start an HPOM for Windows tool on an HPOM for UNIX managed node, the corresponding HPOM for Windows scripts have to be available on the HPOM for UNIX managed node. When you want to start an HPOM for UNIX application on an HPOM for Windows managed node, the corresponding HPOM for UNIX instrumentation files must be available on the HPOM for Windows node.

Communicate with DCE agents installed by HP Operations Manager for UNIX

The instructions below explain how to configure a DCE agent installed by HP Operations Manager for UNIX so that it will send messages to, and accept instruction from, an HP Operations Manager for Windows management server. (HP Operations Manager for Windows will not be able to install policies on the managed node.) Depending on the template that is used, all messages, or a subset of messages can be sent to the HP Operations Manager for Windows management server. In this example, messages are sent to different management servers based on their message group.

TIP:
To configure an HTTPS agent installed by HPOM for UNIX so that it sends messages to an HPOM for Windows management server, switch the agent's primary management server. (See Switch the primary management server .)

To communicate with DCE agents installed by HP Operations Manager for UNIX

1. Add the node to HP Operations Manager for Windows. (See, Create new nodes).
2. Ensure that the agent is up to date.
3. As user root, copy the template below to the working directory:

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

```
# This template sets the following configuration:
#
#   - send messages with message group OpC to Unix management server
#   - send messages with message group VPW to Windows management server
#   - allow both servers to run actions on the node
#

TIMETEMPLATES
# none

RESPMGRCONFIGS
 RESPMGRCONFIG
 DESCRIPTION "responsible mgrs for messages and agents"
  SECONDARYMANAGERS
       SECONDARYMANAGER
     NODE IP 0.0.0.0 "unix.bbn.hp.com"
```

```
      DESCRIPTION "HP Operations Manager for UNIX management server"
        SECONDARYMANAGER
      NODE IP 0.0.0.0 "windows.bbn.hp.com"
      DESCRIPTION "HP Operations Manager for Windows management server"
             ACTIONALLOWMANAGERS
             ACTIONALLOWMANAGER
      NODE IP 0.0.0.0 "unix.bbn.hp.com"
      DESCRIPTION "HP Operations Manager for UNIX management server"
        ACTIONALLOWMANAGER
      NODE IP 0.0.0.0 "windows.bbn.hp.com"
      DESCRIPTION "HP Operations Manager for Windows management server"
   MSGTARGETRULES
        MSGTARGETRULE
        DESCRIPTION "Unix responsibility"
             MSGTARGETRULECONDS
                              MSGTARGETRULECOND
                              DESCRIPTION "Unix messages"
                                 MSGGRP "OpC"
        MSGTARGETMANAGERS
          MSGTARGETMANAGER
                                 TIMETEMPLATE "$OPC_ALWAYS"
     OPCMGR IP 0.0.0.0 "unix.bbn.hp.com"
        MSGTARGETRULE
        DESCRIPTION "Windows responsibility"
      MSGTARGETRULECONDS
                              MSGTARGETRULECOND
                              DESCRIPTION "Windows messages"
                                 MSGGRP "VPW"
      MSGTARGETMANAGERS
         MSGTARGETMANAGER
                                 TIMETEMPLATE "$OPC_ALWAYS"
     OPCMGR IP 0.0.0.0 "windows.bbn.hp.com"
        MSGTARGETRULE
               DESCRIPTION "Rest of News"
       MSGTARGETRULECONDS
       MSGTARGETMANAGERS
          MSGTARGETMANAGER
                                  TIMETEMPLATE "$OPC_ALWAYS"
      OPCMGR IP 0.0.0.0 "$OPC_PRIMARY_MGR"
```

Rename the file either allnodes , if the file applies to all nodes, or the IP address of an individual managed node in Hex notation, generated using the command `/opt/OV/bin/OpC/install/opc_ip_addr` (use `opc_ip_addr -h` for more information).

4. Run the HP Operations Manager for UNIX template validation tool opcmomchk(1) on the finished configuration file to ensure that your changes are correct:

```
/opt/OV/bin/OpC/opcmomchk <file_name>
```

See the man page opcmomchk(1) for more information.

5. As user root, copy the validated file to the configuration directory: `cp <file_name> /etc/opt/OV/share/conf/OpC/mgmt_sv/respmgrs/`

   NOTE:
   If several, but not all, managed nodes have the same configuration, you can apply a symbolic link to the Hex file name of the related managed node. In addition, the same configuration directory can contain both an allnodes file and files for specific managed nodes. If configuration files for specific nodes are present, they are used in preference to the allnodes file for those nodes.

   HP Operations Manager for UNIX distributes the management responsibility configuration file from this directory as part of the standard template distribution process.

6. Install the template(s) on the appropriate managed nodes:
   a. Select the nodes in the Node Bank window or other submap, then select Actions:Agents → Install/Upgrade SW & Config...
   b. Select the Templates check box in the Install/Update Software and Configuration window.
   c. Click OK .

Server-based flexible management

Server-based flexible management enables you to forward messages between multiple management servers . After a management server forwards a message, you can keep it up to date on all management servers by configuring them to forward message operations, for example, ownership changes and acknowledgements. Management servers can also forward action responses, which contain information about the success or failure of operator-initiated and automatic actions.

By default, management servers communicate using the HTTPS protocol, but you can configure communication using the DCE protocol if necessary. HPOM for Windows 8.10 supports server-based flexible management with the following management server versions:

	HTTPS-based	DCE-based
HPOM for Windows 7.50	N	Y
HPOM for Windows 8.10	Y	Y
HPOM for UNIX 7.10 patch level A.07.17 (PHSS_29548) or higher	N	Y
HPOM for UNIX 8.10 or higher	Y	Y

Configuring server-based flexible management involves multiple tasks. The following tasks are mandatory:

- Configure prerequisites for server-based flexible management
- Create a server-based flexible management policy

The following tasks are optional:

- Configure forwarding options for server-based flexible management
- Configure duplicate suppression for server-based flexible management
- Configure the character set for server-based flexible management
- Configure HPOM for UNIX compatibility mode
- Configure communication protocols for server-based flexible management
- Security for server-based flexible management
- Migrate from ForwardToVP-based forwarding
- Configure firewall and NAT for DCE-based server communication

After you deploy a valid server-based flexible management policy to a management server, the management server begins to forward messages and message operations. To subsequently stop the management server forwarding messages and message operations, disable the server-based flexible management policy.

NOTE:
If you have two servers set up as target servers for each other (server A and server B) there can be some communication overhead. If every management server is set up to forward messages to every other server, this could be an issue. As the administrator, you will set up the forward configuration file which contains the rules controlling which messages to forward and where to send them.

Related Topics:

- Disable policy

Configure prerequisites for server-based flexible management

When you deploy a valid server-based flexible management policy to a management server , the management server begins to forward messages and message operations. Before you do this, you must ensure that management servers and nodes meet a number of prerequisites. Some tasks are mandatory, and other tasks are optional.

To configure prerequisites for server-based flexible management

1. Configure trusted certificates for multiple management servers

   Before management servers can communicate with each other using the HTTPS protocol, you must exchange their trusted certificates. Also, to enable a management server to communicate with HTTPS nodes that another management server owns (for example to start actions or deploy policies), you must update the trusted certificates on the nodes.
2. Exchange node configurations

   Management servers immediately discard all messages from unknown nodes. Therefore, before a management server can accept forwarded messages, you must upload the appropriate node configurations.
3. *Optional.* Configure action-allowed and secondary managers

   By default, only a node's primary management server can start actions on the node. To enable other management servers to start actions on a node, you must specify action-allowed management servers in a flexible management policy and deploy it to the node.

   To enable other management servers to become a node's primary management server, you can specify secondary management servers in the same policy. If a node has the HTTPS agent, the secondary management servers can also deploy policies and packages to the node.

You can also exchange other configuration data between management servers. This is only necessary if you need to have policies and instructions, user roles, tools, services, or instrumentation available on other management servers.

Related Topics:

- Exchange configuration data between management servers

Configure forwarding options for server-based flexible management

If you use server-based flexible management, the management server maintains queues of messages, message operations, and action responses to forward to other management servers. There is one queue for all management servers that support HTTPS communication. If you have older management servers that support only DCE communication, there is additional queue for each of these servers.

You can configure options for these queues using the Server Configuration dialog.

To configure forwarding options for server-based flexible management

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog appears.
2. Click Namespaces , and then click Server-based Flexible Management .
3. Select the Expert mode check box. The full list of values appears.
4. Configure the queue sizes and retry intervals. The following table lists the values to configure.

Value	Description
Forwarding queue size warning threshold	When the size of a queue exceeds this threshold, the management server creates a warning event in the event log. Specify a warning threshold in megabytes. The default is 40 MB.
Forwarding queue size maximum	When the size of a queue exceeds this threshold, the management server creates an error event in the event log. The management server does not add to the queue until the queue size falls below the queue maximum setting. The destination management server or management servers for this queue do not receive messages, message operations, or action responses that occur while the queue size exceeds this threshold. Specify a queue size maximum in megabytes. The default is 50 MB.
Forwarding retry interval	This value specifies the number of seconds to wait before trying to reconnect to an unreachable management server. Specify an interval in seconds. The default is 60 seconds.

NOTE:
The management server creates entries in the event log that relate to queue sizes and unreachable management servers. You can monitor event logs on the management server with the self-manager policy VP_SM-Server_EventLogEntries.

5. *Optional.* Configure the message operations that you want this management server to forward and accept. The following table lists the values to configure.

Operations to be forwarded	This setting configures which types of message operations to forward to other management servers. For example, you can configure the management server not to forward message acknowledgements. You configure this setting with a bit mask consisting of the possible values below: 1 Acknowledge 2 Unacknowledge 4 Own 8 Disown 16 Severity change 32 Message counter change 64 Annotation change 128 Text change 256 Action response 512 Action state change Add all the operations that you want the server to forward. The default is 1023, which means that the management server forwards all operations.
Forwarded operations to be accepted	This setting configures which types of message operations to accept from other management servers. For example, you can configure the management server not to accept message acknowledgements. You configure this setting with a bit mask consisting of the possible values below: 1 Acknowledge 2 Unacknowledge 4 Own 8 Disown 16 Severity change 32 Message counter change 64 Annotation change 128 Text change 256 Action response 512 Action state change Add all the operations that you want the server to accept. The default is 1023, which means that the management server accepts all operations.

6. Click **Apply** .

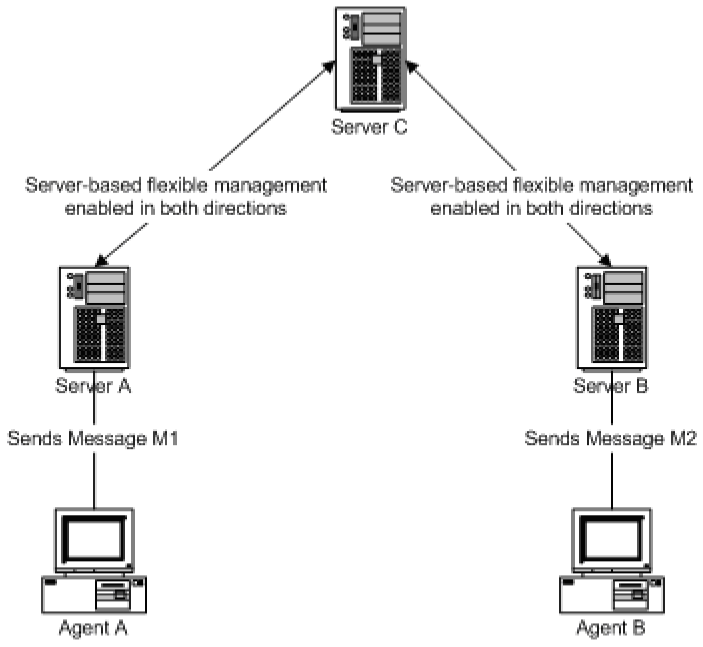
Related Topics:

- Configure communication protocols for server-based flexible management
- Self-management policies

Configure duplicate suppression for server-based flexible management

When duplicate suppression and server-based flexible management are used together, inconsistencies between the servers may arise. Most of the inconsistencies can be resolved by configuring the duplicate suppression on each server in exactly the same way. However, there are some configurations possible, when inconsistencies can only be resolved by disabling duplicate suppression for forwarded messages or changing the forwarding configuration.

Scenario 1


Server C
Server-based flexible management enabled in both directions
Server-based flexible management enabled in both directions
Server A
Server B
Sends Message M1
Sends Message M2
Agent A
Agent B

Server A forwards Message M1 to Server C.
Server B forwards Message M2 to Server C.
Message M1 arrives first at Server C.

Message M2 is recognized as a duplicate to M1 and is being attached to M1 as a duplicate annotation and the duplicate count is being increased (or depending on the configuration just as a duplicate count increase).

Inconsistency 1

1. The duplicate count of M1 on Server C is incremented.
2. Server C forwards the duplicate count of M1 to Server A and Server B.

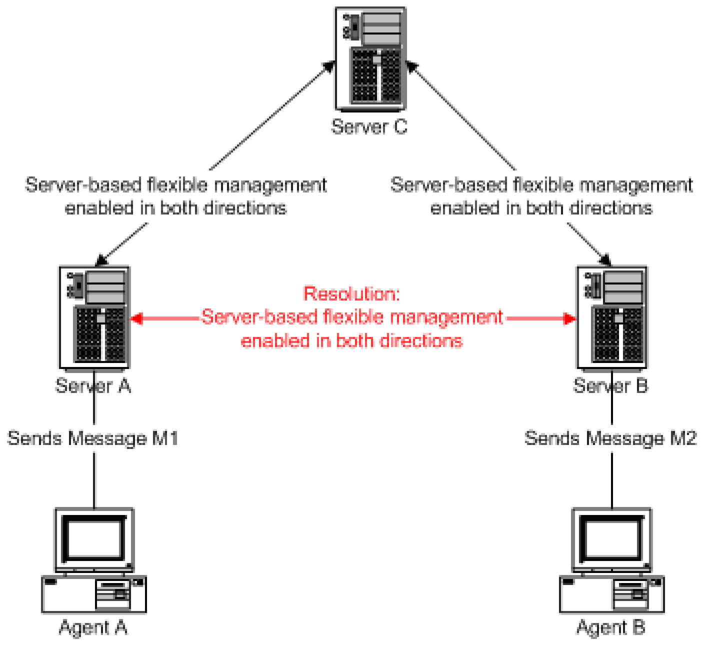
3. Server A increments the duplicate count of M1.
4. Server B discards the operation as it does not have message M1.

Inconsistency 2

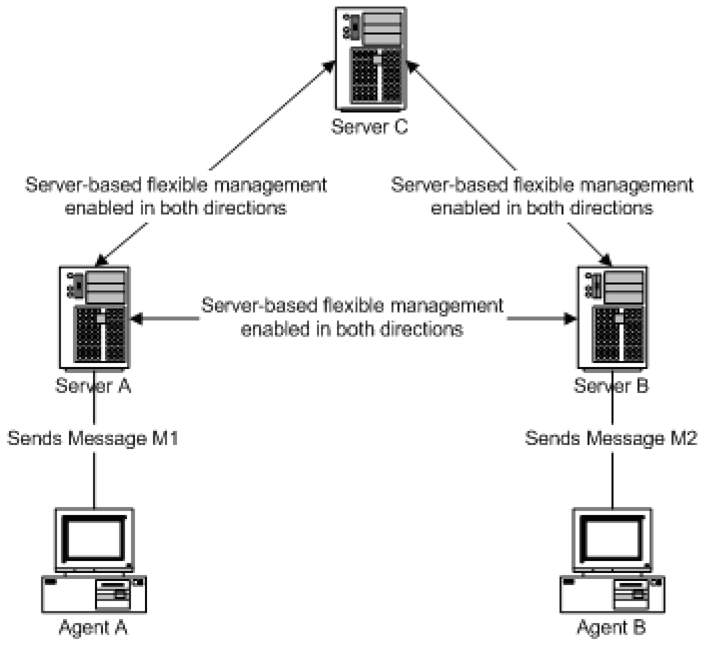
1. An Administrator on Server B acknowledges Message M2.
2. This operation is being forwarded to Server C. Server C does not know about a Message M2, because it was discarded by duplicate suppression.
3. The operation is being discarded without any action.
4. The original message M1 still exists as well as the duplicate annotation created by M2.

Resolution for both inconsistencies

Introduce bi-directional forwarding between Server A and Server B. Configure duplicate suppression on each server in the same way.


Server C
Server-based flexible management enabled in both directions
Server-based flexible management enabled in both directions
Resolution: Server-based flexible management enabled in both directions
Server A
Server B
Sends Message M1
Sends Message M2
Agent A
Agent B

Scenario 2


Server C
Server-based flexible management enabled in both directions
Server-based flexible management enabled in both directions
Server-based flexible management enabled in both directions
Server A
Server B
Sends Message M1
Sends Message M2
Agent A
Agent B

Server A forwards Message M1 to Server C and Server B.
Message M2 arrives on Server B. It is NOT discarded as a duplicate.
Server B forwards Message M2 to Server C and Server A.
Message M2 is recognized on Server C and Server A as a duplicate. Server C and Server A forward the duplicate count change operation to Server B, which increments the duplicate count of Message M1. Message M2 however is still in the message browser.

Inconsistency

Server B has Message M2 twice, once as real message and again as a duplicate annotation (or depending on the configuration just as a duplicate count increase).

Resolution

Configure the duplicate suppression on all servers the same way. Then Message M2 gets discarded already on Server B and all servers have a consistent view in the message browser.

To disable duplicate suppression for forwarded messages

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog appears.

2. Click Namespaces , and then click Server-based Flexible Management .

3. Set the value of Duplicate detection on forwarded msgs to false. This disables duplicate message for forwarded messages. This may help to resolve inconsistencies caused by duplicate suppression, but

may increase the number of messages in the message browser.

(Setting this value to true may help reduce the number of messages in the message browser, but has a small performance impact.)

4. Click Apply .
5. Complete the same

Related Topics:

- Duplicate message suppression

Configure the character set for server-based flexible management

By default, HPOM for Windows uses the HTTPS protocol to communicate with other management servers securely. HPOM for Windows always uses the UTF-8 character set internally and for HTTPS-based communication.

HPOM for Windows also supports the UTF-8 character set for DCE-based communication. However, if you need to forward messages to HPOM for UNIX management servers that support only the DCE protocol, you must configure the HPOM for Windows management server to use a different character set.

Before you configure the character set for server-based flexible management, consider the following details:

- The character set must support all characters that may occur in the forwarded messages, message operations, and action responses. For example, it is not possible to forward Japanese characters using the ascii character set.
- An HPOM for UNIX management server can only receive DCE-based communications in the specific character set that the HPOM for UNIX administrator configures it to use.
- HPOM for Windows can receive DCE-based communications in any character set that HPOM for UNIX supports. HPOM for Windows converts the characters to UTF-8 on arrival for use internally. Therefore, HPOM for Windows can receive DCE-based communications from several HPOM for UNIX servers that each use different character sets.
- An HPOM for Windows management server uses the same character set for all DCE-based communications with other management servers (including both HPOM for Windows and HPOM for UNIX management servers). Therefore it is not possible to use DCE-based communication with several HPOM for UNIX servers that each use different character sets.

To configure the character set for server-based flexible management

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog appears.
2. Click Namespaces , and then click Server-based Flexible Management .
3. Set the value of Character code set of target servers . This value configures the character set to which the management server converts messages, message operations, and action responses before forwarding them to other management servers. Select one of the following character sets:
   - ascii
   - eucJP
   - eucKR

    - gb2312
    - iso88591
    - iso885915
    - iso88592
    - iso88595
    - roman8
    - sjis
    - utf8 (default)

4. Click Apply .

Related Topics:

- Configure communication protocols for server-based flexible management

Configure communication protocols for server-based flexible management

By default, with HPOM for Windows 8.00 and above, the management server uses the HTTPS protocol to communicate with other management servers securely. If you need to forward messages to management servers that support only the DCE protocol, you can change this default, or set exceptions.

To configure communication protocols for server-based flexible management

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog appears.
2. Click Namespaces , and then click Server-based Flexible Management .
3. *Optional.* Set the value Default protocol to DCE. The management server uses DCE-based message forwarding and synchronization for all management servers that are not listed in Protocol exceptions .
4. *Optional* . Set the value of Protocol exceptions to contain a comma-separated list of management server names, for which you need to use the opposite of the Default protocol .

   For example, if most of your management servers support HTTPS, leave Default protocol set to HTTPS. List the older management servers that support only DCE in Protocol exceptions .

Related Topics:

- Configure the character set for server-based flexible management

Configure HPOM for UNIX compatibility mode

NOTE:
With HPOM for UNIX 7.24 or 8.12 and higher, HPOM for UNIX compatibility mode is not necessary.

Older versions of HPOM for UNIX impose restrictions on server-based flexible management. The restrictions are fixed with the patches HPOM for UNIX 7.24 (PHSS_32403) and HPOM for UNIX 8.12 (PHSS_32820). If you want to use server-based flexible management with older HPOM for UNIX management servers , you must turn on HPOM for UNIX compatibility mode.

Older versions of HPOM for UNIX cannot handle user names with a length of more than 20 characters, so forwarded messages and operations that contain longer user names will be discarded by the HPOM for UNIX management server. HPOM for Windows user names are built from the domain name and user name, and so they can easily exceed this limit.

Older versions of HPOM for UNIX are not aware of user rights on the HPOM for Windows management server, so they do not allow an HPOM for Windows administrator to force a disown operation on a message that another user owns.

If you configure HPOM for UNIX compatibility mode, the HPOM for Windows user names are reduced to the first 20 characters. For disown operations, the user name of the owning user is always sent with the operation, even if the operation is performed by an HPOM for Windows administrator.

HPOM for UNIX compatibility mode causes limitations for the handling of messages on multiple HPOM for Windows management servers. For example, if a user owns a message on one HPOM for Windows management server, the management server reduces the user name to 20 characters when it forwards the own operation to a second HPOM for Windows management server. When the same user tries to change the message state on the second HPOM for Windows management server, an error occurs. The error states that the user is not the owner of this message, because the second HPOM for Windows management server cannot map the shortened name to the full user name.

To configure UNIX compatibility mode

1. In the console tree, right-click **Operations Manager** , and then click **Configure→Server...** . The Server Configuration dialog appears.
2. Click **Namespaces** , and then click **Server-based Flexible Management** .
3. Select the **Expert mode** check box. The full list of values appears.
4. Set the value of **HPOM for UNIX compatibility mode** to true.
5. Click **Apply** .

Security for server-based flexible management

Security for message communication between management servers is the same as that provided for the agent to server communication. By default, HPOM for Windows management servers use the HTTPS protocol to communicate with other management servers securely. If you need to forward messages to management servers that support only the DCE protocol, the communication is less secure.

NOTE:
HPOM for Windows cannot communicate with an HPOM for UNIX server that runs Advanced Network Security (ANS).

By default, an HPOM for Windows management servers expects every message that it receives from another management server to contain a known agent ID. You can configure several settings that relate to agent ID checks.

To configure security for server-based flexible management

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box appears.
2. *Optional.* Click Namespaces , and then click Message Action Server Message Filter . A list of values appears. The following table lists the values to configure.

Allow actions in forwarded messages with no agent ID	To increase security, set this value to false so that the management server removes actions from forwarded messages that have an empty agent ID field. Most messages contain the agent ID that uniquely identifies the node that sends the message. The management server checks that the agent ID is valid for each message. However, HPOM for UNIX deployed DCE agents that have never received a policy from an HPOM for Windows management server send messages that have an empty agent ID field. By default, the management server removes all actions from this kind of message, except for forwarded messages.
Disable agent ID check for forwarded messages	This value configures whether the message filter skips the agent ID check for messages that it receives from another management server. If this value is false and the management server receives a message that contains an unknown agent ID but a known node name, the management server contacts the node to check the agent ID. If the management server cannot reach the node due to a firewall, the request for the agent ID takes several seconds to timeout. To increase performance, set this value to true so the management server skips this check.

Ignore empty agent ID on proxy messages forwarded from HPOM for UNIX	This value configures whether the message filter allows proxy messages without agent IDs (for example, SNMP traps), which HPOM for UNIX management servers can forward. If you set this value to true, the message filter does not discard the proxy messages with empty agent IDs that HPOM for UNIX forwards.

3. Click Apply .

Related Topics:

- Configuring HTTPS communication through firewalls
- Configure communication protocols for server-based flexible management

Migrate from ForwardToVP-based forwarding

Previous versions of HPOM for Windows use agent policies on the management server to forward messages to other management servers:

- A Windows Management Interface (WMI) policy creates messages with the message type ForwardToVP.
- A flexible management policy forwards messages that have the message type ForwardToVP.
- Another WMI policy forwards ownership changes and acknowledgements.

Server-based flexible management offers several advantages:

- You can keep messages up to date on all management servers by forwarding more message operations, for example changes of severity, changes to annotations, and action responses.
- The object attribute of the forwarded message remains unchanged. It is not overwritten with the originating node name, as is the case with the ForwardToVP-based method.
- You can forward messages, message operations, and action responses to multiple management servers.
- The message action server forwards the messages, so the load on the management server's agent is much smaller.
- Server-based flexible management is faster and more reliable, especially if there is a high load on the management server.
- You can forward all types of messages, not just messages of with the message type ForwardToVP.

If you use ForwardToVP-based message forwarding, you can stay with that solution or manually migrate to server-based flexible management. Server-based flexible management is the recommended message forwarding solution. ForwardToVP-based message forwarding is only available for backward compatibility.

To migrate from ForwardToVP-based message forwarding to server-based flexible management

1. Copy the content of the agent-based flexible management policy that to forwards all messages of the type ForwardToVP to other management servers. Use it to create a server-based flexible management policy.
2. In the server-based flexible management policy, remove the SECONDARYMANAGERS section. The section is not needed, because each management server maintains an internal list of the other management servers to which it has forwarded each message. The management server forwards message operations and action responses to only the management servers in this list.
3. In the policy, remove the ACTIONALLOWMANAGERS section. The section is not needed for server-based flexible management.

4. In the MSGTARGETRULES section, remove the MSGTARGETRULE that forwards all messages to the source server itself.
5. In the MSGTARGETRULES section, modify the MSGTARGETRULE that forwards messages to destination servers. Replace MSGTARGETRULECOND MSGTYPE "ForwardToVP" with the conditions (if any) from the WMI policy that creates messages with the message type ForwardToVP.
6. The keywords ACKNONLOCALMGR, MSGCONTROLLINGMGR, and NOTIFYMGR are not supported for server-based flexible management policies. If your existing policy contains these keywords, remove them.
7. Remove the WMI policy that creates messages with the message type ForwardToVP, so that it is no longer deployed to the agent on the management server.
8. Remove the agent-based flexible management policy that forwards messages, so that it is no longer deployed to the agent on the management server.
9. Remove the WMI policy that forwards ownership changes and acknowledgements.
10. Configure any prerequisites for server-based flexible management and then deploy the new server-based flexible management policy to the management server.

Related Topics:

- Configure prerequisites for server-based flexible management
- Syntax of server-based flexible management policies

Configure firewall and NAT for DCE-based server communication

Server-based flexible management allows message forwarding and synchronization between management servers that are separated by a firewall, a network address translation (NAT) router, or both.

NOTE:
In HPOM for Windows version 8.10, server-based flexible management uses the HTTPS communication protocol. If you have two HPOM for Windows version 8.10 management servers, you can configure server-to-server communication through a firewall in the same way that you configure agent communication through a firewall. For more information, see Configuring two-way communication .

By default, HPOM for Windows 8.10 uses the HTTPS protocol for server-to-server communication. If you need to forward messages to management servers that support only the DCE protocol, you can change this default. For more information, see Configure communication protocols for server-based flexible management .

The configuration information provided here is for server-based flexible management between HPOM for Windows management servers using DCE communication protocol.

For more information, see the *Firewall Concepts and Configuration Guide* , which is available in the `/Documentation/Whitepapers` folder on the HPOM installation media and also from the HP Software Product Manuals web page.

Firewall and NAT basics

A firewall is a router system between two or more subnets. In addition to the routing, the firewall also filters all communication. Only packets that pass at least one filter rule are allowed to pass the firewall. All other packets are discarded. A filter rule usually consists of the following items:

- A protocol type, such as TCP, UDP, or ICMP
- A direction ("inside → outside" or "outside → inside")
- A source port
- A destination port

Instead of a specific port, you can give a port range. In a typical remote communication, a client uses the source port to connect to a server, which is listening on the destination port on a remote system. For firewall configuration, it is important to know which system initiates the communication (client) and which receives communication requests (server), so that the firewall rules can be set up accordingly.

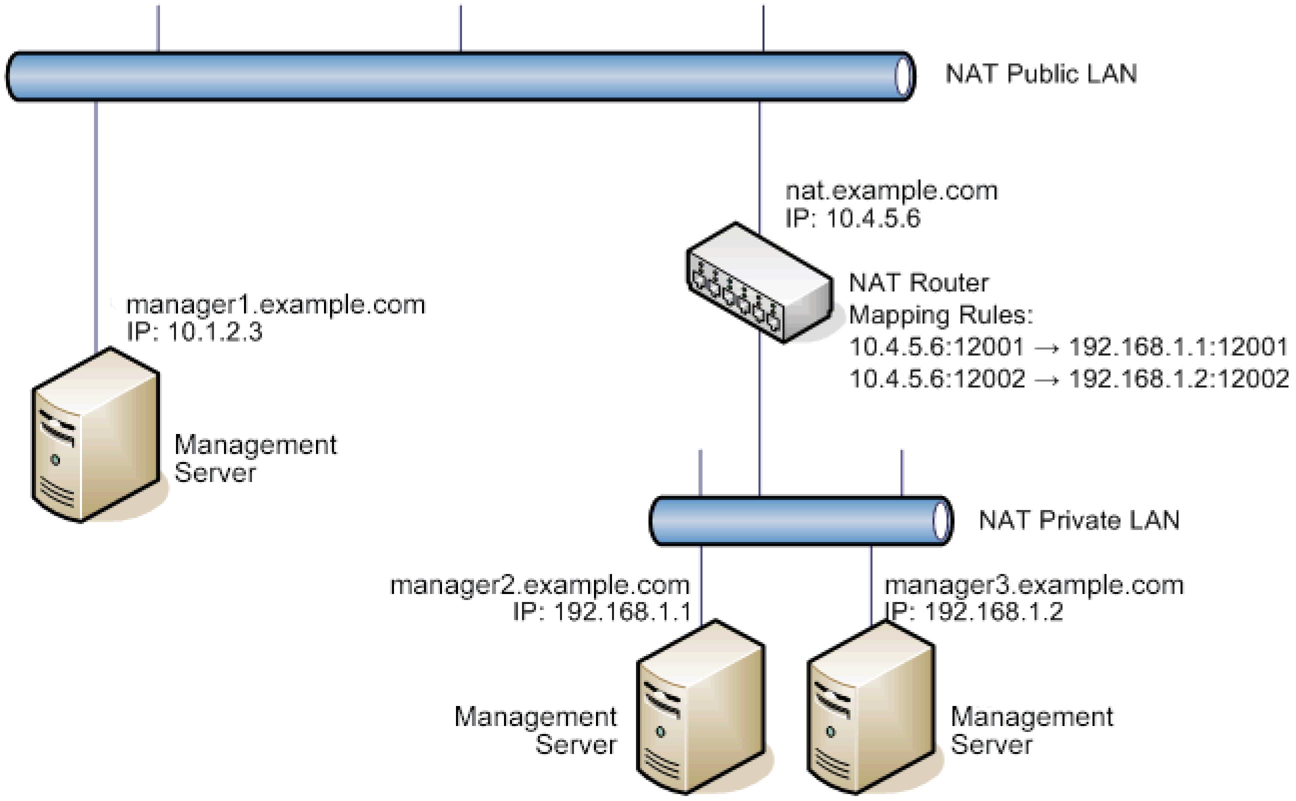
A Network Address Translation (NAT) router connects two subnets, a public one and a private one. The NAT router has an IP address on the public subnet and translates this public IP address to one or more IP addresses on the private subnet, based on a given set of rules. The private IP addresses are not directly accessible on the public subnet, so an IP packet from the public subnet has to be rewritten to the private subnet by the NAT router, exchanging the public IP address with an IP address of the private subnet.

There are two kinds of NAT:

- Basic NAT (static NAT): translates each public IP address to a private IP address; so for each private IP address there needs to be one public IP address.
- Port Address Translation (PAT): has only a single public IP address and maps it to multiple private IP addresses based on the ports used.

Configuration scenario

This example explains the necessary configuration steps based on a PAT scenario. It should be easily possible to derive the necessary configuration steps for any real server-based flexible management environment from this example. The example shows one management server (manager1.example.com) directly connected to a LAN; the other two management servers are hidden behind a NAT router. The configuration task is to set up server-based flexible management between manager1.example.com and manager2.example.com.



Server-based flexible management works using RPC requests from the source management server (acting as DCE RPC client) to the message receiver on the target management server (acting as DCE RPC server). With these RPC requests, HPOM for Windows forwards messages and message operations. To get the port number of the message receiver on the target management server, the client has to request it from the DCE RPC endpoint mapper on the fixed port 135 of the target server system.

As RPC requests from inside a NAT or firewall (in this example: from manager2.example.com to manager1.example.com) are usually allowed without special configuration, this example concentrates on the configuration for RPC requests from manager1.example.com to manager2.example.com. For environments where both servers are behind a firewall or inside separate NATs, you have to repeat the configuration done for RPC requests from manager1.example.com to manager2.example.com also for the other direction.

To configure firewall communication using PAT

Follow these steps to configure firewall communication using PAT.

1. Provide the correct target server name in the server-based flexible management policy. If you want to forward messages and message operations to a management server hidden by a NAT, you must specify the public network name of the NAT router in the policy, not the hidden name of the management server. In the example illustrated above, the server-based flexible management policy on manager1.example.com contains nat.example.com as target server in the MSGTARGETMANAGERS sections for all messages that should be forwarded to manager2.example.com.
2. Force the target management server to use a fixed port for the message receiver RPC server, so that you can open this port in the firewall respectively map this port in the NAT configuration.
   a. In the console tree on the target management server, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box appears.
   b. Click Namespaces , and then click Message Action Server General . A list of values appears.
   c. Set the value of DCE RPC server port to the port number you want to use.
3. Restart the RPC server on the target management server (manager2.example.com) by restarting the OvEpMessageActionServer service.

Verify RPC port usage

You can check the RPC server port usage using the opcrpccp utility which is located in `<InstallDir>\Installed Packages\{790C06B4-844E-11D2-972B-080009EF8C2A}\contrib\OpC\opcrpccp.exe` . The following command lists all RPC servers on the local system:

```
# opcrpccp show mapping
```

A list having many entries similar to the following will be printed:

```
<object> nil
<interface id> 6d63f833-c0a0-0000-020f-887818000000,7.0
<string binding> ncadg_ip_udp:15.136.123.62[12001] <-- port used
<annotation> OvEpRpcDataRcvr
```

Next, open the configured port in the firewall and respectively map this port in the NAT configuration to the target management server system. Make sure that port 135 for the DCE RPC endpoint mapper is opened in the firewall respectively mapped in the NAT configuration to the target management server system, so that the RPC client on the source management server can request the message receiver port of the target management server system.

Communication without DCE RPC endpoint mapper

In many environments, opening the DCE RPC endpoint mapper port 135 in the firewall is considered a security risk. Using a PAT (Port Address Translation) router, it may also not be possible to map port 135 to a hidden system because the PAT router needs port 135 for its own purposes. Server-based flexible management can do without port 135.

For this purpose, the RPC client on the source management server (manager1.example.com) needs to know on which port the message receiver on the target management server (manager2.example.com) is listening. The client can get this information from a port configuration file instead of from the DCE RPC endpoint mapper.

1. Specify the location of the port configuration file on the source management server:
   a. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box appears.
   b. Click Namespaces , and then click Message Action Server General . A list of values appears.
   c. Configure the value DCE RPC server port specification file with the full path of the port configuration file. For example: `C:\restricted\ports.txt`

      Of course you can also use other file locations and names for the port configuration file. For security reasons you should restrict the file access rights for this file (especially write).
2. Create the port specification file and specify which port should be used for a given target server. If you want to access a management server hidden by a NAT, you have to specify the network name of the NAT router as the node name. An example of the server port specification file for the given scenario is shown below:

```
#
# SelectionCriteria SrvType Port Node
# -------------------------------------------------------------------
NODE_NAME opcmsgrd 12001 nat.example.com
NODE_NAME opcdistm 12001 nat.example.com
```

NOTE:
`somename.example.com` matches <*>somename.example.com<*>

`^somename.hp.com matches$` matches somename.hp.com exactly.

Restrictions

In an environment with Port Address Translation (PAT), only one management server in the private network of the PAT router can be accessed from a management server in the public (outside) network. If you use the

RPC endpoint mapper, then only the one management server is accessible, to which port 135 of the PAT router is mapped. If you turned off the use of the RPC endpoint mapper, then only the management server is accessible, to which the port is mapped that is specified in the port specification file.

Related Topics:

- DCE RPC Communication without using the Endpoint Mapper

Create a server-based flexible management policy

Server-based flexible management enables you to forward messages , message operations, and action responses from one management server to another. To configure the types of message that you want to forward, and the management servers to which you want to forward the messages, you create a server-based flexible management policy . You then deploy this policy to the management server.

To subsequently stop the management server forwarding messages and message operations, disable the server-based flexible management policy.

To create a server-based flexible management policy

1. In the console tree, under Policy management → Server policies grouped by type , right-click Server-based Flexible Management and then click New→Policy . The server-based flexible management policy editor appears.

2. In the General tab, type the server-based flexible management policy. The following topics provide more information on the policy syntax:

   - Example server-based flexible management policies
   - Syntax of server-based flexible management policies

   TIP:
   If you are modifying a large policy, you can split the text area vertically and horizontally by dragging the split controls. (The split controls are at the top of the vertical scroll bar and at the left of the horizontal scroll bar.)

   NOTE:
   In HP Operations Manager for Windows version 7.5, the `MsgForwarding.ini` file controls server-based flexible management. This file is in the folder `<`*`data_dir`*` >\shared\conf\MsgActSrv\` on the management server. If you have an existing `MsgForwarding.ini` file, you can copy its text into the policy editor. The syntax is the same.

   Optionally, you can remove any SECONDARYMANAGER entries from the new policy to leave the SECONDARYMANAGERS section empty. The entries are no longer needed, because the management server maintains an internal list of the management servers to which it has forwarded each message. The management server forwards message operations and action responses to only the management servers in this list.

3. *Optional.* To check the policy's syntax, click Check Syntax . A message appears, which gives details of any errors.

4. Save the policy, and then deploy it to the management server that you want to configure.

Related Topics:

- Server-based flexible management policy examples
- Syntax of server-based flexible management policies
- Save a policy
- Deploy a policy or policy group

Syntax of server-based flexible management policies

A management server compares every incoming message to the server-based flexible management policy. If the message attributes match the rules, the management server forwards the message to the other management servers specified in the policy. A policy is designed according to syntax rules. The flexible management policy syntax is in EBNF (Extended Backus-Naur Form). The server-based flexible management policy editor enables you to check that a policy has valid syntax.

NOTE:
Management servers evaluate only the first matching MESSAGETARGETRULE in the policy. For example, you have a message target rule A with the condition to match on critical messages and the rule to forward these kinds of messages to server X. You have another message target rule B, with the condition to match on messages with application "test" and the rule to forward these kinds of messages to server Y.

If a critical message arrives with application "test," then this message matches rules A and B, but the management server evaluates only the first matching rule and so forwards the message only to server X and not to server Y.

If you do not want this behavior, you must write an additional message target rule that matches only on critical messages with application "test" and forward these messages to server X and server Y; then you must place this rule before rule A.

NOTE:
HPOM for UNIX allows time templates but HPOM for Windows does not. The syntax below does not contain time templates, although you will find them in the HPOM for UNIX example files.

EBNF notation

```
<configfile>
                ::= <epsilon>
                | RESPMGRCONFIGS <respmgrconfigs>
<respmgrconfigs>
                ::= <epsilon>
                | <respmgrconfigs> RESPMGRCONFIG DESCRIPTION <string> <respmgrconds>
<respmgrconds>
                ::= SECONDARYMANAGERS <secondmgrs> ACTIONALLOWMANAGERS <actallowmgrs>
                | SECONDARYMANAGERS <secondmgrs> ACTIONALLOWMANAGERS <actallowmgrs> MSGTARGETRU
                | MSGTARGETRULES <msgtargetrules>
<secondmgrs>
                ::= <epsilon>
                | <secondmgrs> SECONDARYMANAGER NODE <node>
                | <secondmgrs> SECONDARYMANAGER NODE <node> DESCRIPTION <string>
<actallowmgrs>
                ::= <epsilon>
                | <actallowmgrs> ACTIONALLOWMANAGER NODE <node>
                | <actallowmgrs> ACTIONALLOWMANAGER NODE <node> DESCRIPTION <string>
<msgtargetrules>
                ::= <epsilon>
                | <msgtargetrules> MSGTARGETRULE DESCRIPTION <string> <msgtargetrule>
<msgtargetrule>
                ::= MSGTARGETRULECONDS <mtrconditions> MSGTARGETMANAGERS <msgtargetmgrs>
                | MSGTARGETRULECONDS <mtrconditions> MSGTARGETMANAGERS <msgtargetmgrs>
<mtrconditions>
                ::= <epsilon>
                | <mtrconditions> MSGTARGETRULECOND DESCRIPTION <string> <mtrcond>
<mtrcond>
                ::= <epsilon>
                | <mtrcond> SEVERITY <severity>
                | <mtrcond> NODE <nodelist>
                | <mtrcond> APPLICATION  <string>
                | <mtrcond> MSGGRP <string>
                | <mtrcond> OBJECT <string>
                | <mtrcond> MSGTYPE <string>
                | <mtrcond> TEXT <string>
                | <mtrcond> SERVICE_NAME <string>
                | <mtrcond> MSGCONDTYPE <msgcondtype>
<msgtargetmgrs>
                ::= <epsilon>
                | <msgtargetmgrs> MSGTARGETMANAGER TIMETEMPLATE "$OPC_ALWAYS" OPCMGR <node>
<severity>
                ::= Unknown | Normal | Warning | Minor | Mayor | Critical
<msgcondtype>
                ::= Match | Suppress
<nodelist>
                ::= <node>
                | <nodelist> <node>
<node>
                ::= IP <ipaddress>
                | IP <ipaddress> <string>
<string>
                ::= "any alphanumeric string"
<ipaddress>
                ::= <digits>.<digits>.<digits>.<digits>
```

NOTE:

For detailed descriptions of the attributes used in the syntax described here, see the help topic Keywords for flexible management policies .

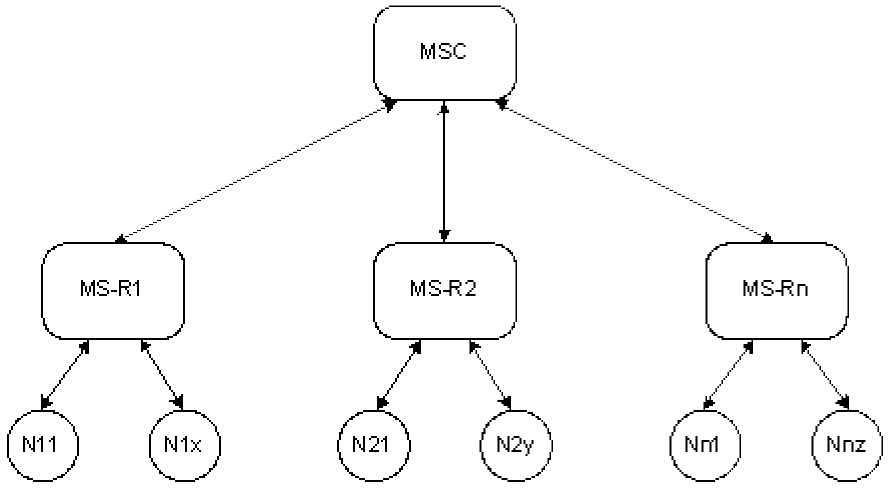
NOTE:
The keywords `ACKNONLOCALMGR` , `MSGCONTROLLINGMGR` , and `NOTIFYMGR` are not supported for server-based flexible management policies.

NOTE:
The syntax check allows the keywords `SECONDARYMANAGERS` and `ACTIONALLOWMANAGERS` in server-based flexible management policies for backwards compatibility with `MsgForwarding.ini` files. However, these keywords have no effect in server-based flexible management policies.

Example server-based flexible management policies

Hierarchy

The figure below shows a hierarchy scenario of managed nodes and management servers.



This hierarchy scenario consists of managed nodes Nxy that report to their regional management servers MS-Rx. Regional management servers control managed nodes in a specific region (for example, a LAN). Operators managing a region are responsible for keeping the managed nodes up and running. All critical problems and problems originating from the regional management servers are reported to the central management server (MSC). Operations are synchronized between regional management servers and central management server.

Regional management server

(see "Hierarchy – Regional Server" Server Policy in the "Samples\Server-based flexible management" policy group)

```
#
# Example hierarchy forward configuration for HPOM for Windows Server-based flexible management
# Forwarding from regional HPOM for Windows management server MS-Rx to central HPOM for Windows
# - Forward all messages from MS-Rx to MSC
# - Forward all critical messages from managed nodes to MSC
# - Synchronize operations with MSC
#
# Note: Adapt the server names and deploy this policy
#       on the regional management server (MS-Rx) if you want to use it.
```

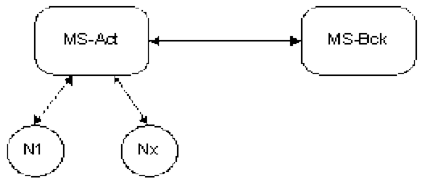
```
#

TIMETEMPLATES
# none

RESPMGRCONFIGS
      RESPMGRCONFIG DESCRIPTION "MS-Rx - Hierarchical Forward Configuration"
      SECONDARYMANAGERS
      ACTIONALLOWMANAGERS

      MSGTARGETRULES
            MSGTARGETRULE DESCRIPTION "messages from node MS-Rx"
                  MSGTARGETRULECONDS
                        MSGTARGETRULECOND
                        DESCRIPTION "messages from node MS-Rx"
                        NODE IP 0.0.0.0 "MS-Rx"
                  MSGTARGETMANAGERS
                        MSGTARGETMANAGER
                        TIMETEMPLATE "$OPC_ALWAYS"
                        OPCMGR IP 0.0.0.0 "MSC"

            MSGTARGETRULE DESCRIPTION "critical messages"
                  MSGTARGETRULECONDS
                        MSGTARGETRULECOND
                        DESCRIPTION "critical messages"
                        SEVERITY Critical
                  MSGTARGETMANAGERS
                        MSGTARGETMANAGER
                        TIMETEMPLATE "$OPC_ALWAYS"
                        OPCMGR IP 0.0.0.0 "MSC"
```

Central management server

(no server-based flexible management configuration required on the Central HPOM for Windows Management Server MSC)

Backup server

The figure below shows a backup server scenario of managed nodes and management servers.



This backup server scenario consists of managed nodes Nx that report to their management server MS-Act. This management server reports all messages to the backup management server MS-Bck and synchronizes all operations to the backup management server. This way MS-Act and MS-Bck are completely kept in synch with regards to messages and the message states. MS-Bck is set up as secondary manager on the managed nodes Nx so that it can get control over them if it needs to.

If MS-Act cannot perform its tasks as management server any longer (for example due to a hardware failure) then MS-Bck can take over its tasks immediately by telling the managed nodes to switch their management server to MS-Bck, for example using the command "opcragt –primmgt".

Actual management server

(see Backup – Actual Server" Server Policy in the "Samples\Server-based flexible management" policy group)

```
#
# Example backup forward configuration for HPOM for Windows Server-based flexible management
# Forwarding from actual HPOM for Windows management server MS-Act to backup HPOM for Windows m
# - Forward all messages from MS-Act to MS-Bck
# - Synchronize operations between MS-Act and MS-Bck
#
# Note: Adapt the server names and deploy this policy
#       on the actual management server (MS-Act) if you want to use it.
#

TIMETEMPLATES
# none

RESPMGRCONFIGS
      RESPMGRCONFIG DESCRIPTION "MS-Act - Backup Forward Configuration"
      SECONDARYMANAGERS
      ACTIONALLOWMANAGERS

      MSGTARGETRULES
            MSGTARGETRULE DESCRIPTION "forward all messages to MS-Bck"
                  MSGTARGETRULECONDS
```

```
                        MSGTARGETMANAGERS
                              MSGTARGETMANAGER
                              TIMETEMPLATE "$OPC_ALWAYS"
                              OPCMGR IP 0.0.0.0 "MS-Bck"
```

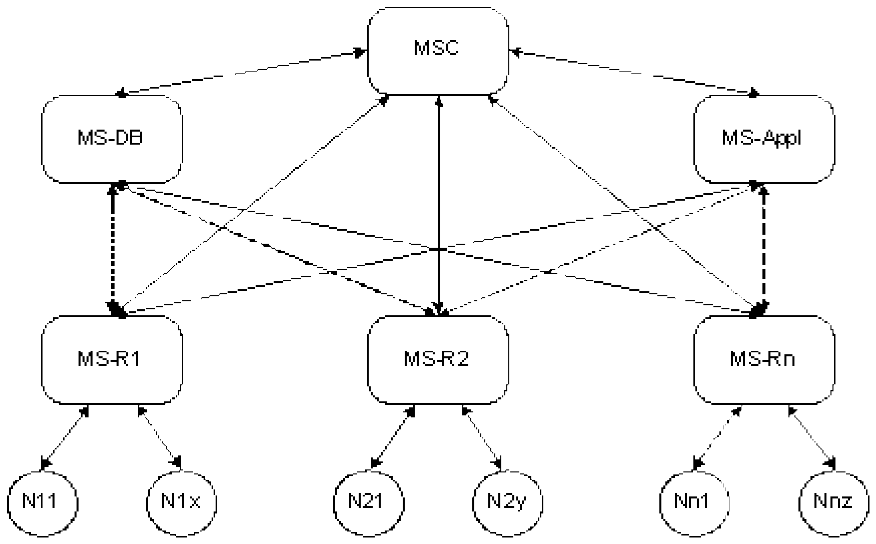
Backup management server

(see Backup – Backup Server" Server Policy in the "Samples\Server-based flexible management" policy group)

```
#
# Example backup forward configuration for HPOM for Windows Server-based flexible management
# Forwarding from backup HPOM for Windows management server MS-Bck to actual HPOM for Windows m
# - Forward all messages from MS-Bck to MS-Act
# - Synchronize operations between MS-Bck and MS-Act
#
# Note: Adapt the server names and deploy this policy
#       on the backup management server (MS-Bck) if you want to use it.
#

TIMETEMPLATES
# none

RESPMGRCONFIGS
      RESPMGRCONFIG DESCRIPTION "MS-Bck - Backup Forward Configuration"
      SECONDARYMANAGERS
      ACTIONALLOWMANAGERS

      MSGTARGETRULES
            MSGTARGETRULE DESCRIPTION "forward all messages to MS-Act"
                  MSGTARGETRULECONDS
                  MSGTARGETMANAGERS
                        MSGTARGETMANAGER
                        TIMETEMPLATE "$OPC_ALWAYS"
                        OPCMGR IP 0.0.0.0 "MS-Act"
```

Competence center

The figure below shows a competence center scenario of managed nodes and management servers.



This competence center scenario consists of managed nodes Nxy that report to their regional management servers MS-Rx. Regional management servers report database problems to the competence server MS-DB and application problems to the competence server MS-Appl. All critical problems and problems originating from the regional management servers and the competence servers are reported to the central management server (MSC). Operations are synchronized between regional, competence center management servers and central management server.

Regional management server

(see "Competence Center – Regional Server" Server Policy in the "Samples\Server-based flexible management" policy group)

```
#
# Example competence center forward configuration for HPOM for Windows Server-based flexible ma
# Forwarding from regional HPOM for Windows management server MS-Rx to competence center manage
# - Forward database messages to MS-DB
# - Forward application messages to MS-Appl
# - Forward critical messages to MSC
# - Synchronize operations with MS-DB, MS-Appl and MSC
#
# Note: Adapt the server names and deploy this policy
#       on the regional management server (MS-Rx) if you want to use it.
#


TIMETEMPLATES
# none
```

```
RESPMGRCONFIGS
      RESPMGRCONFIG DESCRIPTION "MS-Rx - Competence Center Forward Configuration"
      SECONDARYMANAGERS
      ACTIONALLOWMANAGERS

      MSGTARGETRULES
            MSGTARGETRULE DESCRIPTION "database messages"
                  MSGTARGETRULECONDS
                       MSGTARGETRULECOND
                        DESCRIPTION "database messages"
                        MSGGRP "DATABASE"
                  MSGTARGETMANAGERS
                        MSGTARGETMANAGER
                        TIMETEMPLATE "$OPC_ALWAYS"
                        OPCMGR IP 0.0.0.0 "MS-DB"

            MSGTARGETRULE DESCRIPTION "application appl"
                  MSGTARGETRULECONDS
                        MSGTARGETRULECOND
                        DESCRIPTION "application appl"
                        APPLICATION "appl"
                  MSGTARGETMANAGERS
                        MSGTARGETMANAGER
                        TIMETEMPLATE "$OPC_ALWAYS"
                        OPCMGR IP 0.0.0.0 "MS-Appl"

            MSGTARGETRULE DESCRIPTION "critical messages"
                  MSGTARGETRULECONDS
                        MSGTARGETRULECOND
                        DESCRIPTION "critical messages"
                        SEVERITY Critical
                  MSGTARGETMANAGERS
                        MSGTARGETMANAGER
                        TIMETEMPLATE "$OPC_ALWAYS"
                        OPCMGR IP 0.0.0.0 "MSC"
```

Database management server / application management server

(see "Competence Center – Competence Center Server" Server Policy in the "Samples\Server-based flexible management" policy group)

```
#
# Example competence center forward configuration for HPOM for Windows Server-based flexible ma
# Forwarding from competence center management servers MS-DB and MS-Appl to central HPOM for Wi
# - Forward critical messages to MSC
# - Synchronize operations with MS-R1, MS-R2, MS-Rn and MSC
#
# Note: Adapt the server names and deploy this policy
#       on the competence center management servers (MS-DB and MS-Appl) if you want to use it.
#
TIMETEMPLATES
# none

RESPMGRCONFIGS
      RESPMGRCONFIG DESCRIPTION "MS-DB - Competence Center Forward Configuration"
      SECONDARYMANAGERS
      ACTIONALLOWMANAGERS

      MSGTARGETRULES
            MSGTARGETRULE DESCRIPTION "critical messages"
                  MSGTARGETRULECONDS
                        MSGTARGETRULECOND
                        DESCRIPTION "critical messages"
                        SEVERITY Critical
                  MSGTARGETMANAGERS
                        MSGTARGETMANAGER
                        TIMETEMPLATE "$OPC_ALWAYS"
                        OPCMGR IP 0.0.0.0 "MSC"
```

Central management server

(no server-based flexible management configuration required on the Central HPOM for Windows Management Server MSC)

Configuring Management Servers

There are several ways to change a management servers' configuration. The following configuration options are available:

- Generic server configuration
- Duplicate message suppression
- Remote Action Security Policies
- Create a server-based flexible management policy
- Server-based MSI Policies

Generic server configuration

The server configuration dialog contains a generic server configuration tab, which enables you to configure many different aspects of a management server's configuration. This eliminates the need to manually configure values on a management server using registry editors or commands.

Related Topics:

- Change server configuration values
- Find help on server configuration values
- ovowconfigutil

Change server configuration values

The server configuration dialog enables you to change many different values, which control many different aspects of a management server's configuration.

TIP

Click Find to open a find dialog box where you can search for all or part of a value name in all namespaces.

To change server configuration values

1. In the console tree, right-click Operations Manager , and then click Configure→ Server... . The Server Configuration dialog appears.
2. Select a Namespace . A list of values appears. Each namespace contains a group of related values.
3. *Optional.* To see all available values, select Expert mode . Otherwise, the dialog shows only the most commonly used values.
4. Click the value that you want to change. Depending on the type of value, you can either type a new value, or select a value from a list that appears.
5. *Optional.* If a value appears in bold , the default value has been changed. To reset the value to its default, right-click the value, and then click Set to default .
6. If the value name ends with an asterisk (*), you must restart one or more management server components. Read the Description for the exact requirements.
7. Click OK .

Related Topics:

- Find help on server configuration values

Find help on server configuration values

The server configuration dialog enables you to change many different values, which control many different aspects of a management server's configuration. When you select a configuration value, a description appears in the dialog. More detailed online help is also available for many configuration values.

To find help on server configuration values:

1. In the Server Configuration dialog, click the Generic server configuration tab, and then select the appropriate Namespace .
2. *Optional.* To see all available values, select Expert mode .
3. Right-click the value that you need help on, and then click Help .

NOTE:
If the help menu item is unavailable, no appropriate help topics exist.

Duplicate message suppression

As messages arrive from various managed nodes, they are compared to existing messages on the management server. If a duplicate is found, the system increments the original message's duplicate count, discards the new message, and forwards notification of the change to the original message's duplicate count to all interested message browsers.

You can configure duplicate message suppression to store the most important attributes of the new message with the original message before the duplicate is discarded. A maximum of 10 duplicate annotations, plus 1 duplicate annotation containing the original message attributes, is stored by default with the original message. You can also configure the management server to add automatic action responses as annotations to original messages.

If you do not want to suppress message duplicates indefinitely, you can configure the management server to create a new message after a specific interval, or after a specific number of duplicates.

By default, the management server does not start remote automatic actions for duplicate messages, but you can change this behavior.

Message suppression and new policy creation and testing

When creating and testing new policy versions, you should do one of the following:

- Disable message suppression temporarily
- Acknowledge all messages which were created by the old policy version.

This prevents messages created by the new policy from being recognized as duplicates of the messages created by the old policy.

NOTE:
If server-based flexible management is enabled, all participating servers have to be configured in exactly the same way concerning message suppression to prevent inconsistencies between the servers.

Related Topics:

- Configure duplicate message suppression
- Specify duplicate message criteria
- Configure maximum suppression thresholds
- Configure automatic actions for duplicate messages

Configure duplicate message suppression

The message suppression tab in the server configuration dialog enables you to configure how the management server handles duplicate messages.

To configure duplicate message suppression

1. In the console tree, right-click Operations Manager, and then click Configure → Server... . The Server Configuration dialog appears.
2. Click the Message Suppression tab.
3. Check the Suppress and count duplicate messages check box. The number of duplicate messages appears in the message browser headline in the Duplicates column. Selecting this option can improve performance if a message storm occurs. Suppressing messages delays their arrival in the browser for the specified number of seconds and can prevent message storms. One message is sent to the browser and the count indicates whether a storm is in progress.
4. To specify the criteria for a duplicate message, click Message Fields to open the Duplicate Message Comparison Fields dialog box and use it to indicate your choice.
5. Specify the severity level, the message text, or both, to display in the browser using the options in the Display the severity of: list box. You can display the following options:
   - Severity and message text of the original message
   - Severity and message text of the most recent duplicate
   - Severity of the duplicate and the message text of the original message

   If messages are not matched by severity, message text, or both, (for example, if they contain a message key) you could have a duplicate with different severity or message text, or both. You would want to see the data of the most recent duplicate, for example, if the severity had changed from Major to Normal. You should not change the default Display the severity and the text of the most recent duplicate because some SPI-generated messages would cause the console to show the wrong severity and message text.
6. Delay duplicate message notification (in seconds): Specify the time interval at which the duplicate counter is refreshed in the message browser, rather than the real-time arrival of messages. This helps to control message storms.
7. Maximum number of duplicates cached before notification: Specify the number of cached duplicates to be consolidated on the server side before they are sent to the console. This setting is only functional when Delay duplicate message notification is set to a value higher than 0.

   Whenever the time interval for adding duplicates to the browser expires, or the maximum number of duplicates cached is reached, the duplicates are sent consolidated to the console. If you have specified

5 seconds, for example, and 1,000 duplicates, if 5 seconds is reached before 1,000 duplicate have arrived, the duplicates are posted anyway. If 1,000 duplicates arrive before the 5 second interval is up, they will be posted even though the interval has not been exceeded.

8. Duplicate Message Storage Settings: Check the box Store duplicate messages if you want to include the duplicate as a duplicate annotation in the Message Browser dialog box Duplicates tab.
9. Maximum number of duplicates to store: Enter a number in the field for the maximum number of duplicates you want to keep. The Unlimited choice is usually not recommended because the number could become extremely high.

Related Topics:

- Specify duplicate message criteria
- Configure maximum suppression thresholds
- Specify console data presentation properties

Specify duplicate message criteria

For a message to be considered a duplicate of an existing message, selected criteria must match those same criteria in the original message. By specifying characteristics such as node name, you can specify which criteria must be matched. By doing so, you can control the number of duplicate messages received in the message browser.

Messages with identical message keys are always considered duplicates. For messages without message keys, you can specify one or all of the following criteria that a message must meet to be considered a duplicate. By default, all options are selected.

- Severity
- Service ID
- Message Text
- Condition
- Message Group
- Node Name
- Application
- Object

To select the criteria a duplicate message must possess:

1. Check the check box for each criteria that must match the original message. Click Select All to check all boxes. All checked criteria or the Message Key (if available) will be used for duplicate message suppression. Click Clear All if the Message Key should be the only criteria for duplicate message suppression.
2. Click OK to close the dialog box and confirm your changes. When you return to the message browser, the options you checked will take effect.

Configure maximum suppression thresholds

If an original message remains unacknowledged, you may not want to suppress its duplicates indefinitely. You may want a new message after a specific interval, or after a specific number of duplicates. For example, you can configure the management server to suppress duplicate messages for a maximum of 24 hours. Alternatively, you can configure the management server to suppress a maximum of 20 duplicate messages.

You can also specify both thresholds, so that the management server creates a new message in either case.

To configure maximum suppression thresholds

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog appears.
2. Click Namespaces , and then click Message Suppression . A list of values appears.
3. *Optional.* Set the value of Maximum duplicate time interval . This value configures the number of minutes after the arrival of an original message that it is possible for another message to be its duplicate. Messages can only be duplicates of an original message if they arrive within this interval. Use 0 to specify an infinite interval.
4. *Optional.* Set the value of Maximum duplicate count . This value configures the maximum number of duplicate messages that an original message can have. Use 0 to specify an infinite number of duplicates.
5. Click Apply .

Related Topics:

- Configure duplicate message suppression

Configure automatic actions for duplicate messages

By default, if a duplicate message contains a remote automatic action request, the management server ignores this request. The remote automatic action runs for the original message only. However, if you want the action to run for duplicate messages as well, you can enable this. However, the remote action security policy may still block the remote actions.

NOTE:
Local automatic actions always run on the node for duplicate messages.

By default, if the management server discards a duplicate message, it also discards any automatic action response. This applies for both local and remote automatic actions. However, you can configure the management server to add automatic action responses as annotations to the original message.

To enable remote automatic actions for duplicate messages

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Message Suppression . A list of values appears.
3. *Optional.* Set the value of Enable automatic actions on duplicate messages to true. This configures the management server to start remote automatic actions on messages that are duplicates.
4. *Optional.* Set the value of Enable action annotations on duplicate messages to true. This configures the management server to annotate the original messages with responses to the actions of duplicate messages. This may have a small performance impact.
5. Click Apply .

Related Topics:

- Remote Action Security Policies
- Configure duplicate message suppression

Remote Action Security Policies

A node can run a remote automatic action on another node by sending a message to a management server . The message specifies the automatic action and a target node.

By default, any node can send a message with a remote automatic action, and the management server runs that action on the remote node. However, you can configure a management server to allow or deny remote automatic action requests. You do this using a remote action security policy. The policy can contain exceptions, so that the management server allows or denies remote automatic action requests for specific nodes.

For example, you might want to restrict remote automatic actions in the following situations:

- If local administrators of managed nodes cannot be trusted, you may want to deny remote automatic actions from these nodes.
- If you are managing multiple customers, organizations, or departments, you may want to allow automatic actions only between nodes in the same domain.

After you create a remote action security policy, you must deploy it to the management server that you want to configure.

In previous versions of HPOM, you could configure remote action security using registry keys. After you upgrade the management server, these registry keys still take effect. You can reconfigure them using the Server Configuration dialog. However, if you deploy a remote action security policy to the management server, this will override the existing registry keys.

NOTE:
Remote action security policies determine whether a management server starts remote automatic actions. Remote action security policies do not prevent users from starting remote automatic actions manually. They also do not prevent users from starting remote operator-initiated actions. You can configure user roles to prevent users from starting actions on specific nodes.

You can also disable message actions completely as follows:

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Message Action Server Message Filter .
3. Select the Expert mode check box. A full list of values appears.
4. Set the value of Remove all message actions to true. The management server removes automatic and operator-initiated actions (including remote automatic actions) from all incoming messages.
5. Click OK .

Related Topics:

- Create a remote action security policy
- Set exceptions for a remote action security policy
- Reconfigure registry keys for remote action security
- Configuring user roles
- Configure automatic actions for duplicate messages

Create a remote action security policy

You can configure a management server to allow or deny remote automatic action requests using a remote action security policy.

To create a remote action security policy

1. In the console tree, under Policy management → Server policies grouped by type , right-click Remote Action Security and then click New→Policy . The remote action security policy editor opens.
2. In the General tab, click one the following options:
   - Deny all actions

     No remote automatic actions are allowed.
   - Allow actions

     Remote automatic actions are allowed.

     Optional. HTTPS agents run policies that the management server secures using certificates. If someone tampers with a policy or the action that it contains, the policy becomes invalid. To allow remote automatic actions from HTTPS agents only, select the Only certified check box. No remote automatic actions from DCE agents are allowed.
3. *Optional.* Set exceptions. For more information, see Set exceptions for a remote action security policy .
4. Save the policy, and then deploy it to the management server that you want to configure.

Related Topics:

- Save a policy
- Deploy a policy or policy group

Set exceptions for a remote action security policy

A remote action security policy can contain exceptions, which allow or deny remote automatic actions for specific source and target nodes. These exceptions override your choices on the General tab.

After you the deploy the remote action security policy to a management server, the management server evaluates the remote automatic actions in incoming messages against the exceptions. It evaluates the exceptions in the order that you specify, and applies the first exception that matches. If no exception matches, the management server allows or denies the action according to your choices on the General tab.

To set exceptions for a remote action security policy

1. In the remote action security policy editor, click the Exceptions tab.
2. Click New... . The Exception properties dialog box opens.
3. *Optional.* Type a short Description to identify the exception.
4. Specify source nodes. You can either select specific nodes or node groups, or specify a pattern that matches the names of many nodes or node groups.

   - To select specific nodes:
     1. Click Add Nodes ... . A dialog box opens that contains a tree of nodes and node groups.
     2. Select individual nodes or node groups. (You can select node groups in this dialog box to quickly add all the individual nodes in that are *currently* in the group.)
     3. Click OK .
   - To select specific node groups:
     1. Click Add Group ... . A dialog box opens that contains a tree of node groups.
     2. Select node groups, and then click OK .
   - To specify a pattern to match the names of nodes or node groups:
     1. Click Add Pattern... . The Edit Node / Group dialog box appears.
     2. Type a Matching string . The string can contain any characters and pattern matching expressions. You can type the pattern or insert expressions by clicking the > button, and then clicking Matching expressions . Click one of the expressions that appears.
     3. Click Type to select the whether to apply the match string to node name or node group .
     4. Click OK .

   To change an existing match string, click it in the list, and then click Edit... .

5. Click the Target tab, and then specify target nodes. In the same way that you specify source nodes, you can either select specific target nodes or node groups, or specify a pattern that matches the names of many nodes or node groups.

6. To allow actions if the target node is the same as the node that sent the message, in the Target tab, select the on source node check box.

   Normally, if the target node of an automatic-action is the same as the node that is sending the message, the agent runs the action immediately on the node, and the management server receives the action response. However, in some cases the agent does not run the action automatically and the management server is responsible for starting the automatic-action remotely. This can happen, for example, if the action contains variables that the management server must resolve, or if a message passes through the message stream interface on the agent. This check box enables you to allow this type of remote action.

7. In Policy action choose one of the following options:

   - Click Deny to disallow remote automatic actions when the source and target node match the criteria in this exception.

   - Click Allow to allow remote automatic actions when the source and target node match the criteria in this exception.

     Optional. HTTPS agents run policies that the management server secures using certificates. If someone tampers with an policy or the action that it contains, the policy becomes invalid. To allow remote automatic actions from HTTPS agents only, select the Only certified check box. No remote automatic actions from DCE agents are allowed.

8. Click OK . You return to the remote action policy editor.

9. *Optional.* Add further exceptions. To reorder the exceptions, click an exception in the list, and then click Move Up or Move Down .

10. Save the policy, and then deploy it to the management server that you want to configure.

Related Topics:

- Pattern-matching details
- Save a policy
- Deploy a policy or policy group

Reconfigure registry keys for remote action security

This version of HPOM enables you to configure remote action security using server policies. In previous versions of HPOM, you could configure remote action security using registry keys. After you upgrade the management server , these registry keys still take effect. You can reconfigure them using the Server Configuration dialog. However, if you deploy a remote action security policy to the management server, this will override the existing registry keys.

To reconfigure registry keys for remote action security

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Remote Action Handling . A list of values appears.
3. *Optional.* Set the value of Enable remote actions to true or false. If you set this value to false, the management server denies all remote automatic actions. The management server allows automatic actions on the node that the message originates from. This does not prevent users from starting remote operator-initiated actions.
4. *Optional.* In Allow remote actions from specific node s , type a comma-separated list of primary node names without spaces. The management server allows remote automatic actions from these nodes, even if the value of Enable remote remote actions is false .

   The default value is an empty string, which means that the management server denies all remote automatic actions.
5. Click Apply .

Related Topics:

- Configure network information for managed nodes

Server-based MSI Policies

The server-based message stream interface (MSI) enables external applications to read and change incoming messages on the management server before they are stored in the database.

NOTE:
The policy that generates a message must specify that the management server diverts or copies the message to the MSI. Otherwise, the message bypasses the MSI.

Each external application must be registered with the MSI on the management server using a server-based MSI policy. Normally, if a Smart Plug-in (SPI) uses server-based MSI, it includes the policy for the external applications that it provides.

If you develop your own MSI applications using the C and COM APIs provided, you must install the applications on the management server. You must then create or modify a server-based MSI policy to register them, and then deploy the policy to the management server that you want to configure.

To create a server-based MSI policy:

1. In the console tree, under Policy management → Server policies grouped by type , right-click Server-based MSI and then click New→Policy . The server-based MSI policy editor appears.
2. To add details of an application instance, click New . The MSI instance dialog appears.
3. *Optional.* Type a Description of the instance for future reference.
4. Specify an Order number in the range -127 to 127. This controls the order in which application instances can read and change the stream of incoming messages. An application instance with a lower order number is able to read, change, and delete messages before an application instance with a higher order number.

   If multiple application instances have the same order number, the management server forwards messages to all the application instances at the same time.

   CAUTION:
   Forwarding messages to multiple application instances at the same time can have undesirable results with diverted messages, because there will be two messages in the message stream with the same message ID.

   The same problem can also occur with messages that are copied to the MSI, if at least two application instances receive the same message in parallel and do not modify it.

5. Type the Instance name of the external application.
6. Click the Application type that corresponds to the API that the application uses to interact with the MSI. This can either be the Legacy UNIX API (which is the C API), or COM .
7. Click OK . You return to the server-based MSI policy editor.
8. *Optional.* Add further application instances. To reorder the instances, click an instance in the list, and then click Move Up , Move Down , or type a number and click Move to .
9. Save the policy, and then deploy it to the management server that you want to configure.

Related Topics:

- Send messages to the message stream interface
- Server Message Stream Interface (MSI)
- Save a policy
- Deploy a policy or policy group

Configuring outage information

At times the systems in your managed environment may need some maintenance work done. This can be unexpected and can occur randomly. In this case the system is experiencing an unplanned outage. If the outage is known in advance and planned, for example, to install a new Service Pack on the system, the system is undergoing a scheduled or planned outage.

The same applies to services. A service can experience an unplanned outage, for example because the system that hosts the service goes down. It can also experience a planned outage, for example because an application must be upgraded to a newer version.

During the outage, the HP Operations agent on the system may continue to send messages, even though the problem is already known to the administrator or even expected. To avoid unnecessary messages, you can instruct HPOM to temporarily put the managed node, node group, external node, or service into maintenance mode.

When a node or service is in maintenance mode, its status by default does not affect the status of its parent node groups or parent services.

Related Topics:

- Put nodes or node groups into unplanned outage mode
- Schedule an outage for a node or node group
- Put services into unplanned outage mode
- Schedule an outage for a service
- Configure status calculation for nodes and services in maintenance mode

Put nodes into unplanned outage mode

Use the Unplanned Node Outage dialog box to specify how you want to handle incoming messages and heartbeat polling in the event that a node experiences a failure. As the name suggests, an unplanned outage is unexpected and might occur due to power failure, hardware failure, network problems, and similar situations.

To put a node into unplanned outage mode

1. In the console tree, right-click the managed node, external node, or node group that you want to put into outage mode. This opens the context menu.
2. Select All Tasks → Set unplanned outage → On . The Unplanned Node Outage dialog box opens.
3. Select one of the following options:
   - Select Delete to remove any incoming messages during the outage.
   - Select Acknowledge to place the messages in the acknowledged messages browser for later consideration.
4. Click the check box to disable heartbeat polling during the outage.

   NOTE:
   This setting is not available for external nodes.

5. Click OK to confirm your choices and close this dialog box.

Related Topics:

- Schedule an outage for a managed node
- View node outage properties
- View external node outage properties

Schedule an outage for a node

A scheduled outage is planned to happen and recurs at regular intervals for maintenance. You can define a scheduled node outage using two scheduled task policies:

- The first scheduled task policy defines the following:
  - Start date and time
  - Managed node, external node, or node group to put into scheduled outage
  - Whether to delete or acknowledge messages during outage
  - Whether to enable or disable heartbeat polling during outage
- The second scheduled task policy defines the following:
  - End date and time
  - Managed node, external node, or node group to move out of scheduled outage

Both policies use the `ovownodeutil` command line tool.

To schedule an outage for a managed node, external node, or node group

1. Create a scheduled task policy.
2. In the **Task** tab, choose **Command** as task type.
3. Select the `%OvInstallDir%\bin\ovownodeutil.cmd` command line tool with the following options:
   - **Managed nodes or node groups** :

     `-outage_node`
     : Sets the outage state for the node.

     `-node_name <node_name>` or `-node_id <node id>`
     : The primary name or the ID of the node to move into scheduled outage mode.

     `-group_path <hierarchy path>` or `-group_id <group id>`
     : The hierarchical path or the ID of the node group to move into scheduled outage mode.

     `-scheduled`
     : Sets the scheduled outage state.

     `-on`
     : Enables the outage state for the node or node group.

     `-delete_msgs`
     : Specifies that messages from the node are deleted during the scheduled outage.

`-disable_heartbeat`
: Specifies heartbeat polling to be switched off during the scheduled outage.

- External nodes or node groups :

  `-outage_exnode`
  Sets the outage state for the external node or node group.

  `-exnode_path <external node hierarchical path>` or `-exnode_id <node id>`
  The hierarchical path or the ID of the external node to move into scheduled outage.

  `-group_path <hierarchy path>` or `-group_id <group id>`
  The hierarchical path or the ID of the node group to move into scheduled outage mode.

  `-scheduled`
  Sets the scheduled outage state.

  `-on`
  Enables the outage state for the external node or node group.

  `-delete_msgs`
  Specifies that messages from the external node are deleted during the scheduled outage.

4. In the Schedule tab, specify the start date and time for the outage.
5. Save the policy.
6. Create a second scheduled task policy.
7. In the Task tab, choose Command as task type.
8. Select the `%OvInstallDir%\bin\ovownodeutil.cmd` command line tool with the following options:

- Managed nodes or node groups :

  `-outage_node`
  Sets the outage state for the node or node group.

  `-node_name <node_name>` or `-node_id <node id>`
  The primary name or the ID of the node to move out of scheduled outage mode.

  `-group_path <hierarchy path>` or `-group_id <group id>`
  The hierarchical path or the ID of the node group to move out of scheduled outage mode.

  `-scheduled`
  Sets the scheduled outage state.

  `-off`
  Disables the outage state for the node or node group.

- External nodes or node groups :

  `-outage_exnode`
  Sets the outage state for the external node or node group.

`-exnode_path <external node hierarchical path>` or `-exnode_id <node id>`
: The hierarchical path or the ID of the external node to move out of scheduled outage mode.

`-group_path <hierarchy path>` or `-group_id <group id>`
: The hierarchical path or the ID of the node group to move out of scheduled outage mode.

`-scheduled`
: Sets the scheduled outage state.

`-off`
: Disables the outage state for the external node or node group.

9. In the Schedule tab, specify the end date and time for the outage.
10. Save the policy.
11. Deploy both policies to the management server node.

Related Topics:

- Put nodes or node groups into unplanned outage mode
- View node outage properties
- View external node outage properties
- ovownodeutil

Put services into unplanned outage mode

Use the Unplanned Service Outage dialog box to specify how you want to handle incoming messages in the event that a service experiences a failure. As the name suggests, an unplanned outage is unexpected and might occur due to power failure, hardware failure, network problems, and similar situations.

To put a service into unplanned outage mode

1. In the console tree, right-click the service that you want to put into outage mode. This opens the context menu.
2. Select All Tasks → Set unplanned outage → On . The Unplanned Service Outage dialog box opens.
3. Select one of the following options:
   - Select Delete to remove any incoming messages during the outage.
   - Select Acknowledge to place the messages in the acknowledged messages browser for later consideration.
4. Click the check box to also put all subservices into maintenance mode.
5. Click OK to confirm your choices and close this dialog box.

Related Topics:

- Schedule an outage for a service
- View service outage properties for a selected service

Schedule an outage for a service

A scheduled outage is planned to happen and recurs at regular intervals for maintenance. You can define a scheduled service outage using two scheduled task policies:

- The first scheduled task policy defines the following:
  - Start date and time
  - Service to put into scheduled outage
  - Whether to delete or acknowledge messages during outage
- The second scheduled task policy defines the following:
  - End date and time
  - Service to move out of scheduled outage

Both policies use the `ovowserviceutil` command line tool.

To schedule an outage for a service

1. Create a scheduled task policy.
2. In the Task tab, choose Command as task type.
3. Select the `%OvInstallDir%\bin\ovowserviceutil.cmd` command line tool with the following options:

   -outage_service
   : Sets the outage state for the service.

   -service_name <service_name>
   : The name of the service to move into scheduled outage.

   -scheduled
   : Sets the scheduled outage state.

   -on
   : Enables the outage state for the service.

   -delete_msgs
   : Specifies that messages from the service are deleted during the scheduled outage.
4. In the Schedule tab, specify the start date and time for the outage.
5. Save the policy.
6. Create a second scheduled task policy.
7. In the Task tab, choose Command as task type.

8. Select the `%OvInstallDir%\bin\ovowserviceutil.cmd` command line tool with the following options:

   -outage_service
   : Sets the outage state for the service.

   -service_name <service_name>
   : The name of the service to move into scheduled outage.

   -scheduled
   : Sets the scheduled outage state.

   -off
   : Disables the outage state for the service.

9. In the Schedule tab, specify the end date and time for the outage.

10. Save the policy.

11. Deploy both policies to the management server node.

Related Topics:

- Put services into unplanned outage mode
- ovowserviceutil
- View service outage properties for a selected service

Configure status calculation options

The status of a service is determined by the status of its subordinate services and by the status of the messages for that service. By default, services in maintenance mode do not affect the status of their parent services.

You can, however, configure HPOM to consider the status of subservices in maintenance mode when calculating the status of the parent service. If so configured, the highest severity of the subordinate services (including subordinate services in maintenance mode) and of the messages for the parent service are used when calculating the status of the parent service.

1. In the console tree, right-click Operations Manager , and then click Configure Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Status Engine . A list of values appears.
3. Use elements in maintenance in threshold calculation is by default set to False . Choose True if you want HPOM to also consider the status of subservices in maintenance mode when calculating the status of parent services.
4. Click Apply , and then OK to save your changes and close the dialog box.
5. Restart the `OvEpStatusEngine` service.

Related Topics:

- Schedule an outage for a managed node
- Put nodes into unplanned outage mode
- Schedule an outage for a service
- Put services into unplanned outage mode

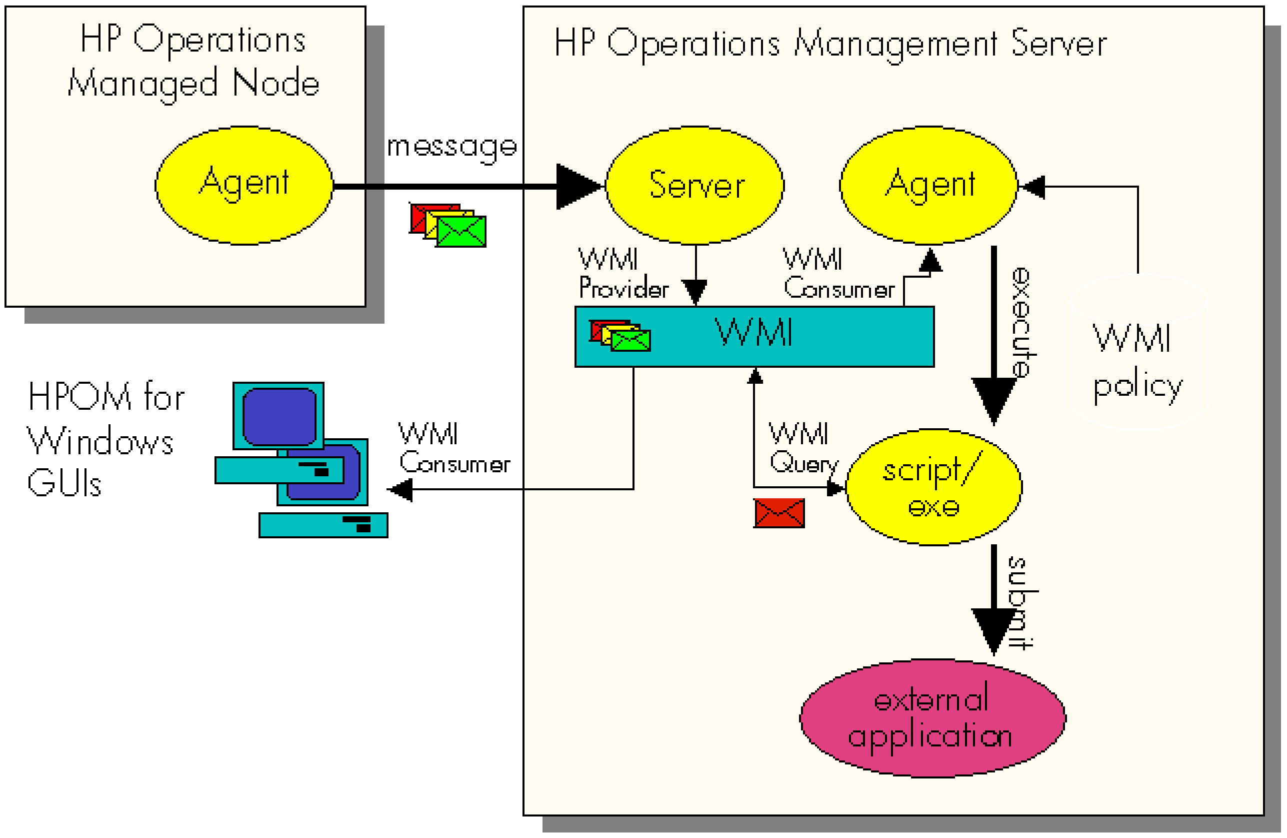
Forwarding Messages to External Applications

HPOM for Windows provides options to forward messages to external applications, including external trouble-ticket or notification software, using the standard interfaces of HPOM.

HPOM for Windows does not provide a dedicated trouble-ticket or notification interface as HPOM for UNIX does. However, a major part of the functionality can be implemented easily by using HPOM for Windows and underlying platform services, particularly the WMI infrastructure on the HPOM for Windows management server. On the server, all HPOM for Windows objects (nodes, services, messages, ...) are represented by WMI objects in HPOM for Windows' own namespace:

You can use regular HPOM for Windows WMI policies to monitor these objects so that any new message that arrives on the management server can trigger an automatic action that forwards the message to an external application in the following sequence:

- The WMI policy catches events triggered by the creation of an instance of the class OV_Message on the HP Operations management server.
- Then the new message can be forwarded to any other party using a script or program bound to the policy rule as an automatic command, as shown in the illustration.



The script itself can perform whatever operations are necessary. In this example, the operation would be to either submit a trouble-ticket or trigger a notification system. The WMI policy can contain as many rules as required, and therefore can match arbitrary messages. The executed export script may be different for each rule.

WMI and HPOM for Windows

WMI, as Microsoft's implementation of WBEM, is used heavily as infrastructure on the HP Operations management server. In addition to the basic CIM objects provided by the operating system, many HPOM for Windows data objects are represented as WMI objects that can be accessed using WMI policies, scripts, or programs. However, these classes are considered internal to HP. They may be modified by HP in subsequent releases and they should be strictly read-only.

HP does not provide any detailed documentation about these classes other than what is available in the HPOM for Windows online help.

The WMI class and instance browsers which are integrated into the HPOM for Windows WMI policy editor can be used to browse through the WMI classes and instances. You can also use the Microsoft tool, wbemtest, to explore WMI classes.

Related topics:

Detailed information:

- Using WMI policies to intercept incoming messages
- Using automatic actions to forward messages

Examples:

- Forward messages using email
- Example policies and scripts for message forwarding

Advanced topics:

- Forwarding message changes

Using WMI policies to intercept incoming messages

Whether a message is passed to some notification interface, a trouble-ticket system, or any other location, the configuration within HPOM for Windows is always the same. You will configure a WMI policy and its rules for forwarding messages to notification systems such as email and eternal help desk systems when an instance of the class OV_message has been created on the HPOM management server.

To configure the WMI policy, follow these steps:

1. In the console tree, select Policy Management → Policies grouped by type → Windows Management Interface .
2. Right-click to open the context menu. Select New → Policy to open the WMI policy editor.
3. In the WMI Namespace box, type the namespace name:

   `ROOT\HewlettPackard\OpenView\Data`
4. Set the Object type to Instance .
5. In the Instance class name box, type `OV_Message` .
6. In the Type of query field, select Query the intrinsic event for these instances .
7. For all other fields, keep the defaults.

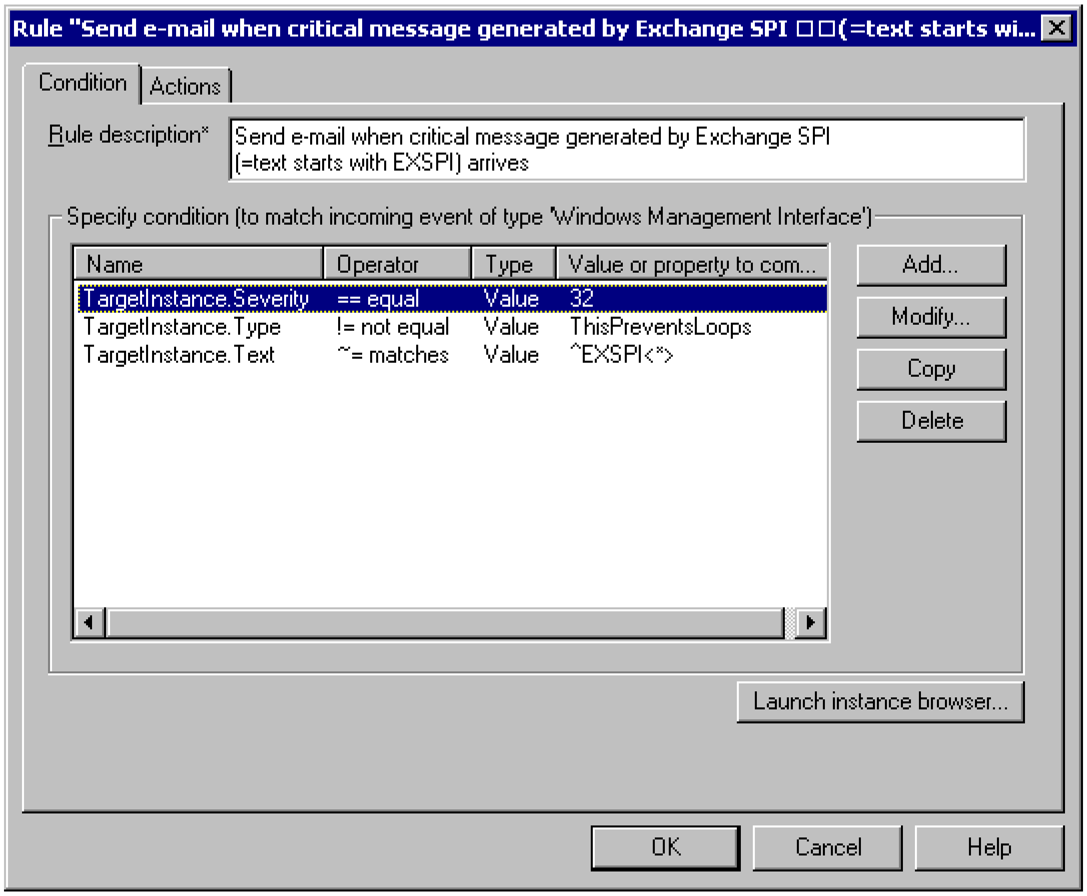
After specifying these general settings, configure the rules that define which messages should be forwarded:

NOTE:
Forwarding all incoming messages will increase the load on the management server and might affect the overall system performance.

To configure WMI policy rules for message forwarding

Follow these steps to specify which messages you want to forward. You can identify these by specifying one or more message attributes. The following example shows conditions which check to see if a message has a severity of 32 (critical) and if the message text starts with 'EXSPI'. The last condition is used to prevent loops.

1. In the WMI policy editor, select the Rules tab . Click New to open the New rule dialog box.
2. Use the Condition tab of the Rule dialog box to specify arbitrary conditions to determine which messages should be forwarded.



Rule "Send e-mail when critical message generated by Exchange SPI □□(=text starts wi...

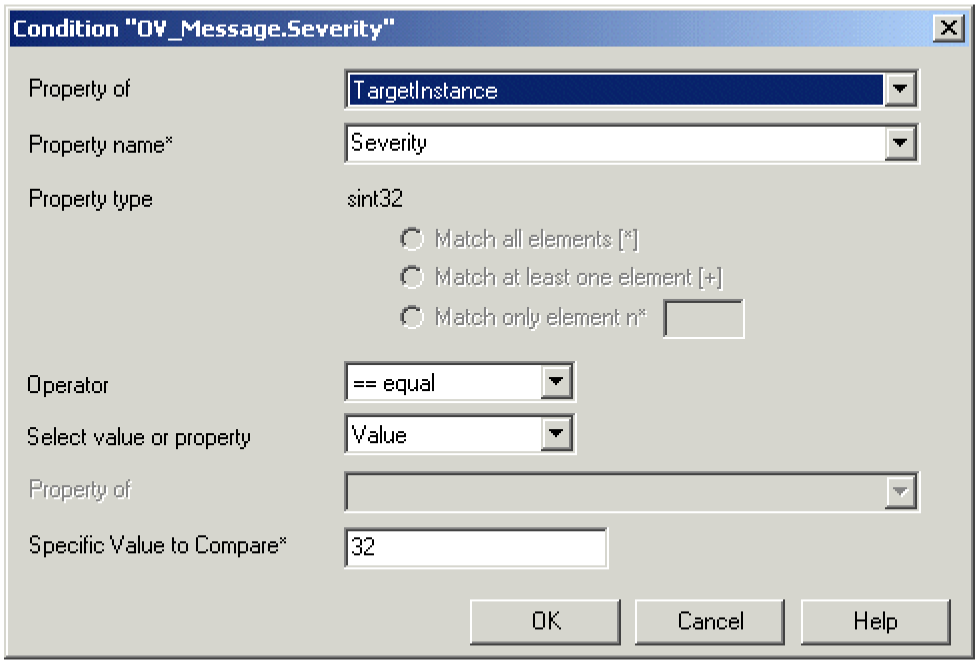
Condition | Actions

Rule description*: Send e-mail when critical message generated by Exchange SPI (=text starts with EXSPI) arrives

Specify condition (to match incoming event of type 'Windows Management Interface')

Name	Operator	Type	Value or property to com...
TargetInstance.Severity	== equal	Value	32
TargetInstance.Type	!= not equal	Value	ThisPreventsLoops
TargetInstance.Text	~= matches	Value	^EXSPI<*>

3. To create a new rule condition, click Add to open the New condition dialog box.



Condition "OV_Message.Severity"

Property of	TargetInstance
Property name*	Severity
Property type	sint32
Operator	== equal
Select value or property	Value
Property of	
Specific Value to Compare*	32

4. From the Property name list, select the name of the property to be used to filter messages for forwarding. For example, select Severity from the list.
5. Select from the Operator list. For example, select ==equal.
6. In the Specific value to compare box, enter the number 32, which represents the severity level "Critical". The other severity levels are represented by the following number equivalents:

   Major=16
   Minor=8
   Warning=4
   Normal=2
   Unknown=1
7. Click OK to return to the New rule dialog box, where the condition you just created appears in the list.
8. Click OK again to return to the Rule tab, where your newly created rule is summarized.
9. Configure the automatic command using the instructions in Using automatic actions to forward messages .
10. To prevent loops, see How to prevent loops for details.
11. After you have created the WMI policy, deploy it to the HP Operations management server (It will not work on any other system).
12. Send a critical test message (or any other test message which matches the rule configured earlier). The message should show up in the HPOM console as usual and the forwarding command should launch. If something went wrong, an additional error message should appear in the HPOM console. This message would be generated by the WMI policy if the automatic command failed. If no error message appears, check to see if the message was forwarded correctly to the external application.

    Related Topics:

    - Forwarding Messages to External Applications
    - Using automatic actions to forward messages

Using automatic actions to forward messages

You can forward messages using an automatic command. This command will differ, depending on the external application to which you want to forward the message:

- To forward a message to an email system, use the FwdMsgAsEMail.vbs VBScript.
- To forward messages to the HP Operations Service Desk product, use OVO-Sd.vbs (which is part of the Service Desk integration pack).
- To forward a message to your own application, this application must offer a command line interface or scripting interface to feed in HP Operations message data. With the automatic command, you can execute this interface and pass in any message attribute that you want.
- To forward messages to a trouble ticket system, see Example policies and scripts for message forwarding .
- To forward messages to a notification system, see Example policies and scripts for message forwarding .

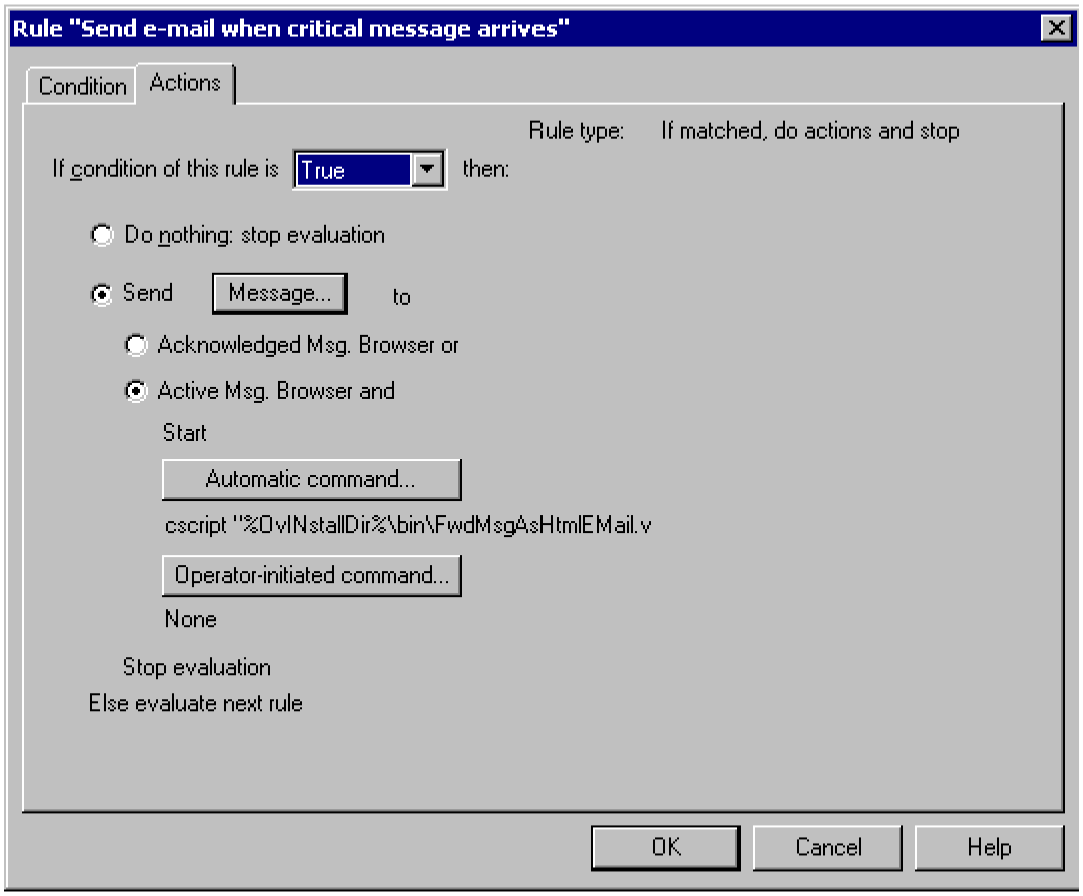
To configure an automatic action

1. First, configure the message, as described in the topic Using WMI policies to intercept incoming messages . Use a message text like "Could not forward message to ..." and set a property to prevent loops. See How to prevent loops for details. This message is primarily for troubleshooting purposes. If the forwarding succeeds, then no additional message will be generated.
2. Configure the automatic action by following these steps.

   From the WMI policy editor, select the **Rules** tab.

   Select **New** to open the **New Rule** dialog box.

   Select the **Actions** tab.


Rule "Send e-mail when critical message arrives"

Rule type: If matched, do actions and stop

If condition of this rule is True then:

( ) Do nothing: stop evaluation

(●) Send Message... to

( ) Acknowledged Msg. Browser or

(●) Active Msg. Browser and

Start

Automatic command...

cscript "%OvINstallDir%\bin\FwdMsgAsHtmlEMail.v

Operator-initiated command...

None

Stop evaluation

Else evaluate next rule

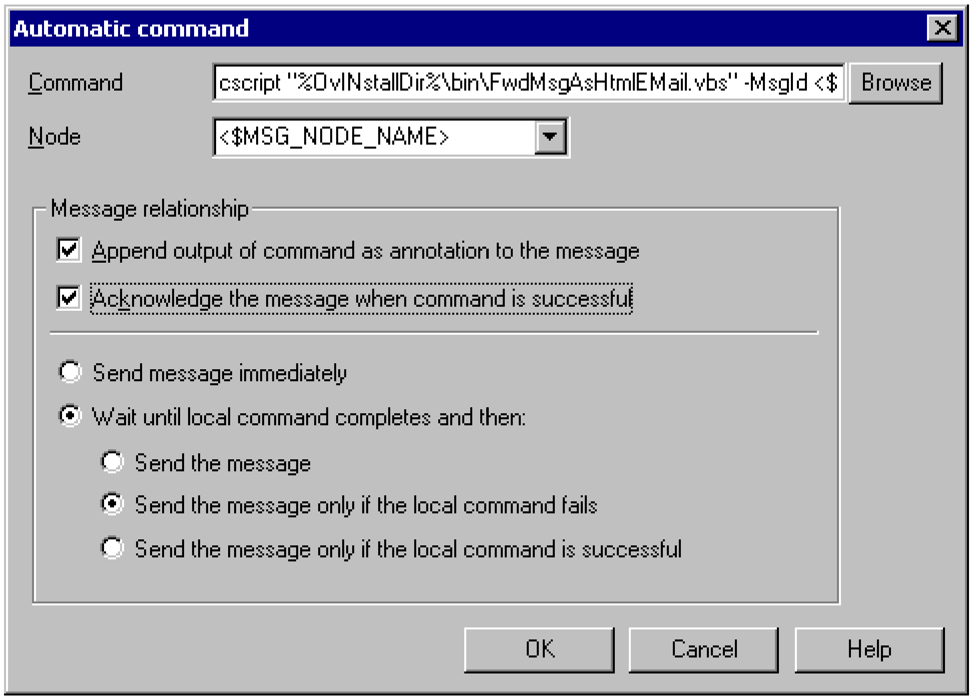
Configure a VB script. It gets the message ID as a parameter:

```
cscript.exe "%OvInstallDir%\bin\MyForwardingScript.vbs" "<$WBEM:TargetInstance.Id>"
```

The following example shows the call of an executable which receives additional attributes of the message as parameters:

```
"C:\Program Files\Test\Forward.exe" "<$WBEM:TargetInstance.Id>"
"<$WBEM:TargetInstance.Text>" "<$WBEM:TargetInstance.Object>"
```

3. Click Automatic command to open the Automatic Command dialog box. Be sure that Wait until local command completes and then and send the message only if the local command fails are selected, so that the error message you configured above is only generated when the forwarding command fails.



Automatic command

Command: cscript "%OvINstallDir%\bin\FwdMsgAsHtmlEMail.vbs" -MsgId <$

Node: <$MSG_NODE_NAME>

Message relationship

- [✓] Append output of command as annotation to the message
- [✓] Acknowledge the message when command is successful

- ( ) Send message immediately
- (•) Wait until local command completes and then:
  - ( ) Send the message
  - (•) Send the message only if the local command fails
  - ( ) Send the message only if the local command is successful

NOTE:

It is also a good idea to select Acknowledge the message when command is successful because it might happen that the forwarding fails due to a network problem or other external cause. You will then get a corresponding error message. After the external problem is solved, you can restart the automatic command, initiating the forwarding again. If it then succeeds, it will automatically acknowledge the error message in your message browser.

Related Topics:

- Example policies and scripts for message forwarding
- Forward messages using email

Example policies and scripts for message forwarding

Several example policies and scripts are available on the HPOM for Windows management server. These can be used as templates for your own forwarding policies and scripts.

Example policies can be found in the Samples policy group.

Example Policy: Forwarding to a trouble ticket system

Policy	Details
opcmsgTTNS	Example opcmsg policy to submit messages tagged with the Forward-to-TT or Notification flag. This can be used to generate messages to be caught with the SubmitTTByFlag or SubmitNSByFlag policies, shown below.
SubmitTTByFlag	WMI policy that forward messages tagged as DoNotification to the script SubmitNS.vbs. With the policy opcmsgTTNS, use: `opcmsga=TT o=oo msg_t=hallo` `or opcmsg a=TT-ack o=oo msg_t=hallo` , which automatically acknowledges the message after successful submission.

SubmitTTByRule	WMI policy that forwards messages which have the message group TT-Rule to the script SubmitTTvbs. With the policy opcmsgTTNS use:
	`opcmsg a=aa o=oo msg_t=hallo`
	`msg_g=TT-Rule`

UpdateTT	WMI policy listening for generic message change events. Calls the script`UpdateTTvbs` .
UpdateTTStateChange	WMI policy listening for message state change events (triggered by own, acknowledge,...). Calls the script `UpdateTTvbs` with options -s and the new state.

Example policy: Forwarding to a notification system

Policy	Details
SubmitNSByFlag	WMI policy that forwards messages tagged as DoNotification to the script SubmitNS.vbs. With the policy opcmsgTTNS use: `opcmsg a=NS o=oo msg_t hallo`

SumitNSByRule	WMI policy that forwards messages to the script SubmitNS.vbs which have the message group NS-Rule. With the policy opcmsgTTNS use:
	`opcmsg a=NS o=oo msg_t hallo`
	msg_g=NS-Rule

Example Policy: Forward messages to Service Desk

See the help topic Example: Forward messages to Service Desk for details.

Example Policy: Forward messages using Email

See the help topic Sending Messages through Email for details.

Example Scripts

Example scripts can be found in this location:

`<InstallDir>/examples/OvOW/MessageForwarding`

Example Script: Forwarding to a trouble ticket system

Script	Details
OvOWSubmitTT.vbs	Submits the message to `C:\temp\test.out` as a simulation.
OvOWUpdateTT.vbs	Logs message updates in `C:\temp\test.out` as a simulation.

Example Script: Forwarding to a notification system

Script	Details
OvOWSubmitNS.vbs	Collects the 14 notification parameters as HPOM for UNIX does and adds an entry in `C:\temp\test.out` as a simulation.

Example Script: Forward messages to Remedy ARS

Script	Details
OvOWSubmitARS.vbs	Submits the message to Remedy ARS.
OvOWSubmitAR.arq.	Macro necessary for Remedy ARS submission.

Example Script: Forward messages to Service Desk

See the help topic Example: Forward messages to Remedy ARS for details.

Some of these scripts log the message into the file `C:\temp\test.out` for simulation purposes. Sample content of this file is shown below. Your output might differ depending on the policies you use.

Submitted TT from message: 'Test message' ID: '096aba60-55c7-71d5-13cf-0f8878020000' Node: CARROT (management server) CreatedAt: 14:33:04 2001/05/31

Notification message: 'MessageId: e7796780-55c1-71d5-1394-0f8878020000 NodeName: CARROT

NodeType: Pentium Windows 2003 DateCreated: 2001/05/31 TimeCreated: 14:38:57 DateReceived: 2001/05/31 TimeReceived: 14:38:57 Application: NS MessageGroup: Object: abc Severity: Normal Resp. Users: n/a Text: hallo10 Instructions: Some instructions.'

Message with Id: '096aba60-55c7-71d5-13cf-0f8878020000' is now in state Owned.

Message with Id: 'db2e2e20-55c6-71d5-13cf-0f8878020000' is now in state Acknowledged.

Example Script: Forward messages using Email

See the help topic Sending Messages through Email for details.

Related Topics:

- Example: Forward messages to Remedy ARS
- Example: Forward messages to Service Desk

Forward Messages to Service Desk

You can forward messages to the Service Desk application following the instructions in the Service Desk manual *HP Operations Manager Integration Administrator's Guide* , which contains a detailed description of the Service Desk integration, including message forwarding, bi-directional synchronization, and other features.

The necessary policies and scripts are part of the Service Desk Integration pack.

The service packs and the latest versions of publications are available at http://support.openview.hp.com/patches/patch_index.jsp and http://ovweb.external.hp.com/lpe/doc_serv respectively.

The Service Desk integration with HP Operations Manager makes it possible to:

- Import nodes and services into Service Desk.
- Send events from HP Operations Manager to Service Desk.
- Reflect HP Operations Manager updates in Service Desk.
- Send acknowledgment messages and message annotations from Service Desk to HP Operations Manager.
- View Service Desk configuration items and incidents from HP Operations Manager
- View HP Operations service state from a browser.
- Generate an HP Operations message from Service Desk.
- Monitor Service Desk processes and error log files.

Related Topics:

- Example policies and scripts for message forwarding

Forward messages to Remedy ARS

Remedy ARS provides a program (`runmacro.exe`) that can be used to externally create a trouble ticket or operate on an existing trouble ticket. This program can be called by a trouble ticket submission script such as the examples shown below:

Example 1:

```
ExecPath="runmacro.exe -eSubmit AR -hg:\ProgramFiles\remedy" &_
         " -p Node=' " & ' " -p MsgId=" &" -p Text=' &Text
```

Example 2:

```
WscriptEchoExecPath
Set WshShell=WScriptCreateObject("WScript.Shell")
RunError=Wsh.ShellRun(ExecPath,1, TRUE)
```

Additionally, an ARS macro is required. In this example, it is `SubmitAR` , located in the macro default directory `c:\Program Files\remedy\arcmds` , that actually does the job (runmacro only triggers the macro). We pass the message attributes as macro parameters (marked by the -p options).

The macro itself may look like the example shown below (but this depends on the ARS form structure and how the trouble ticket entries are structured).

```
SubmitAR
Set-schema:DemoHelpDesk localhost
Submit:
DemoHelp Desk_carrot|8=$MsgId$_100000000=$Text$_100000001=Demo_100000005_10000001
4=Software_100000038=Demo_100000039=Other_2=$-1$_7=0_100000006=0_100000013=0_
end
```

The AR will be submitted into the form DemoHelpDesk, and the parameters Text and MsgId will be used as initial values for the Short Description and Details fields.

NOTE:
You cannot simply copy this text into a text file; it contains some internal structure with control characters. You must use the Remedy ARS User tool to create this macro yourself or use the file SubmitAR.arq in the directory `<InstallDir>/examples/OvOW/MessageForwarding` .

See the Remedy documentation for more details on how to use runmacro and how to create ARS macros.

Related Topics:

- Example policies and scripts for message forwarding

Forwarding Messages as in HPOM for UNIX

Both notification and trouble-ticket systems can be integrated into HPOM for UNIX by providing a script or command and registering it with HPOM for Windows. The script is called with a set of 14 standard parameters (the main message attributes plus some additional context information). See the HPOM for UNIX documentation for details.

An example script, `SubmitNS.vbs` has been provided which collects the 14 parameters as in HPOM for UNIX and then logs these parameters to `C:\temp\test.out` as simulation. You might find this useful if you want to reuse a script you have already used with HPOM for UNIX.

The following restrictions apply:

- The information about 'responsible users' is not available with HPOM for Windows.
- Do not use MS Windows batch files for this purpose. They can handle at most 10 arguments.
- It is not possible to execute programs from a VB script with an overall command line length of more than about 2048 characters. Therefore, if necessary, the text has to be truncated.

Related Topics:

- Example policies and scripts for message forwarding

Forwarding message changes

HPOM for Windows allows you to modify a message in various ways:

- Add and modify annotations
- Change severity and message text
- Own, disown, acknowledge, and unacknowledge messages

All these modifications can be forwarded to the trouble-ticket system that has received the original copy of the message. On the HPOM for Windows side this is fairly simple; the HP Operations server generates change events which can be caught by another WMI policy. Then, these change events must be associated with the created trouble ticket which must meet the following requirements:

- The trouble-ticket must contain the HP Operations Message ID as attribute
- There must be an external mapping mechanism which finds the TT ID for an HP Operations Message ID

Because this has to be done in the interface script/program attached to the WMI export policy (outside of HPOM for Windows) and also depends on the trouble-ticket software itself, it will not be discussed here.

See Example policies and scripts for message forwarding for an example of such a message modification policy (UpdateTT policy) and the example script, `UpdateTT.vbs` .

The UpdateTT policy takes advantage of WMI events generated by the HP Operations server whenever a message is modified. To register for change events, use either the generic class OV_Message_ChangeEvent (to register for any change events) or the derived event classes (which represent specific change events):

- OV_Message_SeverityChangeEvent
- OV_Message_TextChangeEvent
- OV_Message_NumberOfAnnotationsChangeEvent
- OV_Message_StateChangeEvent
- OV_MessageAction_StateChangeEvent

Either the generic or derived classes can be used to listen for message modifications. You must create a separate WMI policy for each change event class, because WMI policies cannot be bound to multiple objects or events in parallel.

As in the initial submission policy, there must be a script called as an automatic command. The script must verify itself whether the change event applies to a message which has been submitted to a trouble-ticket system in the first place. There is no way to specify policy rules filtering for certain message attributes (like the TT-Flag or application, object, ...).

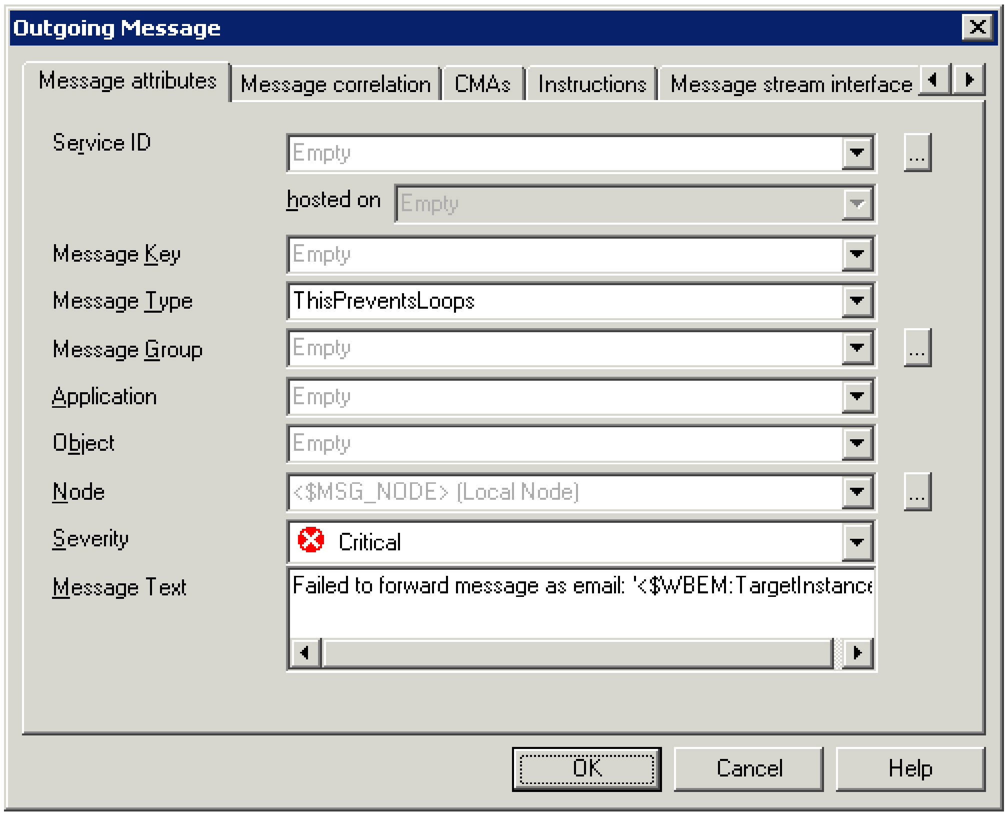
Related Topics:

- Example policies and scripts for message forwarding

How to prevent loops

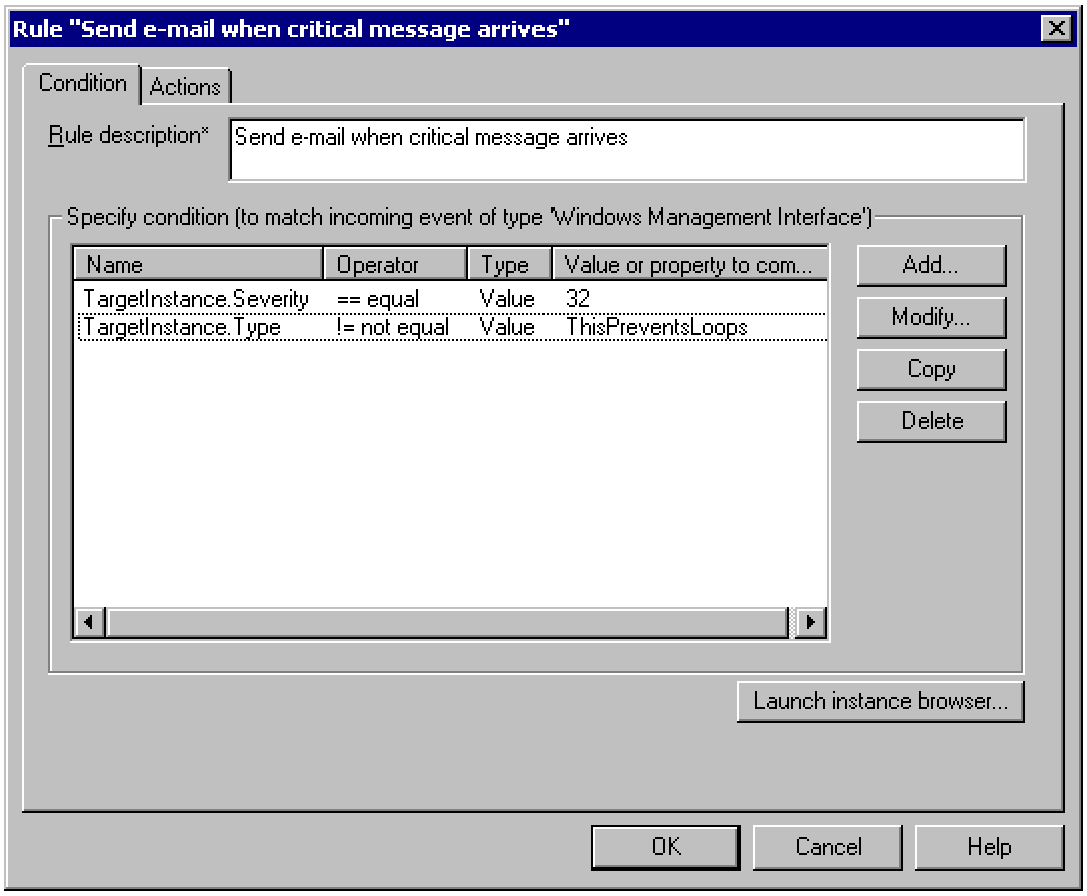
Message forwarding might fail under certain conditions and the operators should be informed about such an error. However, the system must not attempt to forward this exact error message that tells the operator that forwarding has failed, because forwarding it will most probably fail again, creating an endless loop which floods the message browser with error messages.

To prevent such a loop, set a certain message property to a unique value. For example, set the Type or Application property of the *Could not forward message...* message that you are creating in the WMI policy to something like *Forwarder* or *ThisPreventsLoops* or any other unique value.



Service ID	Empty
hosted on	Empty
Message Key	Empty
Message Type	ThisPreventsLoops
Message Group	Empty
Application	Empty
Object	Empty
Node	<$MSG_NODE> (Local Node)
Severity	Critical
Message Text	Failed to forward message as email: '<$WBEM:TargetInstance

Next, create a corresponding condition so that messages with such a value are NOT forwarded, as shown in the following example:



Rule "Send e-mail when critical message arrives"

Rule description*: Send e-mail when critical message arrives

Specify condition (to match incoming event of type 'Windows Management Interface')

Name	Operator	Type	Value or property to com...
TargetInstance.Severity	== equal	Value	32
TargetInstance.Type	!= not equal	Value	ThisPreventsLoops

Related Topics:

- Forwarding messages to external applications
- Using WMI policies to intercept incoming messages
- Using automatic actions to forward messages

HPOM Application Integration Guide

The HPOM Application Integration Guide (AIG) provides the following information for integrating your applications into HPOM for Windows. Follow the links below for integration details.

- Function-naming Conventions
- Libraries on the Managed Nodes
- Using APIs in Internationalized Environments
- HP Software Partnerships

Review the legal disclaimer notice at the end of this help topic before using any of the materials contained in the HPOM Application Integration Guide.

NOTE:
The HP Operations agent API includes support for C/C++ and Java, as well as for every language that supports DCOM automation (for example, VB, VBScript, JScript, and so on). However, the agent message stream interface supports C APIs only. All of the APIs are built using Microsoft Visual Studio 2005.

The Application Integration Guide is divided into several categories:

- Application Programming Interfaces
- Data API
- Interface API
- Agent Message API
- Agent Monitor API
- Data Structures
- Message Stream Interfaces
- Database Schema for Message Action/Server
- Java API
- Command-Line Utilities

HPOM for Windows Application Integration Guide

The HPOM for Windows Application Integration Guide (AIG) provides the following information for integrating your applications into HPOM for Windows. Follow the links below for integration details.

YOUR USE OF THE AIG IS SUBJECT TO THE SOFTWARE LICENSE AGREEMENT STATED BELOW. YOU MAY ONLY USE THE AIG IF YOU READ AND AGREE TO THE SOFTWARE LICENSE AGREEMENT AND THE WARRANTY STATEMENT.

SOFTWARE LICENSE AGREEMENT

Licensed Software . The "Licensed Software" is part of the HP Operations Manager Software. It consists of a collection of APIs, source code and binary code that permits licensee to develop licensee's programs that interoperate with HP Operations Manager.

The HPOM AIG contains:

- Application Programming Interfaces ("APIs"), including Data API, Interface API, Agent Message API, Agent Monitor API, and Java API;
- Data Structures;
- Message Stream Interfaces;
- Database Schema for Message Action/Server;
- Command-line Utilities; and,
- Policy Management and Deployment (PMAD) (separate help document).

License Grant . HP grants you a non-exclusive, non-transferable license to (i) use the Licensed Software solely for internal development of applications that integrate with HP Operations Manager for Windows and (ii) reproduce and distribute only the library of binary linking images contained in the Licensed Software and then, only as integrated into the application developed under this license grant.

Ownership . The Licensed Software is owned and copyrighted by HP or its third party suppliers. Your license confers no title or ownership in the Licensed Software and is not a sale of any rights in the Licensed Software. HP's third party suppliers may protect their rights in the event of any violation of these License Terms.

WARRANTY STATEMENT

NO WARRANTY . YOU AGREE THAT THE HPOM AIG IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY DESCRIPTION. HP SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. YOU ASSUME THE ENTIRE RISK RELATING TO THE USE OR PERFORMANCE OF THE HPOM AIG.

LIMITATION OF LIABILITY . IN NO EVENT SHALL HP BE LIABLE FOR ANY DIRECT, INDIRECT, GENERAL, SPECIAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOSS OF PROFITS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY.

Agent Application Programming Interface Overview

The Agent Application Programming Interface (AIG) allows you to integrate you own applications and programs with HPOM. The guide includes the following sections:

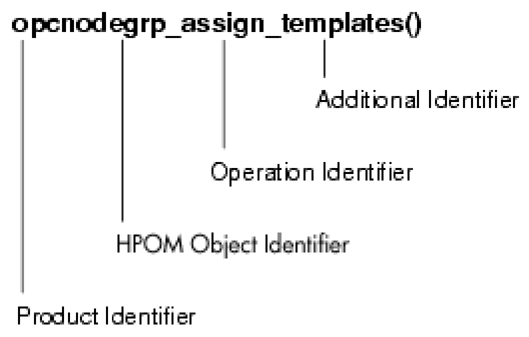
- Agent Application Programming Interface
- Message Stream Interface (MSI)
- Database Schema for Message/Action Server
- Database Schema for Policy Management
- Java API
- Command Line Utilities
- WMI Methods

NOTE:
The HP Operations agent API includes support for C/C++ and Java, as well as for every language that supports DCOM automation (for example, VB, VBScript, JScript, and so on). However, the agent message stream interface supports C APIs only. All of the APIs are built using Microsoft Visual Studio 2005.

Function-naming Conventions

The functions of the HPOM APIs have consistent names which reflect the operation they perform and the HPOM object on which they perform it. See Figure: Naming the HPOM API Functions for an example of how the HPOM API functions are named.

Naming the HPOM API Functions



The function names consist of the following parts:

product identifier
: Identifies the product, in HPOM this is always opc .

HPOM object identifier
: Identifies the HPOM object on which the function performs the operation.

operation identifier
: Identifies the operation which the function performs.

additional identifier
: Additional description to identify what the function does or expects.

Table: Overview of Function-name Combinations gives an overview of all available identifiers.

NOTE:
Not all operations are available on all HPOM objects and not every addition is available for each

operation.

Overview of Function-name Combinations

Product Identifier	HPOM Object Identifier	Operation Identifier	Additional Identifier	Additional Identifier
opc	appl	_add	_all	_layoutgrps
	applgrp	_modify	_list	_nodes
	data	_delete	_node	
	if	_get	_nodes	
	msg	_assign	_nodegrps	
	msggrp	_deassign	_templates	
	msgregrp	_move	_templgrps	
	node		_nodehier	
	nodegrp		_layoutgrp	
	nodehier		_layoutgrps	
	profile		_appls	
	reg		_applgrps	
	sync		_parentusers	
	templ		_profiles	
	templfile		_resps	
	templgrp		_defaults	
	transact			
	user			

Table: HPOM Objects gives an overview of all available HPOM objects which can be manipulated with the APIs. The opcdata type must be used to describe the objects when using the APIs. See HPOM Data Structures for more information about the opcdata types.

HPOM Objects

HPOM Object	Description	opcdata Type
Action Request	Action request to start an action on a managed node. Used by the Legacy Link Interface.	`OPCDTYPE_ACTION_REQUEST`
Action Response	Action response from a previously started action on a managed node. Used by the Legacy Link Interface.	`OPCDTYPE_ACTION_RESPONSE`
Annotation	Message annotation.	`OPCDTYPE_ANNOTATION`
Application	Application used in HPOM	`OPCDTYPE_APPLIC`
Application Configuration	Configuration of an HPOM application. This object type is used to configure HPOM applications.	`OPCDTYPE_APPL_CONFIG`
Application Group	Application group; an application group is a container of applications and other application groups.	`OPCDTYPE_APPL_GROUP`
Application Response	An application response is the response of a previously started HPOM application. Application responses can be received using the Application Response Interface; see also Interface API .	`OPCDTYPE_APPLIC_RESPONSE`
Container	A container contains a list of objects of one type.	`OPCDTYPE_CONTAINER`
Layout Group	A layout group contains a list of layout elements in a node hierarchy.	`OPCDTYPE_LAYOUT_GROUP`
Message	A message is the central management information element of the managed nodes.	`OPCDTYPE_MESSAGE`
Message Event	A message event is sent when a message was changed.	`OPCDTYPE_MESSAGE_EVENT`
Message Group	A message group is a grouping criteria of incoming messages.	`OPCDTYPE_MESSAGE_GROUP`
Message ID	A message ID contains the unique identifier of a message.	`OPCDTYPE_MESSAGE_ID`
Monitor Message	A monitor message is a monitor value which can be sent using the Agent Monitor API	`OPCDTYPE_MONITOR_MESSAGE`
Node	A node is an HPOM managed node.	`OPCDTYPE_NODE`
Node Configuration	A node configuration is the configuration of an HPOM managed node. It contains all necessary parameters to specify a node with all its characteristics.	`OPCDTYPE_NODE_CONFIG`

Node Group	A node group collects several nodes.	`OPCDTYPE_NODE_GROUP`
Node Hierarchy	A node hierarchy is a tree structure containing node layout elements and nodes as its leaves.	`OPCDTYPE_NODEHIER`
Regroup Condition	A regroup condition regroups messages matching the specified condition.	`OPCDTYPE_REGROUP_COND`
Template	A template is used to configure message conditions on managed nodes.	`OPCDTYPE_TEMPLATE_INFO`
Template Group	A template group collects several templates and other template groups. Template groups are handled like templates.	`OPCDTYPE_TEMPLATE_INFO`
Template File	A template file contains the complete configuration of a template including its conditions. Template files are only used by the template file API.	[char *]
Template Info	A template info object contains the name, description, and type of a template. It can be used to get a list of all available templates instead of the complete template configuration.	`OPCDTYPE_TEMPLATE_INFO`
User Configuration	A user configuration contains the properties of an HPOM user.	`OPCDTYPE_USER_CONFIG`
User Profile	A user profile contains the properties of users and is assigned to users so that the user takes over the properties defined in the profile.	`OPCDTYPE_USER_PROFILE`

Libraries on the Managed Nodes

Instrumentation programs which use the HP Operations Agent APIs must be developed on a system with an HP Operations Agent installed, so that the HPOM shared libraries and `opcapi.h` header files are both available.

Examples of how to use the API functions from C, Java, and VB Script are available in the following folder on the management server:

`%OvInstallDir%\examples\OVOW\DevelopmentKit\Agent`

Platforms that support multi-threaded environments, for example DCE, must also supply reentrant system calls that work in this environment. Some platforms only supply reentrant libraries which also work for single-threaded applications. Some have separate libraries-a standard library and a reentrant library; for example, `libc` and `libc_r` , or `libsocket` and `libsocket_r` .

On platforms with two sets of libraries, it is important to link the application using the standard library to the `crt0` object file, and the reentrant library using the `crt0_r` object file. `crt0` and `crt0_r` contain code that is executed before `main()` and is responsible for setting up or initializing the environment before calling any of the library APIs. Mixing reentrant and non-reentrant `crt0` and libraries is not allowed.

The HP Operations agent for DCE is a multi-threaded application and thus requires and uses reentrant libraries. Consequently, the HPOM library `libopc_r` is using reentrant calls from various libraries. To use the APIs in the HPOM library, they must be linked correctly. Applications that use the DCE HPOM library must be linked as a multi-threaded application; see below for details for each managed node platform.

Include file locations

Operating system	DCE agents	HTTPS agents
Windows	`%OvAgentDir%\include\opcapi.h`	`%OvInstallDir%include\opcapi.h`
HPUX	`/opt/OV/include/opcapi.h`	`/opt/OV/include/opcapi.h`
Linux	`/opt/OV/include/opcapi.h`	`/opt/OV/include/opcapi.h`
Solaris	`/opt/OV/include/opcapi.h`	`/opt/OV/include/opcapi.h`
Tru64	`/usr/opt/OV/include/opcapi.h`	`/usr/opt/OV/include/opcapi.h`
AIX	Not applicable.	`/usr/lpp/include/opcapi.h`

Libraries for DCE agents

Platform	Library	Link options
Windows	`%OvAgentDir%\lib\opcapi.lib`	Add library path `$(OvAgentDir)\lib` and library module `opcapi.lib` to the Microsoft Visual Studio project file.
HPUX PA	`/opt/OV/lib/libopc_r.sl`	`-L/opt/OV/lib -lopc_r -lnsp`
HPUX IA	`/opt/OV/lib/hpux32/libopc_r.so`	`-L/opt/OV/lib/hpux32 -lopc_r -lnsp`
Linux	`/opt/OV/lib/libopc_r.so`	`-L/opt/OV/lib -lopc_r -lnsp`
Solaris	`/opt/OV/lib/libopc_r.so`	`-L/opt/OV/lib -lopc_r -lnsp`
Tru64	`/usr/opt/OV/lib/libopc_r.so`	`-L/usr/opt/OV/lib -lopc_r -lnsp`

Lightweight libraries for HTTPS agents

HTTPS agents version 8.53 or higher provide lightweight libraries, which use less memory and provide better performance than previous libraries. Link the lightweight libraries if you develop new applications that use HP Operations Agent APIs.

The lightweight libraries provide the same interfaces as the previous libraries. Therefore, you can recompile existing applications to link the lightweight libraries.

Examples of how to use the lightweight libraries are available in the following folder on nodes that have the HTTPS agent version 8.53 or higher:

```
<OvInstallDir>/examples/copcagtapi
```

Operating system	Libraries	
Windows [a] [b]	32 bit	`%OvInstallDir%\bin\libopcagtapi.dll`
	64 bit	`%OvInstallDir%\bin\win64\libopcagtapi.dll`
HPUX PA [b]	`/opt/OV/lib/libopcagtapi.sl`	
HPUX IA [b]	`/opt/OV/lib/hpux32/libopcagtapi.so`	
Linux [a] [b]	32 bit	`/opt/OV/lib/libopcagtapi.so`
	64 bit [c]	`/opt/OV/lib64/libopcagtapi.so`

Solaris [b]	32 bit	`/opt/OV/lib/libopcagtapi.so`
	64 bit [d]	`/opt/OV/lib64/libopcagtapi.so`
Tru64 [b]	`/usr/opt/OV/lib/libopcagtapi.so`	
AIX [b]	32 bit	`/usr/lpp/OV/lib/libopcagtapi.a`
	64 bit [d]	`/usr/lpp/OV/lib64/libopcagtapi.a`

[a] On operating systems for which the agent provides both 64 bit and 32 bit lightweight libraries, link the appropriate library for your program (for example, link the 32 bit libraries to a 32 bit program, even if the program runs on a 64 bit operating system).

[b] To use the lightweight library on UNIX and Linux operating systems, you must also link the following HP BTO Software shared library:

Operating system	Libraries	
HPUX PA	`/opt/OV/lib/libOvXpl.sl`	
HPUX IA	`/opt/OV/lib/libOvXpl.so`	
Linux	32 bit	`/opt/OV/lib/libOvXpl.so`
	64 bit	`/opt/OV/lib64/libOvXpl.so`
Solaris	32 bit	`/opt/OV/lib/libOvXpl.so`
	64 bit [d]	`/opt/OV/lib64/libOvXpl.so`
Tru64	`/usr/opt/OV/lib/libOvXpl.so`	
AIX	32 bit	`/usr/lpp/OV/lib/libOvXpl.so`
	64 bit [d]	`/usr/lpp/OV/lib64/libOvXpl.so`

[c] The 64 bit libraries that are included with 32 bit Linux agents do not support message stream interface functions. To compile a 64 bit application that uses message stream interface functions, link the 64 bit libraries from a 64 bit Linux agent.

[d] Available in HTTPS agents version 8.60 or higher.

Legacy libraries for HTTPS agents

HTTPS agents version 8.52 or lower provide libraries that use more resources than the lightweight libraries. HP intends to remove these legacy libraries in a forthcoming version of the HTTPS agent. You should therefore only use the legacy libraries if your application must be compatible with HTTPS agent version 8.52 or lower.

Operating system	Libraries	
Windows [1] [2]	32 bit	`%OvInstallDir%\bin\opcapi.dll` `%OvInstallDir%\bin\libopc.dll`
	64 bit	`%OvInstallDir%\bin\win64\opcapi.dll` `%OvInstallDir%\bin\win64\libopc.dll`
HPUX PA [3]	`/opt/OV/lib/libopc_r.sl`	
HPUX IA [3]	`/opt/OV/lib/hpux32/libopc_r.so`	
Linux [2] [3]	32 bit	`/opt/OV/lib/libopc_r.so`
	64 bit [4]	`/opt/OV/lib64/libopc_r.so`
Solaris [3]	`/opt/OV/lib/libopc_r.so`	
Tru64 [3]	`/usr/opt/OV/lib/libopc_r.so`	
AIX [3]	`/usr/lpp/OV/lib/libopc_r.a`	

[1] On Windows operating systems, `libopc.dll` is the agent library, and `opcapi.dll` is the agent API library.

[2] On operating systems for which the agent provides both 64 bit and 32 bit libraries, link the appropriate libraries for your program (for example, link the 32 bit libraries to a 32 bit program, even if the program runs on a 64 bit operating system).

[3] To use the legacy libraries on UNIX and Linux operating systems, you must also link the following HP BTO Software shared libraries:

Operating system	Libraries
HPUX PA	`/opt/OV/lib/libOvBbc.sl` `/opt/OV/lib/libOvConf.sl` `/opt/OV/lib/libOvCtrl.sl` `/opt/OV/lib/libOvCtrlUtils.sl` `/opt/OV/lib/libOvDepl.sl` `/opt/OV/lib/libOvSecCm.sl` `/opt/OV/lib/libOvSecCore.sl` `/opt/OV/lib/libOvXpl.sl`
HPUX IA Solaris	`/opt/OV/lib/libOvBbc.so` `/opt/OV/lib/libOvConf.so` `/opt/OV/lib/libOvCtrl.so`

	/opt/OV/lib/libOvCtrlUtils.so /opt/OV/lib/libOvDepl.so /opt/OV/lib/libOvSecCm.so /opt/OV/lib/libOvSecCore.so /opt/OV/lib/libOvXpl.so	
Linux	32 bit	/opt/OV/lib/libOvBbc.so /opt/OV/lib/libOvConf.so /opt/OV/lib/libOvCtrl.so /opt/OV/lib/libOvCtrlUtils.so /opt/OV/lib/libOvDepl.so /opt/OV/lib/libOvSecCm.so /opt/OV/lib/libOvSecCore.so /opt/OV/lib/libOvXpl.so
	64 bit	/opt/OV/lib64/libOvBbc.so /opt/OV/lib64/libOvConf.so /opt/OV/lib64/libOvCtrl.so /opt/OV/lib64/libOvCtrlUtils.so /opt/OV/lib64/libOvDepl.so /opt/OV/lib64/libOvSecCm.so /opt/OV/lib64/libOvSecCore.so /opt/OV/lib64/libOvXpl.so
Tru64	/usr/opt/OV/lib/libOvBbc.so /usr/opt/OV/lib/libOvConf.so /usr/opt/OV/lib/libOvCtrl.so /usr/opt/OV/lib/libOvCtrlUtils.so /usr/opt/OV/lib/libOvDepl.so /usr/opt/OV/lib/libOvSecCm.so /usr/opt/OV/lib/libOvSecCore.so /usr/opt/OV/lib/libOvXpl.so	
AIX	/usr/lpp/OV/lib/libOvBbc.so /usr/lpp/OV/lib/libOvConf.so /usr/lpp/OV/lib/libOvCtrl.so /usr/lpp/OV/lib/libOvCtrlUtils.so /usr/lpp/OV/lib/libOvDepl.so /usr/lpp/OV/lib/libOvSecCm.so /usr/lpp/OV/lib/libOvSecCore.so /usr/lpp/OV/lib/libOvXpl.so	

[4] The 64 bit libraries that are included with 32 bit Linux agents do not support message stream interface functions. To compile a 64 bit application that uses message stream interface functions, link the 64 bit libraries from a 64 bit Linux agent.

Using APIs in Internationalized Environments

All HPOM API functions are internationalized. This means that they will initialize the language setting, check the codeset for compatibility, and convert codesets if necessary, provided your API programs support Native Language Support (NLS) environments.

When writing API programs for internationalized environments, you must ensure that your programs do select the appropriate locale. In C programs, you do this by calling the function `setlocale()` at the beginning of your program.

It is recommended to use `setlocale(LC_ALL,"")` . The category `LC_ALL` names the program's entire locale. `""` adopts the setting of the current shell.

HP Software Partnerships

The major benefit resulting from an integration with HPOM is the increased customer value of the integrated solution. HPOM is the standard for problem management and supports a wide range of platforms which have either been developed internally, or by partners. When you integrate a solution with HPOM, it becomes part of a comprehensive management solution which meets customers' requirements for a unified system management approach. This increases the value your solution provides to customers, making it attractive to market segments that it couldn't previously address. A partner program has been established by Hewlett-Packard to support your integration efforts.

Integrations created by solution partners can be validated and certified by Hewlett-Packard to achieve the status of HP Software Platinum Partner . Validation ensures that the integration is well-behaved and does not conflict with other integrated solutions. As an HP Software Platinum Partner, your solution is recommended by HP sales channels, you can leverage from the HP Software name, and you receive immediate market exposure for your solution through HP market awareness and selling tools.

For more information about the HP Software Partner programs, see our web site at `http://openview.hp.com` , and select `partners` .

HP Software Developer Assist

HP Software Developer Assist support that increases the speed, ease, and cost effectiveness of integrating with HPOM. For additional documentation and ordering information, see our web site at `http://www.openview.hp.com` , select `partners` , `developers' and third-party applications` , and `developer support services` .

Data API

The HPOM Data API provides a set of functions to set and get information in the form of HPOM data structures. Direct access to HPOM objects is not supported. This API is used for:

- HPOM Data Structures

  (See HPOM Data Structures for more information.)

Usage

To use the functions, include the header file `opcapi.h` in your application. Each routine returns an error/status code.

Prerequisites

The API functions can be issued by any user. For some attribute values a maximum length applies as noted with the appropriate attribute selector described in HPOM Data Structures .

Multithread Usage

All functions of the HPOM Data API are safe to be called by multithreaded applications, and are thread-safe for both POSIX Threads and DCE User Threads. They are neither async-cancel, async-signal, nor fork-safe, and cannot be safely called kernel threads.

opcdata_clear()

```
#include opcapi.h

int opcdata_clear (
        opcdata * data      /*in/out*/
                  );
```

Parameters

data

Points to the data area that will be cleared.

Description

Frees an re-initializes all fields in data .

NOTE:
This function changes the pointer to the data structure.

Return Values

OPC_ERR_OK:

OK

OPC_ERR_INVALID_INPARAM:

parameter data is invalid; probably NULL

OPC_ERR_CANT_INIT:

unable to initialize

OPC_ERR_NO_MEMORY:

memory allocation failed

Versions

HPOM for Windows A.07.50 and later

opcdata_copy()

```
#include opcapi.h

int opcdata_copy(
       const         opcdata data,           /* in */
       opcdata       *copy                   /* out */
                );
```

Parameters

data
: HPOM data structure that will be copied.

copy
: Copy of data .

Description

The API creates a copy of the data area and returns it in copy . It creates a complete copy, that is, the string fields of data are copied and not shared between data and copy . The allocated memory has to be deallocated using opcdata_free() before using this function.

Return Values

OPC_ERR_OK:
: OK

OPC_ERR_INVALID_INPARAM:
: data is NULL or of wrong type

OPC_ERR_INVALID_OUTPARAM:
: copy is invalid; probably NULL

OPC_ERR_NO_MEMORY:
: memory allocation failed

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Data Structures

opcdata_create()

```
   #include opcapi.h

   int opcdata_create(
                    int          data_type,     /* in */
                    opcdata      *data          /* out */
                    );
```

Parameters

`data_type`

Specifies the type of the allocated data area;
see HPOM Data Structures for a list of supported data structures.

`data`

HPOM data structure.

Description

This function allocates and initializes a data structure. To get or set attributes, the respective routines must be called. The memory used for the area must be deallocated by calling opcdata_free() .

Return Values

`OPC_ERR_OK:`

OK

`OPC_ERR_INVALID_OUTPARAM:`

data is invalid; probably NULL

`OPC_ERR_CANT_INIT:`

unable to initialize

`OPC_ERR_NO_MEMORY:`

memory allocation failed

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Data Structures

opcdata_free()

```
#include opcapi.h

int opcdata_free(
              opcdata        *data        /* in/out */
                );
```

Parameters

data
: Pointer to the data area that will be deallocated. data will be reset to NULL.

Description

The function opcdata_free() deallocates memory previously allocated by one of the functions opcif_read() , opcdata_create() , and opcdata_copy() .

Return Values

OPC_ERR_OK:
: OK

OPC_ERR_INVALID_INPARAM:
: data is NULL or of wrong type

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Data Structures

opcdata_generate_id()

```
#include opcapi.h

int opcdata_generate_id (
        opcdata   data       /* in */
       ,const int attribute  /* in */
                        );
```

Parameters

data

Pointer to the opcdata structure.

attribute

Specified which ID should be set.

Description

Generates an HPOM uuid and puts it into the ID field (specified in attribute) of the given opcdata structure.

Return Values

OPC_ERR_OK:

OK

OPC_ERR_INVALID_INPARAM:

Input parameter was not valid.

Versions

HPOM for Windows A.07.50 and later

opcdata_get_double()

```
#include opcapi.h

double *opcdata_get_double(
         const         opcdata data,           /* in */
         int           attribute               /* in */
                         );
```

Parameters

data
: HPOM data structure containing the queried attribute.

attribute
: Specifies the attribute that is queried.

Description

The function opcdata_get_double() returns the value of a double attribute in data .

Return Values

Returns the real value, or, OPC_DOUBLE_UNDEF if no value could be retrieved because data was empty or attribute was not allowed. Note that the real value could also be OPC_DOUBLE_UNDEF.

If an error occurs, for example an invalid attribute is specified, the function returns 0.0. .

Versions

HPOM for Windows A.07.50 and later

See Also

OPCDTYPE_MONITOR_MESSAGE

opcdata_get_error_msg()

```
#include opcapi.h

char* opcdata_get_error_msg (
        const int error_code        /* in  */
       ,char**    p_error_msg       /* out */
       ,int*      p_error_msg_size  /* out */
                            );
```

Parameters

error_code

OPC_ERR_XXX error code.

p_error_msg

Pointer to allocated error message string;
memory must be freed by caller.

p_error_msg_size

Size of allocated memory in bytes.

Description

Returns error text (description) relating to the given error code. The output is localized. This function is thread-safe.

NOTE:
The memory allocated by this function must be freed by the caller.

Return Values

string

String contains error description.

Versions

HPOM for Windows A.07.50 and later

opcdata_get_long()

```
#include opcapi.h

long opcdata_get_long(
         const      opcdata data,          /* in */
         int        attribute              /* in */
                    );
```

Parameters

`data`

Data structure containing the queried attribute.

`attribute`

Specifies the attribute that is queried.

Description

Returns the value of the numeric `attribute` in `data` .

Return Values

Returns the integer value of the attribute; if the routine fails, a value of -1 is returned.

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Data Structures

opcdata_get_str()

```
#include opcapi.h

char *opcdata_get_str(
         const       opcdata data,           /* in */
         int         attribute               /* in */
                     );
```

Parameters

`data`

Data structure containing the queried attribute.

`attribute`

Specifies the attribute that is queried.

Description

Instead of a status value, the function `opcdata_get_str()` returns a pointer to the desired string value. This function can, therefore, be used directly in another function call.

Return Values

Returns a character pointer to the value of the defined attribute in the data area. The pointer points into the internal data area. Modification of the attribute is only allowed using opcdata_set_str() ; direct access to the string is not supported.

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Data Structures

opcdata_report_error()

```
#include opcapi.h

char * opcdata_report_error (
        const int error_code      /* in */
                              );
```

Parameters

error_code

OPC_ERR_XXX error code.

Description

Returns the error text (description) relating to the given error code. On Novell NetWare systems, only the error number is returned in text form.

NOTE:
The memory allocated by this function must be freed by the caller.

Return Values

string

string contains error description

memory must be freed by caller

Versions

HPOM for Windows A.07.50 and later

opcdata_set_double()

```
#include opcapi.h

int *opcdata_set_double(
       opcdata         data,                /* in/out */
       int             attribute,           /* in */
       double          value                /* in */
                       );
```

Parameters

data

HPOM data structure containing the attribute to be set

attribute

Specifies the attribute.

value

Contains the value of the attribute to be set.

Description

The function opcdata_set_double() sets the numeric float attribute in data to value .

Return Values

OPC_ERR_OK:

OK

OPC_ERR_INVALID_INPARAM:

data is NULL, attribute is NULL, value is NULL

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Data Structures

opcdata_set_long()

```
#include opcapi.h

int opcdata_set_long(
                 opcdata      data,         /* in/out */
                 int          attribute,    /* in */
                 long         value         /* in */
                    );
```

Parameters

data
    HPOM data structure containing the attribute to be set.

attribute
    Specifies the attribute.

value
    Contains the value of the attribute to be set.

Description

Use the opcdata_set_long() routine to set the numeric long attribute in data to value .

Return Values

OPC_ERR_OK:
    OK

OPC_ERR_INVALID_INPARAM:
    data is NULL, attribute is invalid

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Data Structures

opcdata_set_str()

```
#include opcapi.h

int *opcdata_set_str(
        opcdata         data,              /* in */
        int             attribute,         /* in */
        const char      *value             /* in */
                    );
```

Parameters

data

HPOM data structure.

attribute

Select the attribute that is set.

value

Specify the value of the attribute.

Description

Use the function opcdata_set_str() to set the string attribute to a copy of value .

Return Values

OPC_ERR_OK:

OK

OPC_ERR_INVALID_INPARAM:

data is NULL, attribute is invalid, value is NULL

OPC_ERR_NO_MEMORY:

memory allocation failed

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Data Structures

opcdata_type()

```
#include opcapi.h

int opcdata_type (
        opcdata data  /* in */
                 );
```

Parameters

`data`

Points to an initialized data area.

Description

Returns the opcdata type in `data` .

Return Values

`OPC_ERR_OK:`

OK

`OPC_ERR_INVALID_INPARAM:`

`data` is invalid; probably NULL

Versions

HPOM for Windows A.07.50 and later

opcreg_copy()

```
#include opcapi.h

int   opcreg_copy (
      const opcregcond     reg_cond,       /* in */
      opcregcond           *copy           /* out */
                  );
```

Parameters

reg_cond
: Pointer to the registration condition to copy.

copy
: Address of the pointer to the copied condition.

Description

The function opcreg_copy() creates a complete copy of a registration condition and returns it. The allocated memory has to be deallocated using opcreg_free() .

Return Values

OPC_ERR_OK:
: No error occurred.

OPC_ERR_NO_MEMORY:
: Memory allocation error.

OPC_ERR_INVALID_OUTPARAM:
: One of the output parameters is NULL or not of the correct type.

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_create()

opcreg_free()

opcreg_get_long()

opcreg_get_str()

opcreg_set_long()

opcreg_set_str()

opcreg_create()

```
#include opcapi.h

int opcreg_create(
       opcregcond           *reg_cond           /* out */
                 );
```

Parameters

`reg_cond`
  HPOM registration condition structure.

Description

This routine creates an empty registration condition. The actual structure of this data area is hidden from the user. To get or set attributes, the respective routines must be called. The memory used for the area has to be deallocated by calling opcreg_free() .

Return Values

`OPC_ERR_OK:`
  OK

`OPC_ERR_INVALID_OUTPARAM:`
  `reg_cond` is invalid

`OPC_ERR_NO_MEMORY:`
  allocation of memory failed

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_copy()

opcreg_free()

opcreg_get_long()

opcreg_get_str()

opcreg_set_long()

opcreg_set_str()

opcreg_free()

```
   #include opcapi.h

   int   opcreg_free (
         opcregcond          *reg_cond          /* in/out */
                          )
```

Parameters

reg_cond

Address of the pointer to the registration condition.

Description

The function `opcreg_free()` deallocates the memory associated with a registration condition.

Return Values

`OPC_ERR_OK:`

No error occurred.

`OPC_ERR_INVALID_OUTPARAM:`

One of the output parameters is NULL or not of the correct type.

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_create()

opcreg_copy()

opcreg_get_long()

opcreg_get_str()

opcreg_set_long()

opcreg_set_str()

opcreg_get_long()

```
#include opcapi.h

int   opcreg_get_long (
      const opcregcond     reg_cond,       /* in */
      int                  field           /* in */
                      );
```

Parameters

`reg_cond`

Registration condition containing the numeric value.

`field`

Specifies the attribute in the registration condition.

Description

Use the function `opcreg_get_long()` to access the attribute values of a condition.

Return Values

Returns the requested long value, or, if not successful -1.

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_create()

opcreg_free()

`opcreg_copy()`

`opcreg_get_str()`

`opcreg_set_long()`

`opcreg_set_str()`

opcreg_get_str()

```
#include opcapi.h

int   *opcreg_get_str (
      const opcregcond     reg_cond,       /* in */
      int                  field           /* in */
                        );
```

Parameters

reg_cond
: Registration condition containing the string.

field
: Specifies the attribute in the registration condition.

Description

Use the function opcreg_get_str() to access the string attribute of a registration condition.

Return Values

Returns a pointer to the requested string or if not successful a NULL pointer.

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_create()

opcreg_free()

opcreg_copy()

opcreg_get_long()

opcreg_set_long()

opcreg_set_str()

opcreg_set_long()

```
#include opcapi.h

int   opcreg_set_long (
      const opcregcond     reg_cond,       /* in/out */
      int                  field,          /* in */
      long                 value           /* in */
                        );
```

Parameters

`reg_cond`

Registration condition containing the string.

`field`

Selects the attribute in the registration condition.

`value`

Specifies the value of the attribute in the registration condition.

Description

The function `opcreg_set_long()` sets the value of a numeric field of a registration condition.

Return Values

`OPC_ERR_OK:`

No error occurred.

`OPC_ERR_INVALID_FIELD:`

Invalid value used for `field` .

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_create()

opcreg_free()

opcreg_copy()

opcreg_get_long()

opcreg_get_str()

opcreg_set_str()

opcreg_set_str()

```
#include opcapi.h

int   *opcreg_set_str (
      const opcregcond     reg_cond,       /* in/out */
      int                  field,          /* in */
      const char           *value          /* in */
                      );
```

Parameters

`reg_cond`
    Registration condition containing the string.

`field`
    Selects the attribute in the registration condition.

`value`
    Specifies the value of the attribute in the registration condition.

Description

The function `opcreg_set_str()` sets the value of a string field of a registration condition.

Return Values

`OPC_ERR_OK:`
    No error occurred.

`OPC_ERR_INVALID_FIELD:`
    Invalid value used for `field` .

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_create()

opcreg_free()

opcreg_copy()

opcreg_get_long()

opcreg_get_str()

opcreg_set_long()

HPOM Interface API

This API provides access to the HPOM Interfaces. HPOM for Windows provides the following interfaces:

- HPOM Message Stream Interface

  This interface makes it possible to read HPOM messages from the internal message stream and to write messages into the internal message stream. The Message Stream Interface (MSI) is available on the HPOM Managed Nodes. All MSI types establish a connection to the HPOM message agent. This interface is divided into the types:

  - `OPCAGTIF_EXTMSGPROC_READ`

    This interface type is used for non-destructive read operations on the HPOM internal message flow. Only messages that are allowed to be output on the Agent MSI are accessible using this interface type. This type of interface is typically used by statistical analysis tools or additional display facilities.

  - `OPCAGTIF_EXTMSGPROC_READWRITE`

    This interface type is used to read messages from HPOM's internal message flow, to modify / create messages, and to write them back to the HPOM processes. Only messages which are allowed to be output on the MSI are accessible using this interface type. Messages tagged with 'copy to' remain in HPOM's message flow, whereas messages tagged with 'divert to' are taken out of the flow. This type of interface could be used by event correlation engines.

  - `OPCAGTIF_EXTMSGPROC_WRITE`

    Interface instances of this type are used for write-only applications, to feed messages into HPOM's message flow. This type of interface could also be used to encapsulate the opcif_write() routine in a command line interface.

Prerequisites

The API functions must be issued by the agent user.

Multithread Usage

All functions of the Interface API are safe to be called by multithreaded applications and are thread safe for both POSIX Threads and DCE User Threads. They are neither async-cancel, async-signal, nor fork-safe, and cannot be called safely i kernel threads.

`opcreg_copy() is not thread safe for POSIX threads or for DCE User Threads.`

Registration conditions of the HPOM Interface API

HPOM provides a user-accessible data type to define registration conditions as the mechanism to register with the HPOM Interfaces.

HPOM provides a set of APIs to create an empty condition, modify or query condition fields and to duplicate or delete a condition definition from memory.

Related Topics:

- Security considerations for the HPOM Interface API

Security Considerations for the HPOM Interface API

Because an event subscription service API is a window for applications to see generic system-wide activity, applications must be prevented from unauthorized snooping of system behavior at this access point. In addition, access to the HPOM message flow in read-write mode allows an external application to discard messages, without a user being made aware that a message was generated. The APIs must, therefore, apply authentication mechanisms to prevent users and applications from unauthorized access to the HPOM message flow.

Automatic/Operator-initiated Actions

One important and critical issue arising from these security considerations is whether external applications using the interfaces are allowed to define automatic actions, operator-initiated actions, or both. If HPOM allows access to these message attributes, any user who is authorized to call the APIs is also able to execute actions on HPOM managed nodes.

According to the current HPOM concept, which regards HPOM as an open application providing a high level of flexibility to integrate applications, HPOM allows external programs to define actions for messages that are passed to the message agent. Event correlation can be seen as an advance on the existing concept of message conditions ("if attributes match then set attributes and actions") to a higher level ("if rule fires then set attributes and actions"). It is, therefore, essential that these external applications are allowed to perform these modifications.

An appropriate authorization mechanism at the API level guarantees that only authorized users can apply the APIs. However, as the checking of a user ID belongs to the OS level with its superuser concept, this conflicts somewhat with the existing HPOM concept where the administrator is responsible for the configuration of user roles.

HPOM for Windows provides a possibility to enable and disable the interface functionality. In addition, you can configure whether actions can be defined by an application that is writing to the interface. This concerns all interface types.

You can also define whether each message is allowed for output to the Message Stream Interface in the HPOM for Windows policy editors. For example, an administrator can prevent the output of certain messages so that external applications do not receive secure information by reading these messages from the HPOM message flow.

Summary

HPOM allows users with a user ID of zero (uid 0), typically root, on UNIX and users that are in the Administrators group on Windows to access the HPOM Interface APIs and to define actions for messages that are sent to the management server. The HPOM for Windows administrator can enable or disable the interface functionality of the interface types concerning the message flow and allow or disallow actions that are read from the interface.

Per default the interfaces are disabled and it is not allowed to define actions.

To enable the message stream interface on a managed node create a nodeinfo policy containing

OPC_AGTMSI_ENABLE TRUE

and deploy it to the managed nodes.

If actions are disallowed, an appropriate error text is added to the annotations field and the action disabled.

To allow the definition of automatic actions add the following to the nodeinfo policy:

OPC_AGTMSI_ALLOW_AA TRUE

To allow the definition of operator initiated actions add the following to the nodeinfo policy:

OPC_AGTMSI_ALLOW_OA TRUE

opcif_open()

```
#include opcapi.h

int opcif_open(
               int          interface_type,     /* in */
               const char   *instance,          /* in */
               int          mode,               /* in */
               int          max_entries,        /* in */
               int          *interface_id       /* out */
              );
```

Parameters

`interface_type`

Specifies the type of interface to use from:

Agent Message Stream Interface

Used by external message processors (for example, event correlation engines):

OPCAGTIF_EXTMSGPROC_READ
OPCAGTIF_EXTMSGPROC_READWRITE
OPCAGTIF_EXTMSGPROC_WRITE

`instance`

Name of the interface instance which is registered for one of the interface types above. The name is limited to a length of 12 alpha-numeric characters because it is also part of the queue file name.

`mode`

To specify whether the interface is opened if the HPOM processes are not running. Use either:

- OPCIF_ALWAYS (default)
- OPCIF_AGT_RUNNING

The following options specify whether the `opcif_read()` API will wait for available data or not; in the WAIT case, the calling process will be blocked until data is available or the process receives an interrupt:

- OPCIF_READ_WAIT (default)

- OPCIF_READ_NOWAIT

To specify the handling of unread messages if the connected process closes the interface or aborts, use one of the following options:

- OPCIF_CLOSE_FORWARD (default)
- OPCIF_CLOSE_DISCARD

In the first case, messages in the read-queue are appended to the write-queue; in the second, these messages are discarded when the external program closes the interface.

It is possible to combine these options using the '|' operator.

max_entries
: Specifies the maximum number of entries in the read-queues. If this number is exceeded, HPOM stops writing to the queue. When the reading process has emptied the queue, it is notified by an error value returned by opcif_read() . The application must then disconnect and reopen the interface. To disable this check, specify 0 for max_entries.

interface_id
: The returned value must be used in subsequent calls to the APIs to refer to this instance of the interface.

Description

Use the function opcif_open() to connect to an instance of one of the following interfaces:

- Agent Message Stream Interface

Return Values

OPC_ERR_OK:
: interface correctly opened

OPC_ERR_INVALID_OUTPARAM:
: pointer to interface_id is invalid

OPC_ERR_ACCESS_DENIED:
: access denied

OPC_ERR_INVALID_INTERFACE_INSTANCE:

instance name contains invalid characters, or is too long

`OPC_ERR_INVALID_INTERFACE_TYPE:`
no such interface type

`OPC_ERR_CANT_INIT:`
initialization of queues failed

`OPC_ERR_CANT_OPEN_READQUEUE:`
unable to open readqueue

`OPC_ERR_CANT_OPEN_WRITEQUEUE:`
unable to open writequeue

`OPC_ERR_CANT_INFORM_MSGA:`
informing message agent failed

`OPC_ERR_NO_MEMORY:`
memory allocation failed

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Interfaces

opcif_get_pipe()

```
#include opcapi.h

int opcif_get_pipe(
                 int       interface_id,       /* in */
                 int       pipefd              /* out */
                  );
```

Parameters

interface_id
: Specifies which interface instance is used.

pipefd
: Returns the file descriptor of the selected output interface queue.

Description

A program reading from several input files needs the pipe file descriptor of its interface input queue. This descriptor is then used as part of the parameters to select(2) or fcntl(2) . The function opcif_get_pipe() returns this value.

For convenience reasons, an application that reads only from the HPOM interface can specify OPCIF_READ_WAIT in the opcif_open() call.

Return Values

OPC_ERR_OK
: OK

OPC_ERR_CANT_INIT
: initialization of queues failed

OPC_ERR_INVALID_OUTPARAM
: pipefd is NULL

`OPC_ERR_INVALID_INTERFACE_ID`
: no such interface opened

Versions

HPOM Windows A.07.50 and later

See Also

HPOM Interfaces

opcif_read()

```
#include opcapi.h

int opcif_read(
              int           interface_id,     /* in */
              opcdata       data              /* in/out */
              );
```

Parameters

`interface_id`
: Specifies which interface instance is used.

`data`
: HPOM data structure.

Description

The function `opcif_read()` reads a message from the queue of the specified interface instance. If the interface instance has been opened with `OPCIF_READ_WAIT` , the calling process is blocked until the information is available. The API returns an error if the application receives an interrupt signal. The data parameter specified in the call must be created with opcdata_create() . Memory for the actual message data is allocated, and if memory was assigned to data before the call to `opcif_read()` , it is deallocated.

Return Values

`OPC_ERR_OK:`
: OK

`OPC_ERR_CANT_INIT:`
: initialization of queues failed

`OPC_ERR_INVALID_INTERFACE_ID:`
: no such interface opened

`OPC_ERR_INVALID_OUTPARAM:`

`data` is NULL or is of wrong type

`OPC_ERR_WRONG_MSITYPE:`
interface assigned to `interface_id` is of wrong type

`OPC_ERR_EINTR:`
reading from pipe failed

`OPC_ERR_MSI_BUF_FULL:`
number of messages exceeds specified number in max_entries while opening

`OPC_ERR_NO_DATA:`
queue is empty

`OPC_ERR_CANT_READ_MSG:`
reading message, event, or response failed

`OPC_ERR_NO_MEMORY:`
memory allocation failed

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Interfaces

HPOM Data Structures

opcif_register()

```
#include opcapi.h

int opcif_register(
           int                interface_id,     /* in */
           const opcregcond   reg_cond ,        /* in */
           long               *cond_id          /* out */
                  );
```

Parameters

`interface_id`

Specifies which interface instance is used.

`reg_cond`

- Messages:

  Defines the combination of message attributes that are checked; NULL registers for all messages
- Message Events:

  Defines an event mask and the restriction of message events of messages for certain operators
- Application Responses

  Defines a certain application response specified by the application response ID

`cond_id`

Returns an ID to reference this condition in a subsequent call to opcif_unregister() ; NULL is allowed if the API user is not interested in the ID (for example, if opcif_unregister() is not called later on).

Description

The function `opcif_register()` is used by an external application to register for the following attributes. See also opcregcond .

- Message Attributes

  HPOM supports registration for message type, message group, node name, object, severity and application attributes. You can also combine attributes (logical AND), and '|' within an attribute (logical OR). Multiple registrations (logical OR of registration conditions) are also possible by using a sequence of

API calls. The following attributes are supported:

- `OPCREG_MSGTYPE`
- `OPCREG_GROUP`
- `OPCREG_NODENAME`
- `OPCREG_OBJECT`
- `OPCREG_SEVERITY`
- `OPCREG_APPLICATION`

Return Values

`OPC_ERR_OK:`
    OK

`OPC_ERR_CANT_INIT:`
    initialization of queues failed

`OPC_ERR_INVALID_INTERFACE_ID:`
    no such interface opened

`OPC_ERR_CANT_INFORM_MSGA:`
    informing message agent failed

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Interfaces

opcif_unregister()

```
#include opcapi.h

int opcif_unregister(
           int              interface_id,        /* in */
           long             cond_id              /* in */
                    );
```

Parameters

`interface_id`

Specifies which interface instance is used.

`cond_id`

Specifies the registration condition to be removed; 0 unregisters for all messages, message events or application responses.

Description

To cancel prior registrations for messages, the external application calls `opcif_unregister()` with the value of reg_cond that was specified in the call to the registration API. As the registration mechanism is a positive filter, removing a registration condition does not mean that messages matched by this condition are filtered out after `opcif_unregister()` is called; instead, just that positive filter condition is cancelled.

By unregistering a condition for message events or application responses, information matching the unregistered condition will no longer received.

Return Values

`OPC_ERR_OK:`

OK

`OPC_ERR_CANT_INIT:`

initialization of queues failed

`OPC_ERR_INVALID_INTERFACE_ID:`
: no such interface opened

`OPC_ERR_CANT_INFORM_MSGM:`
: informing message manager failed

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Interfaces

opcif_write()

```
#include opcapi.h

int opcif_write(
           int                interface_id,     /* in */
           const opcdata      data              /* in */
               );
```

Parameters

`interface_id`
Specifies which interface instance is used.

`data`
HPOM data structure.

Description

Use the function `opcif_write()` to write a message to the HPOM Interface.

Depending on the type of interface, the message is written into the message queue of the message agent.

This function can only be used for interfaces of the following types:

- `OPCSVIF_EXTAGT_MESSAGE`

Return Values

`OPC_ERR_OK:`
OK

`OPC_ERR_CANT_INIT:`
initialization of queues failed

`OPC_ERR_INVALID_INTERFACE_ID:`

no such interface opened

`OPC_ERR_INVALID_OPCDATA_TYPE:`
data must be of type OPCDTYPE_MESSAGE

`OPC_ERR_CANT_WRITE_MSG:`
unable to write message

`OPC_ERR_WRONG_MSITYPE:`
interface type is invalid for this operation

Versions

HPOM for Windows A.07.50 and later

See Also

HPOM Interfaces

HPOM Data Structures

opcreg_copy()

```
#include opcapi.h

int opcreg_copy(
         const opcregcond    reg_cond,           /* in */
         opcregcond          *copy               /* out */
               )
```

Parameters

reg_cond
: HPOM registration condition structure.

copy
: Copy of reg_cond .

Description

The API creates a copy of the condition and returns it in copy. The allocated memory has to be deallocated using opcreg_free() . copy must be freed using opcreg_free() before calling this function.

Return Values

OPC_ERR_OK:
: OK

OPC_ERR_INVALID_OUTPARAM:
: copy is invalid

OPC_ERR_NO_MEMORY:
: allocation of memory failed

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_create()

opcreg_free()

opcreg_get_long()

opcreg_get_str()

opcreg_set_long()

opcreg_set_str()

opcreg_create()

```
   #include opcapi.h

   int opcreg_create(
          opcregcond           *reg_cond           /* out */
                   );
```

Parameters

`reg_cond`
HPOM registration condition structure.

Description

This routine creates an empty registration condition. The actual structure of this data area is hidden from the user. To get or set attributes, the respective routines must be called. The memory used for the area has to be deallocated by calling opcreg_free() .

Return Values

`OPC_ERR_OK:`
OK

`OPC_ERR_INVALID_OUTPARAM:`
`reg_cond` is invalid

`OPC_ERR_NO_MEMORY:`
allocation of memory failed

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_copy()

opcreg_free()

opcreg_get_long()

opcreg_get_str()

opcreg_set_long()

opcreg_set_str()

opcreg_free()

```
#include opcapi.h

int opcreg_free(
           opcregcond        *reg_cond       /* in/out */
               );
```

Parameters

reg_cond
: HPOM registration condition structure.

Description

The function opcreg_free() deallocates memory previously allocated by opcreg_create() , opcreg_copy() , or opcreg_set_...() .

Return Values

OPC_ERR_OK:
: OK

OPC_ERR_INVALID_OUTPARAM:
: reg_cond is invalid

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_copy()

opcreg_create()

opcreg_get_long()

opcreg_get_str()

opcreg_set_long()

opcreg_set_str()

opcreg_get_long()

```
#include opcapi.h

long opcreg_get_long(
         const opcregcond       reg_cond,       /* in */
         int                    field           /* in */
                    );
```

Parameters

`reg_cond`
  HPOM registration condition structure.

`field`
  Selects the attribute that is queried.

Description

Use the routine `opcreg_get_long()` to access the attribute values of a condition.

Return Values

Returns the integer value of the attribute; if the routine fails, - 1 is returned.

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_copy()

opcreg_create()

opcreg_free()

opcreg_get_str()

opcreg_set_long()

opcreg_set_str()

opcreg_set_long()

```
#include opcapi.h

int opcreg_set_long(
        opcregcond      reg_cond,            /* in/out */
        int             field,               /* in */
        long            value                /* in */
                    );
```

Parameters

`reg_cond`
: HPOM registration condition structure.

`field`
: Select the attribute that is set.

`value`
: Specify the value of the attribute.

Description

Use the function `opcreg_set_long()` to set attributes to a certain value.

Return Values

`OPC_ERR_OK`
: OK

`OPC_ERR_INVALID_FIELD`
: `field` is invalid

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_copy()

opcreg_create()

opcreg_free()

opcreg_get_long()

opcreg_get_str()

opcreg_set_str()

opcreg_set_str()

```
#include opcapi.h

int *opcreg_set_str(
        opcregcond        reg_cond,           /* in/out */
        int               field,              /* in */
        const char        *value              /* in */
                   );
```

Parameters

reg_cond

HPOM registration condition structure.

field

Select the attribute that is set.

value

Specify the value of the attribute.

Description

Use the function opcreg_set_str() to set attributes to a certain value.

Return Values

OPC_ERR_OK

OK

OPC_ERR_INVALID_FIELD

field is invalid

Versions

HPOM for Windows A.07.50 and later

See Also

opcreg_copy()

opcreg_create()

opcreg_free()

opcreg_get_long()

opcreg_get_str()

opcreg_set_long()

Agent Message API

HPOM provides a set of APIs to handle messages on managed nodes. These functions enable you, for example, to send messages and acknowledge them at a later time. See Agent Monitor API for functions to send monitor values.

Data Structures

OPCDTYPE_MESSAGE_ID

OPCDTYPE_MESSAGE

Usage

The managed node processes must be running. To use the functions, include the header file `opcapi.h` in your application.

Prerequisites

Each opdata structure must be allocated using opcdata_create() before it can be used in any of these functions. After the execution of your program, each opcdata structure must be freed using opcdata_free().

Multithread Usage

All function of the Agent Message API are safe to be called by multithreaded applications, and are thread-safe for both POSIX Thread and DCE User Threads. They are neither async-cancel, async-signal, nor fork-safe, and cannot be safely called in kernel threads.

Agent Configuration

Operations on messages out of managed nodes require to send these message operations to the manager. Unfortunately it is not possible to deliver the responsible manager of a message from the message ID. Additionally, the configuration could be changed since the message was sent so that it is necessary to send the message operation to all managers. This can produce a lot of network load.

To prevent this, the message agent holds information about the manager to which the messages were sent. After a defined time, the information is deleted to save memory, disk space, and processing time. This time is configurable with a nodeinfo policy using the parameter OPC_STORE_TIME_FOR_MGR_INFO. The specified value is the time in hours, with a default setting of one hour if this parameter is not changed.

```
OPC_STORE_TIME_FOR_MGR_INFO 2
```

The storage of the manager information must be enabled for each message to be sent by setting the message parameter OPCDATA_DATA_INFO to OPC_REMARK_FOR_ACK.

```
opcdata_set_long(message, OPCDATA_DATA_INFO, OPC_REMARK_FOR_ACK);
```

opcmsg()

opcagtmsg_send()

opcagtmsg_ack()

opcagtmsg_ack()

```
#include opcapi.h

int opcagtmsg_ack (
               opcdata     message_id     /* in */
                 );
```

Parameters

message_id
: Message ID of type OPCDTYPE_MESSAGE_ID .

Description

Use the function opcagtmsg_ack() to acknowledge a message out from a managed node. A message operation will be sent to the message agent.

If the message attribute OPCDATA_DATA_INFO of a previously sent message was set to OPC_REMARK_FOR_ACK , the message agent holds the information about the responsible manager in its memory. If this attribute was not set, the message operation will be sent to all managers.

Return Values

OPC_ERR_OK:
: OK

OPC_ERR_INVALID_INPARAM:
: message_id is NULL

OPC_ERR_INVALID_OPCDATA_TYPE:
: message_id is not of type OPCDTYPE_MESSAGE_ID

OPC_ERR_INCOMPLETE_PARAM:
: message ID is not set

`OPC_ERR_NO_MEMORY:`
memory allocation failed

Versions

HPOM for Windows A.06.00 and later

See Also

opcagtmsg_send()

opcmsg()

OPCDTYPE_MESSAGE_ID

opcagtmsg_send()

```
   #include opcapi.h

   int opcagtmsg_send (
        opcdata     message     /* in/out */
                      );
```

Parameters

message
: Message of type OPCDTYPE_MESSAGE .

Description

Use the function opcagtmsg_send() to send a message, created on the managed node, to its responsible manager. The message must be of type OPCDTYPE_MESSAGE . The message ID can be retrieved from the message object using opcdata_get_str() immediately after the send call was executed.

Only the message attributes Severity, Application, Message Group, Object, Message Text, Option Strings and Node are used in opcagtmsg_send() .

If you want to save the information about the responsible manager, remark the message to be acknowledged later. To do this, set OPCDATA_DATA_INFO to OPC_REMARK_FOR_ACK .

After opcagtmsg_send() was called with OPC_REMARK_FOR_ACK it is possible to get the ID of the sent message using:

opcdata_get_str() (message, OPCDATA_MSGID)

Return Values

OPC_ERR_OK:
: OK

OPC_ERR_APPL_REQUIRED:

attribute `OPCDATA_APPLICATION` not set

`OPC_ERR_OBJ_REQUIRED:`
attribute `OPCDATA_OBJECT` not set

`OPC_ERR_TEXT_REQUIRED:`
attribute `OPCDATA_MSGTEXT` not set

`OPC_ERR_INVAL_SEVERITY:`
set severity invalid

`OPC_ERR_MISC_NOT_ALLOWED:`
message group 'misc' not allowed

`OPC_ERR_INVALID_INPARAM:`
message is NULL
message is not of type OPCDTYPE_MESSAGE

`OPC_ERR_WRONG_OPTION_VARS:`
The field OPCDATA_OPTION_VAR of the message has an incorrect format. It can only contain assignments separated by spaces.

`OPC_ERR_NO_MEMORY:`
memory allocation failed

Versions

HPOM for Windows A.06.00 and later

See Also

opcagtmsg_ack()

opcmsg()

OPCDTYPE_MESSAGE

opcmsg()

```
#include opcapi.h

int opcmsg (
            const int      severity,      /* in */
            const char *   application,   /* in */
            const char *   object,        /* in */
            const char *   msg_text,      /* in */
            const char *   msg_group,     /* in */
            const char *   nodename,      /* in */
           );
```

Parameters

severity

Severity level of the new message.

The following severities are supported:
OPC_SEV_NORMAL
OPC_SEV_WARNING
OPC_SEV_MINOR
OPC_SEV_MAJOR
OPC_SEV_CRITICAL.

application

Application of the message source.

object

Object of the message source.

msg_text

Message text.

msg_group

Message group.

nodename

Name of the node originating the message.

Description

Use the function `opcmsg()` to send a message, created on the managed node, to the management server. This function does not return the message ID so that it is not possible to acknowledge the message later, on the managed node.

Return Values

`OPC_ERR_OK:`
: OK

`OPC_ERR_APPL_REQUIRED:`
: The application parameter is not set.

`OPC_ERR_OBJ_REQUIRED:`
: The object parameter is not set.

`OPC_ERR_TEXT_REQUIRED:`
: The msg_text parameter is not set.

`OPC_ERR_INVAL_SEVERITY:`
: The severity parameter value is invalid

`OPC_ERR_MISC_NOT_ALLOWED:`
: message group 'misc' is not allowed

`OPC_ERR_NO_MEMORY:`
: out of memory

Versions

HPOM for Windows A.06.00 and later

See Also

opcagtmsg_ack()

`opcagtmsg_send()`

Agent Monitor API

HPOM provides a set of functions to send monitor values to the monitor agent.

Data Structures

OPCDTYPE_MONITOR_MESSAGE

Usage

To use these functions, the managed node processes must be running. To use the functions, include the header file `opcapi.h` in your application.

Prerequisites

Each opdata structure must be allocated using opcdata_create() before it can be used in any of these functions.

Multithread Usage

All functions of the Agent Monitor API are safe to be called by multithreaded applications, and are thread-safe for both POSIX Threads and DCE User Threads. They are neither async-cancel, async-signal, nor fork-safe, and cannot be safely called in kernel threads.

opcmon()

opcagtmon_send()

opcagtmon_send()

```
#include opcapi.h

int opcagtmon_send (
              opcdata     mon_msg     /* in */
                  );
```

Parameters

mon_msg
: Monitor message/value of type: OPCDTYPE_MONITOR_MESSAGE .

Description

Use the function opcagtmon_send() to send a monitor value, created on the managed node, to the monitor agent. The mon_msg must be of type OPCDTYPE_MONITOR_MESSAGE .

Only the message attributes Monitor Name, Monitor Value, Object and Option String are used in opcagtmon_send() .

Return Values

OPC_ERR_OK:
OK

OPC_ERR_INVALID_INPARAM:
mon_msg is NULL
mon_msg is not of type OPCDTYPE_MONITOR_MESSAGE

OPC_ERR_OBJNAME_REQUIRED:
attribute OPCDATA_MON_VAR not set

OPC_ERR_NO_AGENT:
agent is not running

`OPC_ERR_NO_MEMORY:`
out of memory

`OPC_ERR_WRONG_OPTION_VARS:`
attribute `OPCDATA_OPTION_VAR` not set correctly

Versions

HPOM for Windows A.06.00 and later

See Also

opcmon()

OPCDTYPE_MONITOR_MESSAGE

opcmon()

```
#include opcapi.h

int opcmon (
         const char     *objname,     /* in */
         const double   monval        /* in */
          );
```

Parameters

objname
: Name of the monitored object.

monval
: Actual value of the monitored object.

Description

Use the function opcmon() to send a monitor value, created on the managed node, to its responsible management server.

Return Values

OPC_ERR_OK:
: OK

OPC_ERR_OBJNAME_REQUIRED:
: objname is NULL

OPC_ERR_NO_AGENT:
: agent is not running

OPC_ERR_NO_MEMORY:
: out of memory

Versions

HPOM for Windows A.06.00 and later

See Also

opcagtmon_send()

HPOM Data Structures

HPOM provides a set of data structures which hold information about HPOM objects.

- OPCDTYPE_MESSAGE
- OPCDTYPE_MESSAGE_ID
- OPCDTYPE_MONITOR_MESSAGE

The attributes of each of these data types are described in the following tables. The tables show which of the attributes can only be retrieved and which can be set.

- Functions to retrieve attributes
  - opcdata_get_long()
  - opcdata_get_str()
  - opcdata_get_double()
- Functions to set attributes
  - opcdata_set_long()
  - opcdata_set_str()
  - opcdata_set_double()

The description also includes information about the type of the attribute value (long, string, or double), and, if the attribute value is a string, the maximum character length (for example, `str[32]`).

The available predefined values are defined in the include file `opcapi.h`

OPCDTYPE_MESSAGE

Table: OPCDTYPE_MESSAGE lists the attributes that are available for the Message Attribute data structure.

OPCDTYPE_MESSAGE

Attribute	Scope	Type	Properties			Description
`OPCDATA_DATATYPE`	get	long				Returns the type of the opcdata object.
`OPCDATA_SEVERITY`	get/set	long				Severity of the message. Possible values are: ■ `OPC_SEV_UNCHANGED` ■ `OPC_SEV_UNKNOWN` ■ `OPC_SEV_NORMAL` ■ `OPC_SEV_WARNING` ■ `OPC_SEV_CRITICAL` ■ `OPC_SEV_MINOR` ■ `OPC_SEV_MAJOR`
`OPCDATA_CREATION_TIME`	get/set	long				Time the message was created. The time is in UNIX format (seconds since Epoch). Default: the (local) time when the message was created.
`OPCDATA_RECEIVE_TIME`	get	long				Time the message was received by the management server.
`OPCDATA_AACTION_ACK`	get/set	long				Auto Acknowledge after successful execution of the Automatic Action 0 (default): do not auto-acknowledge 1: auto-acknowledge.
`OPCDATA_AACTION_ANNOTATE`	get/set	long				Defines whether HPOM creates start and end annotations for the automatic action. Possible values for the attribute are:

						0 (default): do not create annotations 1: create annotations.
OPCDATA_AACTION_STATUS	get/set	long				Status of the automatic action: ■ OPC_ACTION_UNDEF (default): no automatic action for this message defined. ■ OPC_ACTION_DEF : the automatic action for this message is defined but was not yet started. ■ OPC_ACTION_STARTED : the automatic action for this message is defined and was already started. ■ OPC_ACTION_FINISHED : the automatic action was started and completed successfully. ■ OPC_ACTION_FAILED : the automatic action was started and failed.
OPCDATA_OPACTION_ACK	get/set	long				Automatically acknowledge a message after a successful execution of the operator-initiated action. Possible values: 0 (default): do not auto-acknowledge 1: auto-acknowledge.
OPCDATA_OPACTION_ANNOTATE	get/set	long				Defines whether HPOM creates start and end annotations for the operator-initiated action. Possible values for the attribute are: 0 (default): do not create annotations 1: create annotations.
OPCDATA_OPACTION_STATUS	get	long				Status of the action: ■ OPC_ACTION_UNDEF ■ OPC_ACTION_DEF ■ OPC_ACTION_STARTED

						■ OPC_ACTION_FINISHED ■ OPC_ACTION_FAILED
OPCDATA_ESCALATED	get	long				Message was escalated: ■ OPC_ESCALATED_TO ■ OPC_ESCALATED_FROM The escalation server can be retrieved using OPCDATA_ESCALATION_SERVER .
OPCDATA_NOTIFICATION	get/set	long				Notification.
OPCDATA_TROUBLETICKET	get/set	long				Forward message to Trouble Ticket system.
OPCDATA_MSG_LOG_ONLY	get/set	long				Message is Server Log Only.
OPCDATA_TROUBLETICKET_ACK	get/set	long				Acknowledge message after forwarding it to the Trouble Ticket System.
OPCDATA_MSI_OUTPUT	get/set	long				The message will be forwarded to the MSI.
OPCDATA_INSTR_IF_TYPE	get/set	long				Type of the Instruction Interface: ■ OPC_INSTR_NOT_SET ■ OPC_FROM_OPC ■ OPC_FROM_OTHER ■ OPC_FROM_INTERNAL
OPCDATA_UNMATCHED	get	long				Defines whether the message matches a condition. Possible values are: 0 (default): the message was sent to the server because it matched a match condition 1: the message did not match a match condition of the assigned templates, but was forwarded nevertheless.
OPCDATA_TIME_ZONE_DIFF	get/set	long				Time difference to GMT in seconds.

OPCDATA_FORWARDED_FROM	get	long				This flag signals whether another management server has forwarded the message.
OPCDATA_IS_READONLY	get	long				The message is read only (TRUE) or can be acknowledged on this server.
OPCDATA_MSGSRC_TYPE	get	long				Defines the Message Source Type, the source which originated this message, for example, the monitor agent. Possible values are: ■ OPC_CONSOLE_SRC : MPE/iX console. ■ OPC_OPCMSG_SRC : opcmsg(1|3) template ■ OPC_LOGFILE_SRC : logfile ■ OPC_MONITOR_SRC : monitor agent ■ OPC_SNMPTRAP_SRC : SNMP trap interceptor ■ OPC_SVMSI_SRC : server MSI ■ OPC_AGTMSI_SRC : agent MSI ■ OPC_LEGLINK_SRC : legacy link interface ■ OPC_SCHEDULE_SRC : scheduler
OPCDATA_TIME_OWNED	get	long				Time an operator took ownership of a message.
OPCDATA_DATA_INFO	get/set	long				Additional information about the message: ■ OPC_REMARK_FOR_ACK
OPCDATA_MSG_STATUS	get	long				Status of the message: ■ OPC_MSG_ACTIVE ■ OPC_MSG_HISTORY
OPCDATA_ACKNOWLEDGE_TIME	get	long				Time when the message was acknowledged.

OPCDATA_NUM_ANNOTATIONS	get	long				Number of annotations.
OPCDATA_APPLICATION	get/set	str			L	Application which produced the message. Default: empty string.
OPCDATA_GROUP	get/set	str				Message group. Default: empty string.
OPCDATA_MSGTEXT	get/set	str			L	Message Text. Default: empty string.
OPCDATA_ORIGMSGTEXT	get/set	str			L	Original Message Text. Allows you to set additional source information for a message. It is only useful if the message text was reformatted but the HPOM operator can access the original text as it appeared before formatting. Default: empty string.
OPCDATA_MSGTYPE	get/set	str				Message Type. This attribute is used to group messages into subgroups (for example, to denote the occurrence of a specific problem). This information may be used by event correlation engines. Default: empty string.
OPCDATA_NODENAME	get/set	str				Name of the node producing the message. The message is only handled by the HPOM manager if this system is part of the HPOM Node Bank. Default: local node name.
OPCDATA_OBJECT	get/set	str			L	Object name to use for the HPOM message. Default: empty string.
OPCDATA_MSGSRC	get	str				Message source. For example, the name of the encapsulated logfile if the message originated from logfile encapsulation or the interface name if the message was sent using an instance of the Message Stream Interface. Default: empty string.
OPCDATA_MSGID	get	str				The unique ID of the message.

OPCDATA_AACTION_NODE	get/set	str				Defines the node on which the automatic action should run. Default: value of OPCDATA_NODENAME .
OPCDATA_AACTION_CALL	get/set	str			L	Command to use as automatic action for the HPOM message. Default: empty string.
OPCDATA_OPACTION_NODE	get/set	str				Defines the node on which the operator-initiated action should run. Default: value of OPCDATA_NODENAME .
OPCDATA_OPACTION_CALL	get/set	str			L	Command to use as operator-initiated action for the HPOM message. Default: empty string. Call of the operator-initiated action.
OPCDATA_INSTR_IF	get/set	str				Name of the external instruction text interface. The external instruction text interface must be configured in HPOM. Default: empty string.
OPCDATA_INSTR_PAR	get/set	str			L	Parameters for a call to the external instruction text interface. Default: empty string.
OPCDATA_OWNED_BY	get	str				Name of the operator who owns the message.
OPCDATA_ESCALATION_SERVER	get	str				The escalation server.
OPCDATA_OPTION_VAR	set	str			L	A string containing the optional parameters used for resolving the $OPTION variables by the message interceptor. The string should have the format [<var>=<value>]* with <var> and <value> not containing spaces or the '=' character.
OPCDATA_ACKNOWLEDGE_OP	get	str				Name of operator who has acknowledged the message.
OPCDATA_MSG_KEY	get/set	str			L	Additional message attribute for customized message handling.

OPCDATA_SERVICE_NAME	get/set	str				Specifies the service name.
OPCDATA_ORIGMSGID	get	str				Unique identifier of the original message. This is set when the message ID was changed because of a message change.
OPCDATA_ESCALATED_BY	get	str				Name of the operator who escalated the message. Default: empty string.
OPCDATA_NUM_DUPLICATES	get	long				Number of duplicate messages of this message.
OPCDATA_LAST_REC_TIME	get	long				Contains the time when the last duplicate message was received.
OPCDATA_MSG_KEY_RELATION	get/set	str			L	Specifies the message key relation. Can contain patterns.
OPCDATA_MSG_KEY_RELATION_ICASE	get/set	long				Case sensitivity of message key relation: 0=case-sensitive, !=0 not case-sensitive.
OPCDATA_MSG_KEY_RELATION_SEPS	get/set	str				Field separators for message key relations.
OPCDATA_MSG_GEN_NODENAME	get	string				Name of the node where the event occurred.
OPCDATA_MSG_GEN_IP_ADDRESS	get	long				IP address of the node where the event occurred.
OPCDATA_MSG_GEN_NETWORK_TYPE	get	long				Network type of the node where the event occurred.

OPCDTYPE_MESSAGE_ID

Table: OPCDTYPE_MESSAGE_ID lists the attributes that are available for the Message ID data structure.

OPCDTYPE_MESSAGE_ID

Attribute	Scope	Type	Properties			Description
OPCDATA_MSGID OPCDATA_ID	get/set	str				Unique identifier of a message (Message ID).

OPCDTYPE_MONITOR_MESSAGE

Table: OPCDTYPE_MONITOR_MESSAGE lists the attributes that are available for the Monitor Message data structure.

OPCDTYPE_MONITOR_MESSAGE

Attribute	Scope	Type	Properties			Description
`OPCDATA_MON_VAR`	set	str				Name of the monitored object.
`OPCDATA_MON_VALUE`	set	double				Monitor value.
`OPCDATA_OPTION_VAR`	set	str			L	A string containing the optional parameters used for resolving the `$OPTION` variables by the monitor agent. The string should have the format [<var>=<value>]* with <var> and <value> not containing spaces or the '=', '(' or ')' characters.
`OPCDATA_OBJECT`	set	str			L	Message object.

opcregcond

HPOM provides a user-accessible data type to define registration conditions as the mechanism to register with the HPOM Interfaces.

Table: opcregcond lists the registration conditions of the function `opcif_register()` , see also opcif_register() for more information.

opcregcond

Attribute	Scope	Type	Properties			Description
`OPCREG_APPLICATION`	get/set	str				Registers for the message attribute application.
`OPCREG_APP_RESPONSE_ID`	get/set	str				Registers for application responses with the ID=OPCREG_APP_RESPONSE_ID.
`OPCREG_GROUP`	get/set	str				Registers for the message attribute message group.
`OPCREG_MSG_EVENT_MASK`	get/set	long				Registers for events matching OPCREG_MSG_EVENT_MASK.
`OPCREG_MSGTYPE`	get/set	str				Registers for the message attribute message type.
`OPCREG_NODENAME`	get/set	str				Registers for the message attribute node.
`OPCREG_OBJECT`	get/set	str				Registers for the message attribute object.
`OPCREG_OPERATOR`	get/set	str				Registers for the message events of certain operators.
`OPCREG_SEVERITY`	get/set	long				Registers for the message attribute severity.

Agent Message Stream Interface (MSI)

The Agent Message Stream Interface allows you to tap the message flow of an HPOM managed node to enable additional message processing by external applications before a message is sent to the management server. This can help to reduce the amount of network traffic considerably. A typical external application might be an event correlation engine, for example ECS.

NOTE:
The HP Operations agent API includes support for C/C++ and Java, as well as for every language that supports DCOM automation (for example, VB, VBScript, JScript, and so on). However, the agent message stream interface supports C APIs only. All of the APIs are built using Microsoft Visual Studio 2005.

Enable the Agent Message Stream Interface

The Agent Message Stream interface is disabled per default on the managed nodes. To allow external programs to use the MSI on the agent, you must first enable it. To enable it, create a nodeinfo policy containing

```
OPC_AGTMSI_ENABLE TRUE
```

on the management server, and deploy it to the managed nodes on which the MSI should be enabled.

Per default it is also not allowed to write messages containing automatic or operator initiated command to the MSI. The message agent discards the actions in the messages.

To allow the definition of automatic actions add the following to the nodeinfo policy:

```
OPC_AGTMSI_ALLOW_AA TRUE
```

To allow the definition of operator initiated actions add the following to the nodeinfo policy:

```
OPC_AGTMSI_ALLOW_OA TRUE
```

Configure messages to be sent to the Agent Message Stream Interface

Even if the Agent MSI is enabled and an application is registered for messages, you need to specify that a message should be sent to the agent MSI. You can do so in the policy editors on the "Message stream interface and external services " tab of the window "Outgoing Message".

To define that a message should be sent to the agent MSI, select "Agent Message Stream Interface" and choose whether message are copied or diverted.

msiconf()

Name

msiconf is the configuration file for the HPOM for Windows message manager

Contents

Synopsis

Server MSI

<SERVER_COFIG_DIR>/msiconf
For example:
`/etc/opt/OV/share/conf/OpC/mgmt_sv/msiconf`

Agent MSI

<AGENT_CONFIG_DIR>/msiconf
For example:
`/var/opt/OV/conf/OpC/msiconf on HP-UX`

Description

The file `msiconf` is an ASCII file containing a list of entries consisting of an HPOM Message Stream Interface (MSI) instance name followed by an order number. Each field is separated by a space, several spaces, or a tab. Each entry is separated from the next by a newline.

The MSI instance name may be a string up to 13 alphanumeric characters. The order number may be an integer value between -127 and 127. Lines or portions of lines beginning with # are assumed to be comments and are ignored. Blank lines are also ignored.

The MSI instance name corresponds to the name of a server MSI application that registers with the HPOM message manager. The order number specifies the order in which the registered MSI application will receive a message from the message manager (lowest to highest). Registered MSI applications that are not listed in the msiconf file are given an order number of 0.

The `msiconf` file is read by the message manager or message agent whenever an MSI instance opens or closes a connection to the MSI.

Example

```
counter -10
opcecm 0
proca 10
proca 10
enhtt 20
```

It is possible for a registered MSI instance to alter or completely suppress a message before writing back to the message stream. The proca and procb entries in the above example demonstrate a parallel MSI configuration, where one message entering the message stream may result in two messages exiting the message stream.

Related Topics:

- opcif_open(3)
- opcif_close(3)
- opcif_read(3)
- opcif_register(3)
- opcif_write(3)
- opcregcond(3)
- opc(5)

Server Message Stream Interface (MSI)

The server-based Message Stream Interface (MSI) enables programs external to the HPOM for Windows Message Action Server to pre-process incoming messages before they are stored in the HPOM for Windows database. These external programs are called MSI clients. MSI clients can change message attributes or suppress entire messages. They can also generate and add new messages to the message pipeline.

You can configure MSI clients in a server policy. To receive messages, an MSI client needs to be registered at the running MSI server.

You perform the general MSI configuration in XplConfig. The configuration parameters consist of name-value pairs. You can see the names but not the values. Each name serves a particular function. Each value configures this function.

The server-based MSI feature supports the following:

- C-API Functions
- COM-API

C-API Functions

The C-API provides access to the HPOM Message Stream Interface (MSI). This interface makes it possible to read HPOM messages from the internal message stream and to write messages into the internal message stream. The MSI is available on HPOM managed nodes. All MSI types establish a connection to the HPOM message agent.

The HPOM Message Stream Interface is divided into the following types:

- `OPCSVIF_EXTMSGPROC_READ`

  This interface type is used for non-destructive read operations on the HPOM internal message flow. Only messages that are allowed to be output on the Agent MSI are accessible using this interface type. This type of interface is typically used by statistical analysis tools or additional display facilities.
- `OPCSVIF_EXTMSGPROC_READWRITE`

  This interface type is used to read messages from the internal message flow of HPOM, to create or modify messages, and to write them back to the HPOM processes. Only messages that can be output on the MSI are accessible using this interface type. Messages tagged with "copy to" remain in the HPOM message flow. Messages tagged with "divert to" are taken out of the flow. This type of interface can be used by event correlation engines.
- `OPCSVIF_EXTMSGPROC_WRITE`

  Interface instances of this type are used by write-only applications to feed messages into the HPOM message flow. This type of interface can also be used to encapsulate the `opcif_write()` routine in a command-line interface.

Prerequisites

The API functions must be issued by the agent user.

Multithread Usage

All functions of the C-API are safe to be called by multi-threaded applications. They are thread-safe for both POSIX Threads and DCE User Threads. They are not async-cancel, async-signal, or fork-safe. They cannot be called safely in kernel threads.

The `opcreg_copy()` routine is not thread-safe for POSIX threads or for DCE User Threads.

Registration Conditions

HPOM provides a user-accessible data type to define registration conditions as the mechanism to register with the HPOM Interfaces.

HPOM provides a set of APIs to create an empty condition, modify or query condition fields and to duplicate or delete a condition definition from memory.

Related Topics:
- C-API Security

C-API Security

An event subscription service API is a window that enables applications to view generic system-wide activity. Applications must be prevented from unauthorized snooping of system behavior at this access point. In addition, access to the HPOM message flow in read-write mode allows an external application to discard messages, without users being made aware that a message was generated. The APIs must, therefore, apply authentication mechanisms to prevent users and applications from unauthorized access to the HPOM message flow.

Automatic and Operator-initiated Actions

One important and critical issue arising from these security considerations is whether external applications using the interfaces are allowed to define automatic actions, operator-initiated actions, or both. If HPOM allows access to these message attributes, any user who is authorized to call the APIs is also able to execute actions on HPOM managed nodes.

According to the current HPOM concept, which regards HPOM as an open application providing a high level of flexibility to integrate applications, HPOM allows external programs to define actions for messages that are passed to the message agent. Event correlation can be seen as an advance on the existing concept of message conditions ("if attributes match then set attributes and actions") to a higher level ("if rule fires then set attributes and actions"). It is, therefore, essential that these external applications are allowed to perform these modifications.

An appropriate authorization mechanism at the API level guarantees that only authorized users can apply the APIs. However, as the checking of a user ID belongs to the OS level with its super user concept, this conflicts somewhat with the existing HPOM concept in which the administrator is responsible for the configuration of user roles.

HPOM for Windows makes it possible for you to enable and disable the interface functionality. In addition, you can configure whether actions can be defined by an application that is writing to the interface. This affects all interface types.

You can also define whether each message is allowed for output to the Message Stream Interface in the HPOM for Windows policy editors. For example, an administrator can prevent the output of certain messages so external applications do not receive secure information by reading these messages from the HPOM message flow.

Enabling and Disabling Interfaces and Actions

HPOM allows users with a user ID of zero (uid 0), typically root, on UNIX, as well as users that are in the Administrators group on Windows, to access the HPOM Interface APIs, and to define actions for messages that are sent to the management server. The HPOM for Windows administrator can enable or disable the interface functionality of the interface types that affect the message flow, and allow or disallow actions that are read from the interface. By default, these interfaces are disabled, and are not allowed to define actions.

To enable the MSI on a managed node, create a nodeinfo policy containing `OPC_AGTMSI_ENABLE TRUE` and deploy it to the managed nodes.

If actions are disallowed, an appropriate error text is added to the annotations field, and the action is disabled.

To allow the definition of automatic actions, add the following to the nodeinfo policy:

```
OPC_AGTMSI_ALLOW_AA TRUE
```

To allow the definition of operator-initiated actions, add the following to the nodeinfo policy:

```
OPC_AGTMSI_ALLOW_OA TRUE
```

opcif_open()

```
#include opcapi.h

int opcif_open(
               int          interface_type,     /* in */
               const char   *instance,          /* in */
               int          mode,               /* in */
               int          max_entries,        /* in */
               int          *interface_id       /* out */
              );
```

Parameters

interface_type

Specifies the type of interface:

Server Message Stream Interface

Used by external message processors (for example, event correlation engines):

- OPCSVIF_EXTMSGPROC_READ
- OPCSVIF_EXTMSGPROC_READWRITE
- OPCSVIF_EXTMSGPROC_WRITE

instance

Name of the interface instance that is registered for one of the interface types. The name is limited to a length of 12 alphanumeric characters because it is also part of the queue file name.

mode

To specify whether the interface is opened if the HPOM processes are not running.

Use one of the following:

- OPCIF_ALWAYS (default)
- OPCIF_AGT_RUNNING

To specifiy whether the opcif_read() API waits for available data, use one of the following options:

- OPCIF_READ_WAIT (default)
- OPCIF_READ_NOWAIT

If you use the WAIT option, the calling process is blocked until data is available or the process receives an interrupt.

To specify the handling of unread messages if the connected process closes the interface or aborts, use one of the following options:

- OPCIF_CLOSE_FORWARD (default)
- OPCIF_CLOSE_DISCARD

With the FORWARD option, messages in the read queue are appended to the write queue. With the FORWARD option, these messages are discarded when an external program closes the interface. You can combine these options using the pipe (|) operator.

max_entries

Specifies the maximum number of entries in the read queues. If this number is exceeded, HPOM stops writing to the queue. When the reading process has emptied the queue, opcif_read() returns an error value. The application must then disconnect and reopen the interface. To disable this safeguard, specify 0 for max_entries .

interface_id

The returned value must be used in subsequent calls to the APIs to refer to this instance of the interface.

Description

Use the function opcif_open() to connect to an instance of one of the following interfaces:

- Server Message Stream Interface

Return Values

OPC_ERR_OK

Interface opened correctly.

OPC_ERR_INVALID_OUTPARAM

Pointer to interface_id is invalid.

OPC_ERR_ACCESS_DENIED

Access denied.

OPC_ERR_INVALID_INTERFACE_INSTANCE

Instance name contains invalid characters or is too long.

`OPC_ERR_INVALID_INTERFACE_TYPE`
: No such interface type.

`OPC_ERR_CANT_INIT`
: Initialization of queues failed.

`OPC_ERR_CANT_OPEN_READQUEUE`
: Unable to open the read queue.

`OPC_ERR_CANT_OPEN_WRITEQUEUE`
: Unable to open the write queue.

`OPC_ERR_CANT_INFORM_MSGA`
: Informing message agent failed.

`OPC_ERR_NO_MEMORY`
: Memory allocation failed.

Versions

HPOM for Windows 8.00 and higher

opcif_get_pipe()

```
#include opcapi.h

int opcif_get_pipe(
                 int       interface_id,       /* in */
                 int       pipefd              /* out */
                  );
```

Parameters

`interface_id`
: Specifies which interface instance is used.

`pipefd`
: Returns the file descriptor of the selected output interface queue.

Description

A program reading from several input files needs the pipe file descriptor of its interface input queue. This descriptor is then used as part of the parameters to `select(2)` or `fcntl(2)` . The function `opcif_get_pipe()` returns this value.

For convenience reasons, an application that reads only from the HPOM interface can specify `OPCIF_READ_WAIT` in the opcif_open() call.

Return Values

`OPC_ERR_OK`
: OK.

`OPC_ERR_CANT_INIT`
: Initialization of queues failed.

`OPC_ERR_INVALID_OUTPARAM`
: `pipefd` is NULL.

`OPC_ERR_INVALID_INTERFACE_ID`
: No such interface opened.

Versions

HPOM for Windows 8.00 and higher

opcif_read()

```
#include opcapi.h

int opcif_read(
              int          interface_id,     /* in */
              opcdata      data              /* in/out */
              );
```

Parameters

`interface_id`
: Specifies which interface instance is used.

`data`
: HPOM data structure.

Description

The function `opcif_read()` reads a message from the queue of the specified interface instance. If the interface instance has been opened with `OPCIF_READ_WAIT` , the calling process is blocked until the information is available. The API returns an error if the application receives an interrupt signal. The data parameter specified in the call must be created with `opcdata_create()` . Memory for the actual message data is allocated, and if memory was assigned to data before the call to `opcif_read()` , it is deallocated.

Return Values

`OPC_ERR_OK`
: OK.

`OPC_ERR_CANT_INIT`
: Initialization of queues failed.

`OPC_ERR_INVALID_INTERFACE_ID`
: No such interface opened.

`OPC_ERR_INVALID_OUTPARAM`
: `data` is NULL or is of the wrong type.

`OPC_ERR_WRONG_MSITYPE`

Interface assigned to `interface_id` is of the wrong type.

`OPC_ERR_EINTR`
Reading from the pipe failed.

`OPC_ERR_MSI_BUF_FULL`
Number of messages exceeds the specified number in `max_entries` while opening.

`OPC_ERR_NO_DATA`
Queue is empty.

`OPC_ERR_CANT_READ_MSG`
Reading message, event, or response failed.

`OPC_ERR_NO_MEMORY`
Memory allocation failed.

Versions

HPOM for Windows 8.00 and higher

opcif_register()

```
#include opcapi.h

int opcif_register(
          int                interface_id,     /* in */
          const opcregcond   reg_cond,         /* in */
          long               *cond_id          /* out */
                );
```

Parameters

`interface_id`

Specifies which interface instance is used.

`reg_cond`

- Messages

  Defines the combination of message attributes that are checked. NULL registers for all messages
- Message Events

  Defines an event mask and the restriction of message events of messages for certain operators.
- Application Responses

  Defines an application response specified by the application response ID.

`cond_id`

Returns an ID to reference this condition in a subsequent call to opcif_unregister() . NULL is allowed if the API user does not require the ID (for example, if opcif_unregister() is not called subsequently).

Description

The function `opcif_register()` is used by an external application to register for the following attributes:

- Message Attributes

  HPOM supports registration for message type, message group, node name, object, severity, and application attributes. You can also combine attributes (logical AND), and pipe (|) within an attribute (logical OR). Multiple registrations (logical OR of registration conditions) are also possible using a sequence of API calls.

  The following attributes are supported:

  - `OPCREG_MSGTYPE`

- `OPCREG_GROUP`
- `OPCREG_NODENAME`
- `OPCREG_OBJECT`
- `OPCREG_SEVERITY`
- `OPCREG_APPLICATION`

Return Values

`OPC_ERR_OK`
: OK.

`OPC_ERR_CANT_INIT`
: Initialization of queues failed.

`OPC_ERR_INVALID_INTERFACE_ID`
: No such interface opened.

`OPC_ERR_CANT_INFORM_MSGA`
: Informing message agent failed.

Versions

HPOM for Windows 8.00 and higher

opcif_unregister()

```
#include opcapi.h

int opcif_unregister(
           int             interface_id,       /* in */
           long            cond_id             /* in */
                   );
```

Parameters

interface_id

Specifies which interface instance is used.

cond_id

Specifies the registration condition to be removed. 0 unregisters for all messages, message events, or application responses.

Description

To cancel prior registrations for messages, the external application calls opcif_unregister() with the value of reg_cond that was specified in the call to the registration API. As the registration mechanism is a positive filter, removing a registration condition does not mean that messages matched by this condition are filtered out after opcif_unregister() is called. Instead, just the positive filter condition is cancelled.

By unregistering a condition for message events or application responses, information matching the unregistered condition will no longer received.

Return Values

OPC_ERR_OK

OK.

OPC_ERR_CANT_INIT

Initialization of queues failed.

OPC_ERR_INVALID_INTERFACE_ID

No such interface opened.

OPC_ERR_CANT_INFORM_MSGM

Informing message manager failed.

Versions

HPOM for Windows 8.00 and higher

opcif_write()

```
#include opcapi.h

int opcif_write(
           int                interface_id,     /* in */
           const opcdata      data              /* in */
               );
```

Parameters

`interface_id`
: Specifies which interface instance is used.

`data`
: HPOM data structure.

Description

Use the function `opcif_write()` to write a message to the HPOM Interface. Depending on the type of interface, the message is written into the message queue of the message agent.

This function can only be used for interfaces of the following types:

- `OPCSVIF_EXTAGT_MESSAGE`

Return Values

`OPC_ERR_OK`
: OK.

`OPC_ERR_CANT_INIT`
: Initialization of queues failed.

`OPC_ERR_INVALID_INTERFACE_ID`
: No such interface opened.

`OPC_ERR_INVALID_OPCDATA_TYPE:`
: data must be of the type `OPCDTYPE_MESSAGE` .

`OPC_ERR_CANT_WRITE_MSG`

Unable to write message.

`OPC_ERR_WRONG_MSITYPE`

Interface type is invalid for this operation.

Versions

HPOM for Windows 8.00 and higher

opcreg_copy()

```
#include opcapi.h

int opcreg_copy(
         const opcregcond    reg_cond,           /* in */
         opcregcond          *copy               /* out */
               )
```

Parameters

reg_cond
: HPOM registration condition structure.

copy
: Copy of reg_cond .

Description

The API creates a copy of the condition and returns it in copy . The allocated memory has to be deallocated using opcreg_free() . copy must be freed using opcreg_free() before calling this function.

Return Values

OPC_ERR_OK
: OK.

OPC_ERR_INVALID_OUTPARAM
: copy is invalid.

OPC_ERR_NO_MEMORY
: Allocation of memory failed.

Versions

HPOM for Windows 8.00 and higher

See Also

opcreg_create()
opcreg_free()

opcreg_get_long()
opcreg_get_str()
opcreg_set_long()
opcreg_set_str()

opcreg_create()

```
   #include opcapi.h

   int opcreg_create(
          opcregcond           *reg_cond            /* out */
                    );
```

Parameters

reg_cond
HPOM registration condition structure.

Description

This routine creates an empty registration condition. The actual structure of this data area is hidden from the user. To get or set attributes, the respective routines must be called. The memory used for the area has to be deallocated by calling opcreg_free() .

Return Values

OPC_ERR_OK
OK.

OPC_ERR_INVALID_OUTPARAM
reg_cond is invalid.

OPC_ERR_NO_MEMORY
Allocation of memory failed.

Versions

HPOM for Windows 8.00 and higher

See Also

opcreg_copy()
opcreg_free()
opcreg_get_long()
opcreg_get_str()
opcreg_set_long()
opcreg_set_str()

opcreg_free()

```
#include opcapi.h

int opcreg_free(
           opcregcond        *reg_cond       /* in/out */
               );
```

Parameters

reg_cond
    HPOM registration condition structure.

Description

The function opcreg_free() deallocates memory previously allocated by opcreg_create() , opcreg_copy() , or opcreg_set_*() .

Return Values

OPC_ERR_OK
    OK.

OPC_ERR_INVALID_OUTPARAM
    reg_cond is invalid.

Versions

HPOM for Windows 8.00 and higher

See Also

opcreg_copy()
opcreg_create()
opcreg_get_long()
opcreg_get_str()
opcreg_set_long()
opcreg_set_str()

opcreg_get_long()

```
#include opcapi.h

long opcreg_get_long(
         const opcregcond       reg_cond,       /* in */
         int                    field           /* in */
                    );
```

Parameters

`reg_cond`
: HPOM registration condition structure.

`field`
: Selects the attribute that is queried.

Description

Use the routine `opcreg_get_long()` to access the attribute values of a condition.

Return Values

Returns the integer value of the attribute. If the routine fails, `-1` is returned.

Versions

HPOM for Windows 8.00 and higher

See Also

opcreg_copy()
opcreg_create()
opcreg_free()
opcreg_get_str()
opcreg_set_long()
opcreg_set_str()

opcreg_get_str()

```
   #include opcapi.h

   char *opcreg_get_str(
       const opcregcond         reg_cond,           /* in */
       int                      field               /* in */
                       );
```

Parameters

`reg_cond`
: HPOM registration condition structure.

`attribute`
: Selects the attribute that is queried.

Description

Use the routine `opcdata_get_str()` to access the attribute values of a condition.

Return Values

Returns a character pointer to the value of the defined attribute in the data area. The pointer points into the internal data area. Modification of the attribute is only allowed using opcreg_set_str() . Direct access to the string is not supported. However, it is not possible to prevent the user from committing direct modifications.

Versions

HPOM for Windows 8.00 and higher

See Also

opcreg_copy()
opcreg_create()
opcreg_free()
opcreg_get_long()
opcreg_set_long()
opcreg_set_str()

opcreg_set_long()

```
#include opcapi.h

int opcreg_set_long(
        opcregcond      reg_cond,            /* in/out */
        int             field,               /* in */
        long            value                /* in */
                    );
```

Parameters

`reg_cond`
HPOM registration condition structure.

`field`
Select the attribute that is set.

`value`
Specify the value of the attribute.

Description

Use the function `opcreg_set_long()` to set attributes to a certain value.

Return Values

`OPC_ERR_OK`
OK.

`OPC_ERR_INVALID_FIELD`
`field` is invalid.

Versions

HPOM for Windows 8.00 and higher

See Also

opcreg_copy()
opcreg_create()

opcreg_free()
opcreg_get_long()
opcreg_get_str()
opcreg_set_str()

opcreg_set_str()

```
#include opcapi.h

int *opcreg_set_str(
        opcregcond       reg_cond,          /* in/out */
        int              field,             /* in */
        const char       *value             /* in */
                   );
```

Parameters

`reg_cond`
: HPOM registration condition structure.

`field`
: Select the attribute that is set.

`value`
: Specify the value of the attribute.

Description

Use the function `opcreg_set_str()` to set attributes to a certain value.

Return Values

`OPC_ERR_OK`
: OK.

`OPC_ERR_INVALID_FIELD`
: `field` is invalid.

Versions

HPOM for Windows 8.00 and higher

See Also

opcreg_copy()
opcreg_create()

opcreg_free()
opcreg_get_long()
opcreg_get_str()
opcreg_set_long()

COM-API

The COM-API is a Windows-like interface that enables you to write new programs for the server-based Message Stream Interface (MSI) more easily. This interface makes it possible to read HPOM messages from the internal message stream, and to write messages into the internal message stream. The MSI is available on HPOM managed nodes. All MSI types establish a connection to the HPOM message agent.

This section describes the following:

- COM-API Documentation

  Describes the COM-based MSI API for HPOM for Windows.

- COM-API Development Practices

  Sample MSI COM client. Although it is an EXE file, it could be a DLL as well. Any MSI client you write supports the `CLSCTX_LOCAL_SERVER` instantiation flag. The HPOM for Windows MSI COM interface allows only clients that run in their own processes.

Server-based Message Stream Interface (MSI) API Documentation

HP Operations Manager for Windows 8.00

This documentation describes the COM- and server-based MSI API for HP Operations Manager for Windows (HPOM for Windows). The API contains the following interfaces:

IOvOWMsiServer is the public interface to access the HPOM for Windows MSI server. You cannot instantiate it directly. Instead, use the IOvOWMsiLocator to get the interface to the IOvOWMsiServer .

IOvOWMsiLocator is the interface used to retrieve an interface pointer to the HPOM for Windows MSI server. You can instantiate it using one of the following:

- `CLSID_OvOWMsiLocator` class id
- `OvOWMsi.OvOWMsiLocator` prog id

IOvOWServerMessage is the interface to an HPOM server message. By using this interface, you can change server message attributes. For details, refer to the interface API documentation. You can instantiate a new server message using one of the following:

- `CLSID_OvOWServerMessage` class id
- `OvOWMsi.OvOWServerMessage` prog id

IOvOWMsiClient is the interface a COM-based MSI client must implement to be recognized by the HPOM for Windows MSI server. For details, refer to the interface API documentation.

IOvOWRegisterCondition is an interface that enables you to set specific conditions for messages you want to register for on the HPOM for Windows MSI server. It is used in the IOvOWMsiServer::registerClient (...) call. You can instantiate a Register Condition using one of the following:

- `CLSID_OvOWRegisterCondition` class id
- `OvOWMsi.OvOWRegisterCondition` prog id

Includes OvOWExTypes.idl .

Author:

Copyright 1993-2007 Hewlett-Packard Development Company, L.P.

Since:

8.00

COM-API Development Practices

This section includes the following articles:

- Creating a New COM-based MSI Client
- Working with MSI Interfaces
- Configuring DCOM Security for Your MSI Client
- Using Tracing in Your COM Application

Creating a New COM-based MSI Client

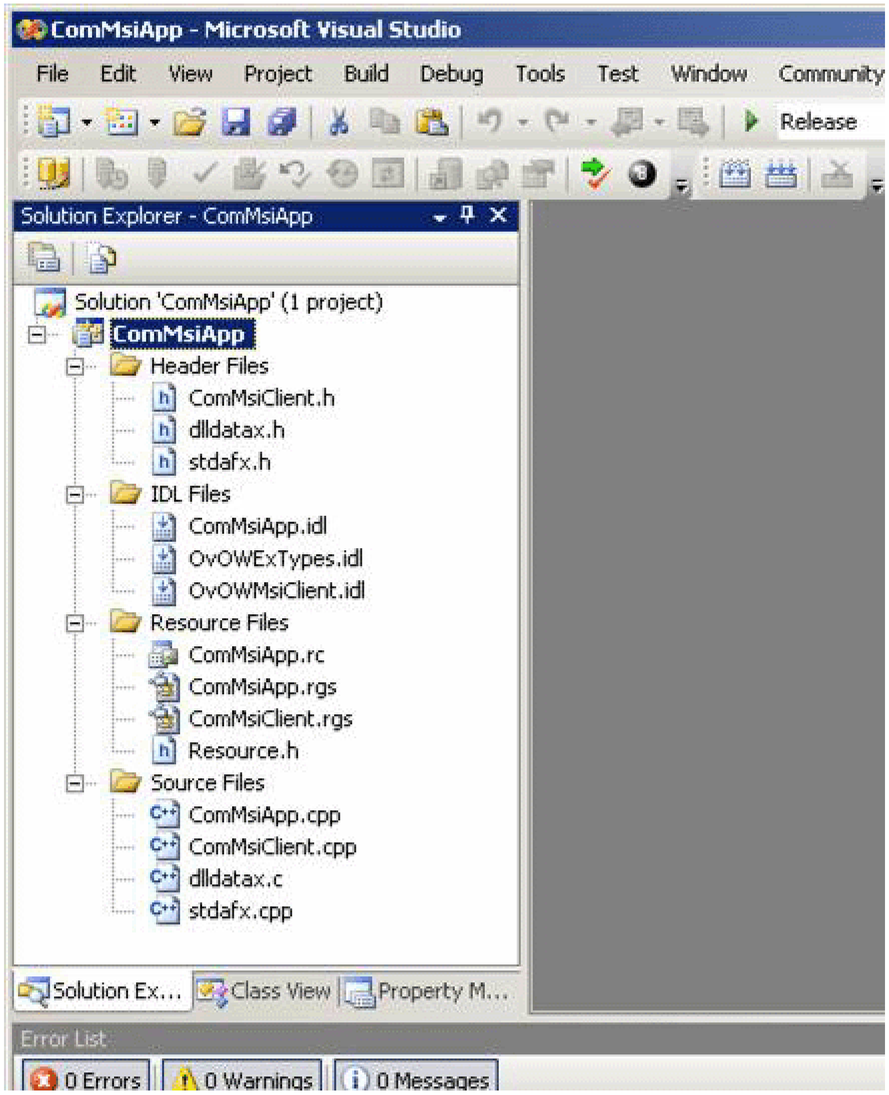
NOTE:
You can perform the first two steps of this procedure in a few minutes using the wizards built into Visual Studic

To create a new COM-based MSI client, follow these steps:

1. Create a COM application that can act as a host for your MSI Client class.

   This application might be an executable (EXE) or a DLL, which is designed to run in a surrogate process. Ei
   local server (`CLSCTX_LOCAL_SERVER`).

   This figure shows an example project setup in Microsoft Visual Studio 2005. The Application is called "ComI



2. Inside the new COM application, create the new MSI Client class.

   In the example, this class is the "ComMsiClient" class.

3. Include `IOvOWMsiClient.idl` and `OvOWExTypes.idl` in your project, so that it can be processed by the MIL

   Including these files will generate the type library as well as a header file and marshalling code.

4. Declare the IOvOWMsiClient interface in your COM client IDL file.

```
import "oaidl.idl";
import "ocidl.idl";

import "OvOWMsiClient.idl";

//...
// interface definition for your ComMsiClient

[
 uuid(52463E05-58AB-4D33-B675-E20D6CCA2F0D),
 version(1.0),
 helpstring("ComMsiApp 1.0 Type Library")
]
library ComMsiAppLib
{
 importlib("stdole2.tlb");
 [
  uuid(48A4B84E-9A15-43BC-8260-60152C2482EE),
  helpstring("ComMsiClient Class")
 ]
 coclass ComMsiClient
 {
  [default] interface IComMsiClient;
  interface IOvOWMsiCLient;
 };
};
```

5. Declare the IOvOWMsiClient interface in your COM client header file.

```
// ComMsiClient.h : Declaration of the CComMsiClient

#pragma once
#include "resource.h"

#include "ComMsiApp.h"

// CComMsiClient
class ATL_NO_VTABLE CComMsiClient :
 public CComObjectRootEx<CComMultiThreadModel>,
 public CComCoClass<CComMsiClient, &CLSID_ComMsiClient>,
 public ISupportErrorInfo,
 public IComMsiClient,
 public IOvOWMsiClient
{
```

```
//...
};
```

In Microsoft Visual Studio 2005, you can reuse large parts of the wizard-generated code. You just need to impler succeeds, you are done with the basics.

You can find the MSI client example in the following directory:

`%OvInstallDir%\examples\OVOW\DevelopmentKit\Server\MSI\CppCOM"`

Working with MSI Interfaces

When working with MSI interfaces, you can do the following:

- Connecting to the MSI Server
- Registering for Messages
- Receiving and Processing Messages
- Adding a New Message to the MSI Server
- Removing a Specific Registration
- Removing All Registrations
- Disconnecting from the MSI Server
- Calling serverInit() and serverShutdown()

Connecting to the MSI Server

To connect to the MSI server, follow these steps:

1. Retrieve the MSI server.

   To retrieve the MSI server in a method, you can use code such as the following.

```
STDMETHODIMP CComMsiClient::retrieveMsiServer(IOvOWMsiServer **ppMsiServer)
{
  CComPtr<IOvOWMsiLocator> pMsiLocator;
  HRESULT hr = CoCreateInstance(CLSID_OvOWMsiLocator, NULL, CLSCTX_INPROC_SERVER, IID_IOv
  if (FAILED(hr))
  {
    // maybe trace the hr; something went wrong
    return hr;
  }
  if (pMsiLocator == NULL)
  {
    return E_FAIL;
  }
  CComPtr<IOvOWMsiServer> pMsiServer;
  hr = pMsiLocator->connectToMsiServer(&pMsiServer);
  if (FAILED(hr))
  {
    // MSI Locator returned an error
    return hr;
  }
  return pMsiServer.CopyTo(ppMsiServer);
}
```

You can also use easier and less cryptic ways of instantiating the MSI Locator. Instead of using the CLSID, MSI components. For details, refer to the HPOM for Windows server-based MSI API documentation.

2. Call `connectClient(...)` .

   The following method shows what a generic `connectClient(...)` call might look like. You may have fixed therefore no need for all the arguments used in this example.

```
STDMETHODIMP CComMsiClient::connectClient(BSTR appName,
                                          int readWriteMode,
                                          IOvOWMsiServer **ppMsiServer,
                                          int *interfaceId)
{
  HRESULT hr;
  if (m_pMsiServer == NULL)
  {
    m_pMsiServer.Release();
    hr = this->retrieveMsiServer(&m_pMsiServer);
    if (FAILED(hr))
    {
      return hr;
    }
  }
  hr = m_pMsiServer->connectClient(appName, this,
  readWriteMode, ppMsiServer, interfaceId);
  if (interfaceId != NULL)
  {
    m_interfaceId = *interfaceId; // remember the interfaceId for the later call to disco
  }
  return hr;
}
```

The `interfaceId` identifies the connection between your MSI Client and the HPOM for Windows MSI server later.

For the exact definition of the MSI client connection modes, refer to the HPOM for Windows COM API documentat

Registering for Messages

To set up a register condition, you can use code such as the following.

```
CComPtr<IOvOWRegisterCondition> regCond;
hr = regCond.CoCreateInstance(CLSID_OvOWRegisterCondition);
```

The register condition does not have any specifics, so it will match all messages. To register for messages with c

```
hr = regCond->put_MessageSeverity(OVEP_C_SEV_CRIT);
```

You can register the MSI client at the MSI server with the register condition you just prepared.

```
STDMETHODIMP CComMsiClient::registerClient(int interfaceId,
                                   IOvOWRegisterCondition *condition,
                                   int *conditionId)
{
  if (m_pMsiServer == NULL)
  {
    return E_FAIL;
  }
  HRESULT hr = m_pMsiServer->registerClient(interfaceId,
  condition, conditionId);
  m_lastMsiAppEvents.registerClientCallReturn = hr;
  return hr;
}
```

The returned conditionId identifies this specific registration call. You can use it later to unregister the specific reg
before you can register for messages.

Receiving and Processing Messages

After you connect to the MSI server and register your MSI client to receive messages, your client will receive me
command-line tool to generate HPOM messages that match at least one of your registrations.

When your MSI client is eligible to receive a message, the MSI server calls the `receiveMessage(...)` method of

You can now read the properties of the message and, as an example, store them in a file. If the message was div
the MSI server. In that case, the message will not reach the HPOM for Windows Message Server Database, and i

In the following code sample, an HPOM Message is received and reinserted into the MSI server without any proce

```
STDMETHODIMP CComMsiClient::receiveMessage(IOvOWServerMessage *message)
{
  // you can do anything with the Message (for example, read the Message Text property)
  CComBSTR msgText;
  message->get_Text(&msgText);

  // now return the Message to the MSI Server
  if (m_pMsiServer == NULL)
  {
    return E_FAIL;
  }
  HRESULT hr = m_pMsiServer->insertMessage(m_interfaceId, message);
  if (FAILED(hr))
  {
    // maybe trace the hr; something went wrong
    return hr;
  }
  return S_OK;
}
```

Adding a New Message to the MSI Server

If the MSI Client has connected to the MSI Server with the appropriate MSI Client Connection Mode, you can alsc

The following code samples shows a method that creates a message with normal severity and the specified Mess
server.

```
STDMETHODIMP CComMsiClient::feedNormalMessage(BSTR messageText)
{
  if (m_pMsiServer == NULL)
  {
    return E_FAIL;
  }
  if (messageText == NULL)
  {
    return E_INVALIDARG;
  }
  CComPtr pNewMessage;
  HRESULT hr = CoCreateInstance(CLSID_OvOWServerMessage, NULL, CLSCTX_ALL, IID_IOvOWServerMess
  if (FAILED(hr))
  {
    // Check for hr; something went wrong
    return hr;
  }
  pNewMessage->put_Text(messageText);
  pNewMessage->put_Severity(OVEP_C_SEV_NORMAL);

return m_pMsiServer->insertMessage(m_interfaceId, pNewMessage);
}
```

Removing a Specific Registration

If you have stored the ID of the register condition you want to remove, and if m_pMsiServer is not NULL , you ca

```
HRESULT hr = m_pMsiServer->unregisterClient(conditionId);
```

Removing All Registrations

You can remove all Registrations of your MSI client by calling the following.

```
HRESULT hr = m_pMsiServer->removeAllRegistrations(interfaceId);
```

In this example, the interface ID is used because it is easier than remembering all condition IDs.

Disconnecting from the MSI Server

You can disconnect an MSI client from the MSI server by calling the following.

```
HRESULT hr = m_pMsiServer->disconnectClient(interfaceId);
```

This call also causes the removal of all registrations made by this MSI client during the last connection. Expect th
server.

Calling serverInit() and serverShutdown()

For calls to serverInit() or serverShutdown() , do the following:

- serverInit()

  Call to serverInit() signals that the HPOM for Windows MSI server is running and (based on your connectio messages from the MSI client, or both. It is usually called when the MSI client connects to the MSI server suc

- `serverShutdown()`

  Call to `serverShutdown()` signals the disconnect of the MSI client. This may result from either a direct call to Windows MSI server (for example, during a shutdown or a restart of the HPOM for Windows Message server). connection-specific cleanup. Most likely, the next method called in the MSI client will be `FinalRelease()` , wh

Configuring DCOM Security for Your MSI Client

Since HP Operations Manager for Windows 8.10, the Message/Action Server, which hosts the MSI server, runs ur for DCOM access to allow instantiation and communication between server and client.

To configure MSI COM clients for DCOM access, follow these steps:

1. Start the DCOM Configuration utility `dcomcnfg` :
   a. In the text field, click Start′ Run .
   b. Type **`dcomcnfg`** .
   c. Press Enter .
2. Open the following folders:
   - `Component Services`
   - `Computers`
   - `DCOM Config`
   - `My Computer`
3. From the list of registered COM applications on your systems, look for your COM application.

   The application name is typically the prefix of your COM client Prog ID. For example, if your client ProgID is in the list.
4. On the list item that represents your COM application, right-click and select Properties .
5. In the Launch and Activation Permissions section of the Security tab, select the Customize radio button.
6. Click Edit .
7. Add the user account under which the `OvEpMsgActSrv.exe` process is running to the list of users.

   By installation default, the user account is `HP-OVE-User` . If you chose a different user name during installa Manager from the Task Bar, and check the User Name entry for the process.
8. For the newly-added user, select Local Launch (already selected) and Local Activation .

   Note that the Deny fields remain unselected.
9. Click OK .
10. Close the Properties Window by clicking OK again.

If you get further errors about insufficient access rights, add Access Permissions for `HP-OVE-User` (or the user na

Using Tracing in Your COM Application

It is up to you to chose any trace tool you prefer to instrument your COM MSI Application. The performance of th

Tracing is generally useful to see what your application is doing and can be helpful to find bugs in early developn

To display trace messages of the MSI server, you need to use the built-in trace client of HPOM.

To enable trace messages, follow these steps:

1. On the command line, execute **`<InstallDir>\support\ovtrcadm.exe -a localhost`** , where `<InstallDi`
2. Execute **`<InstallDir>\support\ovtrcgui.exe`** .
   The Trace Client GUI appears.
3. Cancel the wizard.
4. Select File ′ New ′ Trace Configuration .
5. Select localhost for the machine to configure.
6. To create a new configuration, double-click <add application to trace> .
7. From the list of available applications, select OvEpMsgActSrv .
8. From the list of available traces, select those starting with **`OvOWMsi`** .
9. Set the attribute mask to Max .
10. Click OK .
11. Select File ′ New ′ Trace Monitor .
12. Select localhost for the machine to monitor.
    A list view appears.

New MSI Server traces are shown in the list view. Use Edit ′ Options to configure which attributes should be dis

To close the Trace Client GUI, follow these steps:

1. Do one of the following:
   - Select File ′ Exit
   - Press ALT +F4 .
2. When asked if you want a quick close, select Yes .

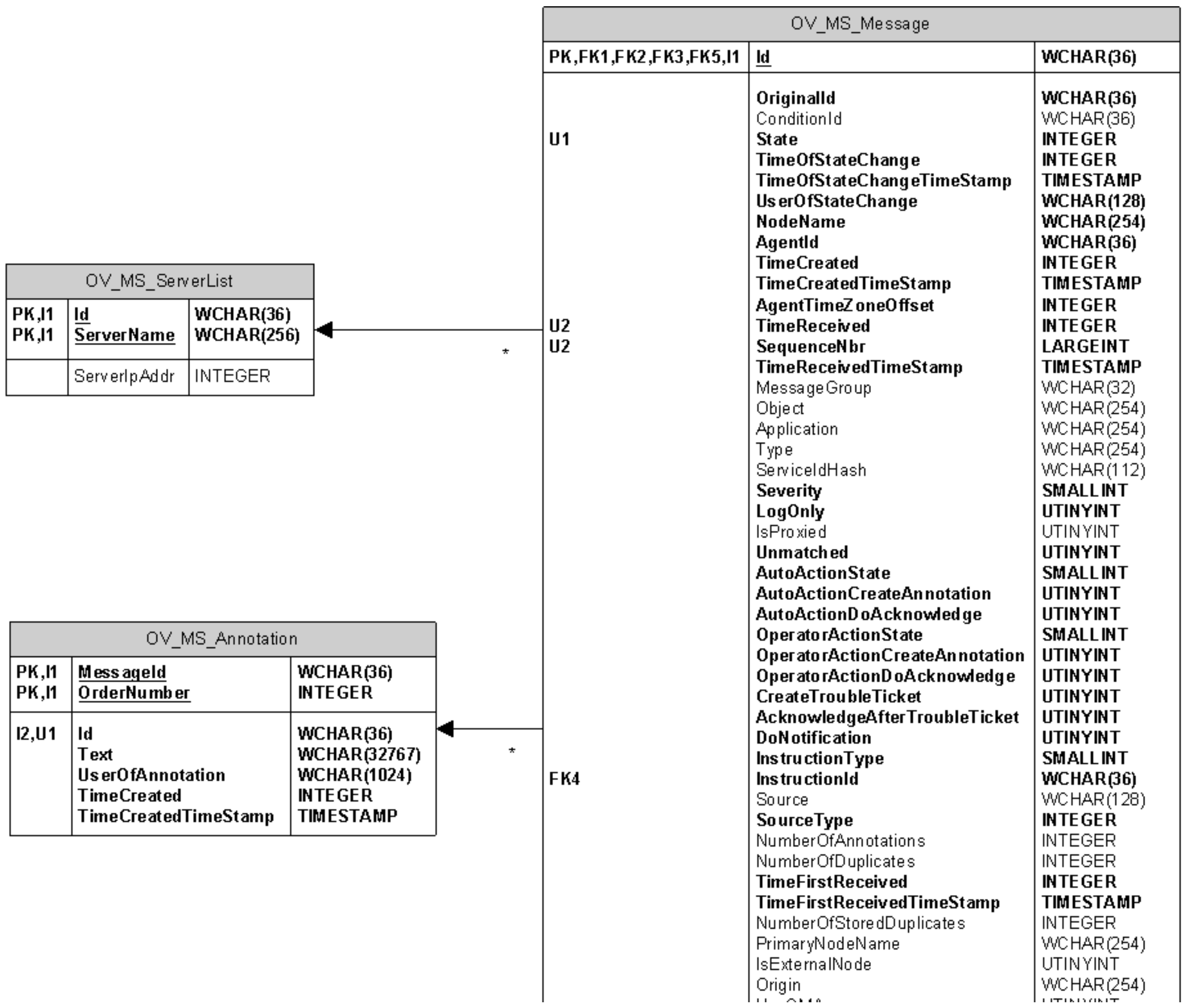
Database schema for message/action server

The ERM diagram illustrates the contents of the message tables.

NOTE:

The HP Operations agent API includes support for C/C++ and Java, as well as for every language that supports DCOM automation (for example, VB, VBScript, JScript, and so on). However, the agent message stream interface supports C APIs only. All of the APIs are built using Microsoft Visual Studio 2005.

ERM diagram of the message tables



OV_MS_ServerList

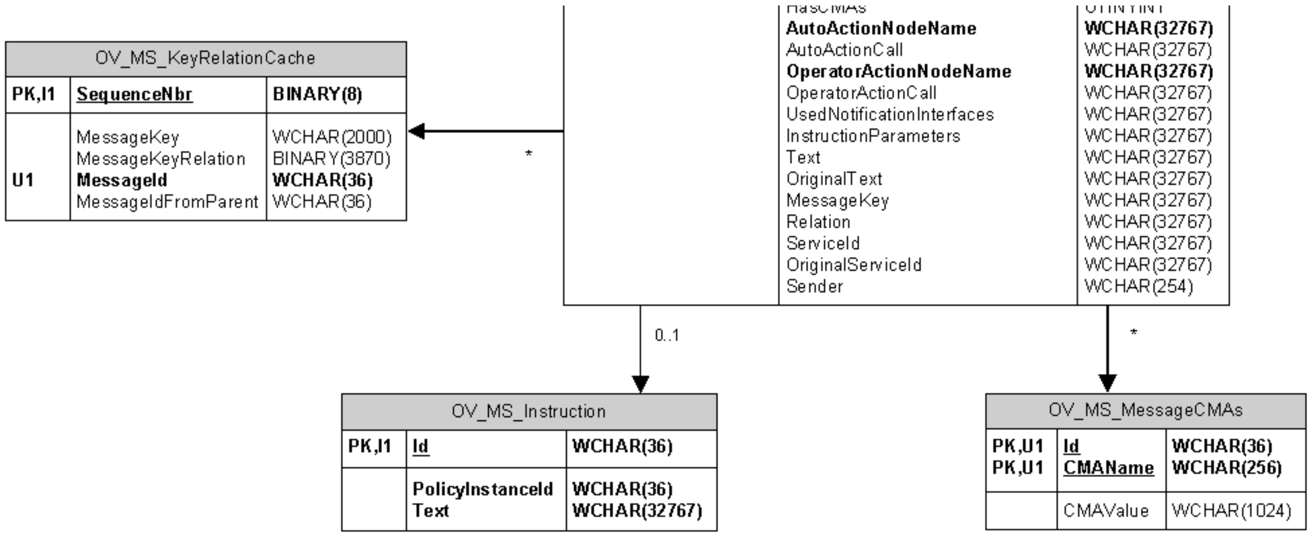
PK,I1	**Id**	**WCHAR(36)**
PK,I1	**ServerName**	**WCHAR(256)**
	ServerIpAddr	INTEGER

OV_MS_Annotation

PK,I1	**MessageId**	**WCHAR(36)**
PK,I1	**OrderNumber**	**INTEGER**
I2,U1	**Id**	**WCHAR(36)**
	Text	**WCHAR(32767)**
	UserOfAnnotation	**WCHAR(1024)**
	TimeCreated	**INTEGER**
	TimeCreatedTimeStamp	**TIMESTAMP**

OV_MS_Message

PK,FK1,FK2,FK3,FK5,I1	**Id**	**WCHAR(36)**
	OriginalId	**WCHAR(36)**
	ConditionId	WCHAR(36)
U1	**State**	**INTEGER**
	TimeOfStateChange	**INTEGER**
	TimeOfStateChangeTimeStamp	**TIMESTAMP**
	UserOfStateChange	**WCHAR(128)**
	NodeName	**WCHAR(254)**
	AgentId	**WCHAR(36)**
	TimeCreated	**INTEGER**
	TimeCreatedTimeStamp	**TIMESTAMP**
	AgentTimeZoneOffset	**INTEGER**
U2	**TimeReceived**	**INTEGER**
U2	**SequenceNbr**	**LARGEINT**
	TimeReceivedTimeStamp	**TIMESTAMP**
	MessageGroup	WCHAR(32)
	Object	WCHAR(254)
	Application	WCHAR(254)
	Type	WCHAR(254)
	ServiceIdHash	WCHAR(112)
	Severity	**SMALLINT**
	LogOnly	**UTINYINT**
	IsProxied	UTINYINT
	Unmatched	**UTINYINT**
	AutoActionState	**SMALLINT**
	AutoActionCreateAnnotation	**UTINYINT**
	AutoActionDoAcknowledge	**UTINYINT**
	OperatorActionState	**SMALLINT**
	OperatorActionCreateAnnotation	**UTINYINT**
	OperatorActionDoAcknowledge	**UTINYINT**
	CreateTroubleTicket	**UTINYINT**
	AcknowledgeAfterTroubleTicket	**UTINYINT**
	DoNotification	**UTINYINT**
	InstructionType	**SMALLINT**
FK4	**InstructionId**	**WCHAR(36)**
	Source	WCHAR(128)
	SourceType	**INTEGER**
	NumberOfAnnotations	INTEGER
	NumberOfDuplicates	INTEGER
	TimeFirstReceived	**INTEGER**
	TimeFirstReceivedTimeStamp	**TIMESTAMP**
	NumberOfStoredDuplicates	INTEGER
	PrimaryNodeName	WCHAR(254)
	IsExternalNode	UTINYINT
	Origin	WCHAR(254)


HasCMAs UTINYINT
AutoActionNodeName WCHAR(32767)
AutoActionCall WCHAR(32767)
OperatorActionNodeName WCHAR(32767)
OperatorActionCall WCHAR(32767)
UsedNotificationInterfaces WCHAR(32767)
InstructionParameters WCHAR(32767)
Text WCHAR(32767)
OriginalText WCHAR(32767)
MessageKey WCHAR(32767)
Relation WCHAR(32767)
ServiceId WCHAR(32767)
OriginalServiceId WCHAR(32767)
Sender WCHAR(254)
OV_MS_KeyRelationCache
PK,I1 SequenceNbr BINARY(8)
MessageKey WCHAR(2000)
MessageKeyRelation BINARY(3870)
U1 MessageId WCHAR(36)
MessageIdFromParent WCHAR(36)
*
0..1
*
OV_MS_Instruction
PK,I1 Id WCHAR(36)
PolicyInstanceId WCHAR(36)
Text WCHAR(32767)
OV_MS_MessageCMAs
PK,U1 Id WCHAR(36)
PK,U1 CMAName WCHAR(256)
CMAValue WCHAR(1024)

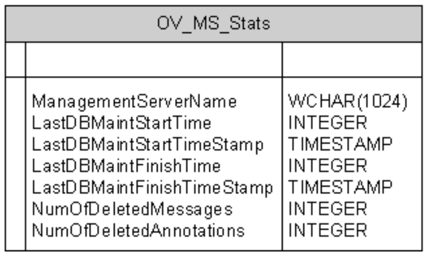

OV_MS_Stats
ManagementServerName WCHAR(1024)
LastDBMaintStartTime INTEGER
LastDBMaintStartTimeStamp TIMESTAMP
LastDBMaintFinishTime INTEGER
LastDBMaintFinishTimeStamp TIMESTAMP
NumOfDeletedMessages INTEGER
NumOfDeletedAnnotations INTEGER

Table OV_MS_Message

In HPOM for Windows, all active and acknowledged messages are stored in one table, OV_MS_Message, for faster acknowledge and unacknowledge operations.

Column Name	Con-straint	Column Type	Description
Id	P, I	nvarchar(36)	Key field to identify the message.
OriginalId		nvarchar(36)	Contains the original message ID when the field "Id" was changed (for example, by the message stream interface).
ConditionId	N	nvarchar(36)	Field to identify the condition matching the message.
State	I	int(4)	The current state of the message. Possible values: 1 : Unknown 2 : Not owned by anyone 3 : Owned 4 : Acknowledged 5 : Node deleted
TimeOfStateChange		int(4)	Time when the last state change occurred. Format: in seconds since 00:00 GMT on 1 Jan 1970
TimeOfStateChangeTimeStamp		datetime(8)	Can be used for queries instead of TimeOfStateChange. Format: YYYY-MM-DD HH:MM:SS.ttt
UserOfStateChange		nvarchar(36)	String of the user that was responsible for the last state change.
Nodename	F	nvarchar(36)	UUID of the nodename from which the message was sent.
AgentId		nvarchar(36)	Contains the ID of the agent.
TimeCreated		int(4)	Time at which the message was created on the agent. Format: in seconds since 00:00 GMT on 1 Jan 1970

TimeCreatedTimeStamp		datetime(8)	Can be used for queries instead of TimeCreated. Format: YYYY-MM-DD HH:MM:SS.ttt
AgentTimeZoneOffset		int(4)	Contains the time offset of the agent time zone to GMT.
TimeReceived	I	int(4)	Time at which the message was created on the server. If duplicate detection is enabled and there are duplicates, this field contains the time at which the last duplicate arrived on the server.
SequenceNbr	I	largeint(8)	Used internally to improve the performance of large queries.
TimeReceivedTimeStamp		datetime(8)	Can be used for queries instead of TimeReceived. Format: YYYY-MM-DD HH:MM:SS.ttt
TimeFirstReceived		int(4)	If duplicate detection is enabled, the field contains the time at which the message was received the first time. If it is disabled, TimeReceived and TimeFirstReceived contain the same data.
TimeFirstReceivedTimeStamp		datetime(8)	Can be used for queries instead of TimeFirstReceived. Format: YYYY-MM-DD HH:MM:SS.ttt
NumberOfDuplicates	N	int(4)	If duplicate detection is enabled, it contains the number of duplicates of the current message. Otherwise, it is 0.
NumberOfStoredDuplicates	N	int(4)	Number of duplicate annotations stored with the message.
MessageGroup	N	nvarchar(32)	MessageGroup attribute of the message.
Object	N	nvarchar(254)	Object attribute of the message.
Application	N	nvarchar(254)	Application attribute of the message.
Type	N	nvarchar(254)	Defines the type of a message. This attribute is used to define filters for the message stream interface.
ServiceId	N	nText(16)	String that defines the service in the service tree that is affected by the message. In the Service ID, the parameters (for example,

			<$MSG_NODE_ID) are already replaced with the values.
OriginalServiceId	N	nText(16)	Contains the Service ID without parameter replacement.
Severity		smallint(2)	Severity attribute of the message. Possible values: 1 : Unknown 2 : Normal 4 : Warning 8 : Minor 16 : Major 32 : Critical
LogOnly		tinyint(1)	Message was sent as log-only to the management server. It is stored immediately as a history message with message state=4. Possible values: 0 : No 1 : Yes
IsProxied	N	tinyint(1)	Flag is 1 if the agent node is different from the node on which the problem occurred.
Unmatched		tinyint(1)	Message does not match any condition. Possible values: 0 : No 1 : Yes
AutoActionState		smallint(2)	Status of the automatic action. Possible values: 2 : Failed 8 : Started 9 : Finished 11 : Defined 12 : No action
AutoActionCreateAnnotation		tinyint(1)	Automatic action generates annotation. Possible values: 0 : No 1 : Yes
AutoActionDoAcknowledge		tinyint(1)	Automatic action acknowledges message. Possible values: 0 : No 1 : Yes

OperatorActionState		smallint(2)	Status of the operator-initiated action. Possible values: 2 : Failed 8 : Started 9 : Finished 11 : Defined 12: No action
OperatorActionCreateAnnotation		tinyint(1)	Operator-initiated action generates annotation. Possible values: 0 : No 1 : Yes
OperatorActionDoAcknowledge		tinyint(1)	Operator-initiated action acknowledges message. Possible values: 0 : No 1 : Yes
CreateTroubleTicket		tinyint(1)	Generate trouble ticket for the message. Possible values: 0 : No 1 : Yes
AcknowledgeAfterTroubleTicket		tinyint(1)	Acknowledge message after generation of the trouble ticket. Possible values: 0 : No 1 : Yes
DoNotification		tinyint(1)	Trigger notification for the message. Possible values: 0 : No 1 : Yes
InstructionType		smallint(2)	Type of instruction Possible values: 0 : No instruction 1 : Instruction text
InstructionId	F	nvarchar(36)	Foreign key field to identify the instruction text.
Source	N	nvarchar(128)	Contains the name and version of the policy that created the message.
SourceType		int(4)	Message source type. Possible values: 0 : Unknown 1 : Console

			2 : Opcmsg 4 : Logfile 8 : Monitor 16 : SNMP 32 : Event correlation / Server Message Stream Interface 64 : Agent Message Stream Interface 128 : External Agent Interface 256 : Scheduled action 4096 : Internal message 8192 : Sub-Product message
NumberOfAnnotations	N	int(4)	Number of annotations attached to the message.
AutoActionNodeName		nText(16)	Identifies the node where the automatic action runs.
AutoActionCall	N	nText(16)	Program call for automatic action.
OperatorActionNodeName		nText(16)	Identifies the node on which the operator-initiated action runs.
OperatorActionCall	N	nText(16)	Program call for operator-initiated action.
InstructionParameters	N	nText(16)	Instructions parameters passed to the instruction text interface.
Text	N	nText(16)	Message text attribute of the message.
OriginalText	N	nText(16)	Text that was intercepted from the agent (for example, the exact line of a log file).
MessageKey	N	nText(16)	Message key to identify a certain type of message.
Relation	N	nText(16)	Messages with message keys that are stored here are acknowledged by this message.
ServiceIdHash	N	nvarchar(112)	Hash value calculated from ServiceId.
PrimaryNodeName	N	nvarchar(254)	Clear text version of the primary node name.
isExternalNode		tinyint(1)	Determines whether the message came from an external node. Possible values: 0 : No 1 : Yes

Sender	N	nvarchar(254)	Last sender management server of a forwarded message.
Origin	N	nvarchar(254)	The first management server that received the forwarded message from the agent.
HasCMAs		tinyint(1)	Determines whether the message has CMAs in the OV_MS_MessageCMAs table. Possible values: 0 : No 1 : Yes

P :Primary Key
I :Index
N :NULL values possible
F :Foreign Key

Table OV_MS_KeyRelationCache

This table is maintained by the good/bad message correlation component of the Message Action Server. It is used for acknowledging messages by message key if the memory cache size exceeds a threshold.

Column Name	Con-straint	Column Type	Description
SequenceNbr	P,I	binary(8)	Artificial primary key.
MessageId	I, F	nvarchar(36)	Message Id to which the relation belongs.
MessageIdFromParent	N	nvarchar(36)	Message Id of the original message if message is a duplicate. Used to prevent the duplicate message from acknowledging the original message.
MessageKey	N	nvarchar(2000)	Message key to identify a certain type of message.
MessageKeyRelation	N	varbinary(3950)	Messages with message keys that are stored here are acknowledged by this message.

P :Primary Key
I :Index
N :NULL values possible
F :Foreign Key

Table OV_MS_Annotation

Annotations that belong to messages are stored in this extra table.

Column Name	Con-straint	Column Type	Description
MessageId	P,I	nvarchar(36)	Message Id to which the annotation belongs.
OrderNumber	P,I	int(4)	Order number of the annotation.
Id	P,I	nvarchar(36)	Unique Id of an annotation.
Text		nText(16)	Text attribute of an annotation.
UserOfAnnotation		nvarchar(1024)	User who created the annotation.
TimeCreated		int(4)	Time when the annotation was created. Format: in seconds since 00:00 GMT on 1 Jan 1970
TimeCreatedTimeStamp		datetime(8)	Can be used for queries instead of TimeCreated. Format: YYYY-MM-DD HH:MM:SS.ttt

P :Primary Key
I :Index
N :NULL values possible
F :Foreign Key

Table OV_MS_Instruction

All instruction texts are stored in this table.

Column Name	Con-straint	Column Type	Description
Id	P,I	nvarchar(36)	Key field that identifies an instruction.
PolicyInstanceId		nvarchar(36)	Id of the policy that contains the instruction text.
Text		nText(16)	Instruction text attribute.

P :Primary Key
I :Index
N :NULL values possible
F :Foreign Key

Table OV_MS_MessageCMAs

All custom message attributes (CMAs) belonging to messages are stored in this table.

Column Name	Con-straint	Column Type	Description
Id	P,I	nvarchar(36)	Message Id to which the CMA belongs.
CMAName	P,I	nvarchar(256)	Name of the CMA.
CMAValue		nvarchar(1024)	Value of the CMA.

P :Primary Key
I :Index
N :NULL values possible
F :Foreign Key

Table OV_MS_ServerList

All servers on which a forwarded message is available are stored in this table.

Column Name	Con-straint	Column Type	Description
Id	P,I	nvarchar(36)	Message Id to which the server belongs.
Servername	P,I	nvarchar(256)	Name of the server.
ServerIpAddr		int(4)	IP address of the server.

P :Primary Key
I :Index
N :NULL values possible
F :Foreign Key

Table OV_MS_Stats

This table is used for statistical purposes only. It has no direct linkage with the other OV_MS_* tables. It might be interesting for building reports on database maintenance.

Column Name	Con-straint	Column Type	Description
ManagementServerName	N	nvarchar(1024)	Management server name used for database maintenance.
LastDBMaintStartTime	N	int(4)	Start time of the last database maintenance job.
LastDBMaintStartTimeTimeStamp	N	datetime(8)	String value of LastDBMaintStartTime for easier use on queries. Format: YYYY-MM-DD HH:MM:SS.ttt
LastDBMaintFinishTime	N	int(4)	Finish time of the last database maintenance job.
LastDBMaintFinishTimestamp	N	datetime(8)	String value of LastDBMaintFinishTime for easier use on queries. Format: YYYY-MM-DD HH:MM:SS.ttt
NumOfDeletedMessages	N	int(4)	Number of messages that were deleted during the maintenance task.
NumOfDeletedAnnotations	N	int(4)	Number of annotations that were deleted during the maintenance task.

P :Primary Key
I :Index
N :NULL values possible
F :Foreign Key

Database Schema for Policy Management and Deployment

The entity-relationship diagram illustrates the contents of the database tables populated as a result of changes and actions in Policy Management and Deployment (PMAD). Each of the tables in the diagram is described in detail in its own topic. To display more information about a particular table, click the table you want to investigate in the PMAD entity-relationship diagram below.

Available information includes the following:

- Column Name
- Constraints
- Column Type
- Description

PMAD Entity Relationships: All | Platform | Node | Policy

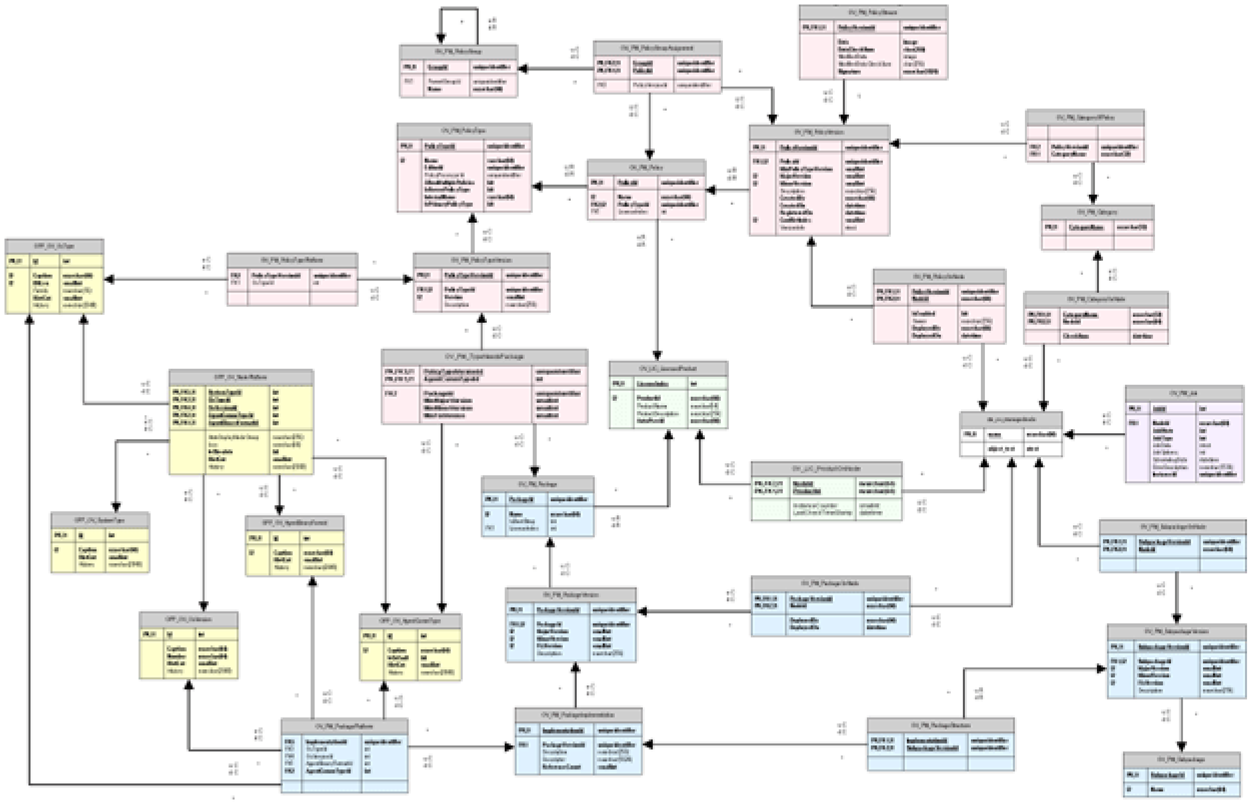


Table OPP_OV_AgentBinaryFormat

The OPP_OV_AgentBinaryFormat table contains all agent binary formats supported by HPOM for Windows 8.

Column Name	Constraint	Column Type	Description
Id	PK, I1	int	ID of the supported agent binary format.
Caption	I2	nvarchar(64)	Non-localized caption string of the supported agent binary formats.
HistCnt		smallint	History count. See History.
History		nvarchar(2048)	Whenever an agent binary format is added, updated, or removed, the history count column (HistCnt) as well as this column is updated, so changes to the platform model may be added and removed in any order.

Table OPP_OV_AgentCommType

The OPP_OV_AgentCommType table contains all agent communication types (DCE or HTTPS) supported by HPOM for Windows 8.

Column Name	Constraint	Column Type	Description
Id	PK, I1	int	ID of the supported agent communication type.
Caption	I2	nvarchar(64)	Non-localized caption string of the supported agent communication types (DCE or HTTPS).
IsDefault		bit	Set to true for HTTPS, and false for DCE.
HistCnt		smallint	History count. See History.
History		nvarchar(2048)	Whenever an agent communication type is added, updated, or removed, the history count column (HistCnt) as well as this column is updated, so changes to the platform model may be added and removed in any order.

Table OPP_OV_NodePlatform

The OPP_OV_NodePlatform table contains all valid combinations of system type, OS type, OS version, agent binary format, and agent communication type that are either supported or marked as obsolete as a node platform in HPOM for Windows 8.00.

Column Name	Constraint	Column Type	Description
SystemTypeId	PK, FK5, I1	int	ID of the referenced system type.
OsTypeId	PK, FK3, I1	int	ID of the referenced OS type.
OsVersionId	PK, FK4, I1	int	ID of the referenced OS version.
AgentCommTypeId	PK, FK2, I1	int	ID of the referenced agent communication type.
AgentBinaryFormatId	PK, FK1, I1	int	ID of the referenced agent binary format.
AutoDeployNodeGroup		nvarchar(256)	Product-defined node group to which a node of this platform is added automatically at creation time.
Icon		nvarchar(64)	Icon to be used in the UI for a node of this platform.
IsObsolete		bit	If set to true, the platform is marked as obsolete in HPOM for Windows 8. In this case, it is no longer possible to create a new node with this platform in the node editor.
HistCnt		smallint	History count. See History.
History		nvarchar(2048)	Whenever a node platform is added, updated, or removed, the history count column (HistCnt) as well as this column is updated, so changes to the platform model may be added and removed in any order.

Table OPP_OV_OsType

The OPP_OV_OsType table contains all operating system types supported by HPOM for Windows 8.

Column Name	Constraint	Column Type	Description
Id	PK, I1	int	ID of the supported OS type.
Caption	I2	nvarchar(64)	Non-localized caption string of the supported OS types.
BitLen	I2	smallint	Bit length of the OS.
Family		nvarchar(16)	OS family to which the OS type belongs. It is set to UNIX for all UNIX-derived OS types, and NULL for all other OS types.
HistCnt		smallint	History count. See History.
History		nvarchar(2048)	Whenever an OS type is added, updated, or removed, the history count column (HistCnt) as well as this column is updated, so changes to the platform model may be added and removed in any order.

Table OPP_OV_OsVersion

The OPP_OV_OsVersion table contains all operating system versions supported by HPOM for Windows 8.

Column Name	Constraint	Column Type	Description
Id	PK, I1	int	ID of the supported OS version.
Caption		nvarchar(64)	Non-localized caption string of the supported OS versions.
Number		nvarchar(64)	Version number of the supported OS version.
HistCnt		smallint	History count. See History.
History		nvarchar(2048)	Whenever an OS version is added, updated, or removed, the history count column (HistCnt) as well as this column is updated, so changes to the platform model may be added and removed in any order.

Table OPP_OV_SystemType

The OPP_OV_SystemType table contains all system types supported by HPOM for Windows 8.

Column Name	Constraint	Column Type	Description
Id	PK, I1	int	ID of the supported system type.
Caption	I2	nvarchar(64)	Non-localized caption string of the supported system types.
HistCnt		smallint	History count. See History.
History		nvarchar(2048)	Whenever a system type is added, updated, or removed, the history count column (HistCnt) as well as this column is updated, so changes to the platform model may be added and removed in any order.

Table OV_LIC_LicensedProduct

The OV_LIC_LicensedProduct table contains information about the licensed products to which a policy or package can be linked. The tables OV_PM_Policy and OV_PM_Package are linked to the licensed product using the LicenseIndex column. They verify license validity before policy or package deployment.

Column Name	Constraint	Column Type	Description
LicenseIndex	PK, I1	int	Primary key of the licensed product. It is referenced by all policies and packages that belong to this product. The value of this column can be different on different HPOM for Windows servers for that same product. Only the product Id is guaranteed to be unique on all HPOM for Windows servers.
ProductId	I2	nvarchar(64)	ID of the licensed product.
ProductName		nvarchar(64)	Name of the licensed product.
ProductDescription		nvarchar(256)	String that describes the licensed product.
AutoPassId		nvarchar(64)	Used only for migration from HPOM for Windows 7.50 to map a licensed product to the new product ID.

Table OV_LIC_ProductOnNode

The OV_LIC_ProductOnNode table keeps the inventory information about all the products that are installed on the nodes of the managed environment. Records in this table are cascaded, and so removed automatically when the managed node is removed.

Column Name	Constraint	Column Type	Description
ProductId	PK, FK2, I1	nvarchar(64)	ID of the product that is installed on the referenced node.
NodeId	PK, FK1, I1	nvarchar(64)	ID of the node where the referenced product is installed.

Table OV_PM_Category

The OV_PM_Category table is a catalog of all categories that are currently referenced by all policies.

Column Name	Constraint	Column Type	Description
CategoryName	PK, I1	nvarchar(32)	Name of the category.

Table OV_PM_CategoryOfPolicy

There is one entry in the OV_PM_CategoryOfPolicy table for every category that is assigned to a policy version.

Column Name	Constraint	Column Type	Description
PolicyVersionId	FK2	uniqueidentifier	ID of the policy version.
CategoryName	FK1	nvarchar(32)	Name of the category.

Table OV_PM_CategoryOnNode

The OV_PM_CategoryOnNode table represents the policy categories that are deployed on a managed node. However, it is not used to expose a category inventory to PMAD clients. The main purpose of this table is to store the category CheckSum (a directory timestamp) that is calculated during instrumentation deployment. This feature is used by PMAD to support the automatic instrumentation redeployment. Records in this table are cascaded, and so removed automatically when the corresponding category or managed node is removed.

Column Name	Constraint	Column Type	Description
CategoryName	PK, FK1, I1	nvarchar(32)	Name of the category.
NodeId	PK, FK2, I1	nvarchar(64)	ID of the node.
CheckSum		datetime	Timestamp of the instrumentation directory (with the same name as the referenced category) at the time the policy was deployed to the node.

Table OV_PM_Job

The table OV_PM_Job contains data about all the deployment jobs that are currently in the queue. Records in this table are cascaded, and so removed automatically when the corresponding managed node is removed.

Column Name	Constraint	Column Type	Description
JobId	PK, I1	int	ID of the deployment job.
NodeId	FK1	nvarchar(64)	ID of the node for which the job is scheduled.
JobState		int	Current state of the deployment job.
JobType		int	Type of the deployment job (for example, install policy, remove package, redeploy node, and so on).
JobData		ntext	Any data that is needed to recover the job from the DB when the Policy Management server is started.
JobOptions		int	Options that have been specified for this deployment job.
SchedulingData		datetime	Date and time when the job was added to the queue.
ErrorDescription		nvarchar(1536)	If the job has failed for some reason, this column contains the error description string.
InstanceId		uniqueidentifier	ID of the object for which this job is scheduled (for example, policy or package version ID).

Table OV_PM_Package

Every row of the OV_PM_Package table represents exactly one logical package. It contains the name of the package (which is common for all versions of the same package) and a reference to the licensed product (if the package is licensed).

Column Name	Constraint	Column Type	Description
PackageId	PK, I1	uniqueidentifier	ID of the package.
Name	I2	nvarchar(64)	Name of the package.
IsBootStrap		bit	Set to true for the package that contains all required infrastructure to deploy additional packages. It is set to true for the HP Operations agent package. For all other packages, it is set to false.
LicenseIndex	FK1	int	Reference to the licensed product to which this package belongs.

Table OV_PM_PackageImplementation

The OV_PM_PackageImplementation table represents the various implementations (binaries) of a package version. The Descriptor field contains a full path to the XML-based descriptor file.

Column Name	Constraint	Column Type	Description
ImplementationId	PK, I1	uniqueidentifier	ID of the package implementation.
PackageVersionId	FK1	uniqueidentifier	ID of the referenced package version.
Description		nvarchar(256)	String that describes this package implementation.
Descriptor		nvarchar(1024)	Location of the XML-based package descriptor file in the file system of the management server.
ReferenceCount		smallint	Number that is increased each time the same package implementation (binary) is registered with Policy Management. It is required for patch handling.

Table OV_PM_PackageOnNode

The OV_PM_PackageOnNode table keeps the inventory information about all the packages installed on the nodes of the managed environment. Records in this table are cascaded, and so removed automatically when the corresponding package or managed node are removed.

Column Name	Constraint	Column Type	Description
PackageVersionId	PK, FK1, I1	uniqueidentifier	ID of the package version that is installed on the referenced node.
NodeId	PK, FK2, I1	nvarchar(64)	ID of the node where the referenced package version is installed.
DeployedBy		nvarchar(64)	Name of the user who has deployed the package to the node.
DeployedOn		datetime	Deployment time of the package.

Table OV_PM_PackagePlatform

The OV_PM_PackagePlatform table references the node platforms that are supported by a certain package implementation.

Column Name	Constraint	Column Type	Description
ImplementationId	FK5	uniqueidentifier	ID of the package implementation.
SystemTypeId			ID of the system type that the referenced package implementation supports. It can be NULL, if all system types are supported.
OSTypeId	FK3	int	ID of the OS type that the referenced package implementation supports. It can be NULL, if all OS types are supported.
OSVersionId	FK4	int	ID of the OS versions that the referenced package implementation supports. It can be NULL, if all OS versions are supported.
AgentBinaryFormatId	FK1	int	ID of the Agent Binary Format that the referenced package implementation supports. It can be NULL, if all Agent Binary Formats are supported.
AgentCommTypeId	FK2	int	ID of the Agent Communication Type that the referenced package implementation supports. It can be NULL, if all Agent Communication Types are supported.

Table OV_PM_PackageStructure

The OV_PM_PackageStructure table stores the subpackage list of a certain package implementation (that is, all subpackages that are part of a package on a certain platform). Records in this table are cascaded, and so removed automatically when a corresponding package implementation is removed from the table OV_PM_PackageImplementation .

Column Name	Constraint	Column Type	Description
ImplementationId	PK, FK1, I1	uniqueidentifier	ID of the package implementation.
SubpackageVersionId	PK,FK2,I1	uniqueidentifier	ID of the subpackage version that belongs to the referenced package implementation.

Table OV_PM_PackageVersion

Each row in the OV_PM_PackageVersion table represents a package version. The main package identifier is a set of four values: PackageId, MajorVersion, MinorVersion, and FixVersion. The PackageVersionId is an internal unique identifier for every package version. The package version is linked to the logical package it belongs to using the PackageId.

Column Name	Constraint	Column Type	Description
PackageVersionId	PK, I1	uniqueidentifier	ID of this package version.
PackageId	FK1, I2	uniqueidentifier	ID of the logical package.
MajorVersion	I2	smallint	Major version number of the package.
MinorVersion	I2	smallint	Minor version number of the package.
FixVersion	I2	smallint	Fix version number of the package.
Description		nvarchar(256)	String that describes the purpose of this package version.

Table OV_PM_Policy

Every row of the OV_PM_Policy table represents exactly one logical policy. It contains all properties that are common for all versions of the same policy.

Column Name	Constraint	Column Type	Description
PolicyId	PK, I1	uniqueidentifier	ID of the policy.
Name	I2	nvarchar(64)	Name of the policy.
PolicyTypeId	FK2, I2	uniqueidentifier	ID of the policy type to which the policy belongs.
LicenseIndex	FK1	int	Reference to the licensed product to which this policy belongs.

Table OV_PM_PolicyGroup

The OV_PM_PolicyGroup table contains all policy groups. The ParentGroupId field links every policy group to its parent, and so creates the policy group hierarchy. Records in this table are cascaded. That is, all child policy groups are removed automatically when the parent policy group is removed.

Column Name	Constraint	Column Type	Description
GroupId	PK, I1	uniqueidentifier	ID of the policy group.
ParentGroupId	FK1	uniqueidentifier	Reference to the parent policy group, if it exists.
Name	FK1	nvarchar(64)	Name of the policy group.

Table OV_PM_PolicyGroupAssignment

The OV_PM_PolicyGroupAssignment table implements the policy to group assignment relation. PolicyVersionId is an optional field referring to the policy version assigned to the group. If this field is empty, the latest policy version is always taken. Records in this table are cascaded, and so automatically removed when the corresponding records are deleted from the OV_PM_Policy , OV_PM_PolicyVersion , or OV_PM_PolicyGroup table.

Column Name	Constraint	Column Type	Description
GroupId	PK, FK2, I1	uniqueidentifier	ID of the referenced policy group.
PolicyId	PK, FK1, I1	uniqueidentifier	ID of the referenced policy.
PolicyVersionId	FK3	uniqueidentifier	Optional. ID of the referenced policy version.

Table OV_PM_PolicyOnNode

The OV_PM_PolicyOnNode table keeps the inventory information about policies installed on different nodes. The IsEnabled flag shows the enabled/disabled status of a policy on a managed node. The Owner field specifies the management server system that owns a policy on the managed node. Records in this table are cascaded, and so are removed automatically when the corresponding policy or managed node is removed.

Column Name	Constraint	Column Type	Description
PolicyVersionId	PK, FK1, I1	uniqueidentifier	ID of the policy version that is installed on the referenced node.
NodeId	PK, FK2, I1	nvarchar(64)	ID of the node where the referenced policy version is installed.
IsEnabled		bit	If true, the policy version is enabled on the node.
Owner		nvarchar(256)	Name of the server that has deployed the policy to the node.
DeployedBy		nvarchar(64)	Name of the user who has deployed the policy to the node.
DeployedOn		datetime	Deployment time of the policy.

Table OV_PM_PolicyStream

The OV_PM_PolicyStream table contains the data stream for the policy version, if it exists. It is possible that a policy version does not have an entry in this table. In HPOM for Windows 8.00, it is possible to have only one policy data part per policy version, but it might be necessary to support multiple policy data parts in future product releases.

If the policy data contains action strings, they will be signed when the policy is created. PMAD will save the new modified data stream in an extra column.

A checksum is calculated over the policy data and a set of policy properties (for example, the policy version ID and license index). The checksum is used to verify data consistency and to enforce licensing.

Records in this table are cascaded and removed automatically when a corresponding policy version is removed from the table OV_PM_PolicyVersion . The signature is calculated over the modified policy data stream if it exists. If not, it is calculated over the unmodified policy data stream (column Data). If a modified data stream exists, it is deployed to the HTTPS-based agent while the unmodified policy data is deployed to the DCE-based agent.

Column Name	Constraint	Column Type	Description
PolicyVersionId	PK, FK1, I1	uniqueidentifier	ID of the policy version to which the policy data belongs.
Data		image	Policy data stream stored as BLOB.
DataCheckSum		char(256)	Checksum for the policy data stream.
ModifiedData		image	Modified (signed) policy data stream stored as BLOB.
ModifiedDataCheckSum		char(256)	Checksum for the modified (signed) policy data stream.
Signature		nvarchar(1024)	Signature over the modified policy data stream (column ModifiedData), if it exists. If not, the signature is calculated over the column Data.

Table OV_PM_PolicyType

Every row of the OV_PM_PolicyType table represents exactly one logical policy type. It contains all properties that are common for all versions of the same policy type. The Name field stores the policy type name. The EditorId field contains a class ID of the ActiveX component implementing the Policy Editor of the given policy type, which is called when a policy is created or edited. The PolicyProcessorId stores a class ID of a COM component, which is called before the policy is saved, deleted, and deployed. The IsServerPolType flag allows distinguishing policies, which users cannot deploy to a remote managed node. Users use them only for the management server configuration (server policies).

Column Name	Constraint	Column Type	Description
PolicyTypeId	PK, I1	uniqueidentifier	ID of the policy type.
Name	I2	varchar(64)	Name of the policy type that appears in the GUI. It is not localized.
EditorId		uniqueidentifier	Class ID of the ActiveX component implementing the Policy Editor.
PolicyProcessorId		uniqueidentifier	Class ID of a COM component, which is called before the policy is saved, deleted, and deployed.
AllowMultiplePolicies		bit	If true, users can deploy more than one policy of this type to a managed node.
IsServerPolicyType		bit	If true, users can deploy policies of this type only to the local management server itself (so-called server policies).
InternalName		varchar(64)	Name of the policy type that the HP Operations agent can recognize. For example, the Logfile Entry and the Windows Event Log policy type in HPOM for Windows map to the policy type "le" (logfile encapsulator) of the HP Operations agent.
IsPrimaryPolicyType		bit	Used for HTTPS-based nodes that are managed by both an HPOM for Windows and an HPOM for UNIX server. HPOM for UNIX 8.x recognizes the policy types that the agent

			supports. If an "le" policy is deployed to the node from HPOM for UNIX, and afterwards the user calls the synchronize operation for this node in HPOM for Windows, Policy Management must determine if this policy should be displayed as a Logfile Entry or a Windows Event Log policy. The property IsPrimaryPolicyType determines which of the two HPOM for Windows policy types takes precedence.

Table OV_PM_PolicyTypePlatform

The OV_PM_PolicyTypePlatform table specifies the operating system, if policies of a certain type make sense only for a particular OS type. For example, you can deploy Windows Event Log policies only to a Windows managed nodes. Hence, the Windows Event Log policy type is limited to the Windows OS type. If there is no entry for a certain policy type version in this table, then all OS types are supported by this policy type.

Column Name	Constraint	Column Type	Description
PolicyTypeVersionId	FK2	uniqueidentifier	ID of the policy type version.
OSTypeId	FK1	int	ID of the operating system type.

Table OV_PM_PolicyTypeVersion

Every record in the OV_PM_PolicyTypeVersion table represents one policy type version. The main policy type identifier consists of two values: PolicyTypeId and Version. The PolicyTypeVersionId is an internal unique identifier for every policy type version. The policy type version is linked to the logical policy type it belongs to using the PolicyTypeId. A description string can be set for every version of the policy type. The policy type version number is a single-digit number that maps to the minimum syntax version of the policy that the associated agent package must support on the managed node.

Column Name	Constraint	Column Type	Description
PolicyTypeVersionId	PK, I1	uniqueidentifier	ID of the policy type version.
PolicyTypeId	FK1, I2	uniqueidentifier	ID of the logical policy type.
Version	I2	smallint	Minimize policy syntax version number that the agent package that is linked to this policy type version must support.
Description		nvarchar(256)	Description string for this policy type version.

Table OV_PM_PolicyVersion

Every record in the OV_PM_PolicyVersion table represents one policy version. The main policy identifier is a set of three values: PolicyId, MajorVersion, and MinorVersion. The PolicyVersionId is an internal unique identifier for every policy version. The policy version is linked to the logical policy it belongs to using the PolicyId.

The MinPolicyTypeVersion specifies the minimal version of the Policy Type that can interpret the syntax of the policy version.

Column Name	Constraint	Column Type	Description
PolicyVersionId	PK, I1	uniqueidentifier	ID of this policy version.
PolicyId	FK1,I2	uniqueidentifier	ID of the logical policy.
MinPolicyTypeVersion		smallint	Syntax version of this policy version. It corresponds to the minimum required policy type version, which determines the minimum agent package version that must be installed on the node to interpret the syntax of the policy.
MajorVersion	I2	smallint	Major version of this policy version.
MinorVersion	I2	smallint	Minor version of this policy version.
Description		nvarchar(256)	Purpose of this policy version.
CreatedBy		nvarchar(64)	Name of the user who created this policy version.
CreatedOn		datetime	Time when this policy version was created.
RegisteredOn		datetime	Time when this policy was added to (registered with) the server.
ConflictIndex	I2	smallint	If the value of this column is greater than zero (for example, one or two), there are two or more instances of the same policy version registered with the policy management server. Although rare, this could happen if the

			same policy version is modified on different management servers, saved under the same version number, and then imported on the same management server. If, for example, you modify a SPI policy, save it under the next higher major version number, and then a new version of this SPI is installed on the server that registers the same policy version, you run into this conflict scenario. In such a case, the policy version with a conflict index of one or two appears in the UI with one or two asterisks (* or **) at the end of the policy name. You must decide which of the two or more instances of the policy version should be deleted.

Table OV_PM_Subpackage

Every row of the OV_PM_Subpackage table represents a patchable unit that is part of a package (for example, HPOVXPL as part of the Operations HTTPS agent package). It contains the subpackage name that is common for all versions of the same subpackage.

Column Name	Constraint	Column Type	Description
SubpackageId	PK, I1	uniqueidentifier	ID of the logical subpackage.
Name	I2	nvarchar(64)	Name of the subpackage.

Table OV_PM_SubpackageOnNode

The OV_PM_SubpackageOnNode table keeps the inventory information about all the subpackages installed on the nodes of managed environment. Records in this table are cascaded, and so removed automatically when the corresponding subpackage or managed node is removed.

Column Name	Constraint	Column Type	Description
SubpackageVersionId	PK, FK1, I1	uniqueidentifier	ID of the subpackage version that is installed on the referenced node.
NodeId	PK, FK2, I1	nvarchar(64)	ID of the node where the referenced subpackage version is installed.

Table OV_PM_SubpackageVersion

Every row of the OV_PM_SubpackageVersion table represents a specific subpackage version.

Column Name	Constraint	Column Type	Description
SubpackageVersionId	PK, I1	uniqueidentifier	ID of this subpackage version.
SubpackageId	FK1, I2	uniqueidentifier	ID of the logical subpackage.
MajorVersion	I2	smallint	Major version number of this subpackage.
MinorVersion	I2	smallint	Minor version number of this subpackage.
FixVersion	I2	smallint	Fix version number of this subpackage.
Description		nvarchar(256)	String that describes the purpose of this subpackage version.

Table OV_PM_TypeNeedsPackage

The OV_PM_TypeNeedsPackage table implements the dependency of a policy type to its agent package version. This dependency is always defined using the minimum required version of the agent that must be installed on the node to interpret the policy syntax. The specified minimum required package version are not required in the database if higher package versions are available. There are at most two records in this table for every policy type: one for the DCE and one for the HTTP-based agent. Records in this table are cascaded, and removed automatically when a corresponding policy type is removed.

Column Name	Constraint	Column Type	Description
PolicyTypeVersionId	PK, FK3, I1	uniqueidentifier	ID of the policy type version.
AgentCommTypeId	PK, FK1, I1	int	ID of the supported Agent Communication Type (either DCE or HTTPS).
PackageId	FK2	uniqueidentifier	ID of the agent package.
MinMajorVersion		smallint	Major version number of the minimum agent package that must be installed on the node.
MinMinorVersion		smallint	Minor version number of the minimum agent package that must be installed on the node.
MinFixVersion		smallint	Fix version number of the minimum agent package that must be installed on the node.

Table sto_ov_managednode

The sto_ov_managednode table contains all nodes (excluding external nodes) of the managed environment. Except for the node ID, all properties of a managed node can be accessed only using the WMI interface. The tables of the policy management component are linked to this table using the node id (the "name" column).

Column Name	Constraint	Column Type	Description
name	PK, I1	nvarchar(64)	ID of the managed node.

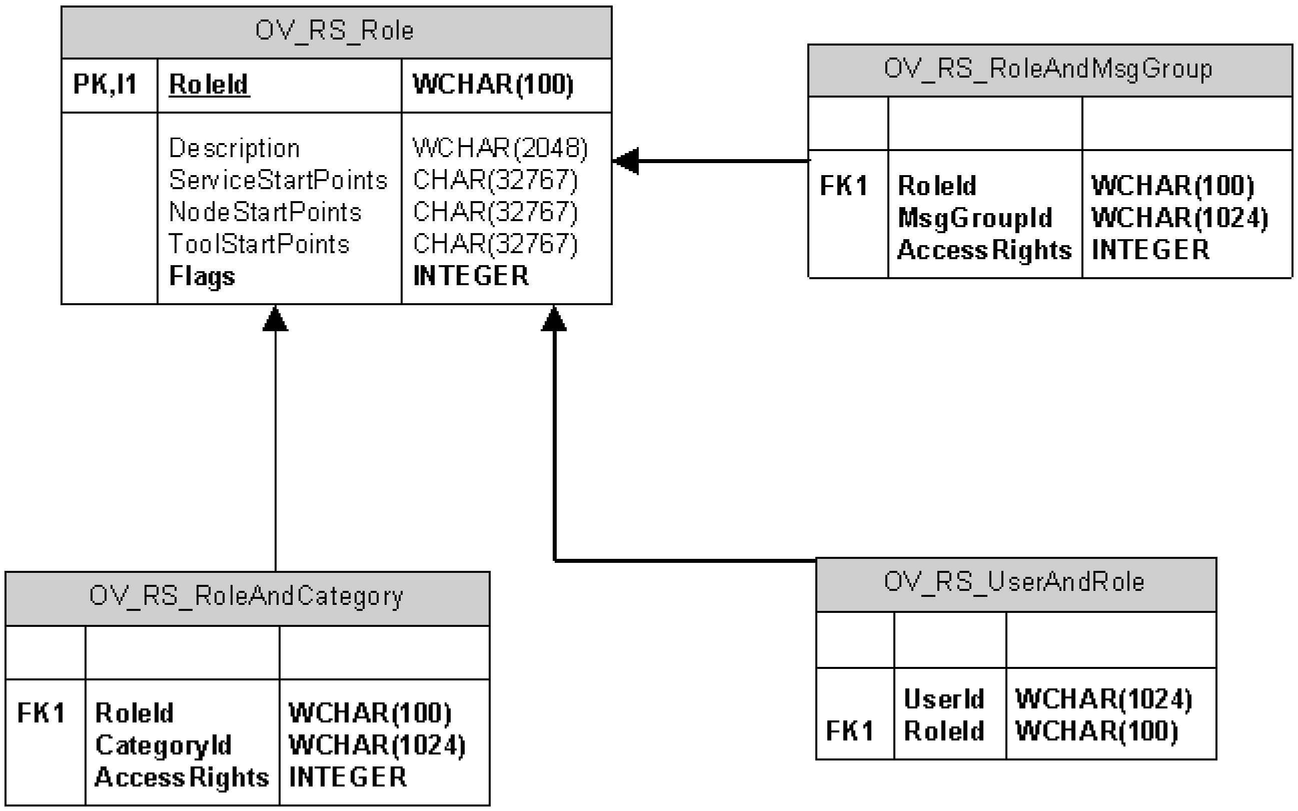
Database Schema for the Security Server

The following ERM diagrams illustrate the contents of Security Server tables:

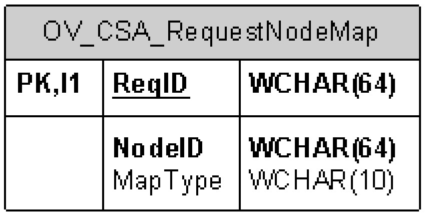
- User roles
- Certificate requests

NOTE:
The HP Operations agent API includes support for C/C++ and Java, as well as for every language that supports DCOM automation (for example, VB, VBScript, JScript, and so on). However, the agent message stream interface supports C APIs only. All of the APIs are built using Microsoft Visual Studio 2005.

ERM diagram of the user role tables



ERM diagram of the certificate request tables



OV_CSA_RequestNodeMap		
PK,I1	**<u>ReqID</u>**	**WCHAR(64)**

	NodeID
	MapType
		WCHAR(64)
		WCHAR(10)

Table OV_RS_Role

In HPOM for Windows, all user roles are stored in the table OV_RS_Role.

Column Name	Constraint	Column Type	Description
RoleId	P, I	wchar(100)	Key field to identify the user role.
Description		wchar(2048)	Description of the user role.
ServiceStartPoints		char(32767	Start points of the services available for this user role. Format: [<length> <id>] Example: 1 31 ov_service.name="root_services"
NodeStartPoints		char(32767)	Start points of the node groups available for this user role. Same format as ServiceStartPoints.
ToolStartPoints		char(32767)	Start points of the tools available for this user role. Same format as ServiceStartPoints.
Flags		integer	Stores additional rights for this user role. This is a bit field with the following values: R_NONE (0) No additional rights available. R_HIDE_REPORTS (0x40) Hide reports. R_HIDE_GRAPHS (0x80) Hide graphs. R_PMAD_CONNECT (0x100) View policy management. R_PMAD_GROUP (0x200) Policy group handling. R_PMAD_JOB (0x400)

			PMAD job handling. R_PMAD_CONFIG (0x800) Administer policies and packages. R_PMAD_IGNOREPOLICYOWNER (0x1000) Ignore policy owner. R_ADMIN (0xffff) All current and future rights.

P= Primary Key
I = Index
N = NULL values possible
F = Foreign Key

Table OV_RS_RoleAndMsgGroup

This table contains the available message groups for the roles.

Column Name	Constraint	Column Type	Description
RoleId	F	wchar(100)	Key field to identify the user role.
MessageGroupId		wchar(1024)	ID of the message group. The message group ID F655A23E-5661-444b-B9E6-C9062B0B179A is the default message group.
AccessRights		integer	Stores the access rights for this message group. This is a bit field with the following values: RAM_NONE (0) No rights available. RAM_OWN (0x1) Own messages. RAM_DISOWN (0x2) Disown messages. RAM_ACKNOWLEDGE (0x4) Acknowledge messages. RAM_UNACKNOWLEDGE (0x8) Unacknowledge messages. RAM_CHANGESEVERITY (0x10) Change severity. RAM_RELAUNCHAUTOCMD (0x20) Relaunch automatic commands. RAM_LAUNCHOPERINITCMD (0x40) Launch operator-initiated commands. RAM_CHANGETEXT (0x80) Change message text. RAM_VIEW (0x100) Messages are visible. RAM_ADMIN (0xffff)

			All current and future rights.

P= Primary Key
I = Index
N = NULL values possible
F = Foreign Key

Table OV_RS_UserAndRole

This table stores the relations between users and user roles.

Column Name	Constraint	Column Type	Description
UserId		wchar(1024)	SID of the Windows user and user group assigned to this role.
RoleId	F	wchar(100)	Key field to identify the user role.

P= Primary Key
I = Index
N = NULL values possible
F = Foreign Key

Table OV_RS_RoleAndCategory

This table contains the available policy categories for the roles.

Column Name	Constraint	Column Type	Description
RoleId	F	wchar(100)	Key field to identify the user role.
CategoryId		wchar(1024)	ID of the policy category.
AccessRights		integer	Stores the access rights for this message group. This is a bit field with the following values: RAM_NONE (0) No rights available. RAC_READ (0x1) Read policies. RAC_DEPLOY (0x2) Deploy policies. RAC_EDIT (0x10) Edit policies. RAC_DELETE (0x20) Delete policies. RAM_ADMIN (0xffff) All current and future rights.

P= Primary Key
I = Index
N = NULL values possible
F = Foreign Key

Table OV_CSA_RequestNodeMap

This table stores the mapping of ungranted certificate requests to nodes.

Column Name	Constraint	Column Type	Description
ReqId	P, I	wchar(64)	ID of the certificate request.
NodeId		wchar(64)	ID of the node to which this certificate request is mapped.
MapType		wchar(10)	Stores how this mapping was done. Possible values: ■ Manual ■ Automatic

P= Primary Key
I = Index
N = NULL values possible
F = Foreign Key

Java API

HPOM provides a set of Java classes on the HP Operations agent to

- create and send a message to the HPOM management server
- acknowledge a previously sent message
- send a monitor value to the HPOM monitor agent

NOTE:
The HP Operations agent API includes support for C/C++ and Java, as well as for every language that supports DCOM automation (for example, VB, VBScript, JScript, and so on). However, the agent message stream interface supports C APIs only. All of the APIs are built using Microsoft Visual Studio 2005.

JAR files

The JAR files `jopcagtbase.jar` and `jopcagtmsg.jar` that are necessary to use the APIs are installed together with the agent on the managed node.

Examples

Examples of how the API classes can be used from Java are available in the directory

`%OvInstallDir%\examples\OVOW\DevelopmentKit\Agent\Java`

on the HPOM for Windows management server.

On Windows

To use the Java HPOM classes:

- the -classpath parameter used for the javac and java commands must include the jopcagtbase.jar and jopcagtmsg.jar files
- the PATH system variable must include the directory where the shared library files reside. The agent installation does this automatically.

See "%OvAgentDir%/www/htdocs/jdoc_agent/index.html" for a javadoc style class documentation.

To compile and run the example code:

- cd to the `%OvInstallDir%\examples\OVOW\DevelopmentKit\Agent\Java` directory

- compile the example code with

  ```
  javac -classpath "%OvAgentDir%/java/jopcagtbase.jar:%OvAgentDir%/java/jopcagtmsg.jar"
  <java source code file>
  ```

- run the example code with

  ```
  java -classpath ".:%OvAgentDir%/java/jopcagtbase.jar:%OvAgentDir%/java/jopcagtmsg.jar"
  <java class>
  ```

where <java source code file> could be JOpcAgtMsgTest.java or JOpcMonValueTest.java;
<java class> would then be JOpcAgtMsgTest or JOpcMonValueTest

On UNIX

To build the managed node sample program you have to copy the source files to the managed node. The HP Operations agent software must be installed on the managed node - otherwise the HPOM JAR files will not be present. Copy the sample programs to any location (for example, /tmp). To use the Java HPOM API wrapper classes:

- the -classpath parameter used for the javac and java commands must include the jopcagtbase.jar and jopcagtmsg.jar files
- the PATH system variable must include the directory where the shared library files reside. The agent installation does this automatically.

See "/opt/OV/www/htdocs/jdoc_agent/index.html" for a javadoc style class documentation.

To compile and run the example code:

- copy the source code to the managed node into a temporary directory and cd to the directory
- compile the example code with javac -classpath "/opt/jar/jopcagtbase.jar:/opt/jar/jopcagtmsg.jar" <java source code file>
- run the example code with java -classpath ".:/opt/jar/jopcagtbase.jar:/opt/jar/jopcagtmsg.jar" <java class>

where <java source code file> could be JOpcAgtMsgTest.java or JOpcMonValueTest.java;
<java class> would then be JOpcAgtMsgTest or JOpcMonValueTest

See Also

JOpcAgentMessage

JOpcMonValue

com.hp.openview.ib.api.jopc

Class JOpcAgentMessage

```
java.lang.Object
  |
  +--com.hp.openview.ib.api.jopc.JOpcObject
        |
        +--com.hp.openview.ib.api.jopc.JOpcMessage
              |
              +--com.hp.openview.ib.api.jopc.JOpcAgentMessage
```

All Implemented Interfaces:
java.lang.Cloneable, JOpcApiDefinition

---

public class JOpcAgentMessage

extends JOpcMessage

Provides a object oriented java class, corresponding to the HPOM API "Agent Message API". Most of the described behavior in this API documentation is very similar to the one provided by this class.

Native API description:
HPOM provides a set of APIs to handle messages on managed nodes. These functions enable you, for example, to send messages and acknowledge them at a later time.
The managed node processes must be running.

Constructor Summary

JOpcAgentMessage () Constructs a new JOpcAgentMessage.

Method Summary

void	acknowledge () Acknowledge a message out from a managed node.
void	send () Send a message, created on the managed node, to its responsible manager.

Methods inherited from class com.hp.openview.ib.api.jopc.JOpcMessage
`getApplication , getDataInfo , getDataType , getGroup , getMsgid , getMsgtext , getNodename , getObject , getServiceName , getSeverity , setApplication , setDataInfo , setGroup , setMsgid , setMsgtext , setNodename , setObject , setOptionVar , setServiceName , setSeverity`

Constructor Detail

JOpcAgentMessage

```
public JOpcAgentMessage()
                 throws JOpcException
```

Constructs a new JOpcAgentMessage.

Method Detail

send

```
public void send()
          throws JOpcException
```

Send a message, created on the managed node, to its responsible manager.

Native API description:
send a message, created on the managed node, to its responsible manager. The message must be of type OPCDTYPE_MESSAGE. The message ID can be retrieved from the message object using opcdata_get_str() immediately after the send call was executed. Only the message attributes Severity, Application, Message Group, Object, Message Text, Option Strings and Node are used in opcagtmsg_ send(). The API program must run as user opc_op or root (if not, customer program has to setuid). After opcagtmsg_send() was called it is possible to get the ID of the sent message using: opcdata_get_str()(message, OPCDATA_MSGID) If you want to save the information about the responsible manager, remark the message to be acknowledged later. To do this, set OPCDATA_ DATA_INFO to OPC_REMARK_FOR_ACK.

Throws:
`JOpcException` - if native method return != OPC_ERR_OK.

```
JOpcException reasons (negative values):
OPC_ERR_APPL_REQUIRED: attribute OPCDATA_APPLICATION not set
OPC_ERR_OBJ_REQUIRED: attribute OPCDATA_OBJECT not set
```

```
  OPC_ERR_TEXT_REQUIRED: attribute OPCDATA_MSGTEXT not set
  OPC_ERR_INVAL_SEVERITY: set severity invalid
  OPC_ERR_MISC_NOT_ALLOWED: message group  misc.  not allowed
  OPC_ERR_INVALID_INPARAM: message is NULL
                           message is not of type OPCDTYPE_MESSAGE
  OPC_ERR_NO_MEMORY: memory allocation failed
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

acknowledge

```
public void acknowledge()
                 throws JOpcException
```

Acknowledge a message out from a managed node.

Native API description:
Acknowledge a message out from a managed node. Therefore a message operation will be sent to the message agent and forwarded to the message interceptor. If the message attribute OPCDATA_DATA_INFO of a previously sent message was set to OPC_REMARK_FOR_ACK, the message agent holds the information about the responsible manager in its memory. If this attribute was not set, the message operation will be sent to all managers. The API program must run as user opc_op or root. If not, the customer program must set the user ID (setuid).

Throws:
    `JOpcException` - if native method return != OPC_ERR_OK.

```
  JOpcException reasons (negative values):
  OPC_ERR_INVALID_INPARAM: message_id is NULL
  OPC_ERR_INVALID_OPCDATA_TYPE: message_id is not of type OPCDTYPE_MESSAGE_ID
  OPC_ERR_INCOMPLETE_PARAM: message ID is not set
  OPC_ERR_NO_MEMORY: memory allocation failed
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

com.hp.openview.ib.api.jopc

Class JOpcException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--com.hp.openview.ib.api.jopc.JOpcException
```

All Implemented Interfaces:

java.io.Serializable

---

public class JOpcException

extends java.lang.Exception

Used for exception handling within the HPOM Java API classes. No public constructors - instances will be thrown by API only.

Method Summary	
java.lang.String	getMessage () get the corresponding message of the exception reason
JOpcException []	getNestedExceptions () Get nested exceptions.
int	getReason () get the reason of the exception
java.lang.String	getStack ()

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Method Detail

getMessage

```
public java.lang.String getMessage()
```

get the corresponding message of the exception reason

Overrides:

`getMessage` in class `java.lang.Throwable`

Returns:

the message string

---

getReason

```
public int getReason()
```

get the reason of the exception

Returns:

the reason

---

getStack

```
public java.lang.String getStack()
```

---

getNestedExceptions

```
public JOpcException[] getNestedExceptions()
```

Get nested exceptions. This may occur for HPOM API operations on arrays of objects. Then the C API may return OPC_ERR_NOT_COMPLETELY_DONE and the individual objects have to asked for their status. To simplify this, the Java API generates individual exceptions for the offending objects which can be queried with this method. If the nested exceptions are of class JOpcObjException they also point to the offending objects - see JOpcObjException.getObject(). If there are no nested exception the method returns null.

Returns:

the array of nested exceptions.

com.hp.openview.ib.api.jopc
Class JOpcMonValue

```
java.lang.Object
  |
  +--com.hp.openview.ib.api.jopc.JOpcObject
        |
        +--com.hp.openview.ib.api.jopc.JOpcMonValue
```

All Implemented Interfaces:
java.lang.Cloneable, JOpcApiDefinition

---

public class JOpcMonValue

extends JOpcObject

Provides a object oriented java class, corresponding to the HPOM API "Agent Monitor API". Most of the described behavior in this API documentation is very similar to the one provided by this class.

Native API description:

To use these functions, the managed node processes must be running.

Constructor Summary
JOpcMonValue () constructs a new JOpcMonValue

Method Summary

void	`send ()` Submits a monitor value to the local HPOM monitor agent.
void	`setMonValue (double aValue)` Sets the Monitor value.
void	`setMonVar (java.lang.String aValue)` Sets Name of the monitored object..
void	`setObject (java.lang.String aValue)` Sets the Message object.
void	`setOptionVar (java.lang.String aValue)` Sets A string containing the optional parameters used for resolving the $OPTION variables by the monitor agent.

Constructor Detail

JOpcMonValue

```
public JOpcMonValue()
             throws JOpcException
```

constructs a new JOpcMonValue

Method Detail

send

```
public void send()
          throws JOpcException
```

Submits a monitor value to the local HPOM monitor agent.

Native API description:
Use the function opcagtmon_send() to send a monitor value, created on the managed node, to its responsible manager. The message must be of type OPCDTYPE_MONITOR_MESSAGE. Only the message attributes Monitor Name, Monitor Value, Object and Option String are used in opcagtmon_send(). The API program must be run as user opc_op or root. If not, the customer program must set the user ID (setuid).

Throws:
`JOpcException` - if native method return != OPC_ERR_OK.

```
JOpcException reasons (negative values):
OPC_ERR_INVALID_INPARAM: mon_msg is NULL
                         mon_msg is not of type OPCDTYPE_MONITOR_MESSAGE
OPC_ERR_OBJNAME_REQUIRED: attribute OPCDATA_MON_VAR not set
OPC_ERR_NO_AGENT: agent is not running
OPC_ERR_NO_MEMORY: out of memory
OPC_ERR_WRONG_OPTION_VARS: attribute OPCDATA_OPTION_VAR not set correctly
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native api data structure.

---

setMonVar

```
public void setMonVar(java.lang.String aValue)
               throws JOpcException
```

Sets Name of the monitored object..

---

setMonValue

```
public void setMonValue(double aValue)
                 throws JOpcException
```

Sets the Monitor value.

---

setOptionVar

```
public void setOptionVar(java.lang.String aValue)
                  throws JOpcException
```

Sets A string containing the optional parameters used for resolving the $OPTION variables by the monitor agent. The string should have the format [<var>=<value>]* with <var> and <value> not containing spaces, equal signs ("="), open parens ("("), or closed parens (")").

---

setObject

```
public void setObject(java.lang.String aValue)
               throws JOpcException
```

Sets the Message object.

com.hp.openview.ib.api.jopc

Interface JOpcApiDefinition

All Known Implementing Classes:
JOpcObject

---

public interface JOpcApiDefinition

Field Summary	
static int	OPC_ERR_ACCESS_DENIED
static int	OPC_ERR_ACTION_FAILED
static int	OPC_ERR_ACTION_RUNNING
static int	OPC_ERR_ALREADY_DONE
static int	OPC_ERR_APPL_NOT_FOUND
static int	OPC_ERR_APPL_REQUIRED
static int	OPC_ERR_APPLGROUP_NOT_FOUND
static int	OPC_ERR_CANT_ADD_TEMPLATE
static int	OPC_ERR_CANT_CONNECT_DB
static int	OPC_ERR_CANT_CONNECT_DM
static int	OPC_ERR_CANT_DISCONNECT
static int	OPC_ERR_CANT_GET_LOCAL_ADDR

static int	OPC_ERR_CANT_GET_MGMTSV_ADDRESS
static int	OPC_ERR_CANT_INFORM_MSGA
static int	OPC_ERR_CANT_INFORM_MSGM
static int	OPC_ERR_CANT_INFORM_UI
static int	OPC_ERR_CANT_INIT
static int	OPC_ERR_CANT_INIT_MUTEX
static int	OPC_ERR_CANT_LOCK_MUTEX
static int	OPC_ERR_CANT_OPEN_FILE
static int	OPC_ERR_CANT_OPEN_PIPE
static int	OPC_ERR_CANT_OPEN_QUEUE
static int	OPC_ERR_CANT_OPEN_READQUEUE
static int	OPC_ERR_CANT_OPEN_WRITEQUEUE
static int	OPC_ERR_CANT_READ_MSG
static int	OPC_ERR_CANT_READ_QUEUE
static int	OPC_ERR_CANT_WRITE_FILE
static int	OPC_ERR_CANT_WRITE_MSG
static int	OPC_ERR_CANT_WRITE_QUEUE
static int	OPC_ERR_DATABASE_ERROR
static int	OPC_ERR_DB_INCONSISTENT

static int	OPC_ERR_DB_WARNING
static int	OPC_ERR_DEADLOCK
static int	OPC_ERR_DUMMY_FUNCTION_CALLED
static int	OPC_ERR_EINTR
static int	OPC_ERR_ESCALATION_FAILED
static int	OPC_ERR_FUNC_NOT_IMPL_YET
static int	OPC_ERR_GROUP_NAME_EXISTS
static int	OPC_ERR_INCOMPLETE_PARAM
static int	OPC_ERR_INVAL_NODE
static int	OPC_ERR_INVAL_SEVERITY
static int	OPC_ERR_INVALID_ANNOTATION
static int	OPC_ERR_INVALID_APPLICATION
static int	OPC_ERR_INVALID_CHARSET
static int	OPC_ERR_INVALID_COMMAND
static int	OPC_ERR_INVALID_DESCRIPTION_LENGTH
static int	OPC_ERR_INVALID_EXEC_USER
static int	OPC_ERR_INVALID_FIELD
static int	OPC_ERR_INVALID_FILE
static int	OPC_ERR_INVALID_ID

static int	OPC_ERR_INVALID_INPARAM
static int	OPC_ERR_INVALID_INTERFACE_ID
static int	OPC_ERR_INVALID_INTERFACE_INSTANCE
static int	OPC_ERR_INVALID_INTERFACE_TYPE
static int	OPC_ERR_INVALID_INTERVAL
static int	OPC_ERR_INVALID_MESSAGE_GROUP
static int	OPC_ERR_INVALID_MINMAX
static int	OPC_ERR_INVALID_MODE
static int	OPC_ERR_INVALID_MSG_GENERATION
static int	OPC_ERR_INVALID_NAME
static int	OPC_ERR_INVALID_NAME_LENGTH
static int	OPC_ERR_INVALID_NODE_GROUP
static int	OPC_ERR_INVALID_OPCDATA_TYPE
static int	OPC_ERR_INVALID_OUTPARAM
static int	OPC_ERR_INVALID_PARAM
static int	OPC_ERR_INVALID_PATH
static int	OPC_ERR_INVALID_PROG_OR_MIB
static int	OPC_ERR_INVALID_TEMPLATE_TYPE
static int	OPC_ERR_INVALID_USER

static int	OPC_ERR_LAST_REFERENCE
static int	OPC_ERR_LAYOUTGROUP_IS_HOLDING_AREA
static int	OPC_ERR_LAYOUTGROUP_NOT_EMPTY
static int	OPC_ERR_LAYOUTGROUP_NOT_FOUND
static int	OPC_ERR_LOCKED_BY_OTHER
static int	OPC_ERR_MISC_NOT_ALLOWED
static int	OPC_ERR_MLM_NAME_REQUIRED
static int	OPC_ERR_MSG_IS_READONLY
static int	OPC_ERR_MSG_NOT_ACTIVE
static int	OPC_ERR_MSG_OWNED_BY_ANOTHER_USER
static int	OPC_ERR_MSI_BUF_FULL
static int	OPC_ERR_NAME_EXISTS
static int	OPC_ERR_NO_ACTION_DEFINED
static int	OPC_ERR_NO_AGENT
static int	OPC_ERR_NO_ANNOTATIONS
static int	OPC_ERR_NO_DATA
static int	OPC_ERR_NO_ELEMENT
static int	OPC_ERR_NO_ESCALATION_DEFINED
static int	OPC_ERR_NO_LOGIN

static int	OPC_ERR_NO_MEMORY
static int	OPC_ERR_NO_OPERATOR_DEF
static int	OPC_ERR_NO_PIPE_NAME
static int	OPC_ERR_NO_QUEUE_NAME
static int	OPC_ERR_NO_TRANSACTION
static int	OPC_ERR_NODE_NOT_FOUND
static int	OPC_ERR_NODEHIER_NOT_FOUND
static int	OPC_ERR_NOT_ACKNOWLEDGED
static int	OPC_ERR_NOT_COMPLETELY_DONE
static int	OPC_ERR_OBJ_REQUIRED
static int	OPC_ERR_OBJECT_ALREADY_ASSIGNED
static int	OPC_ERR_OBJECT_ALREADY_EXISTS
static int	OPC_ERR_OBJECT_NOT_ASSIGNED
static int	OPC_ERR_OBJECT_NOT_FOUND
static int	OPC_ERR_OBJECT_NOT_UNIQUE
static int	OPC_ERR_OBJNAME_REQUIRED
static int	OPC_ERR_OK
static int	OPC_ERR_OUT_OF_RANGE
static int	OPC_ERR_PROFILE_NOT_FOUND

static int	OPC_ERR_STRING_TOO_LONG
static int	OPC_ERR_SV_NOT_RUNNING
static int	OPC_ERR_SYNTAX_ERROR
static int	OPC_ERR_TEXT_REQUIRED
static int	OPC_ERR_TRANSACTION_ALREADY_OPEN
static int	OPC_ERR_USER_NOT_FOUND
static int	OPC_ERR_WRONG_MSITYPE
static int	OPC_ERR_WRONG_OPTION_VARS
static int	OPC_SEV_CRITICAL
static int	OPC_SEV_MAJOR
static int	OPC_SEV_MINOR
static int	OPC_SEV_NORMAL
static int	OPC_SEV_UNCHANGED
static int	OPC_SEV_UNKNOWN
static int	OPC_SEV_WARNING
static int	OPCDATA_ID
static int	OPCDATA_MESSAGE_ID

Field Detail

OPC_ERR_OK

```
public static final int OPC_ERR_OK
```

OPC_ERR_APPL_REQUIRED

```
public static final int OPC_ERR_APPL_REQUIRED
```

OPC_ERR_OBJ_REQUIRED

```
public static final int OPC_ERR_OBJ_REQUIRED
```

OPC_ERR_TEXT_REQUIRED

```
public static final int OPC_ERR_TEXT_REQUIRED
```

OPC_ERR_INVAL_SEVERITY

```
public static final int OPC_ERR_INVAL_SEVERITY
```

OPC_ERR_OBJNAME_REQUIRED

```
public static final int OPC_ERR_OBJNAME_REQUIRED
```

OPC_ERR_MISC_NOT_ALLOWED

```
public static final int OPC_ERR_MISC_NOT_ALLOWED
```

OPC_ERR_INVAL_NODE

```
public static final int OPC_ERR_INVAL_NODE
```

OPC_ERR_NO_MEMORY

```
public static final int OPC_ERR_NO_MEMORY
```

OPC_ERR_NO_AGENT

```
public static final int OPC_ERR_NO_AGENT
```

OPC_ERR_CANT_INIT

```
public static final int OPC_ERR_CANT_INIT
```

OPC_ERR_NO_QUEUE_NAME

```
public static final int OPC_ERR_NO_QUEUE_NAME
```

OPC_ERR_CANT_OPEN_QUEUE

```
public static final int OPC_ERR_CANT_OPEN_QUEUE
```

OPC_ERR_CANT_WRITE_QUEUE

```
public static final int OPC_ERR_CANT_WRITE_QUEUE
```

OPC_ERR_NO_PIPE_NAME

```
public static final int OPC_ERR_NO_PIPE_NAME
```

OPC_ERR_CANT_OPEN_PIPE

```
public static final int OPC_ERR_CANT_OPEN_PIPE
```

OPC_ERR_CANT_GET_LOCAL_ADDR

```
public static final int OPC_ERR_CANT_GET_LOCAL_ADDR
```

OPC_ERR_CANT_INIT_MUTEX

```
public static final int OPC_ERR_CANT_INIT_MUTEX
```

OPC_ERR_CANT_LOCK_MUTEX

```
public static final int OPC_ERR_CANT_LOCK_MUTEX
```

OPC_ERR_CANT_READ_QUEUE

```
public static final int OPC_ERR_CANT_READ_QUEUE
```

OPC_ERR_INVALID_INPARAM

```
public static final int OPC_ERR_INVALID_INPARAM
```

OPC_ERR_INVALID_OUTPARAM

```
public static final int OPC_ERR_INVALID_OUTPARAM
```

OPC_ERR_INVALID_FIELD

```
public static final int OPC_ERR_INVALID_FIELD
```

OPC_ERR_NO_ELEMENT

```
public static final int OPC_ERR_NO_ELEMENT
```

OPC_ERR_INCOMPLETE_PARAM

```
public static final int OPC_ERR_INCOMPLETE_PARAM
```

OPC_ERR_INVALID_INTERFACE_TYPE

```
public static final int OPC_ERR_INVALID_INTERFACE_TYPE
```

OPC_ERR_INVALID_INTERFACE_INSTANCE

```
public static final int OPC_ERR_INVALID_INTERFACE_INSTANCE
```

OPC_ERR_INVALID_INTERFACE_ID

```
public static final int OPC_ERR_INVALID_INTERFACE_ID
```

OPC_ERR_INVALID_OPCDATA_TYPE

```
public static final int OPC_ERR_INVALID_OPCDATA_TYPE
```

OPC_ERR_CANT_OPEN_READQUEUE

```
public static final int OPC_ERR_CANT_OPEN_READQUEUE
```

OPC_ERR_CANT_OPEN_WRITEQUEUE

```
public static final int OPC_ERR_CANT_OPEN_WRITEQUEUE
```

OPC_ERR_CANT_INFORM_MSGM

```
public static final int OPC_ERR_CANT_INFORM_MSGM
```

OPC_ERR_CANT_INFORM_MSGA

```
public static final int OPC_ERR_CANT_INFORM_MSGA
```

OPC_ERR_CANT_READ_MSG

```
public static final int OPC_ERR_CANT_READ_MSG
```

OPC_ERR_CANT_WRITE_MSG

```
public static final int OPC_ERR_CANT_WRITE_MSG
```

OPC_ERR_WRONG_MSITYPE

```
public static final int OPC_ERR_WRONG_MSITYPE
```

OPC_ERR_NO_DATA

```
public static final int OPC_ERR_NO_DATA
```

OPC_ERR_EINTR

```
public static final int OPC_ERR_EINTR
```

OPC_ERR_MSI_BUF_FULL

```
public static final int OPC_ERR_MSI_BUF_FULL
```

OPC_ERR_SV_NOT_RUNNING

```
public static final int OPC_ERR_SV_NOT_RUNNING
```

OPC_ERR_ACCESS_DENIED

```
public static final int OPC_ERR_ACCESS_DENIED
```

OPC_ERR_NO_LOGIN

```
public static final int OPC_ERR_NO_LOGIN
```

OPC_ERR_CANT_CONNECT_DB

```
public static final int OPC_ERR_CANT_CONNECT_DB
```

OPC_ERR_NO_OPERATOR_DEF

```
public static final int OPC_ERR_NO_OPERATOR_DEF
```

OPC_ERR_CANT_DISCONNECT

```
public static final int OPC_ERR_CANT_DISCONNECT
```

OPC_ERR_INVALID_ID

```
public static final int OPC_ERR_INVALID_ID
```

OPC_ERR_NO_ANNOTATIONS

`public static final int OPC_ERR_NO_ANNOTATIONS`

OPC_ERR_ALREADY_DONE

`public static final int OPC_ERR_ALREADY_DONE`

OPC_ERR_NOT_ACKNOWLEDGED

`public static final int OPC_ERR_NOT_ACKNOWLEDGED`

OPC_ERR_DATABASE_ERROR

`public static final int OPC_ERR_DATABASE_ERROR`

OPC_ERR_CANT_INFORM_UI

`public static final int OPC_ERR_CANT_INFORM_UI`

OPC_ERR_CANT_CONNECT_DM

`public static final int OPC_ERR_CANT_CONNECT_DM`

OPC_ERR_MSG_OWNED_BY_ANOTHER_USER

`public static final int OPC_ERR_MSG_OWNED_BY_ANOTHER_USER`

OPC_ERR_NO_ESCALATION_DEFINED

```
public static final int OPC_ERR_NO_ESCALATION_DEFINED
```

OPC_ERR_ESCALATION_FAILED

```
public static final int OPC_ERR_ESCALATION_FAILED
```

OPC_ERR_ACTION_RUNNING

```
public static final int OPC_ERR_ACTION_RUNNING
```

OPC_ERR_NO_ACTION_DEFINED

```
public static final int OPC_ERR_NO_ACTION_DEFINED
```

OPC_ERR_ACTION_FAILED

```
public static final int OPC_ERR_ACTION_FAILED
```

OPC_ERR_OBJECT_NOT_FOUND

```
public static final int OPC_ERR_OBJECT_NOT_FOUND
```

OPC_ERR_OBJECT_NOT_UNIQUE

```
public static final int OPC_ERR_OBJECT_NOT_UNIQUE
```

OPC_ERR_OBJECT_ALREADY_EXISTS

```
public static final int OPC_ERR_OBJECT_ALREADY_EXISTS
```

OPC_ERR_OUT_OF_RANGE

```
public static final int OPC_ERR_OUT_OF_RANGE
```

OPC_ERR_DB_WARNING

```
public static final int OPC_ERR_DB_WARNING
```

OPC_ERR_DB_INCONSISTENT

```
public static final int OPC_ERR_DB_INCONSISTENT
```

OPC_ERR_INVALID_NODE_GROUP

```
public static final int OPC_ERR_INVALID_NODE_GROUP
```

OPC_ERR_INVALID_MESSAGE_GROUP

```
public static final int OPC_ERR_INVALID_MESSAGE_GROUP
```

OPC_ERR_NOT_COMPLETELY_DONE

```
public static final int OPC_ERR_NOT_COMPLETELY_DONE
```

OPC_ERR_CANT_OPEN_FILE

```
public static final int OPC_ERR_CANT_OPEN_FILE
```

OPC_ERR_CANT_ADD_TEMPLATE

```
public static final int OPC_ERR_CANT_ADD_TEMPLATE
```

OPC_ERR_WRONG_OPTION_VARS

```
public static final int OPC_ERR_WRONG_OPTION_VARS
```

OPC_ERR_SYNTAX_ERROR

```
public static final int OPC_ERR_SYNTAX_ERROR
```

OPC_ERR_CANT_WRITE_FILE

```
public static final int OPC_ERR_CANT_WRITE_FILE
```

OPC_ERR_INVALID_USER

```
public static final int OPC_ERR_INVALID_USER
```

OPC_ERR_TRANSACTION_ALREADY_OPEN

```
public static final int OPC_ERR_TRANSACTION_ALREADY_OPEN
```

OPC_ERR_NO_TRANSACTION

```
public static final int OPC_ERR_NO_TRANSACTION
```

OPC_ERR_STRING_TOO_LONG

```
public static final int OPC_ERR_STRING_TOO_LONG
```

OPC_ERR_NAME_EXISTS

```
public static final int OPC_ERR_NAME_EXISTS
```

OPC_ERR_GROUP_NAME_EXISTS

```
public static final int OPC_ERR_GROUP_NAME_EXISTS
```

OPC_ERR_MSG_NOT_ACTIVE

```
public static final int OPC_ERR_MSG_NOT_ACTIVE
```

OPC_ERR_MSG_IS_READONLY

```
public static final int OPC_ERR_MSG_IS_READONLY
```

OPC_ERR_INVALID_NAME

```
public static final int OPC_ERR_INVALID_NAME
```

OPC_ERR_INVALID_ANNOTATION

```
public static final int OPC_ERR_INVALID_ANNOTATION
```

OPC_ERR_INVALID_APPLICATION

```
public static final int OPC_ERR_INVALID_APPLICATION
```

OPC_ERR_INVALID_PARAM

```
public static final int OPC_ERR_INVALID_PARAM
```

OPC_ERR_LOCKED_BY_OTHER

```
public static final int OPC_ERR_LOCKED_BY_OTHER
```

OPC_ERR_DEADLOCK

```
public static final int OPC_ERR_DEADLOCK
```

OPC_ERR_OBJECT_ALREADY_ASSIGNED

```
public static final int OPC_ERR_OBJECT_ALREADY_ASSIGNED
```

OPC_ERR_OBJECT_NOT_ASSIGNED

```
public static final int OPC_ERR_OBJECT_NOT_ASSIGNED
```

OPC_ERR_NODEHIER_NOT_FOUND

```
public static final int OPC_ERR_NODEHIER_NOT_FOUND
```

OPC_ERR_LAYOUTGROUP_NOT_FOUND

```
public static final int OPC_ERR_LAYOUTGROUP_NOT_FOUND
```

OPC_ERR_NODE_NOT_FOUND

```
public static final int OPC_ERR_NODE_NOT_FOUND
```

OPC_ERR_USER_NOT_FOUND

```
public static final int OPC_ERR_USER_NOT_FOUND
```

OPC_ERR_PROFILE_NOT_FOUND

```
public static final int OPC_ERR_PROFILE_NOT_FOUND
```

OPC_ERR_APPL_NOT_FOUND

```
public static final int OPC_ERR_APPL_NOT_FOUND
```

OPC_ERR_APPLGROUP_NOT_FOUND

```
public static final int OPC_ERR_APPLGROUP_NOT_FOUND
```

OPC_ERR_LAYOUTGROUP_NOT_EMPTY

```
public static final int OPC_ERR_LAYOUTGROUP_NOT_EMPTY
```

OPC_ERR_LAYOUTGROUP_IS_HOLDING_AREA

```
public static final int OPC_ERR_LAYOUTGROUP_IS_HOLDING_AREA
```

OPC_ERR_LAST_REFERENCE

```
public static final int OPC_ERR_LAST_REFERENCE
```

OPC_ERR_CANT_GET_MGMTSV_ADDRESS

```
public static final int OPC_ERR_CANT_GET_MGMTSV_ADDRESS
```

OPC_ERR_MLM_NAME_REQUIRED

```
public static final int OPC_ERR_MLM_NAME_REQUIRED
```

OPC_ERR_INVALID_NAME_LENGTH

```
public static final int OPC_ERR_INVALID_NAME_LENGTH
```

OPC_ERR_INVALID_DESCRIPTION_LENGTH

```
public static final int OPC_ERR_INVALID_DESCRIPTION_LENGTH
```

OPC_ERR_INVALID_COMMAND

```
public static final int OPC_ERR_INVALID_COMMAND
```

OPC_ERR_INVALID_INTERVAL

```
public static final int OPC_ERR_INVALID_INTERVAL
```

OPC_ERR_INVALID_FILE

```
public static final int OPC_ERR_INVALID_FILE
```

OPC_ERR_INVALID_PATH

```
public static final int OPC_ERR_INVALID_PATH
```

OPC_ERR_INVALID_MODE

```
public static final int OPC_ERR_INVALID_MODE
```

OPC_ERR_INVALID_CHARSET

```
public static final int OPC_ERR_INVALID_CHARSET
```

OPC_ERR_INVALID_EXEC_USER

```
public static final int OPC_ERR_INVALID_EXEC_USER
```

OPC_ERR_INVALID_PROG_OR_MIB

```
public static final int OPC_ERR_INVALID_PROG_OR_MIB
```

OPC_ERR_INVALID_MINMAX

```
public static final int OPC_ERR_INVALID_MINMAX
```

OPC_ERR_INVALID_MSG_GENERATION

```
public static final int OPC_ERR_INVALID_MSG_GENERATION
```

OPC_ERR_INVALID_TEMPLATE_TYPE

```
public static final int OPC_ERR_INVALID_TEMPLATE_TYPE
```

OPC_ERR_FUNC_NOT_IMPL_YET

```
public static final int OPC_ERR_FUNC_NOT_IMPL_YET
```

OPC_ERR_DUMMY_FUNCTION_CALLED

```
public static final int OPC_ERR_DUMMY_FUNCTION_CALLED
```

OPC_SEV_UNCHANGED

```
public static final int OPC_SEV_UNCHANGED
```

OPC_SEV_UNKNOWN

```
public static final int OPC_SEV_UNKNOWN
```

OPC_SEV_NORMAL

```
public static final int OPC_SEV_NORMAL
```

OPC_SEV_WARNING

```
public static final int OPC_SEV_WARNING
```

OPC_SEV_CRITICAL

```
public static final int OPC_SEV_CRITICAL
```

OPC_SEV_MINOR

```
public static final int OPC_SEV_MINOR
```

OPC_SEV_MAJOR

```
public static final int OPC_SEV_MAJOR
```

OPCDATA_ID

```
public static final int OPCDATA_ID
```

OPCDATA_MESSAGE_ID

```
public static final int OPCDATA_MESSAGE_ID
```

com.hp.openview.ib.api.jopc

Class JOpcMessage

```
java.lang.Object
  |
  +--com.hp.openview.ib.api.jopc.JOpcObject
        |
        +--com.hp.openview.ib.api.jopc.JOpcMessage
```

All Implemented Interfaces:

java.lang.Cloneable, JOpcApiDefinition

Direct Known Subclasses:

JOpcAgentMessage

---

public abstract class JOpcMessage

extends JOpcObject

Provides a object oriented java class with getter() and setter(), corresponding to the HPOM Data Structure "OPCDTYPE_MESSAGE". This data structure is needed in context of HPOM APIs "Agent Message API" .

Constructor Summary
JOpcMessage ()   constructs a new JOpcMessage

Method Summary	
java.lang.String	getApplication () Will load application which produced the message.
long	getDataInfo () Will load additional information about the message.
long	getDataType () Will load the type of the opcdata object.
java.lang.String	getGroup () Will load message group.

java.lang.String	getMsgid () Will load The unique ID of the message.
java.lang.String	getMsgtext () Will load message text.
java.lang.String	getNodename () Will load Name of the node producing the message.
java.lang.String	getObject () Will load Object name to use for the HPOM message.
java.lang.String	getServiceName () Will load the service name.
long	getSeverity () Will load severity of the message.
void	setApplication (java.lang.String aValue) Sets application which produced the message.
void	setDataInfo (long aValue) Sets additional information about the message.
void	setGroup (java.lang.String aValue) Sets message group.
void	setMsgid (java.lang.String aValue) Set the unique ID of the message.
void	setMsgtext (java.lang.String aValue) Sets message text.
void	setNodename (java.lang.String aValue) Sets Name of the node producing the message.
void	setObject (java.lang.String aValue) Sets Object name to use for the HPOM message.
void	setOptionVar (java.lang.String aValue) Sets A string containing the optional parameters used for resolving the $OPTION variables by the message interceptor.
void	setServiceName (java.lang.String aValue) Sets the service name.
void	setSeverity (long aValue) Sets severity of the message.

Constructor Detail

JOpcMessage

```
public JOpcMessage()
            throws JOpcException
```

constructs a new JOpcMessage

Method Detail

getDataType

```
public long getDataType()
                 throws JOpcException
```

Will load the type of the opcdata object.

---

getSeverity

```
public long getSeverity()
                 throws JOpcException
```

Will load severity of the message.

```
Possible values are:
OPC_SEV_UNCHANGED
OPC_SEV_UNKNOWN
OPC_SEV_NORMAL
OPC_SEV_WARNING
OPC_SEV_CRITICAL
OPC_SEV_MINOR
OPC_SEV_MAJOR
```

---

setSeverity

```
public void setSeverity(long aValue)
                 throws JOpcException
```

Sets severity of the message.

```
Possible values are:
OPC_SEV_UNCHANGED
OPC_SEV_UNKNOWN
OPC_SEV_NORMAL
OPC_SEV_WARNING
OPC_SEV_CRITICAL
```

```
OPC_SEV_MINOR
OPC_SEV_MAJOR
```

---

getDataInfo

```
public long getDataInfo()
                 throws JOpcException
```

Will load additional information about the message.

```
OPC_REMARK_FOR_ACK
```

---

setDataInfo

```
public void setDataInfo(long aValue)
                 throws JOpcException
```

Sets additional information about the message.

```
OPC_REMARK_FOR_ACK
```

---

getApplication

```
public java.lang.String getApplication()
                                throws JOpcException
```

Will load application which produced the message. Default: empty string

---

setApplication

```
public void setApplication(java.lang.String aValue)
                    throws JOpcException
```

Sets application which produced the message. Default: empty string

---

getGroup

```
public java.lang.String getGroup()
                          throws JOpcException
```

Will load message group. Default: empty string.

---

setGroup

```
public void setGroup(java.lang.String aValue)
              throws JOpcException
```

Sets message group. Default: empty string.

---

getMsgtext

```
public java.lang.String getMsgtext()
                            throws JOpcException
```

Will load message text. Default: empty string.

---

setMsgtext

```
public void setMsgtext(java.lang.String aValue)
                throws JOpcException
```

Sets message text. Default: empty string.

---

getNodename

```
public java.lang.String getNodename()
                             throws JOpcException
```

Will load Name of the node producing the message. The message is only handled by the HPOM manager if this system is part of the HPOM Node Bank.

Default: local node name.

---

setNodename

```
public void setNodename(java.lang.String aValue)
                 throws JOpcException
```

Sets Name of the node producing the message. The message is only handled by the HPOM manager if this system is part of the HPOM Node Bank.

Default: local node name.

---

getObject

```
public java.lang.String getObject()
                           throws JOpcException
```

Will load Object name to use for the HPOM message. Default: empty string.

---

setObject

```
public void setObject(java.lang.String aValue)
               throws JOpcException
```

Sets Object name to use for the HPOM message. Default: empty string.

---

getMsgid

```
public java.lang.String getMsgid()
                          throws JOpcException
```

Will load The unique ID of the message.

---

setMsgid

```
public void setMsgid(java.lang.String aValue)
              throws JOpcException
```

Set the unique ID of the message. Applies to operations dealing with just a reference to a message (for example, acknowledge(), own(), and so on). Cannot be used to set a message ID before sending it on the managed node.

---

setOptionVar

```
public void setOptionVar(java.lang.String aValue)
                  throws JOpcException
```

Sets A string containing the optional parameters used for resolving the $OPTION variables by the message interceptor. The string should have the format [<var>=<value>]* with <var> and <value> not containing spaces or the '=' character.

---

getServiceName

```
public java.lang.String getServiceName()
                                throws JOpcException
```

Will load the service name.

---

setServiceName

```
public void setServiceName(java.lang.String aValue)
                    throws JOpcException
```

Sets the service name.

com.hp.openview.ib.api.jopc

Class JOpcObject

```
java.lang.Object
  |
  +--com.hp.openview.ib.api.jopc.JOpcObject
```

All Implemented Interfaces:

java.lang.Cloneable, JOpcApiDefinition

Direct Known Subclasses:

JOpcMessage , JOpcMonValue

---

public abstract class JOpcObject

extends java.lang.Object

implements java.lang.Cloneable, JOpcApiDefinition

base class of opc objects. Supports generic behavior and attribute getting and setting.

Constructor Summary

protected	JOpcObject (int aOpcDataType)

Method Summary

protected long	addListElement (int aAttrType) Native API description: Adds an element of the correct type to the specified list in the opcdata structure.
java.lang.Object	clone () Creates and returns a copy of this object.
protected long	deleteListElement (int aAttrType, long aIndex) Native API description: Deletes an element of the specified list in the opcdata structure.

boolean	equals (java.lang.Object aObj) Indicates whether some other object is "equal to" this one.
protected void	finalize () as fallback, to avoid memory leak in native adapter/API layer in case of non released JOpcObjects
protected long	getListLength (int aAttrType) Returns the number of elements in an embedded list of a HPOM opcdata structure.
protected long	getLong (int aAttrType) Returns the value of the numeric attribute
protected java.lang.String	getString (int aAttrType) Returns the desired string value.
protected java.lang.String	getString (int aAttrType, int aElmType, long aIndex)
protected void	setDouble (int aAttrType, double aAttrValue) sets the numeric float attribute
protected void	setLong (int aAttrType, long aAttrValue) Sets the numeric long attribute.
protected void	setString (int aAttrType, java.lang.String aName) Sets the desired string value.
protected void	setString (int aAttrType, java.lang.String aValue, int aElmType, long aIndex)
void	terminate () release resources in native adapter/API layer allocated while object construction (JOpcObject and derived classes).
java.lang.String	toString () Returns a string representation of all get-methods including method()-name and value.
protected void	type () Native API description: Returns the opcdata type in data.
static java.lang.String	version () Returns the what string of the HPOM library that is used in this version.

Constructor Detail

JOpcObject

```
protected JOpcObject(int aOpcDataType)
           throws JOpcException
```

Method Detail

version

```
public static java.lang.String version()
                                throws JOpcException
```

Returns the what string of the HPOM library that is used in this version.

Returns:
the what string / version

Throws:
`JOpcException` - if native method return != OPC_ERR_OK. or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

getString

```
protected java.lang.String getString(int aAttrType)
                              throws JOpcException
```

Returns the desired string value.

Native API description:
Instead of a status value, the function opcdata_get_str() returns a pointer to the desired string value. This function can, therefore, be used directly in another function call.

Parameters:
`aAttrType` - attribute that is queried

Returns:
the desired string

Throws:
`JOpcException` - if native method return != OPC_ERR_OK. or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

setString

```
protected void setString(int aAttrType,
                         java.lang.String aName)
                  throws JOpcException
```

Sets the desired string value.

Parameters:
- aAttrType - attribute to be set
- aName - value to be set

Throws:
- JOpcException - if native method return != OPC_ERR_OK. or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

type

```
protected void type()
             throws JOpcException
```

Native API description:

Returns the opcdata type in data.

Throws:
- JOpcException - if native method return != OPC_ERR_OK.

```
  JOpcException reasons (negative values):
  OPC_ERR_INVALID_INPARAM: data is invalid; probably NULL
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

getLong

```
protected long getLong(int aAttrType)
                throws JOpcException
```

Returns the value of the numeric attribute

Parameters:
- aAttrType - attribute that is queried

Returns:
- the long value

Throws:

JOpcException - if native method return == -1.

```
  JOpcException reasons (negative values):
  if the routine fails, a value of -1 is returned.
```

---

setLong

```
protected void setLong(int aAttrType,
                       long aAttrValue)
                throws JOpcException
```

Sets the numeric long attribute.

Parameters:

aAttrType - attribute to be set

aAttrValue - value to be set

Throws:

JOpcException - if native method return != OPC_ERR_OK.

```
  JOpcException reasons (negative values):
  OPC_ERR_INVALID_INPARAM: data is NULL, attribute is invalid
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

setDouble

```
protected void setDouble(int aAttrType,
                         double aAttrValue)
                  throws JOpcException
```

sets the numeric float attribute

Parameters:

aAttrType - attribute to be set

aAttrValue - value to be set

Throws:

JOpcException - if native method return != OPC_ERR_OK.

```
  JOpcException reasons (negative values):
```

```
  OPC_ERR_INVALID_INPARAM: data is NULL,
                           attribute is NULL,
                           value is NULL
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

getString

```
protected java.lang.String getString(int aAttrType,
                                     int aElmType,
                                     long aIndex)
                              throws JOpcException
```

---

setString

```
protected void setString(int aAttrType,
                         java.lang.String aValue,
                         int aElmType,
                         long aIndex)
                  throws JOpcException
```

---

getListLength

```
protected long getListLength(int aAttrType)
                      throws JOpcException
```

Returns the number of elements in an embedded list of a HPOM opcdata structure.

Parameters:
: `aAttrType` - Specifies the list in the data structure.

Returns:
: number of elements in the list

Throws:
: `JOpcException` - if native method return < OPC_ERR_OK.

```
  JOpcException reasons (negative values):
  OPC_ERR_INVALID_INPARAM: parameter data is invalid;
                           probably NULL list is invalid
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

addListElement

```
protected long addListElement(int aAttrType)
                       throws JOpcException
```

Native API description:

Adds an element of the correct type to the specified list in the opcdata structure. The element type is specified with the list. After adding the element, the new length of the list will be returned.

Parameters:

`aAttrType` - Specifies the list in the data structure.

Returns:

new length of the list

Throws:

`JOpcException` - if native method return < OPC_ERR_OK.

```
JOpcException reasons (negative values):
OPC_ERR_INVALID_INPARAM: Input parameter was not valid ( < 0 )
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

deleteListElement

```
protected long deleteListElement(int aAttrType,
                                 long aIndex)
                          throws JOpcException
```

Native API description:

Deletes an element of the specified list in the opcdata structure. This function returns the new number of elements in the list.

Parameters:

`aAttrType` - Specifies the list in the data structure.

`aIndex` - Specifies the element in the list.

Returns:

the new number of elements in the list.

Throws:

`JOpcException` - if native method return < OPC_ERR_OK.

```
  JOpcException reasons (negative values):
  OPC_ERR_INVALID_INPARAM: parameter data is invalid;
                           probably NULL
                           list or index is invalid
```

or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

terminate

`public void terminate()`

release resources in native adapter/API layer allocated while object construction (JOpcObject and derived classes).

---

finalize

`protected void finalize()`

as fallback, to avoid memory leak in native adapter/API layer in case of non released JOpcObjects

Overrides:

`finalize` in class `java.lang.Object`

---

equals

`public boolean equals(java.lang.Object aObj)`

Indicates whether some other object is "equal to" this one.

Overrides:

`equals` in class `java.lang.Object`

Parameters:

`aObj` - the reference object with which to compare.

Returns:

true if this object is the same as the obj argument; false otherwise.

---

clone

`public java.lang.Object clone()`

`throws java.lang.CloneNotSupportedException`

Creates and returns a copy of this object. Native API description:

The API creates a copy of the data area and returns it in copy. It creates a complete copy, that is, the string fields of data are copied and not shared between data and copy. The allocated memory has to be deallocated using opcdata_free() before using this function. This function cannot be used to copy a data area of type OPCDTYPE_CONTAINER. If it is necessary to copy a whole container, the application must do this using iterator and container functions.

Overrides:

`clone` in class `java.lang.Object`

Returns:

a clone of this instance.

Throws:

`java.lang.CloneNotSupportedException` - if the object's class does not support the Cloneable interface. or if a field contains an improper value, the exception reason is a positive value corresponding to the JOpcApiDefinition interface identifying the attribute name of the native API data structure.

---

toString

`public java.lang.String toString()`

Returns a string representation of all get-methods including method()-name and value. The format of the returned string is <method-name> = <value>; separated by a new line (\n).

```
Further / detailed requirements:
- method name has to start with get
- method must not have a parameter
- method must have return type String or Long
- method must not be static
```

Overrides:

`toString` in class `java.lang.Object`

Returns:

a string of all get-methods

Command-Line Utilities

HPOM provides a number of command-line utilities. Some allow you to perform the same functions available in the user interface by using the command line. Others offer additional functionality.

Utilities include both agent and server utilities.

Agent Command-Line Utilities

These utilities can be executed on the HP Operations agent.

bbc.ini

NAME

bbc.ini

- configuration file for HTTPS communication.

DESCRIPTION

`bbc.ini` is the configuration file of a node using HTTPS communication.

It is located in the following directory:

`/<` *`OvDataDir`* `>/conf/confpar`

The file 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`

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, thereby restricting access to a limited part of the file system. The default is < *`OvDataDir`* > . This parameter is ignored on Microsoft Windows and Sun Solaris 7 systems. For details about `chroot` , see the `chroot` man page.

`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 allows only 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 port is 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. The default is `INADDR_ANY` .

`bbc.cb.ports`

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 BBC CBs is 383.

You can use the following parameters:

`string PORTS`

This configuration parameter must be the same on all nodes. To change the port number of a BBC CB on a particular host, you must add the hostname to this parameter (for example, `name.hp.com:8000`). You can use an asterisk (`*`) as a wildcard to denote an entire network (for example; `*.hp.com:8001`). You should use either a comma (`,`) or a semicolon (`;`) to separate entries in a list of hostnames.

Example:

`name.hp.com:8000, *.hp.com:8001` .

In these examples, all hostnames ending in `hp.com` configure their BBC Communication Broker to use port 8001, except the host name , which uses port 8000. All other hosts use the default port 383.

You can also use IP addresses and the asterisk wildcard (`*`) to specify hosts.

Example:

`15.0.0.1:8002` , `15.*.*.*:8003`

`bbc.http`

HTTP Namespace for node-specific configuration. For application-specific settings, see the section `bbc.http.ext.*` . Note that application-specific settings in `bbc.http.ext.*` override 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 <*app_Name*> to listen for requests. Note that it makes sense to explicitly set this parameter only in the `bbc.http.ext.<` *app_Name* `>` 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. The 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. The default is port `0` . The operating system assigns the first available port.

Microsoft Windows systems do not immediately release ports for reuse. On Microsoft Windows systems, this parameter should be a large range.

`string CLIENT_BIND_ADDR = <`*address*`>`

Bind address for the client port. The 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 is used.

b

List of hostnames, separated by a comma or a semicolon, for which the proxy is *not* used.

The first matching proxy is chosen.

It is also possible to use IP addresses, instead of hostnames. As a result, `15.*.*.*` or `15:*:*:*:*:*:*:*` would be valid as well. However, you *must* specify the correct number of dots or colons. IP version 6 support is not currently available, but will be available in the future.

`string DOMAIN`

Defines the default DNS domain to use if no domain is specified for a target host. This domain name is appended to hostnames not containing a DNS domain name if a match for the hostname alone cannot be found. This is done for PROXY lookups and lookups in the `[cb.ports]` table. (For example, if you specify the hostname `merlin` and `DOMAIN = "bbn.hp.com"` , the `[cb.ports]` entries are first searched for the match of `merlin` . If there is no match found for the hostname `merlin` , a search is made for `"merlin.bbn.hp.com"` , `"*.bbn.hp.com"` , `"*.hp.com"` , `"*.com"` , and `""` , in that order.

`bbc.fx`

BBC File-Transfer Namespace for node-specific configuration. For application-specific settings, see the section `bbc.fx.ext.*` . Note that application-specific settings in `bbc.fx.ext.*` override 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. The default directory is < *`OvDataDir`* > .

`string FX_TEMP_DIRECTORY = <`*`directory path`*`>`

Temporary directory where uploaded files are placed while the upload is in progress. After the upload completes, the file is moved to <*`directory path`*> . The 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. You may override the upload target directory with this configuration parameter. The default directory is `FX_BASE_DIRECTORY` .

`bbc.snf`

BBC Store-and-Forward Namespace for node-specific configuration. For application-specific settings, see the section `bbc.snf.ext.*` . Note that application-specific settings in `bbc.snf.ext.*` override node-specific settings in `bbc.snf` .

You can use the following parameters:

`string BUFFER_PATH = <`*`path`*`>`

Specifies the SNF path where the buffered requests are stored.

Default:

<*`OvDataDir`*> `/datafiles/bbc/snf/` <*`app_Name`*>

`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>* `.` *<app_Name>*
- `bbc.http.ext.` *<app_Name>*

This is the Dynamic External-Communication Namespace for application-specific settings. Note that application-specific settings in `bbc.http.ext.*` override node-specific settings in `bbc.http` .

For a list of the parameters you can use in the `bbc.http.ext.*` namespace, see the section "bbc.http."

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 node-specific settings in `bbc.fx` .

File Transfer External Namespaces:

- `bbc.fx.ext.` *<compID>* `.` *<app_Name>*
- `bbc.fx.ext.` *<app_Name>*

for a list of the parameters you can use in the `bbc.fx.ext.*` namespace, wee the section "bbc.fx."

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 node-specific settings in `bbc.snf` .

Store and Forward External Namespace:

- `bbc.snf.ext.` *<compID>* `.` *<app_Name>*
- `bbc.snf.ext.` *<app_Name>*

For a list of the parameters you can use in the `bbc.snf.ext.*` namespace, see the section "bbc.snf."

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 is used.

SEE ALSO

ovbbccb(1) , bbcutil(1) .

COPYRIGHT

bbcutil

NAME

bbcutil

- a tool for debugging a BBC-based server.

SYNOPSIS

```
bbcutil -h|-help
bbcutil -version
bbcutil -ovrg [<ovrg>]
bbcutil -reg|-registrations [<hostname>|<ip>] [-v|-verbose]
bbcutil -deregister {<path>|*} [-force] [-v|-verbose]
bbcutil -ping {[<hostname>|<ip>[:<port>]] | [<uri>]} [count] [-v|-verbose]
bbcutil -status {[<hostname>|<ip>[:<port>]] | [<uri>} [-v|-verbose]]
bbcutil -migrate {[<namespace>] [<appname>] [<filename>]} [-v|-verbose]
bbcutil -count|-size|-list [-p|-path <path>] [-t|-target <target>] [-v|-verbose]
bbcutil -getcbport [<hostname>|<ip>]
bbcutil -gettarget [<hostname>|<ip>]
```

DESCRIPTION

The `bbcutil` command helps you to debug an HP Operations BBC-based server. You can use the `bbcutil` command to list all applications registered to a HP Operations Communication Broker, to check whether specified communication services are alive, and to display details about the current state of the server.

Parameters

The `bbcutil` command incorporates the options in the following list. For example, the syntax for the [< *hostname* >|< *ip* >][:< *port* >]] string in the options `-registrations` or `-ping` can be a hostname and a port separated by a colon (:).

The string can also be a full URL path (including protocol), such as the following:

```
https://merlin.guilford.mycom.com:383/com.hp.ov.coda
```

`bbcutil` recognizes the following options:

`-h|-help`
: Displays and describes the available options for the `bbcutil` command.

`-version`
: Displays the version of the HPOM communication in use.

`-ovrg <` *`ovrg`* `>`

Executes a `bbcutil` command option in the context of the HPOM Resource Group specified by `<`*`ovrg`*`>`. This is an optional command. You can use it with other `bbcutil` commands. For example, the `bbcutil -ovrg testsrv -getcbport` command returns the Communications Broker port number of the HPOM Resource Group, `testsr v`.

`-reg|-registrations [<` *`hostname`* `>|<` *`ip`* `>]`

Queries a Communications Broker on the node specified by <*hostname*> or <*ip*> , and displays a list of all registered applications. If you do not specify the hostname or IP address, localhost is assumed.

`-deregister {<` *`path`* `>|*} [-force]`

Deregisters the specified path from the Communications Broker on the localhost. You can use an asterisk (*) to denote *all* paths. The specified path is deregistered if the application servicing the specified path is currently running. Use the `-force` option to override this behavior and force the path to be deregistered.

`-ping {[<` *`hostname`* `>|<` *`ip`* `>][:<` *`port`* `>]] | [<` *`uri`* `>]} [count]`

Pings the specified HPOM for Windows server process. To locate the server process to ping, you can provide a hostname or IP address with an optional port number or a URL. If a URL is given with the path of a valid process registered with the Communications Broker, the Communications Broker automatically forwards the ping to the registered process. Count specifies the number of times to execute the ping. You may specify the node with a hostname or IP address. The default for the node is localhost. The default for the port is the Communications Broker port on the specified node. The default count is 1.

`-status {[<` *`hostname`* `>|<` *`ip`* `>[:<` *`port`* `>]] | [<` *`uri`* `>]}`

Displays the status of the specified HPOM for Windows server process. To locate the server process, you can provide a hostname or IP address with an optional port number or a URI. You may specify the node with a hostname or IP address. The default for the node is localhost. The default for the port is the Communications Broker on the specified node.

`-migrate {[<` *`namespace`* `>] [<` *`appname`* `>] [<` *`filename`* `>]} [-v|-verbose]`

Migrates the specified BBC configuration parameters. If you do not specify command parameters, the BBC 2 LLB and the BBC 4 CB parameters are migrated to the namespace `bbc.cb` in the configuration database. The BBC 2/3 DEFAULT parameters are migrated to the namespaces `bbc.http` , `bbc.fx` , and `bbc.snf` . The BBC 4 CB parameters override BBC 2 LLB parameters. The namespace specifies the BBC 2/3/4 namespace to migrate the parameters from. The < *appname* > specifies the application name to use in determining the BBC 5 target namespace. Parameters are migrated to the `bbc.http.ext.<` *`appname`* `>` , `bbc.fx.ext.<` *`appname`* `>` , and `bbc.snf.ext.<` *`appname`* `>` namespaces. The *`filename`* parameter specifies the file to read the parameters from. The default file name is the BBC 2 standard `default.txt` file and the standard BBC 4 Communications Broker `settings.ini` file. The BBC 4 `settings.ini` parameters override the BBC 2 `default.txt` parameters.

`-count`
: Displays the number of requests in a store-and-forward buffer for the specified target, or in the entire buffer if no target is specified.

`-size`
: Displays the size of a store-and-forward buffer. If you specify `-verbose` as well, the size of each individual request displays. If you specify a target, only the size of the requests to this target displays.

`-list`
: Displays all requests in a store-and-forward buffer for the specified target, or in the entire buffer if no target is specified.

`-p|-path` *<path>*
: Defines the path to the store-and-forward buffer. This parameter is used to set the *BUFFER_PATH* parameter.

`-t|-target` *<target>*
: Specifies the target URI whose information you want to display. If you do not specify a target, information for all targets in the buffer displays.

`-verbose`
: Shows more detailed output.

`-getcbport [<` *hostname* `>|<` *ip* `>]`
: Displays the Communications Broker port number of the node specified by < *hostname* > or < *ip* > . If you do not specify the hostname or IP address, localhost is assumed.

`-gettarget [<` *hostname* `>|<` *ip* `>]`
: Displays the IP address of the target node and the Communications Broker port number, or the HTTP Proxy and port number if you configured a proxy for the specified <*hostname*> or <*ip*> .

AUTHOR

`bbcutil` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
: `bbcutil` exited normally with no error.

`1`

Command syntax error encountered. For possible values, see the command syntax.

`2`
: Command partially succeeded.

`3`
: Command failed. For details, see the command output.

`4`
: `bbcutil` could not complete the requested command because of an authorization error.

`100`
: An exception was encountered, causing the Communications Broker to exit.

Corresponding error messages are written to stderror.

EXAMPLES

The following examples show you how to use the `bbcutil` command:

- To show the status of Communication Broker on the local node:

  `bbcutil -status`

- To query the communication server located at `https://merlin.guilford.mycom.com:383/com.hp.ov.coda` for details about the current state of the server:

  `bbcutil -ping https://merlin.guilford.mycom.com:383/com.hp.ov.coda`

SEE ALSO

ovbbccb(1) , bbc.ini(4) .

COPYRIGHT

opcagt

The command opcagt administers the agent processes running on an HP Operations Manager managed node . This command can be integrated in the startup procedure of the computer. If used without any option, opcagt returns the current status of the agent services running on the local system.

Command synopsis

Windows

```
opcagt [ -help | -h | -stop | -kill | -start | -restart | -cleanstart | -status | -type [-
verbose]]
```

Linux

```
./opcagt [-help | -stop | -kill | -start | -restart | -cleanstart | -status | -type [-
verbose]]
```

Options

`-help | -h`
: Displays the opcagt options.

`-stop`
: Stops the Event/Action Agent and Coda.

`-kill`
: Stops the HPOM Core functionality.

`-start`
: Starts the Event/Action Agent and Coda.

`-restart`
: Restarts the HPOM Core functionality.

`-cleanstart`
: Clears the agent's buffers and then starts the agent. This option discards any messages that the agent added to the message buffer but did not send. This option also removes any scheduled tasks that agent added to the action queue but did not start.

`-status`
: Displays the status of Event/Action Agent and Coda.

`-type`
: Displays the type of Event/Action Agent.

`-verbose`
: Displays detailed information about the Event/Action Agent.

`-version`
: Displays version information.

Exit values

This command exits with a value of zero after successful operation. If a failure occurs, the exit value is set to one and an appropriate message displays.

In addition, the HP Operations agents record any warnings or errors in the local HPOM log file:

- Windows

  `'%OvAgentDir%\log\OpC\opcerror'`

- UNIX

  `/var/opt/OV/log/OpC/opcerror`

Restrictions

This command can be run only by a user with administrative rights.

Windows examples

To start all agent services on the local system

```
opcagt -start
```

Related Topics:

- opcmon
- opcmsg
- ovpmutil

opcmon

The command opcmon forwards the current value of the monitored object to the monitoring agent of HP Operations Manager running on a local managed node .
The monitoring agent checks this value against the configured threshold. According to the monitor configuration, the event is locally logged, suppressed, or forwarded to the message agent running on the local system if the threshold is exceeded. The message agent forwards the message to the HP Operations management server , where the message can be reviewed in the HP Operations message browser by the responsible HP Operations operators.
If a local automatic command is set up to occur when the threshold is exceeded, this command is immediately started by the local HP Operations agent. The monitoring agent must be configured and operating on the managed node, otherwise the opcmon command will fail.

Command synopsis

```
opcmon [ -help ] <object_name>[-<shortname>]=<value> [ -object <msg_object> ] [ -option
<variable>=<var_value> ]*
```

Options

- `-help` Print usage message of opcmon. All other parameters are ignored.
- `<object_name>[-<shortname>]=<value>` Object name is the name of the measurement threshold policy. When the measurement threshold policy has been configured for multiple instance data, the short name is used to uniquely identify each instance within the policy.
- `-object <msg_object>` Value of the object text box which is part of an HP Operations message. Setting the object with the opcmon can be used for the object monitoring.
- `-option <variable>=<var_value>` Sets the variable `$OPTION(<variable>)` to `<var_value>`. Within the message conditions this variable can be used to access the value passed with the opcmon call. Special characters must be escaped with a backslash "\".

Exit Values

This command exits with value zero (0) after successful execution. If something is wrong regarding the passed parameters, opcmon exits with value 2 and explain the problem on standard error. For other errors, the exit value is set to 1 and an appropriate error message is returned on standard error.

Restrictions

None.

Examples

```
opcmon cpu_load=78.4
```

```
opcmon DB_STATUS=1
```

```
opcmon disk_util=91.1 -object /tmp -option device=/dev/dsk/c0t6d0
-option addl_auto_actn="bdf /"
```

Related Topics:

- opcagt
- AutomationWrapper: opcmsg

opcmsg

The command opcmsg generates a message for HP Operations Manager. Before the message is submitted, it is interpreted by the HP Operations Message Interceptor on the local managed node where the command is executed. Depending on how you configure the message, the message can be:

- Discarded
- Locally logged
- Forwarded to the management server
- Forwarded to the management server , with local logging.

The behavior of messages depends on the configuration of interceptors (or opcmsg policies). A message may be created, or may be suppressed. For example, you might have a suppress condition in the opcmsg policy, which for example suppresses all messages with application=Test.

If you have such a condition and submit the call:

```
opcmsg application=Test msg_text="Test message"
```

then you will not get a message in the browser (it has been suppressed). The message interceptor must be configured with at least one Open Message interface policy and be running on the managed node , otherwise the opcmsg command will fail.

Command synopsis

```
opcmsg [ -help ] [ -id ] application=<application> object=<object name>
msg_text="<message_text>" [ severity=<severity label> ] [ msg_grp=<message_group> ]
[ node=<node Name> ] [ service_id=<service name> ] [ -option variable=<value> ]*
```

Options

You can specify any unique prefix for the available options. Note that the prefix for the option severity is s while the prefix for the option service_id is ser .

- -help Print usage message of opcmsg. All other options are ignored and no message is submitted.
- -id Return the message ID of the submitted message to stdout. This option also sets the OPCDATA_REMARK_FOR_ACK flag of the message, so that the manager information of the message is held by the message agent.
- severity=<severity label> Specifies the severity of the message. Following severities are supported: normal, warning, minor, major, critical. By default severity normal is applied.
- application=<application name> name of application (or script/program) which is affected by or has detected the event/problem.

- msg_grp=<message group> Default message group to which the message belongs. By default, no message group is assigned.
- object=<object name> Object which is affected by or has detected the event/problem.
- msg_text=<message text> Descriptive text explaining the event/problem in more detail.
- node=<node Name> System on which the event/problem is detected. By default the node name of the current system is applied.
- service_id=<service name> name of the service (as defined in the Service Editor) to which the message is mapped.
- -option variable=<value> Sets the variable $OPTION(variable) to value. Within the message conditions this variable can be used to access the value passed with the opcmsg call.
  Special characters must be escaped.

Exit Values

This command exits with value zero after a message is successfully generated; in case of an internal error, 1 is returned and an error message displays. If a syntax or usage error is detected, 2 is returned and an error message displays.

Restrictions

This command can be run by any user. The message group (msg_grp), the object, and the application parameter should not be longer than 32 bytes, because this is the maximum size HP Operations can handle with these parameters.

Example

To submit a normal message issued when a user logs onto the system, you could set up the following scheduled task

```
opcmsg appl=ScheduledTask obj=login severity=normal msg_g=Security msg_t="%USERNAME% logged onto system %COMPUTERNAME%"
```

Related Topics:

- opcagt
- opcmsg
- AutomationWrapper: opcmon
- Quick start: how to create a policy

opcntprocs

This program is used in a measurement threshold policy to check if a particular process is running on a managed node . The policy must use the source type Program , and run opcntprocs with the following parameters:

```
opcntprocs <policy name> <process name>
```

For example, to receive a message when the process RPCSS is not running, create a policy with the name myPolicyName, set the source type to Program , and type the following in the Program name text box:

```
opcntprocs myPolicyName RPCSS
```

Set the threshold of the policy so that a message is sent if the value returned is less than one (1). Usage example: -v verbose messages -l output sent to c:\temp\opcntprocs_.log -t trace details about API calls to logfile -h print usage information

opctemplate

The command opctemplate enables and disables policies on HP Operations managed nodes .
If used without any option, opctemplate lists all deployed polices with type, name, and status.

The command opctemplate lets you to enable and disable policies programmatically, directly on the managed node, without using the HP Operations Manager for Windows console. This is useful in situations where you want to use scripts or programs to disable policies, for example during regular scheduled outages or when packages in cluster environments switch.

Enabling and disabling policies with opctemplate does not change the status of the node in the inventory. This means that the HP Operations Manager console does not get informed when policies are enabled and disabled locally on the node with opctemplate.

Command synopsis

```
opctemplate [ -help ] [ -list ] [ ( -enable | -disable ) policy_name ... ]
```

Option	Short	Description
-help	-h	Displays the opctemplate options.
-list	-l	Lists all deployed policies with type, name, and status. This is the default
-enable <policy_name>	-e	Enables policies specified by <policy_name> on a managed node.
-disable <policy_name>	-d	Disables policies specified by <policy_name> on a managed node.

The option <policy_name> can be replaced with any of the following symbolic names to specify all policies of a certain type:

Name	Policy type
`-all`	Policies of all types
`-all_logfile`	Windows Event Log policy
`-all_monitor`	Measurement Threshold policy
`-all_snmptrap`	SNMP Interceptor policy
`-all_opcmsg`	Open Message Interface policy
`-all_wbem`	Windows Management Interface policy
`-all_schedule`	Scheduled Task policy
`-all_svcdisc`	Service Discovery policy

Exit Values

This command exits with a value of zero (0) after successful operation. If a failure occurs, the exit value is set to one and an appropriate message displays. In addition, the HP Operations agents record any warning or error in the local HPOM log file `\usr\OV\log\OpC\<node>\opcerror` (or `/var/opt/OV/log/OpC/opcerror` for UNIX operating systems).

Restrictions

This command can only be run by a user with administrative rights.
The HP Operations agent must be running on the managed node to execute this command successfully.

Examples

To list all deployed templates:

```
opctemplate -l
```

To enable the "opcmsg" policy:

```
opctemplate -e opcmsg
```

To disable all SNMP Interceptor and Windows Event Log policies:

```
opctemplate -d -all_snmptrap -all_logfile
```

Related Topics:

- Disable policy
- Enable policy
- Enable policy switching on DCE agents

ovagtrep

NAME

ovagtrep

- Enables configuration and control of the discovery agent and agent repository.

SYNOPSIS

```
ovagtrep  [-clearAll] |
          [-run <policy name>] |
          [-publish]
```

DESCRIPTION

The discovery agent is an extension to the HTTPS agent, which runs service discovery policies that have been deployed from a management server. It stores the services that it discovers in the agent repository, which is a local data store of services that exist on the node.

The agent synchronizes the services in the agent repository with the management server. The management server receives details of new, changed, and removed services only. Details of unchanged services are not resent.

The `ovagtrep` command enables you to configure and control the discovery agent and agent repository. It has the following options:

`-clearAll`

Clears all services from the agent repository. The next time that the discovery agent runs service discovery policies, it will recreate the services. The agent then synchronizes the services with the management server. This is enables you to force the agent to synchronize unchanged services with the management server.

`-run <`*`policy name`*`>`

Runs a service discovery policy. Use this to run a policy at an unscheduled time, to discover any changes immediately. The agent sends details of changes to the management server. You can find the names of installed policies using `ovpolicy` .

`-publish`

Resends details of all the services that are currently in the agent repository to the management server. Use this for troubleshooting if services fail to appear on the management server.

The discovery agent and agent repository are part of a component that is registered with the control service.

You can start and stop the component with the commands `ovc -start agtrep` and `ovc -stop agtrep` .

You can use the command `ovconfchg` to modify the following settings in the agtrep name space:

`ACTION_TIMEOUT` < *`minutes`* >

Sets the maximum number of minutes that a service discovery policy can run. If the policy runs any longer, the discovery agent stops running the policy and logs an error in the system log (< *`data_dir`* `>/log/System.txt`).

`INSTANCE_DELETION_THRESHOLD` < *`value`* >

Sets the number of times that service discovery policies must fail to discover existing services before the agent deletes the services from the agent repository.

If a service discovery policy can no longer discover a service that exists in the agent repository, the discovery agent deletes the service from the agent repository only after the service discovery policy has run the number of times that you specify with this setting.

For example, to set the action timeout to five minutes with the command `ovconfchg -ns agtrep -set ACTION_TIMEOUT 5` .

After you change the action timeout or instance deletion threshold, restart the component with the command `ovc -restart agtrep` .

AUTHOR

`ovagtrep` was developed by Hewlett-Packard Company, LP.

SEE ALSO

ovc(1)

ovpolicy(1)

ovconfchg(1)

ovappinstance

NAME

ovappinstance

- return configuration parameters for application instances.

SYNOPSIS

```
ovappinstance  -h | -help
ovappinstance  -v | -version
ovappinstance  -i | -instance <instance> {-st | -state} | {-h | -host} [-an | -appNamespace <app
ovappinstance  -is | -instances [-an | -appNamespace <appNamespace>]
ovappinstance  -ai | -activeInstances [-an | -appNamespace <appNamespace> ]
ovappinstance  -vc | -verifyConfig
```

DESCRIPTION

The ovappinstance command reads and displays the information contained in the APM XML configuration files. For information about the parameters you can use with the ovappinstance command, see "Parameters." For information about the options you can use with the ovappinstance command parameters, see "Options."

Parameters

The ovappinstance command recognizes the following parameters:

`-h | -help`
: Displays the command parameters and options.

`-v | -version`
: Displays the version of the command.

`-i | -instance` < *instance* >
: Returns information about the specified application instance.

`-is | -instances`
: Returns information about all application instances found.

`-ai | -activeInstances`
: Returns information about all application instances found to be up and running.

`-vc | -verifyConfig`

Check s and report on the validity of the APM XML configuration files.

Options

You can use the following options with the `ovappinstance` command parameters:

`-st | -state`

Displays the outage state of the instance specified in < *instance* > .

`-h | -host`

Gets the virtual IP address of the instance < *instance* > . Alternatively, if the command is executed on a node, which is not configured as part of a high-availability cluster, it gets the FQDN or IP address of the local host.

`-an | -appNamespace`

Specifies the name of the application namespace whose information you want to display.

Return Codes

`ovappinstance` issues the following return codes:

`0`

All steps were completed successfully.

`1`

One or more steps failed.

EXAMPLES

The following examples show how to use the `ovappinstance` command.

- To display a list of all application instances for a given application namespace:

  ovappinstance -instances -appNamespace < *appNamespace* >

- To display a list of all application instances which are active (or running) in a given application namespace:

  ovappinstance -activeInstances -appNamespace < *appNamespace* >

AUTHOR

`ovappinstance` was developed by Hewlett-Packard Company.

SEE ALSO

ovclusterinfo(1) , ovconfpar(1) , ovpolicy(1) .

ovbbccb

NAME

ovbbccb

- control HTTPS communication using HP Operations Communication Broker proxies on local nodes.

SYNOPSIS

```
ovbbccb -h|-help
ovbbccb -version
ovbbccb -install|-remove [-v|-verbose]
ovbbccb -daemon|-nodaemon [-debug] [-v|-verbose]
ovbbccb -start|-stop <ovrg> [<hostname>|<ip>] [-v|-verbose]
ovbbccb -kill|-reinit [<hostname>|<ip>] [-v|-verbose]
ovbbccb -listovrg [<hostname>|<ip>] [-v|-verbose]
ovbbccb -ping {[<hostname>|<ip>[:<port>]] | [<uri>} [-v|-verbose]]
ovbbccb -status {[<hostname>|<ip>[:<port>]] | [<uri>} [-v|-verbose]]
```

DESCRIPTION

You use the ovbbccb command to control HTTPS communication using HP Operations Communication Broker proxies on local nodes. It controls the starting of the Communication Broker as a background daemon process or in normal mode, stopping, and re-initializing of the Communication Broker. You can also use the command to start and stop HPOM Resource Groups in the Communication Broker.

In addition, you can use the command to list all active HPOM Resource Groups and all applications registered to a Communication Broker, to check whether specified communication services are alive, and to display details about the current state of the server.

Parameters

The ovbbccb command incorporates the options in the following list. For example, the syntax for the `[<hostname>|<ip>][:<port>]]` string in the options `-registrations` or `-ping` can be a hostname and a port separated by a colon (:). It can also be a full URL path, including protocol.

Example:

```
https://merlin.guilford.mycom.com:383/com.hp.ov.coda
```

ovbbccb recognizes the following options:

`-h|-help`
Displays and describes the available options for the ovbbccb command.

`-version`
: Displays the version of the HPOM communication in use.

`-install`
: Installs the Communications Broker program as a service on a Microsoft Windows machine.

`-remove`
: Removes the Communications Broker program from the services on a Microsoft Windows machine.

`-daemon`
: Starts the Communication Broker, either as a background daemon process on a UNIX machine, or as a service on a Microsoft Windows machine.

`-nodaemon`
: Starts the Communication Broker as a foreground process (*default*).

`-debug`
: Disable the Control-C signal handler for debugging.

`-verbose`
: Shows more detailed output.

`-start <ovrg> [<hostname>|<ip>]`
: Starts the HPOM Resource Group specified by <*ovrg*> in the Communication Broker on the host specified by <*hostname*> or <*ip*> . If you do not specify the hostname or IP, ovbbccb uses the local host as the host. You must configure the HPOM Resource Group on a cluster node to use this option.

`-stop <ovrg> [<hostname>|<ip>]`
: Stops the HPOM Resource Group specified by <*ovrg*> in the Communication Broker on the host specified by <*hostname*> or <*ip*> . If you do not specify the hostname or IP, ovbbccb uses the local host as the host. You must configure the HPOM Resource Group on a cluster node to use this option.

`-kill [<hostname>|<ip>]`
: Stops the Communication Broker on the host specified by <*hostname*> or <*ip*> . If you do not specify the hostname or IP, ovbbccb used the local host as the host. You must set the *`LOCAL_CONTROL_ONLY`* parameter to `false` to make this option work on a remote node.

`-reinit [<hostname>|<ip>]`
: Communication Broker specified in <*hostname*> or <*ip*> reloads the configuration data and is re-initialized. If you do not specify the hostname or IP, ovbbccb uses the local host as the host.

  You can also use the `SIGHUP` signal on UNIX systems to re-initialize the Communication Broker process.

You must set the *`LOCAL_CONTROL_ONLY`* parameter to `false` to make this option work on a remote node.

`-listovrg [<hostname>|<ip>]`

Displays a list of all active HPOM Resource Group for the Communication Broker on the node specified by <*`hostname`*> or <*`ip`*> . If you do not specify the hostname or IP, ovbbccb uses the localhost as the host. You must set the *`LOCAL_CONTROL_ONLY`* parameter to `false` to make this option work on a remote node.

`-ping {[<` *`hostname`* `>|<` *`ip`* `>[:<` *`port`* `>]] | [<` *`uri`* `>]}`

Pings the specified HPOM for Windows server process. To locate the server process to ping, you can provide a hostname or IP address with an optional port number or a URI. If you provide a URI with the path of a valid process registered with the Communication Broker, the Communication Broker automatically forwards the ping to the registered process. You can specify the node with a hostname or IP address. The default for the node is localhost. The default for the port is the HP Operations Communication Broker port on the specified node.

`-status {[<` *`hostname`* `>|<` *`ip`* `>[:<` *`port`* `>]] | [<` *`uri`* `>]}`

Displays the status of the specified HPOM for Windows server process. To locate the server process, you can provide a hostname or IP address with an optional port number. The default for the node is localhost. The default for the port is the Communication Broker port on the specified node.

AUTHOR

ovbbccb was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`

ovbbccb exited normally with no error.

`1`

Command syntax error was encountered. For possible values, see the command syntax.

`2`

Command partially succeeded.

`3`

Command failed. For details, see the command output.

`4`
: Communications Broker start command failed because a Communications Broker process is already running.

`5`
: Communications Broker failed to start because a Local Location Broker process is already running. The Communications Broker is not supported on systems running the LLB. Stop the LLB before attempting to start the Communications Broker.

`6`
: Communications Broker failed to stop because the Communications Broker process is already stopped.

`7`
: Communications Broker failed to start because of a bind exception on the Communications Broker port to be opened.

`8`
: Communications Broker could not complete the command because of an authorization error.

`100`
: Exception was encountered, causing the Communications Broker to exit.

Corresponding error messages are written to `stderror` .

EXAMPLES

The following examples show you how to use the ovbbccb command:

- To start the Communication Broker as a daemon process on the local system:

  `ovbbccb -daemon`

- To start the HPOM Resource Group `WebCluster1` in the Communication Broker on host `merlin` :

  `ovbbccb -start WebCluster1 merlin`

- To display the status of the specified HPOM for Windows server process:

```
ovbbccb -status
Status: OK
(Namespace, Port, Bind Address, Open Sockets)
<default>    383    ANY    2
HP Operations HTTP Communication Incoming Connections
    localhost:55467  e91b67e4-a337-750a-163c-c3bbd2c257cc  BBC 06.00.030; ovbbccb 06.00.03
HP Operations HTTP Communication Reverse Channel Connections
 Opened:
  ovsolt9.india.hp.com:9090    BBC 06.00.030; ovbbcrcp 06.00.030
```

```
     Failed:
       abc:990 Host Unknown
     Pending:
       gdstgd:9809 Host Unknown
```

SEE ALSO

bbcutil(1) , bbc.ini(4) .

COPYRIGHT

ovbbcrcp

NAME

ovbbcrcp

- a tool to manage the Reverse Channel Proxy (RCP) and monitor RCP connections.

SYNOPSIS

```
ovbbcrcp -h|-help
ovbbcrcp -v|-version
ovbbcrcp -kill
ovbbcrcp -status
```

DESCRIPTION

You can use the ovbbcrcp tool to manage RCPs and monitor RCP connections. All HP BTO Software products that follow a client-server architecture use the Black Box Communication (BBC) component for communication. You can use a Reverse Channel Proxy (RCP) to satisfy the advanced security requirements for communication across trust zones separated by firewalls. An RCP allows you to establish a two-way communication (outbound and inbound) channel across a firewall configured to allow only outbound communication.

The RCP functions as a channel between the BBC server and the requests to the BBC server. An established RCP channel is referred to as a reverse channel. A reverse channel through which RCPs request the BBC server to initiate more reverse channels is referred to as a reverse administration channel.

You can deploy an RCP on one of the following:

- Client systems
- Dedicated RCP server

To establish a reverse channel, you must configure the BBC server, the BBC client, and the RCP.

Configuring a BBC Server to Enable RCP Communication

To enable communication from clients to the BBC server through an RCP, you must configure each BBC server. The BBC server loads the configuration from the bbc.< *server* > namespace, and establishes reverse administration channels during startup.

To configure a BBC server, use the following options:

ENABLE_REVERSE_ADMIN_CHANNELS

You can set this option to `true` to establish a permanent reverse administration channel with the RCPs specified in the `RC_CHANNELS` option. By default, this option is set to `false` for all BBC servers, except for the BBC Communication Broker (BBC CB).

For more information about this option, refer to the following example.

```
[bbc.cb]

ENABLE_REVERSE_ADMIN_CHANNELS=true

RC_CHANNELS=pnode:9090
```

The options specified in the example instructs BBC CB on the management server to contact the RCP on the pnode node and port 9090 when starting up.

`RC_CHANNELS`

Use this option to specify the list of RCPs with which you can establish reverse channels. If you specify the OvCoreID, BBC validates this ID against the core ID of the RCP. You can specify multiple RCPs by separating the RCPs using a semicolon (`;`).

You can specify the list of RCPs in the following format:

<*RCP_hostname*>:<*RCP_port*>[,<*RCP_OvCoreID*>][;<*RCP2*>.....]

In this syntax, <*RCP_hostname*> specifies the RCP host name, <*RCP_port*> specifies the RCP port number, and <*RCP_OvCoreID*> specifies the core ID of the RCP.

You must use the `-ovrg server` option with the ovconfchg command if the HP Operations server runs on a High Availability (HA) cluster. If the HP Operations server runs as an HA resource group, use the `ovconfchg -ovrg server -ns bbc.cb -set RC_CHANNELS` <*value*> command, where <*value*> specifies the RCPs specified in the `RC_CHANNELS` option.

`RC_CHANNELS_CFG_FILES`

Use this option to specify the list of configuration files. A configuration file can contain a list of one or more RCPs with which you can establish reverse channels. You must place the specified configuration files in the <*OvDataDir*>/`conf.bbc` directory, where <*OvDataDir*> specifies the name of the HP Operations data directory. You must use this option in place of the `RC_CHANNELS` option if you use multiple RCPs that require a frequent hostname change.

You can specify a list of configuration files by separating the configuration file names using a comma (`,`) in the following format:

<*filename*>[,<*filename*>....]

In this syntax, <*filename*> specifies the name of the configuration file.

Each line in the configuration file can contain only one RCP name. For each RCP, you must specify a port number. The `OvCoreID` is an optional parameter that you can specify.

It must be separated from the port number by a comma as follows:

< *RCP_hostname* >:< *port* >[,< *RCP_OvCoreID* >]

If you change only a few RCP host names inside one or more files specified in the `RC_CHANNELS_CFG_FILES` option, you must use the ovconfchg command to trigger the BBC server to refresh the configuration, as follows:

```
ovconfchg ns bbc.cb -set ENABLE_REVERSE_ADMIN_CHANNELS true
```

`RETRY_INTERVAL`
: Use this option to specify the retry interval in minutes to establish a reverse channel with an RCP.

Enabling Communication Broker Connections to the RCP

The Communication Broker (ovbbccb) runs with `/var/opt/OV` as the root directory. The name service relevant configuration files that are necessary to open Transmission Control Protocol (TCP) connections are present in the `/etc directory.` This prevents ovbbccb from creating connections to the RCP.

To resolve this problem, you must do the following:

- Create the directory named etc under `/var/opt/OV` .
- Copy the name service relevant configuration files (for example, `resolv.conf` , `hosts` , `nsswitch.conf`) from `/etc` to `/var/opt/OV/etc` .

Alternatively, you can disable the ovbbccb `chroot` feature by running the following command:

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

This method resolves the problem of preventing ovbbccb from creating connections to the RCP.

Configuring a BBC Client to Enable RCP Communication

To configure a BBC client, you must specify the hosts that must be connected through an RCP. You can specify the list of RCPs in the XPL configuration database under the `bbc.http` namespace. Use the syntax of the normal proxy configuration to specify the RCP configuration. If you do not specify the port number of the RCP, it is assumed that BBC CB is running on the current node. If you configure the OvCoreID, the BBC client verifies the OvCoreID of the RCP. If you do not specify the port number of the RCP in the configuration file or the BBC CB, BBC fails to open the connection to RCP.

You can configure a BBC client using the following option:

`PROXY`
: Use this option to specify the RCP and port name for a hostname.

The format to specify this option is shown in the following example:

```
PROXY=pnode.hp.com:9090-(pnode.hp.com,*.noallow.hp.com)+(*.hp.com)
```

In this example, the parameters specified are as follows:

`pnode.hp.com`
: Name of the RCP.

`9090`
: Port number.

`-(*.noallow.hp.com)`
: Specifies that the RCP must not be used to connect to all hostnames ending with `.noallow.hp.com` . You can separate multiple hostnames with commas (`,`) or semicolons (`;`).

`+(*.hp.com)`
: Specifies that the specified RCP must be used to connect to all hostnames ending with `.hp.com` . You can separate multiple hostnames with commas (`,`) or semicolons (`;`).

The BBC client connects to the RCP that first matches the specified set of conditions.

In the example shown in this section, the BBC client connects to any host name that ends with `.hp.com` by using the RCP on the system pnode and the port `9090`.

You can also use IP addresses instead of hostnames to specify the hosts. For example, `+(15.*.*.*)` specifies that the RCP must be used to connect to hosts with an IP address that starts with 15. You may not configure a normal proxy server and an RCP on the same system. You must also make sure that you specify the RCP system name in the list of hostnames for which the RCP must not be used. This helps to ease the communication through the RCP.

Configuring RCP

To configure RCP, you can use the following option in the `bbc.rcp` namespace:

`SERVER_PORT`
: Use this option to specify the RCP port number.

Starting and Stopping RCPs

You can start or stop the RCP process by using the ovc command. This command registers the RCP process as `ovbbcrcp` under the `RCP` category.

By default, the `ovbbcrcp` process is not registered with HP Operations Control (OvCtrl).

You must register the `ovbbcrcp` process with the `ovctrl` daemon by using the following command.

`$OvInstallDir/bin/ovcreg -add $OvInstallDir/newconfig/DataDir/conf/bbc/ovbbcrcp.xml`

`$OvInstallDir` is the directory in which HP Operations Manager is installed.

To start the RCP process, use the following command:

`ovc -start ovbbcrcp`

To stop the RCP process, use the following command:

`ovc -stop ovbbcrcp`

Parameters

The `ovbbcrcp` command recognizes the following options:

`-h|-help`
: Displays and describes the available options for the `ovbbcrcp` tool.

`-v|version`
: Displays the version of the OV RCP.

`-kill`
: Stops the RCP on the local node.

`-status`
: Displays the RCP status.

AUTHOR

`ovbbcrcp` is developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
: `ovbbcrcp` exited normally with no error.

`1`
: Command syntax error was encountered. For possible values, refer to the command syntax.

`2`
: Command partially successful.

`3`

Command failed. For details, see the command output.

`4`

Command to start RCP failed because of an existing RCP process.

`6`

RCP failed to start because of a bind exception on the RCP port to be opened.

`100`

Exception encountered that resulted in an RCP exit.

Corresponding error messages are written to `stderror` .

EXAMPLES

The following example shows you how to use the `ovbbcrcp` tool.

To display the status of the RCP:

```
ovbbcrcp -status

Status: OK

(Namespace, Port, Bind Address, Open Sockets)

  bbc.rcp    9090    ANY    1

 Admin Reverse Channel Connections Accepted
  ovsolt9.india.hp.com:383  e91b67e4-a337-750a-163c-c3bbd2c257cc  BBC 06.00.030; ovbbccb 06.00.

 Admin Reverse Channel Connections Opened

 Normal Connections
Incoming
localhost:55464  e91b67e4-a337-750a-163c-c3bbd2c257cc  BBC 06.00.030; ovbbcrcp 06.00.030

Outgoing

 Queued CONNECT connections
+------------------------------------+---------------------+
|Source Address | Target Address
+------------------------------------+--------------------

 HTTP Tunneled Connections
+---------------------------+---------------------------+--+
| Source Address | Destination Address | Target Address|
+-------------------------+-------------------------+--+
```

See Also

ovbbccb(1)

COPYRIGHT

ovc

NAME

ovc

- perform actions on local components.

SYNOPSIS

```
ovc  -h|-help
ovc  -start [<target> ... ] [-boot]{[-async]|[-verbose]}
ovc  -stop [<target> ... ][-nostart]{[-async]| [-verbose]}
ovc  -restart [<target> ... ]
ovc  -kill [-verbose]
ovc  -status [<target> ... ] [-level <level>]
ovc  -trace [<target> ... ]
ovc  -notify <event> [<target> ...] [-value <value>]
ovc  -version
```

DESCRIPTION

The ovc command controls the starting and stopping, event notification, and status reporting of all components registered with the HP Operations control service.

A component can be a server process belonging to any of the products, such as HP Operations Manager for Windows (HPOM for Windows), HP Operations agents (for example, the Performance Agent or the Discovery Agent), an event interceptor, or an application delivered by an integrator. Each component must have an associated registration file providing HPOM with configuration and process information about the component. For more information about registration, refer to ovcreg(1) .

A target can be either a component or a group of components, defined as a category. The ovc command first tries to initiate action on the category specified in `target` . If the category called *`target`* is not found, ovc then tries the individual component called *`target`* . Note that a category name may not match any component name.

The HP Operations control daemon or service automatically restarts any component that terminates unexpectedly if the `AutoRestart` option in the registration file of the component is set to `true` . If you stop the HP Operations control daemon or service using the `-kill` option, all registered components are stopped as well.

Parameters

The ovc command recognizes the following options:

`-h|-help`
: Displays *all* available options for the ovc command.

`-start [<`*`target`*`> ... ] [-boot ]{[-async]| [verbose]}`
: Starts the selected components. The <*target*> option specifies a component or category. If you do not use <*target*> , all components are started. If you do not use `-boot` , only components that start at boot time are started.

  The -async option starts the components asynchronously. If you use the -verbose option, the ovc command displays the progress of the command execution. You can use the -async or the -verbose option, but you may not include these options together in a command.

`-stop [ <`*`target`*`> ... ] [-nostart] {[-async]| [verbose]}`
: Stops the selected components. The <*target*> option specifies a component or category. If you do not use <*target*> , all components are stopped *except* those components that belong to the CORE component group. If you specify the `-nostart` option, and if the control daemon is not running, the command does not perform any action. If you do not specify the `-nostart` option, the ovc `-stop` command starts the control daemon, and ovbbccb components if these components are not running. The -async option starts the components asynchronously. If you use the -verbose option, the ovc command displays the progress of the command execution. You can use the -async or the -verbose option, but you may not include these options together in a command.

`-restart [ <`*`target`*`> ... ]`
: Stops components before they are restarted. The <*target*> option specifies a component or category. If you do not use <*target*> , all components are stopped and restarted.

`-kill [-verbose ]`
: Stops all components registered with the HP Operations control service. If you use the -verbose option, the ovc command displays the progress of the command execution.

`-notify < `*`event`*` > [< `*`target`*` > ... ] [-value < `*`value`*` >]`
: Sends notification of an event with the value of <*value*> to the component or category specified by the following:

  <*target*> ...

  You can specify the < *value* > to the component that generates the event (event generator) and that sends the event-related information to all components that request the event information (event subscribers). If you do not use < *target* > , the event notification is sent to all components. If you do not use <*value*> , only the event notification is sent.

`-trace <`*`target`*`> ... ]`
: This option is reserved for use by HP Support.

`-status [ <`*`target`*`> ... ] [-level <`*`level`*`> ]`

Reports the status of a component or category specified by <*target*> . The status report contains the component's label, description, category, process ID, and STATE. Components can be in one of the following states: Stopped (0 in numeric format), Starting (1), Initializing (2), Running (3), Stopping (4), N/A (5), or Aborted (6). If you do not specify <*target*> , the status of *all* components is returned.

The <*level*> option specifies the type and quantity of information to display, as follows:

`Level 0`

Status of registered components monitored by HPOM.

`Level 1`

Status of registered components, whether they are monitored by HPOM or not.

`Level 2`

Status of registered components and a dump of their registration information.

`Level 3`

ID of core processes. Zero (0) indicates root. Non-zero indicates non-root ownership.

`Level 4`

Similar to level 0, but the STATE is reported in numeric format.

`Level 5`

Similar to level 1, but the STATE is reported in numeric format.

`Level 6`

Similar to level 0, but the output is not formatted.

`Level 7`

Similar to level 1, but the output is not formatted.

`-version`

Prints the version of ovc .

AUTHOR

ovc was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`	Success.
`1`	Not defined.
`2`	Ignored.
`62`	UNIX daemon or Windows service is not running.
`63`	Control daemon is being initialized.
`64`	Generic error.
`65`	Invalid target.
`67`	Operation aborted.
`69`	Missing prerequisite.
`70`	Authorization error.
`71`	Operation on prerequisite failed.
`73`	Invalid event.

EXAMPLES

The following examples show how to use the ovc command and some of its options to control and display important information about registered components:

- To start the component registered as `opcle` :

  `ovc -start opcle`

  Before `opcle` itself starts, all the components that `opcle` depends on are started.

- To start the component registered as `opcle` and display the progress of the command execution:

  `ovc -start opcle` `-verbose`

  Before `opcle` itself starts, all the components that opcle depends on are started.

- To print the status of all registered components:

  `ovc -status`

- To stop the component registered as `opcle` :

  `ovc -stop opcle -verbose`

  Before `opcle` itself stops, all the components that depend on `opcle` are stopped. This command starts the control daemon and ovbbccb components if these components are not running.

- To stop the component registered as opcle using the `ovc -stop [<target>...] -nostart` option:

  `ovc -stop opcle -nostart`

  Before `opcle` itself stops, all the components that depend on `opcle` are stopped. This command does not perform any action if the control daemon is not running.

- To send the event `RECONFIGURE` to all running components:

  `ovc -notify RECONFIGURE`

- To start all components (and their dependents) belonging to category SERVER and AGENT.

  `ovc -start SERVER AGENT`

- To print the status of the component `opcle` and display the registration details:

  `ovc -status opcle -level 2`

SEE ALSO

ovcreg(1)

ovcert

NAME

ovcert

- manage certificates with the Certificate Client on an HTTPS-based node.

SYNOPSIS

```
ovcert -h|-help
ovcert -importcert -file <file> [-pass <passphrase>] [-ovrg <ov_resource_group>]
ovcert -exportcert -file <file> [-alias <alias>] [-pass <passphrase>] [-ovrg <ov_resource_group>]
ovcert -importtrusted -file <file> [-ovrg <ov_resource_group>]
ovcert -exporttrusted -file <file> [-alias <alias>] [-ovrg <ov_resource_group>]
ovcert -certreq [-instkey <file> [-pass <passphrase>]]
ovcert -list [-ovrg <ov_resource_group>]
ovcert -remove <alias> [-f] [-ovrg <ov_resource_group>]
ovcert -certinfo <alias> [-ovrg <ov_resource_group>]
ovcert -check
ovcert -status
ovcert -updatetrusted
ovcert -version
```

DESCRIPTION

You can use the `ovcert` command to manage certificates with the Certificate Client on an HTTPS-based node. You can execute tasks, such as initiating a new certificate request to the Certificate Server, adding node certificates and importing the private keys, and adding certificates to the trusted root certificates.

Parameters

The `ovcert` command incorporates the following options:

`-h|-help`

Displays usage help for the `ovcert` command options.

`-importcert -file <file>` [-pass <*passphrase*>] [-ovrg <*ov_resource_group*>]

Adds the certificate located in the file <*file*> (in PKCS12 format) as a node certificate, and imports the private key (which must be located in the same file as the private key for the node). You must specify the pass phrase for protecting the exported data, using encryption specified during creation of the data to import, as the parameter <*passphrase*> .

You can specify the optional <*ov_resource_group*> parameter to import an additional certificate on an

HA system. As a result, the specified certificate is not imported to the default location but to the HA default location for the specified package on the shared disk.

-exportcert -file <*file*> [-alias <*alias*>] [-pass <*passphrase*>] [-ovrg <*ov_resource_group*>]

Exports the currently installed node certificate together with its private key to the file system location specified as the parameter <*file*> (in PKCS12 format). You must specify the pass phrase for protecting the exported data, using encryption specified during creation of the data to import, as the parameter <*passphrase*> .

You can specify the optional <*ov_resource_group*> parameter to export an additional certificate on an HA system. As a result, the certificate for the specified HA package from the shared disk, rather than the default node certificate, is exported.

-importtrusted -file <*file*> [-ovrg <*ov_resource_group*>]

Adds the certificate located in the specified file (in PEM format) to the trusted root certificates.

You can specify the optional <*ov_resource_group*> parameter to import an additional root certificate on an HA system. As a result, the specified root certificates is imported to the HA default location for the specified package on the shared disk, rather than to the default location.

-exporttrusted -file <*file*> [-alias <*alias*>] [-ovrg <*ov_resource_group*>

Exports the trusted certificate to the file system location specified as the parameter <*file*> (in PEM format). You must specify the pass phrase for protecting the exported data, using encryption specified during the creation of the data to import, as the parameter <*passphrase*> .

You can specify the optional <*ov_resource_group*> parameter to export an additional certificate on an HA system. As a result, the certificate installed for the specified HA package from the shared disk, rather than the default node certificate, is exported.

-certreq [-instkey <*file*> [-pass <*passphrase*>]]

Initiates a new certificate request that is sent to the Certificate Server.

You can use the optional parameters <*file*> and <*passphrase*> to initiate a certificate request, based on the installation key that is contained in the specified file. You can generated such an installation key file with the ovcm tool on the certificate server.

You can use the installation key to authenticate the node on the certificate server. Such a request may be granted automatically, without human interaction.

-list [-ovrg <*ov_resource_group*>]

Displays the aliases of the installed certificates and trusted certificates.

-certinfo <*alias*> [-ovrg <*ov_resource_group*>]

Displays information (for example, serial number, issuer, subject, and fingerprint) for the certificate specified by <*alias*> .

`-remove <alias> [-ovrg <ov_resource_group> ]`

Removes the certificate specified by *<alias>* .

`-check`

Checks whether all prerequisites for SSL communication are fulfilled (for example, assigned OvCoreId, installed and valid certificate and private key, and installed and valid trusted certificate).

On completion, it displays the components checked and their status, as well as the final result.

`-status`

Contacts the Certificate Client, and displays the current certificate status, which can one of the following possible values:

- Certificate installed
- No certificate
- Pending certificate request
- Certificate request denied
- Undefined (if the Certificate Client cannot be contacted)

`-updatetrusted`

Retrieves the currently trusted certificates from the Certificate Server, and installs them as trusted certificates on the node.

`-version`

Returns the version of the tool (the component version).

AUTHOR

`ovcert` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`

All steps were successful.

`1`

One or more steps were not successful.

Corresponding error messages are written to stderror.

EXAMPLES

The following examples show how to use the `ovcert` command:

- To import the certificate, private key, and trusted certificates located in the file <*file* > to the system's key store:

  ovcert -importcert -file <*file*>

- To add the certificates located in <*file* > to the trusted certificates:

  ovcert -importtrusted -file <*file*>

ovclusterinfo

NAME

ovclusterinfo

- get information about clusters, cluster nodes, or high-availability (HA) resource groups.

SYNOPSIS

```
ovclusterinfo  -h | -help
ovclusterinfo  -v | -version
ovclusterinfo  -a | -all
ovclusterinfo  -c | -cluster {-ty | -type} | {-nm | -name} | {-st | -state} | {-nds | -nodes} |
ovclusterinfo  -n | -node <node> {-id} | {-st | -state}
ovclusterinfo  -g | -group <group> {-id} | {-st | -state} | {-ls | -localState} | {-nds | -node
```

DESCRIPTION

The `ovclusterinfo` command gets information about high-availability clusters, cluster nodes, and resource groups. This information includes the name, status, and type of the cluster, as well as the nodes configured in the cluster. The `ovclusterinfo` command also gets information about high availability (HA) resource groups, including the status, IP address, and nodes contained in the resource group. An HA resource group is a collection of resources (for example, files and processes) that are available on one node in a cluster. They can be switched to another cluster node as a single entity.

Parameters

The `ovclusterinfo` command accepts the following parameters:

`-h | -help`
    Displays all options for the `ovclusterinfo` command.

`-v | -version`
    Displays the version of the installed command.

`-c | -cluster`
    Displays information about the named cluster.

`-a | -all`
    Displays all available information about the named cluster, nodes, and resource groups.

`-a | -node`

Displays all available information about the named node in the cluster.

`-g | -group`
Displays information about the named HA resource group.

Options

You can use the following options with the appropriate command parameters:

`-ty | -type`
Display the type of cluster which is installed.

Possible values:

- Microsoft Clustering Services (Windows)
- MC/ServiceGuard (HP-UX)
- VERITAS Cluster Server (Solaris)
- Sun Cluster (Solaris)
- TRU64 Cluster (TCR)
- Red Hat Advanced Server (RHAS)
- HACMP (AIX)
- Unknown

`-nm |-name`
Name of the cluster.

`-st | -state`
Status of the cluster on the local node.

This status can be one of the following:

- Cluster is up
- Cluster is down
- State unknown

`-nds | -nodes`
Displays the names of the nodes in the cluster on separate lines. The cluster configuration determines how the node information displays (for example, short or long host names, IP address, and so on).

`-rgs | -groups`
All resource groups in the cluster.

`-status`
Status of the HA resource group, defined by <*rgname*> , on the local node.

`-virtualIPaddress`
Virtual IP address of the HA resource group, defined by <*rgname*> .

`-nodes`
The list of all nodes to which the HA resource group, defined by <*rgname*> , can fail over.

`-activeNode`
Node that currently hosts the HA resource group, defined by <*rgname*> .

AUTHOR

`ovclusterinfo` was developed by Hewlett-Packard Company.

EXAMPLES

The following examples show how to use the `ovclusterinfo` command:

- To display the name of the cluster:
  `ovclusterinfo -cluster -name`
- To display the names of all HA resource groups in the cluster:
  `ovclusterinfo -cluster -groups`
- To display the virtual IP address that you configured for the HA resource group haRG:
  `ovclusterinfo -group haRG -virtualIPaddress`
- To display the name of the node where the HA resource group haRG is currently running.
  `ovclusterinfo -group haRG -activeNode`

SEE ALSO

ovappinstance(1) , ovconfpar(1) , ovpolicy(1) .

ovcm

NAME

ovcm

- manage certificates with the Certificate Server in an HTTPS-based environment.

SYNOPSIS

```
ovcm -h|-help
ovcm -version
ovcm -newcacert [-ni]
ovcm -importcacert -file <file> [-pass <passphrase>]
ovcm -exportcacert -file <file> [-pass <passphrase>]
ovcm -listpending [-l]
ovcm -grant <reqid>
ovcm -deny <reqid>
ovcm -remove <reqid>
ovcm -issue -file <file> -name <nodename> [-pass <passphrase>] [-coreid <OvCoreId>] [-ca]
ovcm -genInstKey -file <file> [-context <context>] [-pass <passphrase>.
```

NOTE:
Do not use the `-issue` and `-genInstKey` options on the Windows platform.

DESCRIPTION

You can use the `ovcm` command to manage certificates with the Certificate Server in an HTTPS-based environment. You can execute tasks, such as creating public/private key pairs for signing certificates and granting and issuing signed certificates and the corresponding private keys against certificate requests from HTTPS nodes.

Parameters

The `ovcm` command incorporates the following options:

`-h|-help`
Displays all the command-line options for the `ovcm` command.

`-version`
Returns the version of the tool (the component version).

-newcacert [-ni]

Creates a new public/private key pair for signing certificates. If there is already a public/private key pair in use by the certification authority, you are asked whether this should be replaced. Use this option with care! An initial public/private key pair is automatically created when the Certificate Management component is installed.

The `-ni` non-interactive option creates a new public/private key pair without operator interaction. If a public/private key pair already exists, the request is cancelled.

`-importcacert -file <`*`file`*`>` [-pass <*passphrase*>]

Imports a certificate for signing certificate requests together with its private key (both are contained in one file in PKCS12 format). Use this option with care as the existing certificate and private key are replaced. This option is intended for restoring a backup of the current private key/certificate (for example, if the originals are damaged or destroyed) or for setting up a backup system.

Use <*file*> to specify the name of the file (in PKCS12 format) from which to import.

Use <*passphrase*> to specify the text string you use to protect the data. If you do not use the -pass option, you are prompted to enter the value of the pass phrase.

`-exportcacert -file <`*`file`*`>` [-pass <*passphrase*>]

Exports the certificate and the corresponding private key of the current certification authority to a file. This option is intended to be used for creating backups. The certification authority private key must be handled very carefully because of its importance to the whole communication environment. You should never transmit it over the network or store it in an insecure place.

Use <*file*> to specify the name of the file where the certificate data should be written to (in PKCS12 format).

Use <*passphrase*> to specify the text string you use to protect the data. If you do not use the -pass option, you are prompted to enter the value of the pass phrase.

-listPending [-l]

Displays the request IDs of all pending certificate requests.

With the `-l` option, detailed information on every pending request is listed.

`-grant <`*`reqid`*`>`

Grants the selected certificate request, and sends a signed certificate to the requesting certificate client.

Changes the state of the pending certificate request with the request ID <*reqid*> to `granted` .

`-deny <`*`reqid`*`>`

Denies the selected certificate request, and sends a message to the requesting certificate client.

Changes the state of the pending certificate request with the request ID <*reqid*> to denied .

`-remove <`*`reqid`*`>`

Removes the selected certificate request from the pending pool. No message is sent to the requesting certificate client.

Changes the state of the pending certificate request with the request ID <*reqid*> to removed .

`-issue -file <`*`file`*`> -name <`*`nodename`*`> [-pass <`*`passphrase`*`> ] [-coreid <`*`OvCoreId`*`> ] [-ca]`

Issues a signed certificate and the associated private key for a node, and writes both to the file <*file*> (in PKCS12 format). You can then move the file to a portable medium, and take it to the corresponding node.

You must specify the <*nodename*> as additional information.

You can specify the optional <*OvCoreId*> parameter to specify the unique ID of the certificate. If this parameter is empty, a new `OvCoreId` value is generated for the certificate.

The <*passphrase*> parameter is required to protect the generated certificate data. The pass phrase entered is used to calculate an encryption key that is then used to encrypt the generated certificate data. If you do not use the -pass option, you are prompted to enter the value of the pass phrase.

If you use the -ca option, you can use the issued certificate to sign other certificates. This may be necessary if you want to set up a second Certificate Server, which creates certificates that are trusted by all nodes that trust the root Certificate Server.

`-genInstKey -file <`*`file`*`> [-context <`*`context`*`> ] [-pass <`*`passphrase`*`> ]`

Creates a new installation key, which, together with some additional information, is stored in the file <*file*> . You should then transfer the created file securely to the node system.

On the target node, you can use the file to initiate a new certificate request encrypted with the installation key. The certificate server accepts only one request that is encrypted with this key.

The advantage of this approach is that you generate the certificate request (including the private key) on the node system, and can authenticate the system by using the installation key.

You can use the optional parameter <*context*> to add additional (application- specific) information that is contained in the certificate request.

The <*passphrase*> parameter is required to protect the generated installation key. The pass phrase you enter is used to calculate an encryption key, which is then used to encrypt the generated installation key. If you do not use the -pass option, you are prompted to enter the value of the pass phrase.

AUTHOR

ovcm was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`

All steps were successful.

`1`

One or more steps were not successful.

Corresponding error messages are written to stderror.

EXAMPLES

The following examples show how to use the ovcm command:

- To create a new public/private key pair for the signing of certificates on the management server system:

  `ovcm -newcacert`

- To grant the certificate request < *reqid* > and send a signed certificate to the requesting certificate client:

  `ovcm -grant <reqid>`

ovconfchg

NAME

ovconfchg

- manipulate settings files, update the configuration database, and trigger notification scripts.

SYNOPSIS

```
ovconfchg -h | -help
ovconfchg -version
ovconfchg [-ovrg <OVRG>] [-edit | -job {-ns namespace {-set <attr> <value> | -clear <attr> | -cl
```

DESCRIPTION

Installed HP Operations Manager 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.

The ovconfchg command manipulates the settings in either the system-wide configuration file or the configuration file for the specified HPOM Resource Group, `local_settings.ini` , updates the configuration database, `settings.dat` , and triggers notification scripts. If you call ovconfchg without options, or only with `-ovrg` , no settings are changed, but an update is triggered anyway. This update enables updating after default settings files have been added, removed, or updated.

When ovconfchg runs, all configuration settings are read and merged in memory. Default definitions are used to make corresponding checks, as well as to emit and log warnings in the event of a violation. During this process, file locks are used to prevent parallel updates. A new configuration database is then created containing the merged data.

Parameters

The ovconfchg command recognizes the following options:

-h | -help
    Displays all the options for the ovconfchg command.

-version
    Displays the version of the ovconfchg command.

-ovrg *<OVRG >*
    If the parameter you want to change belongs to an HPOM Resource Group, use `-ovrg` to specify the name of the resource group. Otherwise, system-wide settings files are opened.

-edit

Starts a text editor to edit the settings file, local_settings.ini. The text editor used is determined by the `$EDITOR` environment variable. If `$EDITOR` is not set, vi starts on UNIX, and Notepad starts on Windows.

A temporary copy of the file is created for editing. After you make changes, the file is validated for syntax errors. The syntax rule for validation is that the namespace and attribute names should contain only letters (a-z, A-Z), digits (0-9), period(.) and underscore(_) characters.

If the validation fails, the line number of the error is reported, and you are prompted to correct the file. If Yes, the file is reopened for making the necessary changes. If No, the original settings file remains unchanged. If the validation is successful, the changes are saved into the original settings file.

CAUTION:
Do not configure binary values using this option. This can corrupt the file. It is also recommended to restrict the data entered using this option to US-ASCII (7-bit only) subset.

Do not open the settings file directly in a text editor and change it. This can corrupt the file.

-job

Creates an update job file only. Does not synchronize.

`-ns | -namespace` <*namespace* >

Sets a namespace for the `-set` and `-clear` options.

-set <*attr* > <*value* >

Sets an attribute value in the namespace specified by the `-namespace` option. The local or HPOM resource settings file is updated accordingly.

`-clear` <*attr* >

Clears the local setting for the attribute *attr* in the namespace specified by the `-namespace` option. The local settings file is updated accordingly.

`-clear -all`

Clears all local settings. The local settings file is updated accordingly.

AUTHOR

ovconfchg was developed by Hewlett-Packard Company.

FILES

The `ovconfchg` command uses the following files to store local settings:

- < *DataDir* >/conf/xpl/config/local_settings.ini
- < *ShareDir* >/< *OVRG* >/conf/xpl/config/local_settings.ini

The ovconfchg command uses the following files to store database configuration settings:

- < *DataDir* >/datafiles/xpl/config/settings.dat
- < *ShareDir* >/< *OVRG* >/datafiles/xpl/settings.dat

EXAMPLES

The following examples show how to use the ovconfchg command:

- To assign the value 12 to the attribute COUNT , and assign the value "red blue white" to the attribute COLORS in the namespace, tst.lib :

  ovconfchg -ns tst.lib -set COUNT 12 -set COLORS "red blue white"
- To clear the attribute COUNT in the namespace tst.lib :

  ovconfchg -ns tst.lib -clear COUNT
- To remove all locally configured attributes from the namespace tst.lib :

  ovconfchg -ns tst.lib -clear '*'
- For the HPOM Resource Group server , assign the value 50 to the attribute COUNT in the namespace tst.lib :

  ovconfchg -ovrg server -ns tst.lib -set COUNT 50

SEE ALSO

ovconfget(1)

ovconfget

NAME

ovconfget

- return specified attributes from the configuration database.

SYNOPSIS

```
ovconfget  -h | -help
ovconfget  -version
ovconfget  [-ovrg <OVRG>] [<namespace> [<attr>]]
```

DESCRIPTION

Installed HP Operations components have associated configuration settings files that contain one or more namespaces, and apply system wide or for a specified HPOM 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.

Parameters

The `ovconfget` command recognizes the following options:

`-h | -help`
: Displays the options for the `ovconfget` command.

-version
: Displays the component version.

`-ovrg <` *`OVRG`* `>`
: Specifies the named HPOM Resource Group < *`OVRG`* > .

< *namespace* > < *attr* >
: Gets the specified attribute in the specified namespace for the named HPOM Resource Group < *`OVRG`* > , and writes them to stdout. If you use *namespace* without specifying an attribute, <*`attr`*> , `ovconfget` writes the contents of the database for the specified namespace. If you do not specify <*`attr`*> or <*namespace*> , `ovconfget` writes the complete contents of the configuration database to stdout.

AUTHOR

ovconfget was developed by Hewlett-Packard Company.

FILES

The ovconfget command uses the following files to read configuration-database settings:

- `<` *`DataDir`* `>/datafiles/xpl/config/settings.dat`
- `<` *`ShareDir`* `>/<` *`OVRG`* `>/datafiles/xpl/settings.dat`

EXAMPLES

The following examples show how to use the ovconfget command:

- To return the value of the `Port` attribute in the `tst.settings` namespace:

  ```
  ovconfget tst.settings Port
     9012
  ```

- To return all attributes in the `tst.settings` namespace as multiple lines in the form of *`attr`* `=`*`value`* :

  ```
  ovconfget tst.settings

  Port=9012

  Protocols=HTTP FTP HTTPS

  MaxFileSize=128
  ```

- To return all attributes in all namespaces on multiple lines:

  ```
  ovconfget

  [tst.lib]

  LibraryPath=/opt/OV/lib:/opt/OV/lbin/tst/var/opt/OV/tmp

  [tst.settings]

  Port=9012

  Protocols=HTTP FTP HTTPS

  MaxFileSize=128
  ```

SEE ALSO

ovconfchg(1)

ovconfpar

NAME

ovconfpar

- set and return configuration parameters remotely

SYNOPSIS

```
ovconfpar  -get [-host <hostname> [-targetid [<id>]...] -ovrg <OVRG> -ns <namespace> ]
ovconfpar  -change [-host <hostname> [-targetid [<id>]...] -ovrg <OVRG>] -ns <namespace> [ [-s
ovconfpar  -help
ovconfpar  -version
```

DESCRIPTION

The ovconfpar command reads and sets configuration parameters for installed HP Operations components. For information about the parameters you can use with the ovconfpar command, see "Parameters." For information about the options you can use with the ovconfpar command parameters, see "Options."

Parameters

The ovconfpar command recognizes the following parameters:

`-get <options>`
: Returns the value or values of one or more keys for the specified namespaces.

`-change <options>`
: Sets different key-value pairs for multiple namespaces.

`-version`
: Displays the version of the command.

`-help`
: Displays the help information.

Options

You can use the following options with the ovconfpar command parameters:

`-host <hostname> [-targetid <id> ]`
: Host name and target ID of the remote machine.

`-ovrg` *`<OVRG>`*

If the parameter you want to get or change belongs to an HPOM Resource Group, use `-ovrg` to specify the name of the resource group.

`-ns` *`<namespace>`*

Name of the namespace whose configuration parameters you want to get or change.

`-set <`*`attr`*`> <`*`value`*`> ...`

Sets the named attribute to the specified value for the specified namespace.

`-clear [<`*`attr`*`>] ...`

Clears the named attributes from the specified namespace. If no attribute is specified, all attributes are cleared for the specified namespace.

Return Codes

`ovconfpar` issues the following return codes:

`0`

All steps were completed successfully.

`-1`

One or more steps failed.

EXAMPLES

The following examples show how to use the `ovconfpar` command.

- To set the key ovo_port_range to 12345 in the namespace `ovo.server` :

  `ovconfpar -set -ns ovo.svr01 -set ovo_port_range 12345`

- To set the key ovo_port_range to 12345 in the namespaces `ovo.svr01` and `ovo.svr02` :

  `ovconfpar -set -ns ovo.svr01 -set ovo_port_range 12345 -ns ovo.svr02 -set ovo_port_range 12345`

- To set the key `MaxFileSize` to 128 and the key `Protocol` to HTTP in the namespace `ovo.svr01` :

  `ovconfpar -set -ns ovo.svr01 -set MaxFileSize 128 -ns ovo.svr01 -set Protocol HTTP`

- To display all keys and their values for all namespaces:

  `ovconfpar -g`

- To display the value for `MaxFileSize` in the `ovo.svr01` namespace:

```
ovconfpar -g -ns ovo.svr01 MaxFileSize
```

- To display the values in the `ovo.svr01` namespace:

```
ovconfpar -g -ns ovo.svr01
```

AUTHOR

`ovconfpar` was developed by Hewlett-Packard Company.

SEE ALSO

ovappinstance(1) , ovclusterinfo(1) , ovpolicy(1) .

ovcoreid

NAME

ovcoreid

- manage the unique node identifier `OvCoreId` on the local node.

SYNOPSIS

```
ovcoreid -show [-ovrg <OV_Resource_Group>]
ovcoreid -create [-force] [-ovrg <OV_Resource_Group>]
ovcoreid -set <OvCoreId> [-force] [-ovrg <OV_Resource_Group>]
ovcoreid -version
ovcoreid -h|-help
```

DESCRIPTION

You can use the ovcoreid command to display existing `OvCoreId` values, and to create and set new `OvCoreId` values on the local node.

Parameters

The ovcoreid command accepts the following parameters and options:

`-show [-ovrg <` *`OV_Resource_Group`* `>]`

Displays the current `OvCoreId` of the system (configuration setting `CORE_ID` in namespace `[sec.core]`). This is the default if you do not specify any parameters. If the `OvCoreId` you want to show belongs to an HPOM Resource Group, use the `-ovrg` option to specify the name of the HPOM Resource Group. If you specify an HPOM Resource Group, the corresponding configuration settings are read or modified as well.

If you specify a non-existent HPOM Resource Group, ovcoreid displays the local `OvCoreId` .

`-create [-force] [-ovrg <` *`OV_Resource_Group`* `>]`

Generates a new `OvCoreId` . If a `CORE_ID` value already exists, the existing `OvCoreId` is overridden only if you specify `-force` . If the `OvCoreId` you want to show belongs to an HPOM Resource Group, use the `-ovrg` option to specify the name of the HPOM Resource Group. If you specify an HPOM Resource Group, the corresponding configuration settings are read or modified as well.

If you specify a non-existent HPOM Resource Group, ovcoreid displays an error.

`-set [-force] [-ovrg <` *`OV_Resource_Group`* `>]`

Sets a specific `OvCoreId` . You must use the `-force` option if an `OvCoreId` value has already been set.

If the `OvCoreId` you want to show belongs to an HPOM Resource Group, use the `-ovrg` option to specify the name of the resource group. If you specify an HPOM Resource Group, the corresponding configuration settings are read or modified as well.

`-version`

Returns the version of the tool (the component version).

`-h|-help`

Display all available command options.

AUTHOR

`ovcoreid` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`

All steps were successful.

`1`

If you use `-create` or `-set` without `-force` and a value for `OvCoreId` already exists.

`2`

One or more steps were not successful.

Corresponding error messages are written to stderror.

CAUTION:
Changing the `OvCoreId` of a system is analogous to giving the system a new identity. You should execute this action only if you fully understand the consequences. Changing the `OvCoreId` of a system requires a number of significant changes, including a new certificate and reconfiguration of HPOM servers.

EXAMPLES

The following examples show you how to use the `ovcoreid` command:

- To display the `OvCoreId` for the local node:

  `ovcoreid -show`

- To create and set a new `OvCoreId` on the local node:

  `ovcoreid -create`

- To set the specified `OvCoreId` on the local node:

  ovcoreid -set < ***OvCoreId*** **>**

SEE ALSO

ovconfget(1) , ovconfchg(1) .

ovcreg

NAME

ovcreg

- component registration tool

SYNOPSIS

```
ovcreg -h|-help
ovcreg -check [<filename>]
ovcreg -add [<filename>]
ovcreg -del [<component>]
ovcreg -version
```

DESCRIPTION

You use the `ovcreg` command to register a component with (and de-register the component from) OvCtrl. You can also use `ovcreg` to check a component registration file for syntactical correctness.

If the OvCtrl daemon (ovcd) is running at the time of registration, it is informed about the new component only if you applied the -add option and if you did not start the component. OvCtrl shows the new component the next time the ovc command is called with the `-status` option.

If the OvCtrl daemon (ovcd) is running, the component is stopped if you applied the `-del` (delete) option. Note that this option does *not* stop CORE components, which are denoted by the option CoreProcess in the registration file. You can stop CORE components with ovc command and the `-kill` option.

Parameters

The `ovcreg` command recognizes the following options:

`-h|-help`
: Displays *all* available options for the `ovcreg` command.

`-check [<` *`filename`* `>]`
: Checks the syntax of < *`filename`* > . The < *`filename`* > option may not contain more than one component.

`-add [<` *`filename`* `>]`
: Checks the syntax of < *`filename`* > , and stores a copy in the configuration directory. Adding a component with a name which is already registered with OvCtrl overwrites the original registration with the new one. The < *`filename`* > option may not contain more than one component.

`-del [<` *`component`* `>]`
: Stops and de-registers the specified < *`component`* > from OvCtrl, and deletes the specified < *`component`* > registration file. Note that the delete option does not stop CORE components.

`-version`
: Displays the version of `ovcreg` .

AUTHOR

`ovcreg` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
: Success. The syntax of the file is correct, and the registration file is successfully added or deleted.

`1`
: Wrong usage.

`2`
: Parsing error.

`3`
: Error deleting registration file.

`5`
: Error writing XML file.

`6`
: Component is not registered.

`7`
: Error stopping component.

`8`
: Error deleting component.

FILES

Registration files for components registered with OvCtrl for the supported platforms reside in the following locations:

- AIX, HP-UX, Linux, Solaris:

  `/var/opt/OV/conf/ctrl/*.xml`
- True64:

  `/usr/var/opt/OV/conf/ctrl/*.xml`
- Microsoft Windows:

  `C:\Program Files\HP\HP BTO Software\conf\ctrl\*.xml`

Users can change the specified default location for the registration files on machines running Microsoft Windows.

EXAMPLES

The following examples show how to use the ovcreg command and some of its options to control and display important information about registered components:

- To check the syntax of the component registration file `opcle.xml`:

  `ovcreg -check opcle.xml`
- To check the syntax of the component registration file `opcle.xml` , and add the component defined in the file to OvCtrl:

  `ovcreg -add opcle.xml`
- To stop and de-register the component registered as `opcle` :

  `ovcreg -del opcle`

SEE ALSO

ovc(1)

ovlogdump

NAME

ovlogdump

- dump a specified binary log file as text in the current locale to the console.

SYNOPSIS

```
ovlogdump -h|-help
ovlogdump -version
ovlogdump [<binary_logfile_name>]
ovlogdump -merge -tofile <binary_logfile_name> -fromfiles <binary_logfile1_name> <binary_logfile2_nan
```

DESCRIPTION

The ovlogdump command dumps a binary log file as text in the current locale to the console. To view the contents of a log file, specify its location and name. Otherwise, the `system.bin` file is dumped to the console by default.

By default, all the log files are stored in the following location:

- Windows

  `C:\Documents and Settings\All Users\Application Data\HP\HP BTO Software\log`
- UNIX

  `/var/opt/OV/log`

If permissions are inadequate for the default locations, the log files are stored in the <*OvDataDir* >`/log/public` directory.

During application logging, if multiple log files are created, you can use the -merge option to merge these files into a single binary log file.

Parameters

ovlogdump recognizes the following options:

[<*binary_logfile_name*>]

Name and location of the binary log file to be dumped. If you do not specify the log file name, `system.bin` file in the <*OVDataDir* >`/log/` directory displays on the console by default.

-merge -tofile < *binary_logfile_name* > -fromfiles < *binary_logfile1_name* > <

`*binary_logfile2_name* >....`

Merges application log files specified by < `*binary_logfile1_name*` > .... into a single binary log file specified by <`*binary_logfile_name*` >. This option is not supported for merging system log files.

`-h|-help`

Displays all available options for the `ovlogdump` command.

`-version`

Displays the version of the `ovlogdump` command.

AUTHOR

`ovlogdump` was developed by Hewlett-Packard Company.

ovoreqcheckagt

The tool ovoreqcheckagt enables local HP Operations agent prerequisite checking from the command line on the node. It parses requirements from the local configuration file, checks the node, and prints out the report in either a detailed or summarized format. The default location of the configuration file is the current directory. You can specify a custom file with `-cfg <config_file>` .

UNIX-specific Information

The tool consists of three files: ovoreqcheckagt, ovoreqcheckagt.awk, and ovoreqcheck.cfg. These files must be transferred from the agent package directory on the server (for example, `%OvShareDir%\packages\HPUX\11IA`) to the node where the agent prerequisites will be checked. The ovoreqcheckagt script needs to be made executable with the command `chmod +x ./ovoreqcheckagt` .

Windows-specific Information

The tool consists of two files: ovoreqcheckagt.exe and ovoreqcheck.cfg. These files must be transferred from the agent package directory on the server (for example, `%OvShareDir%\data\packages\Windows\Windows_manual`) to the node where the agent prerequisites will be checked.

Command Synopsis

```
ovoreqcheckagt (-det | -sum) -agt_comm_type <agent_comm_type>
               -agt_bin_format <agent_bin_format> [-cfg <config_file>]
ovoreqcheckagt -req <system> [-agt_comm_type <agent_comm_type>]
               [-agt_bin_format <agent_bin_format>] [-cfg <config_file>]
ovoreqcheckagt -allsystems [-cfg <config_file>]
ovoreqcheckagt -ver
ovoreqcheckagt -help
```

`-help`	Prints tool usage and description.
`-det`	Prints detailed result of requirements and recommendations validation. All prerequisites are displayed regardless if they have passed or failed.
`-sum`	Prints summary result of requirements and recommendations validation. Only failed prerequisites are displayed.
`-req <system>`	Shows requirements and recommendations for the specified operating system.
`-allsystems`	Prints the list of all supported systems.
`-agt_comm_type <agent_comm_type>`	Specifies the agent communication type for this node. If you specify `-det` or `-sum` , this parameter is required. Valid values for `<agent_comm_type>` are `DCE` and `HTTPS` .
`-agt_bin_format <agent_bin_format>`	Specifies the agent binary format for this node. If you specify `-det` or `-sum` , this parameter is required. Valid values for `<agent_bin_format>` are as follows: ■ `Alpha` ■ `IPF32` ■ `IPF64` ■ `PA-RISC` ■ `PowerPC` ■ `SPARC` ■ `x64` ■ `x86`
`-cfg <config_file>`	Specifies a custom configuration file (default file is `.\ovoreqcheck.cfg`).
`-ver`	Prints the version of the ovoreqcheckagt tool.

Exit Values

Exit value indicates prerequisite checking status:

0 - All checked prerequisites (requirements and recommendations) are OK.

1 - All requirements are OK; at least one recommendation has failed.

2 - All requirements are OK; at least one recommendation could not be checked.

3 - At least one requirement has failed.

55 - No prerequisites specified for <system> system.

101 - Client does not have administrative privileges on the node. Prerequisites can be checked only with administrative rights.

102 - At least one requirement could not be checked (it is unknown whether requirement is OK or not); all other successfully checked requirements are OK.

103 - Checked node is not available; either there is no network connection or firewall ports are not opened.

104 - Node (name) cannot be resolved

105 - The platform/OS version on node xxx may not yet be supported - consult the latest support matrix ; if platform is supported ignore prerequisite check and check prerequisites manually.

106 - Node’s platform properties (System Type, Operating System, Version) are not set.

107 - Cannot discover platform.

ovpolicy

NAME

ovpolicy

- install, manage, and remove local and remote policies.

SYNOPSIS

```
ovpolicy -help
ovpolicy -version
ovpolicy -install [-host <hostname> [-targetid [<id>]...]{-enabled|-disabled}
-chkvers -add-category [<cat1>]... {-remove-category [<cat>]...
|-remove-all-categories} -force-cat -add-attribute [<name> <value>]... -remove-attribute [<nam
-force-attr -set-owner <owner> -force-owner -no-notify]
{-file [<file>]...|-dir [<dir>]...} [-ovrg <ov_res_group>]
ovpolicy -remove [-no-notify -host <hostname> [-targetid [<id>]...] [-ovrg <ov_res_group>] <SELI
ovpolicy [-enable |-disable] [-no-notify -host <hostname> [-targetid [<id>]...] [-ovrg <ov_res_
ovpolicy [-addcategory |-removecategory] <cat>... [-no-notify -host <hostname> [-targetid [<id>]
ovpolicy -removeallcategories [<cat>]... [-no-notify -host <hostname> [-targetid [<id>]...] [-ov
ovpolicy [-addattribute |-removeattribute] <name> <value>... [-no-notify -host <hostname> [-tar
ovpolicy -removeallattributes [-no-notify -host <hostname> [-targetid [<id>]...][-ovrg <ov_res_
ovpolicy [-setowner | -removeowner <owner>] [-no-notify -host <hostname> [-targetid [<id>]...][
ovpolicy -notify [-host <hostname> [-targetid [<id>]...][-ovrg <ov_res_group>]]
ovpolicy -list [-level <0|1|2|3|4> -host <hostname> [-targetid [<id>]...][-ovrg <ov_res_group>]]
```

DESCRIPTION

ovpolicy installs, manages, and removes, local and remote policies. A policy is a set of one or more specifications rules and other information that help automate network, system, service, and process management. You can deploy policies to managed systems, providing consistent, automated administration across the network. You can group policies into categories (for example, to assign policies to a special policy group for simple enable and disable actions). Each category can have one or more policies. Policies can also have one or more attributes, an attribute being a name-value pair.

You can use ovpolicy to install, remove, enable, and disable local policies, as well as perform other functions. For information about the parameters supported by the ovpolicy command, see "Parameters." For information about parameter options, see "Options."

Parameters

ovpolicy recognizes the following parameters:

`-install`

Installs one or more policies using a single policy file specified with `-file` or multiple policy files specified with `-dir` .

`-remove`

Removes one or more policies.

`-enable`

Enables one or more policies.

`-disable`

Disables one or more policies. Note that the `-disable` option only disables a policy, it does not remove a policy from the file system.

`-addcategory`

Adds all category strings to the policy. You can add multiple categories using a blank-separated list.

`-removecategory`

Removes the specified category strings from the policy. You can remove multiple categories using a blank-separated list.

`-removeallcategories`

Deletes *all* categories.

`-addattribute`

Adds a category attribute to the policy. You can add multiple attribute names using a blank-separated list.

`-removeattribute`

Removes category attribute from the policy. You can remove multiple attribute names using a blank-separated list.

`-removeallattributes`

Deletes *all* category attributes.

`-setowner`

Sets the owner of a policy.

`-removeowner`

Removes the owner of a policy.

`-list`

Lists the installed policies.

`-notify`

Triggers any notifications to the HP Operations control service, if there are any outstanding or suppressed notifications from previous policy operations.

`-version`

Displays the version number of the command.

`-h | -help`

Displays the help information.

Options

You can use the following options with the allowed `ovpolicy` command parameters:

`-add-attribute`

Adds an attribute < *name* > with the value defined in < *value* > to the specified installed policy.

`-add-category` *<cat1>* [*<cat2> ... <catN>*]

Adds all category strings to the policy. This is a blank-separated list.

`-chkvers`

Checks and compares the version of the already installed policy and the policy you want to install. If yo use `-chkvers` , the new policy is not installed if the current installed version is the same or higher. If you do *not* use `-chkvers` , the new policy overwrites the current policy with the same `policy_id` , regardless of the version number. The `-chkvers` command parameter does not overwrite the categories, owner, or status of a current policy. To overwrite the categories, owner, and status associated with a policy owner, use `-forcecat` and `-forceowner` respectively.

`-dir` < *dirname*>

If you specify a directory name, all policy files from that directory are used. A line is printed to `stdout` for each successfully installed policy.

`-enabled|-disabled`

If you use `-enabled` or `-disabled` , the new policy acquires the status that is defined in the policy header. If you do *not* use `-enabled` or `-displayed` , the new policy acquires the status of the currently installed policy (if any).

Note that this option overwrites the status defined in the policy-header installation file. If the new policy is already installed on the target system, the new version assumes the status of the installed version.

`-file <`*`filename`*`>`

Specifies a policy file name to be used. A line is printed to `stdout` for the successfully installed policy.

`-force-attr`

Enables you to remove category attributes that are set on a current installed policy. By default, the attributes from current installed policies are used. If there is no current installed policy, the attributes set in the header file of the new policy are used.

`-force-cat`

Enables you to remove categories that are set on a current installed policy. By default, the categories from current installed policies are used. If there is no current installed policy, the categories set in the header file of the new policy are used.

`-force-owner`

Overwrites the policy owner regardless of the settings for the installed policy.

`-host <` *`hostname`* `> [-targetid <` *`ids`* `>]`

Specifies the hostname of the managed node. If no hostname is specified, the local host is assumed. `-targetid` specifies one or more target IDs.

`-level`

Specifies the type of information to be returned with the `-list` parameter, as follows:

`0`

Policy type, policy name, status, policy version. This is the default setting.

`1`

Policy type, policy name, status, policy version, policy_ID.

`2`

Policy type, policy name, status, policy version, policy_ID, category.

`3`

Policy type, policy name, status, policy version, policy_ID, category, owner.

`4`

Policy type, policy name, status, policy version, policy_ID, category, owner, attributes.

`-no-notify`

If you use `-no-notify` , `ovpolicy` does not trigger any notifications.

`-remove-category` *<cat1>* [*<cat2> ... <catN>*]
: Removes the specified category strings from the policy. Using the `-remove-category` option with an empty string deletes *all* categories. This is a blank-separated list.

`-remove-all-categories`
: Removes the specified category strings from the policy.

`-remove-attribute`
: Remove the category attribute <*name*> with the value defined in <*value*> from the specified installed policy.

`-remove-all-attributes`
: Allows you to remove *all* category attributes that are set on a current installed policy. If there is no current installed policy, the attributes set in the header file of the new policy are used.

`-set-owner` < *owner* >
: Sets the owner of a policy. Using `-set-owner` with an empty string deletes the owner.

`-ovrg` < *ovrg_res_group* >
: Sets the name of the HPOM Resource Group.

The <*SELECTION*> option is one of the following:

< *SELECTION* >-all|-owner < *owner* >|-owner < *owner* > -polname < *name* >|-polid < *uuid* > |-polname <[*type* :] *name* >|-poltype < *typename* >|-category < *category*> |-attribute < *name* > [value]

`-all`
: All installed policies.

`-owner` <*owner*>
: Policy owner <*owner*> .

`-owner` <*owner*> *-polname* < *name*> .
: Policy owner <*owner*> and the policy name -owner < *name*> .

`-polid` <*id*>
: ID of the policy.

`-polname` [<*policy_type_name*> :]< *policy name*>
: Name of the policy. If you use *policy_type_name* , the section applies to all policies of the specified type.

`-poltype` *<policy_type_name>*
: Name of the type of policy.

`-category` *<category_name>*
: Name of the category to be used.

`-attribute` *<name>* *<value>*
: Name of the policy attribute and value to be used.

Return Codes

`ovpolicy` recognizes the following return codes:

`0`
: All steps were successful.

`1`
: One or more steps were not successful.

AUTHOR

`ovpolicy` was developed by Hewlett-Packard Company.

EXAMPLES

The following examples show you how to use the `ovpolicy` command:

- To list all policies on a node:

  `ovpolicy -list`
- To disable the HP-UX `syslog` policy:

  `ovpolicy -disable -polname "HPUX ovsyslog" -host` *<HPUX_hostname>*
- To enable all trap policies:

  `ovpolicy -enable -poltype ovsnmptrap`
- To install all policies located in the current working directory:

  `ovpolicy -install -dir .`
- To install all policies located in the `/tmp/sap_policies` directory with a status of disabled:

  `ovpolicy -install -disable -dir /tmp/sap_policies`
- To reinstall all policies located in the `/tmp/xyz` directory, independent of the former owner:

  `ovpolicy -install -forceowner -dir /tmp/xyz`

- To remove all policies from the local host:

  ```
  ovpolicy -remove -all
  ```

- To remove all installed policies that are owned by the management server :

  ```
  ovpolicy -remove -owner mgtsvr
  ```

ovrc

NAME

ovrc

- perform actions on *remote* components.

SYNOPSIS

```
ovrc  -h|-help
ovrc  -host <name_or_ip> [-tid <ids>] -start [<target> ... ]
ovrc  -host <name_or_ip> [-tid <ids>] -stop [<target> ... ]
ovrc  -host <name_or_ip> [-tid <ids>] -restart [<target> ... ]
ovrc  -host <name_or_ip> [-tid <ids>] -status [<target> ... ] [-level <level>]
ovrc  -host <name_or_ip> [-tid <ids>] -notify <event> [<target> ...] [-value <value>]
ovrc  -version
```

DESCRIPTION

The `ovrc` command controls the starting, stopping, event notification, and status reporting of all components on remote hosts.

A component can be a server process belonging to any of the products, such as HP Operations Manager (HPOM), HP Operations agents (for example, the Performance Agent or the Discovery Agent), an event interceptor, or an application delivered by an integrator. Each component must have an associated registration file (see ovcreg(1)) providing HP Software with configuration and process information about the component.

In the `-host` option, you can specify the fully qualified hostname or IP address of the remote node. If you do not specify the hostname or IP addressd, the requested operation is performed on the local node, where the command is run.

To perform an action, a target can be either a component or a group of components, defined as a category. The `ovrc` command first tries to run the requested operation on the category specified in *`target`* . If the category called *`target`* is not found, `ovrc` tries to start the individual component *`target`* . A category name must not match any component name. You can also use the `-tid` option to specify a target ID (CORE ID) of the remote host specified in the `-host` option. The CORE ID is a unique identification for a node. You can use the CORE ID with the `-tid` option to make sure that the remote host that receives the request is correct.

Parameters

The `ovrc` command recognizes the following options:

`-h|-help`

Displays *all* available options for the ovrc command.

`-start [<`*`target`*`> ... ]`

Starts the selected components. The <*target*> option specifies a component or category. If you do not use <*target*> , all components are started.

`-stop [ <`*`target`*`> ... ]`

Stops the selected components. The <*target*> option specifies a component or category. If you do not use <*target*> , all components are stopped except components which are CORE processes.

`-restart [<`*`target`*`> ... ]`

Stops components before they are restarted. If you do not use <*target*> , all components are stopped and restarted.

`-notify <`*`event`*`> [<`*`target`*`> ... ] [-value <`*`value`*`>]`

Sends notification of an event with the value of <*value*> to the component or category specified by <*target*> . If you do not use <*target*> , the event notification is sent to all components. If you do not use <*value*> , only the event notification is sent.

`-status [ <`*`target`*`> ... ] [-level <`*`level`*`> ]`

Reports the status of a component or category specified by <*target*> . The status report contains the component's label, description, category, process ID, and STATE. Components can be in one of the following states: Stopped (0 in numeric format), Starting (1), Initializing (2), Running (3), Stopping (4), N/A (5), or Aborted (6). If you do not use <*target*> , the status of *all* components is returned.

The <*level*> option specifies the type and quantity of information to display, as follows:

`Level 0`

Status of registered components monitored by HPOM.

`Level 1`

Status of registered components, whether they are monitored by HPOM or not.

`Level 2`

Status of registered components and a dump of their registration information.

`Level 3`

ID of core processes. Zero (0) indicates root, non-zero indicates non-root ownership.

`Level 4`

Similar to level 0, but the STATE is reported in numeric format.

`Level 5`

Similar to level 1, but the STATE is reported in numeric format.

`Level 6`

Similar to level 0, but the output is not formatted

`Level 7`

Similar to level 1, but the output is not formatted

`-version`

Prints the version of `ovrc` .

AUTHOR

`ovrc` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`

Success

`1`

Not defined

`2`

Ignored

`64`

Generic error

`65`

Invalid target

`67`

Operation aborted

`69`

Missing prerequisite

`70`

Authorization error

71

Operation on prerequisite failed

73

Invalid event

EXAMPLES

The following examples show how to use the `ovrc` command and some of its options to control and display important information about registered components:

- To display the status of category SERVER on a remote system with the hostname `mach.hp.com` :

  `ovrc -host mach.hp.com -status SERVER`

- To stop the component registered as `opcle` on a local host:

  `ovrc -stop opcle`

  Before `opcle` itself stops, components that depend on `opcle` are stopped.

SEE ALSO

ovc(1) , ovcreg(1) .

ovswitchuser

NAME

ovswitchuser

- run the HP Operations Manager agent processes under a non-administrative account (not the root account).

SYNOPSIS

```
ovswitchuser  -h | -help
ovswitchuser  -v | -version
```

NOTE:
Stop all HP Operations processes on the system before applying an ovswitchuser command to change the user account that you want to use for the HP Operations processes.

Use the following command:

`ovc -kill`

UNIX only:

`ovswitchuser.sh  -existinguser <userName> | -existinguserID <userID> -existinggroup <groupName`

Windows only:

NOTE:
On Windows, the user that is specified with ovswitchuser needs to have the `Log On as a Service` permission.

`cscript ovswitchuser.vbs  -existinguser <userName> | -existinggroup <groupName> | [-passwd <pa`

DESCRIPTION

By default, the HP Operations core processes run under the root/administrator account. The ovswitchuser command allows you to run the HP Operations processes under a non-administrative account. The group ownership of all registered HP Operations component product files and directories of < *`OVDataDir`* > is changed. The specified user is added to the group, and the core HP Operations processes are started under this user account. Boot scripts are changed to allow daemons and services to run under non-root, non-administrative accounts, and to modify the operating-system-specific registration of daemons and services, so HP Operations processes start under the specified user.

The ovswitchuser command also stores information about the specified group in the HP Operations configuration file.

The non-root concept relies on the user, under which the agent runs, belonging to a specific UNIX group. As a result, you must set the group bits of any files that are created by HP BTO Software applications. This allows HP BTO Software 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 unmask appropriately for the users that are used to run HP BTO Software applications.

An unmask setting of 02 is preferable. The setting 022 causes problems when multiple applications are run under different users.

If all HP BTO Software applications run under the same user, the unmask setting is not required.

If the HP Operations Communication Broker is running, the port that it uses must be 1024 or greater, or you must set the switchuser bit to ovbbccb . You may need to change the port number on both communication systems. For exact details, refer to product documentation.

To check if the Communication Broker is running, execute the following command:

```
/opt/OV/bin/ovc -status
```

It is running if there is the following entry:

```
ovbbccb          OV Communication Broker    CORE    (****)    Running
```

If the Communication Broker is running, set the port number to 1024 or greater, or set the switchuser bit to ovbbccb .

Example command for the node mynode :

```
ovconfchg -ns bbc.cb.ports -set PORTS mynode:1024
```

For further details, refer to the Communication Broker man pages.

It is also recommended that you specify the domain for the system.

Example:

```
ovconfchg -ns bbc.http -set DOMAIN mydomain.com
```

For further information, refer to the ovconfchg(1) , ovconfget(1) , bbc.ini(4) , and ovbbccb(1) man pages.

CAUTION:
Usage restrictions and further considerations may apply, depending on the HP BTO product being used. Some HP BTO products must be run under the root/administrative account. Do not use the ovswitchuser functionality in these environments. Before attempting to change the user account with the ovswitchuser tool, refer to the product documentation.

Parameters

ovswitchuser recognizes the following options:

-h | -help
: Displays the options for the ovswitchuser command.

-version
: Displays the version number of the cross platform component.

-existinguser < *userName* >
: Specifies an existing user <*userName*> who can run the HP Operations processes.

-existinguserID < *userID* >
: *UNIX only* : Specifies an existing user <*userID*> under which to run the HP Operations processes.

-existinggroup < *groupName* >
: Specifies an existing group <*groupName*> that can run the HP Operations processes. The <*userName*> specified with the -existinguser parameter is added to this group if the <*userName*> does not belong to this group.

-existinggroupID < *groupID* >
: *UNIX only* : Specifies an existing group <*groupID*> under which to run the HP Operations processes.

[-passwd < *passwd* >]
: *Microsoft Windows only* : If you use the -passwd option to specify the password of the user < *userName* > defined in -existinguser , the password is used as a logon for the HP Software services, which are started. For security reasons, a password is required to start the HP Software services. So, if you choose not to specify a password here, you have to enter the password manually in the Services dialog when you start the HP Software services after the ovswitchuser command completes.

-setgroup < *package* >
: Sets group ownership for the specified package defined in the XPL configuration.

AUTHOR

ovswitchuser was developed by Hewlett-Packard Company.

EXAMPLES

To set ownership of all the installed package files to the group defined in < *groupName* >=OV_group and the user defined in < *userID* >=1000 :

ovswitchuser.sh -existinguserID 1000 -existinggroup OV_group

SEE ALSO

ovconfchg(1) , ovconfget(1) , bbc.ini(4) , ovbbccb(1) .

Server Command-Line Utilities

These utilities can be executed on the HPOM management server.

IOvReqCheck

The IOvReqCheck interface provides the following methods.

Method	UseManagementServer

Method declaration	`HRESULT UseManagementServer(`
	`[in]BSTR Server)`

Description	This method must be called before any other method provided by the IOvReqCheck interface. It sets the management server that should be used. NOTE: The console can connect to a different management server, so it is necessary to specify which management server should be used. Internal: Sets the management server name to be used by all other methods. It also transfers the prereq.check config file from the management server to the console if a different (newer or older) version exists on the management server. The config file is located here on the console: <HPOM for Windows data dir>\tmp Access to the config file on the console must be protected with global mutex(CreateMutex(lpMutexAttributes,FALSE,"Global\PrereqCheckConfigFile")). The file can be updated only when it is not used by the OvOWReqCheck component.
Method	**CheckNode**

Method declaration	`HRESULT CheckNode(`
	`[in]BSTR Node,`
	` [out]VARIANT* Report,`
	` [out]VARIANT* ResultDescription,`
	` [in]BSTR Platform,`
	` [in,optional,defaultvalue (RP_TXT)]ReportFormat rpFormat,`
	` [in,optional,defaultvalue(RP_REQ_FAILED)] ReportType rpType,`
	` [in,optional,defaultvalue ("")]BSTR User,`

	`[in,optional,defaultvalue ("")]BSTR Password,`
	`[out,reval]long* Result)`

Description	It checks prerequisites for the node specified with the Node parameter (primary node name or IP address). With the platform parameter, we can define the platform of the checked node. Valid values are "UNIX" and " WINDOWS." The rpType parameter specifies how detailed the report should be. ■ RP_REQ_REC_INF: all prerequisites (requirements, recommendations) together with information data are returned. Each prerequisite has associated state (PASS,FAIL,UNCHECK). Each failed prerequisite has additional error description. ■ RP_REQ_REC: all prerequisites (requirements, recommendations) are returned. Each prerequisite has associated state (PASS,FAIL,UNCHECK). Each failed prerequisite has additional error description. ■ RP_REQ_REC_FAILED: only prerequisites (requirements, recommendations) that have failed are returned. Each prerequisite has associated state (FAIL,UNCHECK). Each failed prerequisite has additional error description. ■ RP_REQ: all requirements are returned. Each requirement has associated state (PASS,FAIL,UNCHECK). Each failed requirement has additional error description. ■ RP_REQ_FAILED: only requirements that have failed are returned. Each requirement has associated state (FAIL,UNCHECK). Each failed requirement has additional error description. *rpFormat* parameter specifies format of report: ■ RP_XML: report is generated in XML format (appropriate for further processing) ■ RP_TXT: report is generated in text format (appropriate for directly displaying to the user) The prerequisite check report (list of requirements/recommendations with the status) is returned as a string with Report parameter. The content/format of the report is determined with rpFormat and rpType parameters. If User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. The call is routed to the `OvOWReqCheckSrv` component on server which actually checks requirements. If the user and Password are not provided, the`OvOWReqCheck` component

impersonates the client and contacts the node directly.

The status of prerequisite checking is returned in the Result parameter:

PREREQUISITES CHECKED:

- ovrcResAllPrereqOK (0) – All checked prerequisites are OK
- ovrcResReqOKRecFailed (1) - All requirements are OK; at least one recommendation has failed
- ovrcResReqOKRecNotCheked (2) - All requirements are OK; at least one recommendation could not be checked
- ovrcReqFailed (3) – At least one requirement has failed
- ovrcResNoPrerequisitesSpecified (55) – No prerequisites specified for <system> system.

PREREQUISTES WERE NOT CHECKED:

- ovrcResReqCheckFailed (100) – not real return code; it is just a code to compare against to check whether check completed of failed;

  Result < ovrcResReqCheckFailed à prerequisite check completed

  Result > ovrcResReqCheckFailed à prerequisite check failed
- ovrcResNoAdminRight (101) – Client does not have administrative privileges on the node. Prerequisites can be checked only with administrative rights.
- ovrcResOneReqNotChecked (102) – At least one requirement could not be checked (it is unknown whether requirement is OK or not); all other successfully checked requirements are OK
- ovrcResNodeNotAvailable (103) – Checked node is not available; either there is no network connection or firewall ports are not opened
- ovrcResNodeNotExists (104) - Node (name) cannot be resolved
- ovrcResPlatformNotSupported (105) - Platform is not supported (Desc for the user: The platform/OS version on node xxx may not yet be supported - consult the latest support matrix ; if platform is supported, ignore the prerequisite check and check prerequisites manually)
- ovrcResPlatformNotDefined (106) – Node's platform properties (System Type, Operating System, Version) are not set.
- ovrcResPlatformNotDiscovered (107) – Cannot discover platform.

Textual description of Result code is returned in `ResultDescription` parameter.

Note: Returned status is 2 if at least one requirement fails and if any other requirement could not be checked.

HRESULT:

	ovrcNoError(S_OK) – success ovrcErrMgmtServerNotSet-SetManagementServer method was not called ovrcErrConfigFileSyntaxError - "Syntax error in line <line>:<err_descr>"
Method	CheckNodeOv

Method declaration	`HRESULT CheckNodeOv(`
	`[in]BSTR Node,`
	`  [in]long SystemType,`
	`  [in]long OsType,`
	`  [in]BSTR OSVersion,`
	`  [in, out]BSTR* Report,`
	`  [out]VARIANT* ResultDescription,`
	`  [in,optional,defaultvalue(RP_TXT)ReportFormat rpFormat,`
	`  [in,optional,defaultvalue(RP_REQ_FAILED)] ReportType rpType,`
	`  {in,optional,defaultvalue(“”)]BSTR User,`
	`  [in,optional,defaultvalue(“”)]BSTR Password,`
	`  [out,retval]long* Result)`

Description	It checks prerequisites for the node specified with the Node parameter (primary node name or IP address). Prerequisites are checked according to provided HPOM for Windows node properties (SystemType, OSType, OSVersion). HPOM for Windows node properties directly relate to the [OVM] tag in the config file. For a description of parameters and return codes, see the CheckNode method.
Method	CheckNodeOv2

Method declaration	`HRESULT CheckNodeOv2(`
	
	`  [in]BSTR NodeID,`
	`  [out]VARIANT* Report,`
	`  [out]VARIANT* ResultDescription,`
	`  [in,optional,defaultvalue(RP_TXT)] ReportFormat rpFormat`
	`  [in,optional,defaultvalue(RP_REQ_FAILED)] rpType,`
	`  [in,optional,defaultvalue(“”)]BSTR User,`
	`  [in,optional,defaultvalue(“”)]BSTR Password,`
	`  [out,retval]long* Result)`

Description	It checks prerequisites for a node specified with the NodeID parameter. Prerequisites are checked according to HPOM for Windows node properties (StystemType, OSType, OSVersion). HPOM for Windows node properties directly relate to the [OVM] tag in the config file.

	If all requirements are OK (recommendations are not important), the method sets the node's PrerequisitesOkay property to TRUE. For a description of parameters, see the CheckNode method.
Method	GetRequirements

Method declaration	`HRESULT GetRequirements(`
	
	`  [in]BSTR System,`
	`  [in,optional,defaultvalue (RP_TXT)] ReportFormat rp Format,`
	`  [out, retval]BSTR* Requirements)`

Description	It returns the list of prerequisites for a specified system as a string formatted (XML/TXT) according to the rpFormat parameter. (System parameter directly relates to the [VER] tag in the config file. ERROR CODES: ovrcErrMgmtServerNotSet – SetManagementServer method was not called ovrcErrSystemNotSupported – Specified system is not supported (not found in config file – VER tag)
Method	GetAllRequirements

Method declaration	`HRESULT GetAllRequirements(`
	
	`  [in,optional,defaultvalue(RP_TXT)] Report Format rpFormat,`
	`  [out, retval ]BSTR* Requirements)`

Description	It returns the list of prerequisites for all supported systems. Format (XML/TXT) of the report is defined with the fpFormat parameter. HRESULT: ovrcErrMgmtServerNotSet - SetManagementServer method was not called ovrcErrSystemNotSupported – Specified system is not supported (not found in config file – VER tag)
Method	GetSupportedSystems

Method declaration	`HRESULT GetSupportedSystems(`
	
	`  [out,retval] VARIANT* SupportedSystems)`

Description	It returns the list of supported OS versions in an array of strings. The list of supported OS versions is retrieved from the OvReqCheck config file ([VER] tag) HRESULT: ovrcErrMgmtServerNotSet - SetManagementServr method was not called
Property	CheckingEnabled

Property declaration	`Propget - CheckingEnabled ([out, retval ] VARIANT_BOOL *pVal );`
	
	`propput - CheckingEnabled ([in] VARIANT_BOOL newVal );`

Description	With this property, you can check whether prerequisite checking is enabled (for GUI and PMAD operations). Prerequisite checking can also be enabled and disabled by setting it to VARIANT_TRUE_/VARIANT_FALSE. HRESULT: ovrcErrMgmtServerNotSet - SetManagementServr method was not called Internal: Call is just forwarded to OvOWReqCheckSrv component on the management server, where the registry key is read and set.

IOvReqCheckSrv

The IOvReqCheckSrv interface provides the following methods.

Method declaration

Method	CheckNode	HRESULT CheckNode [in]BSTR Node, [out]VARIANT* R [out]VARIANT* R [in]BSTR Platfo [in,optional,de [in,optional,de [in,optional,de [in,optional,de [out,retval]lor
Description	It checks prerequisites for a node specified with the Node parameter (primary node name or IP address). With Platform parameter we can define the platform of the checked node. Valid values: "UNIX", "WINDOWS". rpType parameter specifies how detailed report should be: ■ RP_REQ_REC_INF: all prerequisites (requirements, recommendations) together with information data are returned. Each prerequisite has associated state (PASS,FAIL,UNCHECK). Each failed prerequisite has additional error description. ■ RP_REQ_REC: all prerequisites (requirements, recommendations) are returned. Each prerequisite has associated state (PASS,FAIL,UNCHECK). Each failed prerequisite has additional error description. ■ RP_REQ_REC_FAILED: only prerequisites (requirements, recommendations) that have failed are returned. Each prerequisite has associated state (FAIL,UNCHECK). Each failed prerequisite has additional error description. ■ RP_REQ: all requirements are returned. Each requirement has associated state (PASS,FAIL,UNCHECK). Each failed requirement has additional error description. ■ RP_REQ_FAILED: only requirements that have failed are returned. Each requirement has associated state (FAIL,UNCHECK). Each failed requirement has additional error description. *rpFormat* parameter specifies format of report:	

- RP_XML: report is generated in XML format (appropriate for further processing)
- RP_TXT: report is generated in text format (appropriate for directly displaying to the user)

Prerequisite check report (list of requirements/recommendations with the status) is returned as string with Report parameter. A content/format of report is determined with *rpFormat* and *rpType parameters .*

If User and Password parameters are provided prerequisite check is performed in context of provided user otherwise OvOWReqCheck component impersonates the client and contacts node directly.

Status of prerequisite checking is returned in *Result* parameter:

PREREQUISITES CHECKED:

- ovrcResAllPrereqOK (0) – All checked prerequisites are OK
- ovrcResReqOKRecFailed (1) - All requirements are OK; at least one recommendation has failed
- ovrcResReqOKRecNotCheked (2) - All requirements are OK; at least one recommendation could not be checked
- ovrcResReqFailed (3) – At least one requirement has failed
- ovrcResNoPrerequisitesSpecified (55) – No prerequisites specified for <system> system.

PREREQUISITES WERE NOT CHECKED:

- ovrcResReqCheckFailed (100) – not real return code; it is just a code to compare against to check whether check completed of failed; Result < ovrcResReqCheckFailed à prerequisite check completed; Result > ovrcResReqCheckFailed à prerequisite check failed
- ovrcResNoAdminRight (101) – Client does not have administrative privileges on the node. Prerequisites can be checked only with administrative rights.
- ovrcResOneReqNotChecked (102) – At least one requirement could not be checked (it is unknown whether requirement is OK or not); all other successfully checked requirements are OK
- ovrcResNodeNotAvailable (103) – Checked node is not available; either there is no network connection or firewall ports are not opened
- ovrcResNodeNotExists (104) - Node (name) cannot be resolved
- ovrcResPlatformNotSupported (105) - Platform not supported (Desc for the user: The platform/OS version on node xxx may not yet be supported - consult the latest support matrix ; if platform is supported ignore prerequisite check and check prerequisites manually)

	■ ovrcResPlatformNotDefined (106) – Node's platform properties (System Type, Operating System, Version) are not set. ■ ovrcResPlatformNotDiscovered (107) – Cannot discover platform. Textual description of Result code is returned in ResultDescription parameter. Note: Returned status is 2 if at least one requirement fails and if any other requirement could not be checked. HRESULT: ovrcNoError (S_OK) – success ovrcErrConfigFileSyntaxError – "Syntax error in line <line>: <err_descr>"
Method	CheckNodeOv

Method declaration	HRESULT CheckNodeOv(
	`[in]BSTR Node,`
	`   [in]long SystemType,`
	`   [in]long OsType,`
	`   [in]BSTR OS Version,`
	`   [out]VARAINT* Report,`
	`   [out]VARIANT* ResultDescription,`
	`   [in,optional,defaultvalue(RP_TXT)]ReportFormat rpFormat,`
	`   [in,optional,defaultvalue(RP_REQ_FAILED)]ReportType rpType,`
	`   [in,optional,defaultvalue("")]BSTR User,`
	`   [in,optional,defaultvalue("")]BSTR Password,`
	`   [out,retval]long* Result)`

Description	It checks prerequisites for node specified with Node parameter (primary node name or IP address). Prerequisites are checked according to provided HPOM for Windows node properties (SystemType, OSType,OSVersion). HPOM for Windows node properties directly relate to the [OVM] tag in the config file. For a description of parameters and return codes, see the CheckNode method.
Method	CheckNodeOv2

Method declaration	HRESULT CheckNodeOv2(
	`[in]BSTR NodeID,`
	   `[in]long SystemType,`
	   `[out]VARAINT* Report,`
	   `[out]VARIANT* ResultDescription,`
	   `[in,optional,defaultvalue(RP_TXT)]ReportFormat rpFormat,`
	   `[in,optional,defaultvalue(RP_REQ_FAILED)]ReportType rpType,`
	   `[in,optional,defaultvalue("")]BSTR User,`
	   `[in,optional,defaultvalue("")]BSTR Password,`
	   `[out,retval]long* Result)`

Description	It checks prerequisites for node specified with NodeID parameter. Prerequisites are checked according to provided HPOM for Windows node properties (SystemType, OSType,OSVersion). HPOM for Windows node properties directly relate to the [OVM] tag in the config file. If all requirements are all right (recommendations are not important) the method sets the node's PrerequisitesOkay property to TRUE. For a description of parameters, see the CheckNode method.
Property	CheckingEnabled

Property declaration	`Propget - CheckingEnabled([out, retval] VARIANT_BOOL *pVal);`
	
	`propput - CheckingEnabled([in] VARIANT_BOOL newVal);`

Description	With this property you can check whether prerequisite checking is enabled for GUI and PMAD operations. Prerequisite checking can also be enabled and disabled by setting it to `VARIANT_TRUE/VARIANT_FALSE.` Internal: Prerequisite setting is enabled and disabled by setting the registry key value: HKLM\SOFTWARE\Hewlett-Packard\OVEnterprise\Management Server\PrereqCheck\PrereqCheckingEnabled to 0 – disabled and 1 – enabled.

If the username and password are not provided (when calling CheckNode, CheckNodeOv, and CheckNodeOv2) C
set the impersonation level at least to "Impersonate:"

CoInitializeSecurity (NULL, -1, NULL, NULL, RPC_CAUTHN_LEVEL_NONE
RPC_C_IMP_LEVEL_IMPERSONATE, NULL, EOCA_NONE,0);

opcdeploy

The opcdeploy tool allows usage of the following actions from the command line:

- Copying a file to a remote system
- Retrieving a file from a remote system
- Executing a command on a remote system

Command synopsis

```
opcdeploy
   [ -help | -?]
   [ -deploy [ -srv <OVO_server_name> ] -node <nodename> -file <source_file>
             -targetdir <target_dir> ]
   [ -retrieve [ -srv <OVO_server_name> ] -node <nodename> -file <target_file>
               -sourcedir <source_dir> ]
   [ -cmd <command> [ -srv <OVO_server_name> ] -node <nodename>
                    [ -par "<parameters>" ] ]
```

Tool options

`-help | -?`
: Show tool usage and description.

Common options

`-srv <OVO_server_name>`
: Name of the HPOM for Windows server. If the server is not specified, a local machine is used.

`-node <nodename>`
: Name of the managed node. This string must match the string that was used when the node was put under management.

Specific options

`-deploy`
: Uses opcdeploy in deploy mode. Copies a file to a remote system.

`-file <source_file>`

Specifies the source file to be copied.

`-targetdir <target_dir>`

Specifies the target directory to which the file is copied. Specify the target as an absolute path.

`-retrieve`

Uses `opcdeploy` in retrieve mode. Copies a file from a remote system.

`-file <target_file>`

Specifies the target file to be copied.

`-sourcedir <source_dir>`

Specifies the source directory from which the file is copied.

`-cmd`

Uses `opcdeploy` in execute mode.

`<command>`

Specifies the command that should be executed by the shell of the node.

`-par "<parameters>"`

Parameters for the command execution. Enclose all parameters in quotation marks (""). If a parameter contains a quotation mark, escape it with a backslash ("\"").

Examples

```
opcdeploy -cmd copy -srv <OVO_server_name> -node <nodename>
          -par "\"C:\Program Files\test.txt\" C:"
```

Copies the `test.txt` file from `<Program_Files>` directory to the `C:` directory.

opcragt

The tool opcragt allows you to remotely administer agent processes on HP Operations Manager for Windows managed nodes. The tool can be used to:

- Switch the primary management server on specified managed nodes
- Get current status of the agents on the specified managed nodes
- Start or restart configured agents on the specified managed nodes
- Shut down (stop) configured agents except HP Operations Control Agent on the specified managed nodes
- Return the version number of the HP Operations agent software that is currently the installed agent on the specified managed nodes
- Return the setting of the configuration variable or variables on the specified managed nodes
- Set the configuration variables to the specified value on the specified managed nodes
- Remotely execute every action that you can execute locally with opcagt

Synopsis

```
opcragt
     [ -help ] |
     [ -all | [ [ -r ] -nodegrp <group*gt;... ] <node>... ]
     [ ( -start | -stop | -status | -primmgr ) [-id <subagent_id>] ] |
     [ -agent_version ] |
     [ -get_config_var <config_var>[(<process_name>)] ] |
     [ -set_config_var <config_var>[(<process_name>)]=[<value>] ]
```

Description	The command opcragt remotely administers the agent processes running on the managed nodes of HP Operations Manager (HPOM) from the HPOM management server. If called without any option, opcragt returns the current status of all agent processes on the specified system(s). This command is recommended for use in the command line prompt. When an operation is executed on more that one node, opcragt displays status immediately after a remote operation finishes for each node. The tool OvOWRagtS performs the same function as opcragt, but is recommended for use in HPOM tools. OvOWRAgtS displays status when a remote operation finishes on

	all nodes. NOTE: The HPOM Control Agent must always be running on the managed nodes, otherwise the HPOM management server cannot access the agent processes remotely.
`-help`	Displays usage message of opcragt. All other options and parameters are ignored.
`-start`	Start or restart configured agents on the specified system or systems.
`-stop`	Shut down (stop) configured agents except the HP Operations Control Agent on the specified system or systems.
`-status`	Get current status of the agents on the specified system or systems, grouped by subagent_id.
`-primmgr`	Informs the agents on the specified system or systems to change the HPOM primary manager to the calling HPOM management server.
`-id<subagent_id>`	Perform administration on subagents belonging to the <subagent_id>. If no <subagent_id> is specified, administration will be performed for all subagents.
`-get_config_var <config_var>[(<process_name>)]`	Return the setting of the configuration variable specified in <config_var>. If a variable has different values set for different processes, all values are reported. The values are queried from both the nodeinfo and opcinfo files on the managed node. If the values in these files differ, the ones from the opcinfo file are reported.
`-set_config_var <config_var>[(<process_name>)]={<value>]`	Set the configuration variable specified in <config_var> to the value specified in <value>. If required, a value can also be

	set for a specific process specified in (<process_name>). Process-specific settings have higher priority than general settings. To reset a value to its default setting, call `opcragt -set config_var` without any values. Resetting the general settings does not affect the specific settings and vice versa. NOTE: The values are set in the nodeinfo files on the managed node. Different settings in the opcinfo file will overwrite any settings produced with opcragt.
`-all`	Perform administration on all systems belonging to this HPOM environment.
`-r`	Perform administration also on all subgroups belonging to the specified node groups or groups.
`-nodegrp<group>...`	Perform administration on all systems belonging to the HPOM node group or groups. The <group> name must reflect the name specified in the HPOM Node Group Bank.
`<node>...`	System or list of systems where administration will be performed.If no system has been specified, the local system is used. You can specify the system either as a short system name (for example, mycomputer) or as a fully qualified system name (for example, mycomputer.prod.your.com).

Exit Values

This command will exit with value 0 after successful operation; otherwise the exit value will be set to 1 and an appropriate message will be displayed on stderr.

Restrictions

This command can only be issued by HPOM administrators. This command is only available on the HPOM management server.

After you set a configuration variable with `-set_config_var` , you must restart the agents for the new values to take effect.

Related Directories

The nodeinfo file is located in the following directory on the managed nodes:

UNIX:

`/var/opt/OV/conf/OpC/nodeinfo`

AIX:

`/var/lpp/OV/conf/OpC/nodeinfo`

Windows 2000:

```
<HP BTO product base dir>\Installed Packages\{790C06B4-844E-11D2-972B-
080009EF8C2A}\conf\OpC
```

Examples

- Start all agent processes on system 'sales' and 'opcagt.prod.your.com':

  `opcragt -start sales opcagt.prod.your.com`

- Stop all agent packages (except the Control Agent) on system 12.132.123.3:

  `opcragt -stop 12.132.123.3`

- Determine current status of all systems belonging to this HPOM environment:

  opcragt -status -all

- Set the primary HPOM manager on all systems belonging to the HPOM node groups hp_ux :

  `opcragt -primmgr -nodegrp hp_ux`

- Return the version number of the HP Operations agent software that is currently installed on node `'opcagt.prod.your.com'` :

  `opcragat -agent_version opcagt.prod.your.com`

- Return the port range specified for the process `'opcctla'` on node `'opcagt.prod.your.com` .

  `opcragt -get_config_var OPC_COMM_PORT_RANGE (opcctla)`

  opcagt.prod.your.com

- Set the port range for the process `'opcctla'` on node `'opcagt.prod.your.com to the value`

'1234':

```
opcragt -set_config_var OPC_COMM_PORT_RANGE(opcctla)=1234 opcagt.prod.your.com
```

- Reset the port range for all processes on node `'opcagt.prod.your.com'` to their default values:

  ```
  opcragt -set_config_var OPC_COMM_PORT_RANGE opcagt.prod.your.com
  ```

- Reset the port range for the process `'opcctla'` on node `'opcagr.prod.your.com'` to the default value:

  ```
  'opcragt -set_config_var OPC_COMM_PORT_RANGE(opcctla) opcagt.prod.your.com
  ```

ovowconfigexchange

The command ovowconfigexchange enables the exchange of configuration data (some specific subset) between cross-platform HPOM management servers (to support message forwarding), between HPOM for Windows and HPOM for UNIX servers in either direction.

This tool operates only on the HPOM for Windows platform. Its counterparts on HPOM for UNIX are opccfgupld/opccfgdwn and opctempl tools. It can upload data from HPOM for UNIX or download local configuration to files, formatted for upload to HPOM for UNIX. Files must be manually copied between the systems.

The following data exchange is supported:

- Download of instruction texts from HPOM for Windows (-dnl INSTR_TXT)

  A dummy HPOM for UNIX template (ovowhelptext) is generated that can be uploaded to HPOM for UNIX with the following command:

  ```
  opctempl -add ovowhelptext
  ```

  If the template already exists on the target HPOM for UNIX server, it must first be deleted and then uploaded again using the following sequence of commands to upload the ovowhelptext template.

  ```
  opctempl -delete "OVOW_Instruction_Text_Import" MSG
  opctempl -add ovowhelptext
  ```

  NOTE:
  Do not use opctempl -modify . This results in changing the instruction texts IDs, which makes the uploaded texts unusable.

  You do not need to distribute the template to any of the nodes.

- Download of nodes and node groups from HPOM for Windows (-dnl NODES)

  Data is transferred by using HPOM for UNIX node hierarchies that reflect the hierarchical structure of HPOM for Windows node groups. Generated files must be transferred to the HPOM for UNIX server and uploaded there with the opccfigupld tool.

  After the upload, the added nodes are found in the newly created node hierarchy (named CFGX_*) and also in the Node Bank's Holding Area. Make sure that you add these nodes to the operator's responsibilities, so that you are able to see their messages. After using the opccfgupld command on HPOM for UNIX, you must restart HPOM for UNIX server processes with opcsv-start to be able to receive messages for added nodes.

- Upload of nodes, node groups, and node hierarchies from HPOM for UNIX (-upl NODES)

Any node-related data (nodes, node groups, and node hierarchies) can be downloaded from HPOM for UNIX with the `opccfgdwn` tool. When the files (the complete, downloaded directory structure) are transferred to HPOM for Windows, they can be uploaded into HPOM for Windows with this tool.

NOTE:
When new nodes are uploaded to an HPOM for Windows server, then the autodeployment feature automatically deploys some core policies to the newly added nodes. If you do not want this behavior, you can change it by turning off the autodeployment feature, as described in the help topic Disable policy autodeployment . If you plan to turn off autodeployment just for the current node upload, remember to turn it on again afterwards.

The tool can read and write files in several different codesets, most of them being used on UNIX platforms. These are the keywords for codesets, understood by the -src_codeset and -dest_codeset options: ASCII, UTF8, ISO81 (for ISO88591), ISO82 (for ISO88592), ISO85 (for ISO88595), ISO815 (for ISO885915), ROMAN8, SJIS, EUCJP, GB2312, BIG5, EUCTW, EUCKR, and UNICODE (for Windows Unicode). When using files for or from HPOM for UNIX, the selected codeset must be compatible with the HPOM for UNIX server's database codeset.

Command synopsis

```
-ent INSTR_TXT -dnl [<directory>] [-dest_codeset <codeset>]

-ent NODES
     ( -dnl [<directory>] [-src_path <OVO_entity_path>]
                          [-dest_codeset <codeset>]
  [-no_ip_detect])|
     ( -upl <directory> [-dest_path <OVO_entity_path>]
                        [-src_codeset <codeset>] )

-help
```

Options

- -ent <entity> Specifies an entity for download or upload. <entity> can be INSTR_TXT or NODES.
- -dnl <directory> Specifies the download operation and sets the target directory. If the directory is not given, the current directory is used as default. The following hierarchy will be created in given directory:

  ```
  C
  C\NODES
  C\NODEHIER
  ```

- -upl <directory> Specifies the upload operation and sets the source directory. The user must specify the path to the .idx file located in the "language" directory (for example, the C subdirectory).

- `-src_path <OVO_entity_path>` Specifies the HPOM hierarchy path of an entity to be downloaded. This is used to select a specific node group for download.
- `-dest_path <OVO_entity_path>` Specifies the HPOM hierarchy path for the upload of an entity. This is used to select the existing target node group where the new uploaded data will be stored. If this option is not set, a new generic path will be created to hold the uploaded data.
- `-no ip detect` Disables IP detection for nodes through DNS during download from HPOM for Windows. By default, the IP is determined through DNS lookup if there is no explicit IP address available.
- `-dest_codeset <codeset>` Specifies the codeset of the resulting files when data is downloaded from HPOM for Windows and formatted for HPOM for UNIX.
- `-src_codeset <codeset>` Specifies the codeset of the source files when data is uploaded into HPOM for Windows.

Exit Values

None.

Restrictions

HPOM for UNIX does not allow nodes to belong to several different node groups. When uploading such configuration data into HPOM for UNIX, opccfgupld will issue warnings that duplicate nodes were found. A node will only be added to the first node group where it appears.

Related Files

None.

Examples

1. ovowconfigexchange -help

   Outputs help and usage of the tool.
2. ovowconfigexchange -ent INSTR_TXT -dnl c:\data\transf\2 -dest_codeset SJIS

   Downloads instruction texts from the HPOM for Windows server and stores them to a dummy HPOM for UNIX template file `c:\data\transf\2\ovowhelptext` using SJIS encoding.
3. ovowconfigexchange -ent NODES -dnl c:\temp\1 -src_path \cad\koeln3

   Downloads node group \cad\koeln3 (with all its contents) from the HPOM for Windows server and creates the following HPOM for UNIX files:

```
c:\temp\1\C\1.idx
c:\temp\1\C\NODEHIER\nodehier.dat
c:\temp\1\C\NODES\nodes.dat
```

During download, the "language" directory is always set to C, regardless of the chosen destination codeset.

To upload the data into an HPOM for UNIX server, transfer the created files (from the c:\temp\1 directory downward) to the target machine (for example, to /tmp/transf/1 directory) and create the following structure:

```
  /tmp/transf/1/C/1.idx
  /tmp/transf/1/C/NODEHIER/nodehier.dat
  /tmp/transf/1/C/NODES/nodes.dat
```

Run the following command on the UNIX machine:

"`opccfgupld /tmp/transf/1/`" to upload it.

4. ovowconfigexchange -ent NODES -upl c:\temp\5\JP.SJIS\ -dest_path \cad\ulm -src_codeset SJIS

   Uploads the HPOM for UNIX node data into HPOM for Windows, converting the files from the SJIS to the Windows Unicode codeset. HPOM for UNIX configuration files were manually transferred from HPOM for UNIX to HPOM for Windows beforehand and the following structure was created:

   ```
   c:\temp\5\JP.SJIS\5.idx
   c:\temp\5\JP.SJIS\NODEHIER\nodehier.dat
   c:\temp\5\JP.SJIS\NODES\nodes.dat
   ```

   The node hierarchy will be placed into the existing `\cad\ulm` node group on the HPOM for Windows server.

- Configuration interoperability

ovpmpwutil

Use the command-line program ovpmpwutil to change the user name or password for measurement threshold policies, scheduled task policies, and Windows Management Interface policies on the management server where the program runs.

Command synopsis

```
ovpmpwutil [/? | /all | /measure | /schedtask | /wmi] /u <UserName> [/n <NewUserName>] /p
<Password>
```

Options

- *Optional* . Specify one of the following flags:
  - ? : Usage information.
  - all : Change all measurement threshold policies, scheduled task policies, and Windows Management Interface policies.
  - measure : Change measurement threshold policies only.
  - schedtask : Change scheduled task policies only. (This is the default.)
  - wmi : Change Windows Management Interface policies only.
- UserName : The current user name in the policies. Only those policies that contain this user name are changed by the command. This parameter is required.
- NewUserName : The new user name for the policies. This parameter is optional.
- Password : The new password for the policies. This parameter is required.

Examples

- `ovpmpwutil /u User1 /p Pass1`
  Changes the password to `Pass1` for all scheduled task policies with the user name `User1` .
- `ovpmpwutil /all /u User1 /p ""`
  Removes the password for all measurement threshold, scheduled task, and WMI policies with the user name `User1` .
- `ovpmpwutil /measure /u User1 /n Administrator /p "a4f 99c"`
  Changes the password to `a4f 00c` and the user name to `Administrator` for all measurement threshold policies with user name User1. Note that the password contains a space and is therefore enclosed in quotation marks.

NOTE:
A new policy version is produced for each policy that is changed by this command. This means that the action redeploy all will not deploy the new polices, because redeploy all redeploys the policy versions that exist on the managed node . The updated policies will also only be available in policy group where they were added if you configured the group to automatically update to the latest version of the policies it contains.

Related Topics:

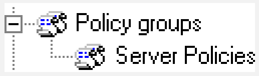
- Update to latest
- Redeploy all

ovpmutil

The tool ovpmutil.exe enables you to perform some configuration tasks from the command line. The tool can be used to do the following:

- Deploy policies and policy groups.
- Download or upload policy, tool, user role, node, service information, policy and package node inventory, message group, or server configuration values from the management server.
- Convert downloaded information between structured storage format and text format.
- Register policies, packages, and policy types.

```
DEP (/p | /pn | /pg) <policyname> (/n | /np | /ng) <nodename> [/c <checkversion>][/e
<enable>][/t <type>][/v <version>][(/user /pass ) | /client]
```

Description	Deploys policies or policy groups on one or more managed nodes.
`/p | /pn | /pg`	This *required* parameter indicates what is deployed. Use /p to indicate a policy with a full policy group path, /pn to indicate a policy without a path (only policy name), or /pg to indicate a policy group.
`<policyname>`	This *required* parameter indicates the name of the policy or policy group to be deployed. For /pn , it is only the name of the policy without full policy group path. Otherwise, it must contain the path to the policy as shown in the console tree, starting under Policy groups . The path must begin with a backslash (\ ). Policy groups within the path are separated with a backslash, but may not end with a backslash. If the name of a policy group contains spaces, the entire path must be enclosed in quotation marks. For example, you may want to distribute a policy named disk_monitor found in a policy group named "Server Policies":  Policy groups Server Policies The correct syntax would be: `ovpmutil dep /p "\Server Policies\disk_monitor" mynode`
`/n | /np | /ng`	This *required* parameter indicates the target type for deployment. Use /n to indicate a node, /np to indicate a node with a primary node name, or /ng to indicate a node group.
`<nodename>`	This *required* parameter indicates the name of the node or node group on which the policy will be deployed. For /np , it is only the primary node name without the full node group path. Otherwise, it must contain the path to the node, as shown in the console tree,

	starting under Nodes. Use /n to indicate a node, or /ng to indicate a node group. All node groups within the path must be preceded with a backslash (\ ), but may not end with a backslash. If the name of a node group contains spaces, the entire path must be enclosed in quotation marks.
`/c <checkversion>`	If this *optional* parameter is set to `FALSE` , no version check is made, and the policy is deployed even if a newer version of that policy is already deployed on the node. If the parameter is `TRUE` (or absent), the policy is deployed only if no newer version of the policy is deployed on the node.
`/e <enable>`	If this *optional* parameter is set to `FALSE` , the policies are disabled on the node after deployment . If this parameter is `TRUE` (or absent), the policy or policy group is enabled after deployment.
`/t <type>`	If this *optional* parameter is set, the policy is deployed only if it is a policy of the given policy type . You may use only the original English names. You must enclose the names quotation marks (for example, "Logfile Entry" or "Windows Management Interface"). This parameter is not valid when a policy group is deployed.
`/v <version>`	If this *optional* parameter is set, the policy is deployed only if it matches the given version. This parameter is not valid when a policy group is deployed.
`/use <username>`	Use this optional parameter to select the name of the account used to access the remote node. You can specify the account as a user principal name (UPN), such as `User@Domain` , or in the down-level log-on name format (`Domain\User`).
`/pass <password>`	The password of the account used to access the remote node. Use this option only together with the `/user` option.
`/client`	This *optional* parameter indicates that the deployment job should run under the security context of the specified user that is impersonated.

```
PCV [/x]|[/c [/v <version>][/n <name>][/i]][/e <encoding>]][<filenamelist> | /d
<dirnamelist>]
```

Description	Converts downloaded policy files (structured storage files) into a header stream (`*.header`) containing the policy management information (for example, name, version, LogicalID, InstanceID, and Checksum) and the data stream (`*.data`) containing the policy definition. The purpose of this functionality is to convert the policy from HPOM for Windows format (structured storage file) to

	HPOM for UNIX format(*.data file), and vice versa.
/x	This *optional* parameter indicates that the command should extract the structured policy store from <filenamelist> or <dirnamelist> , which are lists of policy files (downloaded using ovpmutil CFG DNL) to be converted. You can extract multiple structured storage files by listing them separated with a space or by specifying them with directories of policy files separated with a space (/d option). You can specify the policy file list with <filenamelist> or <dirnamelist> .
/c	This *optional* parameter indicates that the command should combine the *.header and *.data files into a structured policy store. <filenamelist> and <dirnamelist> have the same format as for /x .
/v <version>	This *optional* parameter changes the version information of the policy to <version> during combination. (The InstanceID and Checksum are also updated.) Use this option only together with the /c option.
/n <name>	This *optional* parameter changes the name given to the policy. (The new LogicalID, InstanceID, is created and Checksum is updated). Use this option only together with the **/c** option.
/i	This *optional* parameter indicates that just the InstanceID (and Checksum) information of the policy is updated. If you do not specify it, and the Checksum does not match the data stream, an error is generated. Use this option only together with the **/c** option.
/e <encoding>	This *optional* parameter indicates that policy *.data files are converted from selected to UTF-8 encoding, and then combined into a structured policy store. Use this option to convert a policy file that is not encoded in UTF-8 or ASCII. Supported encodings are ASCII, UTF-8, ISO81, ISO82, ISO85, ROMAN8, SJIS, EUCJP, GB2312, BIG5, EUCTW, and EUCKR. Default is UTF-8. Use this option only together with the **/c** option.
<filenamelist>	This *optional* parameter is a list of policy file names to be converted. The policy that is a structured storage is extracted into the header stream (*.header) containing the policy management information (for example, name, version, LogicalID, InstanceID, and Checksum) and the data stream (*.data) containing the policy definition. If you do not use this parameter, you must use /d <dirnamelist> . File names must not include file extensions.

`/d <dirnamelist>`	This *optional* parameter is a list of directories with policy files to be converted. All policy files located on directories in the list are converted to the appropriate format. If you do not use this parameter, you must use `<filenamelist>`.

```
REG POL <PolicyFileName>[/g <PolicyGroupPath>] [/pid <ProductId>]
```

Description	Registers policies (in the form of structured storage files) on the management server.
`<PolicyFileName>`	This *required* parameter indicates the name of the policy file to be registered.
`/g <PolicyGroupPath>`	This *optional* parameter indicates the group to which the policy should be added. If no group is given, the policy is visible only under Policies grouped by Type. The path must begin with a backslash (\ ). The policy groups within the path are separated with a backslash (\ ). If the name of a policy group contains spaces, the entire path must be enclosed in quotation marks. If the policy group does not exist, the command creates it.
`/pid <ProductId>`	This *optional* parameter allows you to specify the product ID of a licensed OV product. If you supply this parameter, the policy is enabled only if the product is licensed on the management server. If you do not supply the parameter, the policy functions without any license check.

```
UNREG POL <PolName> <PolVersion> <PTName> <PTVersion>
```

Description	Unregisters policies on the management server. This operation is carried out only if the policy is currently not deployed on any managed node.
`<PolName>`	This *required* parameter indicates the name of the policy as it appears in the console.
`<PolVersion>`	This *required* parameter indicates the version of the policy. The format is `n.m` , where `0 = n = 9999` and `0 = m = 9999` .
`<PTName>`	This *required* parameter indicates the original English name of the policy type of the policy.
`<PTVersion>`	This *required* parameter indicates the version of the policy type. The format is `n.m` , where `0 = n = 9999` and `0 = m = 9999` .

```
CFG <conftype> (UPL <configfilename> [/noautodeploy]) | (DNL <configfilename> [ (/a) | (/p
<identifier> [/configuredonly | /externalonly]) | (/excludenodes) ] |
(([/writable][/migration]) | [/changed]) | (RMV <configfilename>)
```

Description	Uploads or downloads the policy, service, user role, node, tool, service type, message group, or policy, package node inventory, and agent profile configuration.
`<conftype>`	This *required* parameter indicates what the command should upload or download. Valid choices are: ■ `POL` = policy configuration of all policies in a group ■ `SVC` = service configuration ■ `TLS` = tools configuration ■ `NDS` = node configuration ■ `USR` = user roles ■ `PPN` = policy and package node inventory ■ `XML` = service configuration in HP Operations Manager for UNIX XML format ■ `SVT` = service type configuration ■ `MSG` = message group configuration ■ `CIV` = configuration values ■ `MDL` = combines `SVC` , `SVT` , `TLS` , `NDS` , and `MSG` options ■ `ALL` = combines `MDL` , `POL` , `USR` , `PPN` , and `CIV` options
`UPL <configfilename>`	This *optional* parameter indicates that the command should upload the management configuration file designated by the file `<configfilename>` . The `<configfilename>` designates the location and name of the management configuration file to be uploaded or downloaded (for example, `c:\test\dnl-svc.mof` or `c:\tmp\config.mm`). Uploading data in HP Operations Manager for UNIX XML format and upload agent profile operations are not supported.
`/noautodeploy`	This *optional* parameter is valid only for the node configuration upload (`NDS | MDL | ALL`) `UPL` option. If you do not specify it, the property `DisableAutoDeployment` is set to `true` for all imported nodes.
`DNL <configfilename>`	This *optional* parameter indicates that the command should download the configuration specified in the `<configtype>` parameter, and save it in the directory designated by `<configfilename>` . The name can include a drive letter and a path. If you do not specify a path, the file is written to the current directory. For conftype `SVC` , `TLS` , `NDS` , `SVT` , `MSG` , and `CIV` , `<configfilename>` indicates the name and location of the downloaded `mof` file. For conftype `XML` and `PPN` , it indicates the

	name and location of a downloaded XML file. For conftype `POL` , it indicates the directory for the downloaded `mm` file and the downloaded structured storage policy files. For conftype `PRF` , it indicates the directory for the downloaded agent profile file. NOTE: For conftype `NDS` and `SVT` , only associations to tools, policy groups, and deployment packages (plus reports and graphs for `NDS`) are downloaded, not objects themselves (which must be downloaded manually). Although all types of policies can be downloaded, only the following policies can be imported into HP Operations Manager for UNIX: ■ Logfile Entry policy type ■ Open Message Interface policy type ■ Windows Event Log policy type ■ SNMP Interceptor policy type ■ Scheduled Command policy type
`/a`	This *optional* parameter is valid only for policy configuration download. If it is specified, all policy versions (assigned to the policy group or not) are downloaded and stored in a specified directory with one mm file. You cannot use this parameter when `/p` is specified.
`/p <identifier>`	This *optional* parameter is only valid for configuration download. For policies, the identifier is a path from which policies are downloaded recursively. It must contain the path to the policy, as shown in the console tree, starting under Policy groups. The path must begin with a backslash (\ ). Policy groups within the path are separated with a backslash (\ ). If the name of a policy group contains spaces, the entire path must be enclosed in quotation marks. If you do not specify the path, all policy groups are downloaded (each in its own subdirectory with its own `mm` file). You cannot use this parameter when `/a` is specified. For services, the parameter is the service ID of the top-level service of the sub-tree to be downloaded. For tools, nodes, or XML configurations, this parameter is the unique id of the item at which the recursive enumeration begins. It defaults to the respective root item.

	For message groups, this parameter specifies the message group ID. If you do not specify this parameter, all message groups are downloaded. For user roles, this parameters specifies the user role ID. If you do not specify this parameter, all user roles are downloaded.
`/excludenodes`	This *optional* parameter is valid only with the `DNL MDL` option. It indicates that the node information should not be downloaded.
`/configuredonly`	This *optional* parameter is valid only with the `DNL NDS` option. It indicates that only the configuration of configured nodes (nodes with defined OS) should be downloaded.
`/externalonly`	This *optional* parameter is valid only with the `DNL NDS` option. It indicates that only the configuration of external nodes should be downloaded.
`/writable`	This *optional* parameter is valid only with the DNL CIV option. It indicates that only writable configuration values should be downloaded.
`/migration`	This *optional* parameter is valid only with the DNL CIV option. It indicates that only configuration values relevant for migration of configuration to another HPOM for Windows server should be downloaded.
`/changed`	This *optional* parameter is valid only with the DNL CIV option. It indicates that only modified configuration values should be downloaded.
`RMV <configfilename>`	Not valid for end-user operations.

```
DNL PRF ( (([/fqdn <FQDN>][/ip <IP>]) | [/nid <ID>]) [/d <dir>] ) | PKG ( <PackName>
[PackVersion] [/d <dir>] ( /nid <ID> | /pnn <PrimaryNodeName> | (/ost <OSType> /osv
<OSVersion> /abf <AgtBinFormat>) ) )
```

Description	Downloads the agent profile (PRF option) or package binaries (PKG option) for manual installation.
`/nid <ID>`	This *optional* parameter enables you to specify an ID of the node that is already managed by the management server.
`/d <dir>`	This *optional* parameter specifies the directory where the agent profile or package binaries are downloaded. If you do not specify this parameter, the current directory is used.
`PRF`	This parameter indicates that the command should download the agent profile for the manual agent installation for the specified node.
`/fqdn <FQDN>`	This *optional* parameter enables you to specify a Fully Qualified Domain Name (FQDN) of the node for which the agent profile should be generated.
`/ip <IP>`	This *optional* parameter enables you to specify an IP address of the node for which the agent profile should be downloaded.
`PKG`	This parameter indicates that the command should download package binaries for manual package installation on the specified node or on a specified platform.
`<PackName>`	This *required* parameter indicates the name of the package you want to download.
`<PackVersion>`	This *optional* parameter indicates the version of the package you to download. The format is `n.m.o` , where `0 <= n,m,o <= 9999` .
`/ppn <PrimaryNodeName>`	This *optional* parameter enables you to specify the primary node name of the node that is already managed by the management server.
`/ost <OSType>`	This *optional* parameter enables you to specify the OS version of a target platform. Use this parameter together with `/osv` and `/abf` .
`/osv <OSVersion>`	This *optional* parameter enables you to specify the OS version of a target platform. Use this parameter together with `/ost` and `/abf` .
`/abf <AgtBinFormat>`	This *optional* parameter enables you to specify the binary format of agent bits for deployment on a target platform. Use this parameter together with `/ost` and `/osv` .

Samples

```
ovpmutil cfg civ dnl c:\test\dnl-civ.mof
ovpmutil cfg civ dnl c:\test\dnl-civ.mof /migration /changed
ovpmutil cfg civ upl c:\test\dnl-civ.mof
ovpmutil cfg mdl dnl c:\test\dnl-mdl.mof /excludenodes
ovpmutil cfg mdl upl c:\test\dnl-mdl.mof
ovpmutil cfg msg dnl c:\test\dnl-msg.mof /p <MessageId>
ovpmutil cfg msg upl c:\test\dnl-msg.mof
ovpmutil cfg nds dnl c:\test\dnl-nds.mof /p Root_Nodes
ovpmutil cfg nds dnl c:\test\dnl-ext.mof /p Root_Nodes /externalonly
ovpmutil cfg nds dnl c:\test\dnl-cfg.mof /p Root_Nodes /configuredonly
ovpmutil cfg nds upl c:\test\dnl-nds.mof
ovpmutil cfg pol dnl c:\test /p \Samples
ovpmutil cfg pol upl c:\test\config.mm
ovpmutil cfg pol rmv c:\test\config.mm
ovpmutil cfg ppn dnl c:\test\dnl-ppn.xml /p Root_Nodes
ovpmutil cfg ppn upl c:\test\dnl-ppn.xml
ovpmutil cfg svc dnl c:\test\dnl-svc.mof /p Root_Services
ovpmutil cfg svc upl c:\test\dnl-svc.mof
ovpmutil cfg svt dnl c:\test\dnl-svt.mof /p root
ovpmutil cfg svt upl c:\test\dnl-svt.mof
ovpmutil cfg tls dnl c:\test\dnl-tls.mof /p Root_Tools
ovpmutil cfg tls upl c:\test\dnl-tls.mof
ovpmutil cfg usr dnl c:\test\dnl-user-roles.xml /p Windows-admin
ovpmutil cfg usr upl c:\test\dnl-user-roles.xml
ovpmutil cfg xml dnl c:\test\dnl-xml.xml /p Root_Services
ovpmutil dnl prf /fqdn <nodename.hp.com>
ovpmutil dnl prf /nid <NodeId> /d "c:\test"
ovpmutil dnl pkg <PkgName> 8.50.10 /ost Solaris /osv 9 /abf SPARC
ovpmutil dnl pkg /nid <NodeId> /d "c:\test"
```

Examples

Example 1

To download a policy and convert it to text (HPOM for UNIX) format:

1. Download the policies in the policy group *Server Policies* to the test directory on the C drive

   ovpmutil CFG POL DNL c:\test /p "\Server Policies"

2. Extract a downloaded structured storage file to a text header and data file:

   ovpmutil PCV /x "C:\test\disk_monitor_CC832F49-A8BC-11D3-A45F-080009DC628C"

3. Combine the header and data file into a structured policy storage file that can be uploaded on the

management server:

```
ovpmutil PCV /c "C:\test\disk_monitor_CC832F49-A8BC-11D3-A45F-080009DC628C"
```

Example 2

To deploy all policies in the policy group Server Policies to the node parsnip, even if a newer version is already deployed on the node:

```
ovpmutil DEP /pg "\Server Policies" /n "\email servers\parsnip" /c FALSE
```

ovowconfigutil

The Server Configuration dialog box in the HPOM console enables you to change many different values, which control many different aspects of a management server's configuration. `ovowconfigutil` enables you to download these server configuration values to a file. This may be useful, for example, for backup purposes or to migrate customized values to another management server.

`ovowconfigutil` downloads the current set of server configuration values into a Management Object Format (MOF) file. You later upload this MOF file using the `mofcomp` command, which Microsoft provides. You can upload the MOF file to management servers with HPOM for Windows version 8.00 or later.

Command synopsis

```
ovowconfigutil
   [ -help ]    |
   [ -check ]   |
   [ -install ] |
   [ -reset ]   |
   [ -download  [ -all | -writable | -migration ] [ -changed_only ] [ -file <filename.mof> ] ]
```

Options

`-help`

Show usage information.

`-check`

Perform a basic validity check of the server configuration values.

`-install`

Creates registry keys to store some of the configuration values. This option is meant for internal use during management server installation. You do not normally need to use this option.

CAUTION:
The `-install` option resets some configuration values to their default values.

`-reset`

Resets all server configuration values to their default values.

`-download`

Downloads server configuration values for backup or migration purposes. You can use the following additional options with -download :

`-all`

Includes all configuration values.

`-writable`
: Omits read-only configuration values.

`-migration`
: Includes only server configuration values that are relevant for migration to another management server.

`-changed_only`
: Includes only server configuration values that are different to their default values.

`-file` *<filename.mof >*
: Specifies the name of the MOF file that the command creates. If you omit this option, the command creates the file `OV_ConfigValue_Instances.mof` in the current folder.

Examples

`ovowconfigutil -download`
: Downloads all server configuration values to the file `OV_ConfigValue_Instances.mof` in the current folder.

`ovowconfigutil -download -all -changed_only -file changed_values.mof`
: Downloads all changed server configuration values to the file `changed_values.mof` in the current folder.

`mofcomp OV_ConfigValue_Instances.mof`
: Uploads server configuration values from the file `OV_ConfigValue_Instances.mof` .

Related Topics:

- Change server configuration values

ovownodeutil

The ovownodeutil tool allows command-line management of nodes and node groups. Specific nodes or node groups are referred to in several different ways.

Nodes

Nodes are referred to by primary node name and Id:

- Primary node name (-node_name parameter)

  Represents the PrimaryNodeName property of the OV_ManagedNode object that uniquely identifies a node.

- Node Id (-node_id parameter)

  Represents the Name property of the OV_ManagedNode object that uniquely identifies a node. It is usually a GUID, but can also be any other unique string.

External nodes

External nodes are referred to by hierarchical path and external node ID:

- Hierarchical path (-exnode_path parameter)

  Represents a directory-like path of node group names that uniquely identifies a group and an external node name. Captions/Display Names are used as node group names. The Caption property of OV_ExternalNode is used as an external node name. A backslash (\ ) is used as a separator (for example, `\rocco\cad\room108`). The path must always start with a single backslash (\ ), which denotes the Root. If the external node name itself contains a backslash, it must be escaped with an additional backslash (\\ ).

- External node ID (-exnode_id parameter)

  Represents the Name property of the OV_ExternalNode object that uniquely identifies a node. It is usually a GUID, but can also be any other unique string.

Node groups

Node groups are referred to by hierarchical path and group Id:

- Hierarchical path (-group_path parameter)

  Represents a directory-like path of node group names that uniquely identifies a group. Captions/Display Names are used as node group names. A backslash (\ ) is used as a separator (for example, `\rocco\cad\room108`). The path must always start with a single backslash (\ ), which denotes the Root. If the node name itself contains a backslash, it must be escaped with an additional backslash (\\ ).

- Group Id (-group_id parameter)

  Represents the Name property of the OV_Nodegroup object that uniquely identifies a node group. It is usually a GUID, but can also be any other unique string.

Command synopsis

```
ovownodeutil
   [ -help ]
   [ -add_node -node_name <primary node name>
             [ -group_path <hierarchy path> | -group_id <group id> ]
             [ -agent_comm_type (DCE | HTTPS) ]
             [ -agent_bin_format <agent binary format> ]
             [ -caption <caption name> ]
             [ -comm_path <communication path> ]
             [ -always_resolve_comm_path (yes | no) ]
             [ -auto_update_comm_path (yes | no) ]
             [ -auto_deploy (yes | no) ]
             [ -osbits (32 | 64)] ]
   [ -add_nodewithtype -node_name <primary node name>
               -ostype <OS type caption>
               -osversion <OS version caption>
               -systemtype <system type caption>
               -osbits (32 | 64)
               -agent_bin_format <agent binary format>
             [ -group_path <hierarchy path> | -group_id <group id> ]
             [ -agent_comm_type (DCE | HTTPS) ]
             [ -caption <caption name> ]
             [ -comm_path <communication path> ]
             [ -always_resolve_comm_path (yes | no) ]
             [ -auto_update_comm_path (yes | no) ]
             [ -auto_deploy (yes | no) ] ]
   [ -delete_node ( -node_name <primary node name> | -node_id <node id> ) ]
   [ -add_group ( -group_path <hierarchy path> ]
   [ -delete_group ( -group_path <hierarchy path> | -group_id <group id> ) ]
   [ -assign_node ( -node_name <primary node name> | -node_id <node id> )
             ( -group_path <hierarchy path> | -group_id <group id> ) ]
   [ -deassign_node (-node_name <primary node name> | -node_id <node id>) ]
             ( -group_path <hierarchy path> | -group_id <group id> ) ]
   [ -list_nodes (-group_path <hierarchy path> | -group_id <group id>)
             [ -recursive ]  ]
   [ -list_groups ]
   [ -change_comm_opt ( -node_name <primary node name> | -node_id <node id> )
             [ -comm_path <communication path> ]
             [ -always_resolve_comm_path (yes | no) ]
```

```
          [ -auto_update_comm_path (yes | no) ] ]
[ -change_comm_opt ( -group_path <hierarchy path> | -group_id <group id> )
          [ -always_resolve_comm_path (yes | no) ]
          [ -auto_update_comm_path (yes | no) ] ]
[ -deassign_tool ( ( -node_name <primary node name> | -node_id <node id> ) |
          ( -group_path <hierarchy path> | -group_id <group id> ) )
          ( -tool_path <tool hierarchy path> | -tool_id <tool id> ) ]
[ -deassign_autodeploy_pg
          (-group_path <hierarchy path> | -group_id <group id>)
          (-pg_path <policy group hierarchy path> | -pg_id <pol. group id>) ]
[ -delete_report ( -group_path <hierarchy path> | -group_id <group id> )
          -report_name <report name> ]
[ -delete_graph ( -group_path <hierarchy path> | -group_id <group id> ) ]
[ -outage_node [-scheduled | -unplanned ]
          [ -node_name {<primary node name>} ] [ -node_id {<node id>} ]
          [ -group_path {<hierarchy path>} ] [ -group_id {<group id>} ]
          -get [ -single_line | -detailed | -show_outage_only ] ]
[ -outage_node [ -scheduled | -unplanned ]
          [ -node_name {<primary node name>} ] [ -node_id {<node id>} ]
          [ -group_path {<hierarchy path>} ] [ -group_id {<group id>} ]
          ( -on | -off | -toggle )
          [ -delete_msgs ] [ -disable_heartbeat ] ]
[ -add_exnode -caption <caption name>
          [ -group_path <hierarchy path> | -group_id <group id> ]
          ( -ip_address <IP address pattern> | -ip_name <IP name pattern>
          | -other <other pattern> ) [ -order <order number> ]
          [ -check_before_managed_nodes ] ]
[ -delete_exnode ( -exnode_path <external node hierarchical path> |
          -exnode_id <unique_node_id> ) ]
[ -assign_exnode ( -exnode_path <external node hierarchical pat> |
          -exnode_id <unique_node_id> )
          ( -group_path <hierarchy path> | -group_id <group id> ) ]
[ -deassign_exnode ( -exnode_path <external node hierarchical pat> |
          -exnode_id <unique_node_id> )
          ( -group_path <hierarchy path> | -group_id <group id> ) ]
[ -list_exnodes [ -group_path <hierarchy path> | -group_id <group id> ]
          [-recursive] ]
[ -outage_exnode [-scheduled | -unplanned ]
          [ -exnode_path <external node hierarchical path> } ]
          [ -exnode_id {<unique_node_id>} ]
          [ -group_path {<hierarchy path>} ] [ -group_id {<group id>} ]
          -get [ -single_line | -detailed | -show_outage_only ] ]
[ -outage_exnode [ -scheduled | -unplanned ]
          [ -exnode_path <external node hierarchical path> } ]
          [ -exnode_id {<unique_node_id>} ]
          [ -group_path {<group_hierarchy_path>} ]
          [ -group_id {<unique_group_id>} ]
          ( -on | -off | -toggle )
          [ -delete_msgs ] ]
```

Tool options

`-help`

Show tool usage and description.

`-add_node`

Adds a node to this management server. Use this when the discovery service can find the node's system information automatically. Otherwise, if the discovery service cannot find the node's system information automatically, use -add_nodewithtype instead.

`-node_name <primary node name>`

Sets the PrimaryNodeName property of the new node.

`-group_path <hierarchy path>`

Sets a parent node group of the new node by hierarchical path. The node group must exist.

`-group_id <group id>`

Sets a parent node group of the new node by its ID. The node group must exist.

`-agent_comm_type (DCE | HTTPS)`

Sets the communication type of the nodes agent.

`-agent_bin_format <agent binary format>`

Sets the agent binary format (x86, x64, IA64, and so on).

`-caption <caption name>`

Sets the Caption property of the new node.

`-comm_path <communication path>`

Sets the CommunicationPath property of the new node. It can contain the DNS name, WINS name, or IP.

`-always_resolve_comm_path (yes | no)`

If set to yes , the communication to the new node is always resolved before contacting the agent (DHCP).

`-auto_update_comm_path (yes | no)`

If set to yes , the CommunicationPath property is updated by the agent automatically.

`-auto_deploy (yes | no)`

If set to yes , the management server attempts to deploy auto-deployment policies or packages of the parent node groups to the node. Otherwise, no automatic deployment occurs (primarily intended for nodes imported from another management server).

`-osbits (32 | 64)`

Sets the operating system bit length. You may need to specify this option for some nodes if the discovery service cannot find the information automatically.

`-add_nodewithtype`

Adds a node to this management server with system information. Use this when the discovery service cannot find the node's system information automatically. Valid values for node system information are visible in the System tab of the Node Properties dialog box.

`-node_name <primary node name>`

Sets the PrimaryNodeName property of the new node.

`-ostype <OS type caption>`

Sets the operating system type.

`-osversion <OS version caption>`

Sets the operating system version.

`-systemtype <system type caption>`

Sets the system type.

`-osbits (32 | 64)`

Sets the operating system bit length.

`-agent_bin_format <agent binary format>`

Sets the agent binary format.

`-group_path <hierarchy path>`

Sets a parent node group of the new node by hierarchical path. The node group must exist.

`-group_id <group id>`

Sets a parent node group of the new node by its ID. The node group must exist.

`-agent_comm_type (DCE | HTTPS)`

Sets the communication type of the nodes agent.

`-agent_bin_format <agent binary format>`

Sets the agent binary format (x86, x64, IA64, and so on).

`-caption <caption name>`

Sets the Caption property of the new node.

`-comm_path <communication path>`

Sets the CommunicationPath property of the new node. It can contain the DNS name, WINS name, or IP.

`-always_resolve_comm_path (yes | no)`

If set to yes , the communication to the new node is always resolved before contacting the agent (DHCP).

`-auto_update_comm_path (yes | no)`

If set to yes , the CommunicationPath property is updated by the agent automatically.

`-auto_deploy (yes | no)`

If set to yes , the management server attempts to deploy auto-deployment policies or packages of the parent node groups to the node. Otherwise, no automatic deployment occurs (primarily intended for nodes imported from another management server).

`-delete_node`

Deletes a node from this management server. All the node associations are deleted as well.

`-node_name <primary node name>`

Specifies a node by its PrimaryNodeName property.

`-node_id <node id>)`

Specifies a node by its ID.

`-add_group`

Adds a node group to this management server.

`-group_path <hierarchy path>`

Specifies the hierarchical path of the new node group. All non-existing groups from the path are created automatically.

`-delete_group`

Deletes a node group from this management server. All the group associations are deleted as well.

`-group_path <hierarchy path>`

Specifies a node group to be deleted by hierarchical path.

`-group_id <group id>`

Specifies a node group to be deleted by its ID.

`-assign_node`

Assigns a node to a node group. Both the node and the node group must already exist.

`-node_name <primary node name>`

Specifies a node by its PrimaryNodeName property.

`-node_id <node id>)`

Specifies a node by its ID.

`-group_path <hierarchy path>`

Specifies a parent node group by hierarchical path.

`-group_id <group id>`

Specifies a parent node group by its ID.

`-deassign_node`

De-assigns a node from a node group. The node is not deleted. Only its association to the node group is removed.

`-node_name <primary node name>`

Specifies a node by its PrimaryNodeName property.

`-node_id <node id>)`

Specifies a node by its ID.

`-group_path <hierarchy path>`

Specifies a parent node group by its hierarchical path.

`-group_id <group id>`

Specifies a parent node group by its ID.

`-list_nodes`

Lists nodes belonging to a specific node group. If no node group is specified, the Root is used as default. Some nodes may be listed more than once because they may belong to more than one node group, including the HPOM for Windows predefined node groups. This is especially the case when the -recursive option is used on a higher-level node group or Root, and a larger number of node groups is affected.

`-group_path <hierarchy path>`

Specifies a node group by hierarchical path.

`-group_id <group id>`

Specifies a node group by its ID.

`-recursive`

Specifies that the child node group is searched recursively for containing nodes.

`-list_groups`

Lists all node groups on this management server.

`-change_comm_opt`

Changes the communication options of a single node or of nodes belonging to specific node group.

`-node_name <primary node name>`

Specifies a node by its PrimaryNodeName property.

`-node_id <node id>)`

Specifies a node by its ID.

`-group_path <hierarchy path>`

Specifies a node group by hierarchical path.

`-group_id <group id>`

Specifies a node group by its ID.

`-comm_path <communication path>`

Sets the CommunicationPath property of the node. It can contain a DNS name, a WINS name, or an IP. If a node group is selected, this option is not allowed.

`-always_resolve_comm_path (yes | no)`

If set to yes , the communication to the node is always resolved before contacting the agent (DHCP).

`-auto_update_comm_path (yes | no)`

If set to yes , the CommunicationPath property is updated automatically by the agent.

`-deassign_tool`

De-assigns a tool (OV_Action object) from a node or node group (but only one entity at a time).

`-node_name <primary node name>`

Specifies a node by its PrimaryNodeName property.

`-node_id <node id>)`

Specifies a node by its ID.

`-group_path <hierarchy path>`

Specifies a node group by hierarchical path.

`-group_id <group id>`

Specifies a node group by its ID.

`-tool_path <tool hierarchy path>`

Specifies a tool to be de-assigned by hierarchical path, which is a directory-like path of tool group names (Captions/Display Name properties) that uniquely identifies a group. A backslash (\ ) is used as a separator. The tool name must appear at the end of the path.

`-tool_id <tool id>)`

Specifies a tool to be de-assigned by its ID.

`-deassign_autodeploy_pg`

De-assigns an auto-deployment policy group from a node group.

`-group_path <hierarchy path>`

Specifies a node group by hierarchical path.

`-group_id <group id>`

Specifies a node group by its ID.

`-pg_path <policy hierarchy path>`

Specifies a policy group to be de-assigned by hierarchical path, which is a directory-like path of policy group names (Captions/Display Name properties) that uniquely identifies a group. A backslash (\ ) is used as a separator.

`-pg_id <policy group id>`

Specifies a policy group to be de-assigned by its ID.

`-delete_report`

Deletes a report from a node group.

`-group_path <hierarchy path>`

Specifies a node group by hierarchical path.

`-group_id <group id>`

Specifies a node group by its ID.

`-report_name <report name>`

Specifies a report to be deleted.

`-delete_graph`

Deletes graphing information from a node group.

`-group_path <hierarchy path>`

Specifies a node group by hierarchical path.

`-group_id <group id>`

Specifies a node group by its ID.

`-outage_node`

Sets or gets the outage state of nodes, node groups, or both.

`-scheduled`

Sets the scheduled outage mode. (Used in case of scheduled outages.)

`-unplanned`

Sets the unplanned outage mode. (Used in case of maintenance.)

`-node_name <primary node name>`

Specifies a node by its PrimaryNodeName property.

`-node_id <node id>`

Specifies a node by its ID.

`-group_path <hierarchy path>`

Specifies a node group by hierarchical path.

`-group_id <group id>`

Specifies a node group by its ID.

`-on`

Enables the outage mode for a specified node or node group.

`-off`

Disables the outage mode for a specified node or node group.

`-toggle`

Switches the node or node group outage state to the opposite state (from `-on` to `-off` , then from `-off` to `-on` , and so on).

`-get`

Returns the outage state (on or `off`) for a specified node or node group. If neither switch (`-scheduled` or `-unplanned`) is defined, the outage states of both modes are returned.

NOTE:
If one of the "set outage state" switches (`-on` , `-off` , or `-toggle`) is specified, at least one of the "external node or node group specification" switches (`-node_name` , `-node_id` , `-group_path` , or `-group_id`) must be specified. If the `-get` switch is specified, and none of the "node or node group specification" switches is specified, the tool returns the outage state for all nodes.

`-delete_msgs`

Specifies a messages to be deleted during an outage. Otherwise, messages are acknowledged automatically.

`-disable_heartbeat`

Specifies heartbeat polling to be disabled during an outage.

`-node_name, -node_id, -group_path, -group_id`

Specifies at least one node or node group. To specify more than one node or node group, you can use multiple switches. Each switch can have one or more parameters.

`-detailed`

Specifies the detailed format of output.

`-single_line`

Specifies the compressed format of output that is printed in one line.

`-show_outage_only`

Specifies the detailed format of output. It is the same as detailed format, except it shows only nodes that are in at least one of the outage modes.

`-add_exnode`

Adds external node to this management server.

`-caption <caption name>`

Sets the Caption property of the new node.

`-group_path <hierarchy path>`

Sets a parent node group of the new node by hierarchical path. The node group must exist.

`-group_id <group id>`

Sets a parent node group of the new node by its ID. The node group must exist.

`-ip_address <IP address pattern>`

Specifies the pattern data type as IP address and sets `<IP address pattern>` as object pattern.

`-ip_name <IP name pattern>`

Specifies pattern data type as IP name and sets `<IP a name pattern>` as an object pattern.

`-other <other pattern>`

Specifies the pattern data type as other and sets `<other pattern>` as an object pattern.

`-order <order number>`

Specifies the order in which the external nodes are evaluated. This is an optional parameter. Default is `0` .

`-check_before_managed_nodes`

Specifies that the external node is checked before the managed nodes. If this parameter is not specified, the external node is not checked before managed nodes.

`-delete_exnode`

Deletes an external node from this management server.

`-exnode_path <external node hierarchical path>`

Specifies an external node by hierarchical path.

`-exnode_id <unique_node_id>`

Specifies an external node by its ID.

`-assign_exnode`

Assigns an external node to a node group.

`-exnode_path <external node hierarchical path>`

Specifies an external node by hierarchical path.

`-exnode_id <unique_node_id>`

Specifies an external node by its ID.

`-group_path <hierarchy path>`

Sets a parent node group of the new node by hierarchical path. The node group must exist.

`-group_id <group id>`

Sets a parent node group of the new node by its ID. The node group must exist.

`-deassign_exnode`

De-assigns an external node from a node group.

`-exnode_path <external node hierarchical path>`

Specifies an external node by hierarchical path.

`-exnode_id <unique_node_id>`

Specifies an external node by its ID.

`-group_path <hierarchy path>`

Sets a parent node group of the new node by hierarchical path. The node group must exist.

`-group_id <group id>`

Sets a parent node group of the new node by its ID. The node group must exist.

`-list_exnodes`

Lists external nodes belonging to a specific node group. If no node group is specified, Root is used as default. Some external nodes may be listed more than once because they may belong to more than one node group, including the HPOM for Windows predefined node groups. This is especially the case when the `-recursive` option is used on a higher-level node group or Root, and a larger number of node groups is affected.

`-group_path <hierarchy path>`

Specifies a node group by hierarchical path.

`-group_id <group id>`

Specifies a node group by its ID.

`-recursive`

Specifies that the child node group is searched recursively for containing external nodes.

`-outage_exnode`

Sets or gets the outage state of nodes, node groups, or both.

`-scheduled`
: Sets the scheduled outage mode. (Used in case of scheduled outages.)

`-unplanned`
: Sets the unplanned outage mode. (Used in case of maintenance.)

`-exnode_path <external node hierarchical path>`
: Specifies an external node by hierarchical path.

`-exnode_id <unique_node_id>`
: Specifies an external node by its ID.

`-group_path <hierarchy path>`
: Specifies a node group by hierarchical path.

`-group_id <group id>`
: Specifies a node group by its ID.

`-on`
: Enables the outage mode for a specified external node or node group.

`-off`
: Disables the outage mode for a specified external node or node group.

`-toggle`
: Switches the external node or node group outage state to the opposite state (from `-on` to `-off` , then from `-off` to `-on` , and so on).

`-get`
: Returns the outage state (on or `off`) for a specified external node or node group. If neither switch (`-scheduled` or `-unplanned`) is defined, the outage states of both modes are returned.

> NOTE:
> If one of the "set outage state" switches (`-on` , `-off` , or `-toggle`) is specified, at least one of the "external node or node group specification" switches (`-node_name` , `-node_id` , `-group_path` , or `-group_id`) must be specified. If the `-get` switch is specified, and none of the "node or node group specification" switches is specified, the tool returns the outage state for all external nodes.

`-delete_msgs`
: Specifies a messages to be deleted during an outage. Otherwise, messages are acknowledged automatically.

`-detailed`
: Specifies the detailed format of output.

`-single_line`
: Specifies the compressed format of output that is printed in one line.

`-show_outage_only`

Specifies the detailed format of output. It is the same as detailed format, except it shows only nodes that are in at least one of the outage modes.

Examples

```
ovownodeutil -help
```

Gets help and usage for the tool.

```
ovownodeutil -add_node -node_name kermit -group_path \cad\ulm
```

Adds a new node with the Primary Node Name `kermit` and assigns it to the existing node group `\cad\ulm` .

```
ovownodeutil -assign_node -node_name kermit -group_path \cad\koeln
```

Assigns an existing node with the Primary Node Name `kermit` to the existing node group `\cad\koeln` .

```
ovownodeutil -change_comm_opt -node_name kermit -always_resolve_comm_path yes
```

Changes the communication option for an existing node with the Primary Node Name `kermit` . Communication to the node is resolved before the agent is contacted.

```
ovownodeutil -outage_node -node_name Node1 Node2 -node_id Node1_ID -group_path
\Group1\SubBroup1 -group_id SubGroup1_ID -scheduled -on -delete_msgs –disable_heart_beat
```

Enables or disables outage of nodes or node groups.

```
ovownodeutil -outage_node -node_name Node2 -unplanned -get
```

Gets the state of the node outage.

OvOWNPReg

The `OvOWNPReg` tool enables you to register and unregister platforms and packages, as well as to update the prerequisite check configuration file.

There are three general scenarios for using the `OvOWNPReg` tool for registering a package:

- You want to register a patched version of an existing package. In this case, run the tool specifying the `-backup <BackupDir>` and `-reinstall` parameter in the command line. This stores the original package in the specified backup directory, and registers the updated package.
- You want to register a newer (higher) version of a package. In this case, run the tool to upload the new package to the server, and to register it. The old version of the package is not affected, and is retained on the server.
- You want to register exactly the same version of an existing package for the second time, because, for instance, you need to repair the application to restore proper functionality. In this case, run the tool with the `-reinstall` parameter, in addition to exactly the same parameters that were used for the first run. In this case, the log file and backup directories are not overwritten if they were created when the package was first registered. Failure to specify the `-reinstall` parameter for any repeated package registration results in serious problems.

NOTE:
Never attempt to register a version of a package that is lower than the version that already exists on the server.

Command synopsis

```
OvOWNPReg
  -reg <ConfigFile> -patch <PatchName> [-sections <sections>]
    [-log <UndoFile> -backup <BackupDir>] [-reinstall]

  -unreg <UndoFile> [-delundo]

  (-? | -help)
```

Tool options

`-help`

Show tool usage and description.

`-reg <ConfigFile>`

Registers new platforms and packages specified in the XML-based `<ConfigFile>` .

`-unreg <UndoFile>`

Unregisters platforms and packages specified in `<UndoFile>` .

`-patch <PatchName>`

String in the format `OVOW_xxxx` (for example, `OVOW_0200`).

`-reinstall`

You must specify this option when reinstalling the patch. Failure to specify the reinstall option when the patch is applied for the second time results in serious problems. (Reference counters are incremented for a second time. During patch uninstallation, platform objects are not removed because counters are not decremented to `0` .)

`-sections <sections>`

If you specify this parameter, only specified sections are read from the configuration file. Available sections are package and `prereq` . You must separate multiple sections with commas (for example, `-sections platform,package`).

`-log <UndoFile>`

If you specify this option, all changes are recorded in `<UndoFile>` . When you uninstall the patch, you can undo all the changes using `<UndoFile>` .

`-backup <BackupDir>`

Specify this option to perform a backup of an existing package before registering a patched version of the same package.

`-delundo`

If you specify this option, `<UndoFile>` is deleted if unregistration ended successfully.

ovowmomchk

The command ovowmomchk is a syntax check tool for "Manager of Manager" configuration files used for server-based flexible management message forwarding, for example, the file `MsgForwarding.ini` .

If the syntax of the configuration file is correct, ovowmomchk returns the message `Syntax of <file> is OK` . If the syntax is not correct, a list of appropriate errors is returned.

The command ovowmomchk does not check text that is entered before the first keyword. You must check text before the first keyword manually.

Command synopsis

```
ovowmomchk.exe [-p] [-help] <file>
```

Options

The ovowmomchk command supports the following options:

- -p Prints verbose information about the configuration file.
- -help Displays the usage text.

Exit Values

This command exits with value 0 if the syntax of the configuration file, starting from the first keyword, is correct. If the syntax is not correct, ovowmomchk exits with value 1 and a list of errors. If not enough memory is available, ovowmomchk returns value 2.

Related Files

```
%OvShareDir%\conf\MsgActSrv\MsgForwarding.ini
```

The `%OvShareDir%` is the shared directory which is defined by the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\HP OpenView\ShareDir` and which depends on the installation path that you provided during the installation of HPOM for Windows.

By default it is:

```
C:\Documents and Settings\All Users\Application Data\HP\HP BTO Software\shared
```

This file holds the currently active message forwarding configuration for server-based flexible management forwarding.

Examples

```
ovowmomchk.exe -p MsgForwarding.ini
```

ovowmsgutil

The `ovowmsgutil` utility allows you to run bulk operations on messages offline without using the HPOM console.

The following operations are possible:

- Change the state of messages (own, unown, acknowledge, unacknowledge)
- Change the severity of messages
- Delete messages from the management server database
- Download messages and related annotations
- Upload previously downloaded messages and related annotations

Because the utility changes the database directly, it stops some HPOM services (OvEpStatusEngine and OvEpMessageActionServer) before the update. After the database is updated, the tool automatically restarts affe HPOM services. This ensures that all HPOM components are provided with the actual data from the database.

The CSV format (MS Excel) is used for storing messages and annotations in files.

NOTE:
The `ovowmsgutil` tool is located in the Support directory of the HPOM installation. To run the tool, go to its par directory or specify its absolute path in the command line.

Operational requirements

- Runs on a HPOM management server
- user must be a member of the HP-OVE-ADMINS group or local administrators group
- The MSSQL$OVOPS service must be running to allow database access and updates (or the HPOM server must configured for the remote DB access)

Node name specifications

One of the possible message attributes for qualifying messages to be affected is the node name. You can specify names in several different ways:

- node's GUID
- node's primary node name
- node's caption
- node's communication path

Sample use cases

- Acknowledge messages generated by particular nodes, by some specified applications, and are older than a sp date and have severity normal
- Change severity for messages which were generated by a specified service on a specified day, from minor to c
- Delete messages which are still active and are older than a specified number of days and were generated on s nodes and have severity normal
- Unacknowledge those messages which belong to specified message groups and which have been acknowledge specified user
- Delete all acknowledged messages
- Delete all active and acknowledged messages
- Download messages coming from specific nodes and fulfilling certain conditions (state, severity, date). If need messages can be removed from HPOM.
- Upload messages (previously downloaded on this or another management server)

Command synopsis

```
ovowmsgutil <TypeOfChange> -exp <QualifyString> [-nodedelim <DelimChar>]
                               OR
ovowmsgutil <TypeOfChange>> -{fa|fu} <SqlFile> [-nodedelim <DelimChar>]
                               OR
ovowmsgutil -dnl <BaseFilename> [-remove] [-cp (UNICODE|MBYTE)]
            [-sep <FieldSeparator>>]
            -exp <QualifyString> [-nodedelim <DelimChar>]
                               OR
ovowmsgutil -dnl <BaseFilename> [-remove] [-cp <CodePage>]
            [-sep <FieldSeparator>]
            -{fa|fu} <SqlFile> [-nodedelim <DelimChar>]
                               OR
ovowmsgutil -upl <BaseFilename> [-replace]
                               OR
ovowmsgutil -help
```

Options

```
 <TypeOfChange>  Type of change you want to make. Possible values are:
                   -ack        Acknowledge qualified messages
```

```
                -unack      Unacknowledge qualified messages
                -own        Own qualified messages
                -disown     Disown qualified messages
                -del        Delete qualified messages
                -sev <Num>  Change severity of messages to level <Num>.
                            Possible values for <Num>:  1  Unknown
                                                        2  Normal
                                                        4  Warning
                                                        8  Minor
                                                       16  Major
                                                       32  Critical

-dnl            Download a subset of messages from HPOM DB.

-upl            Upload messages into HPOM DB. Messages must be stored in a
                file, previously generated with the -dnl option. All messages
                from the file will be uploaded.

<BaseFilename>  Partial file name to be used for download or upload of
                messages and annotations. _MSG.csv and _ANN.csv will be
                appended to the given name to get actual file names.

-remove         Causes removal of downloaded message and their annotations
                from HPOM DB.

-cp             Specifies the codepage of the generated files for messages
                and annotations. Two values are allowed: UNICODE (for Unicode
                encoding) and MBYTE (for MultiByte encoding). It defaults to
                MBYTE.

-sep            Sets the field separator for the resulting CSV files
                (defaults to comma ',').

<FieldSeparator> The field separator character to be used in the CSV files.

-replace        Causes replacement of existing messages and annotations with
                data from the files during upload.

-exp            Expression option to qualify messages you want to change or
                download.
    <QualifyString> String which qualifies the messages to be changed or
                downloaded. Syntax is similar to an SQL expression as used in
                the WHERE clause of an SQL SELECT statement. The expression
                can contain the following items for qualifying the messages:
                State                         Current message state.
                                              Possible values:
                                                 1   Undefined
                                                 2   Unowned
```

```
                                         3   Owned
                                         4   Acknowledged
                                         5   Node of message has been
                                             deleted
                                         6   Message is marked for
                                             deletion
                TimeOfStateChangeTimeStamp    Time of last state change
                UserOfStateChange             Name of user who made the last
                                              state change
                NodeName                      Node name where the message
                                              originates from. Possible node
                                              name specifications are:
                                                 Primary node name,
                                                 Caption,
                                                 Communication path.
                                              Above specifications have to be
                                              included between delimiters.
                                              See -nodedelim command option
                                              below. Alternately, node GUID's
                                              enclosed between {} can be used
                                              as well
                TimeCreatedTimeStamp          Time when message has been
                                              created
                TimeReceivedTimeStamp         Time when message has been
                                              received on server
                MessageGroup                  Message group
                Object                        Object message is associated with
                Application                   Application message is associated
                                              with
                Type                          Message type
                ServiceId                     Service message is associated with

                Severity                      Message severity. For possible
                                              values see above
                LogOnly                       Message is log only
                Unmatched                     Message is unmatched

                NOTE: instead of using an expression you can specify * as ALL
                      operator. In this case all messages will qualify for
                      the selected change.

-nodedelim      Specifies a character which is used as delimiter for node names.
                Node names have to be included between 2 delimiters. Default
                delimiter is the '@' character.
                If node GUID's are specified instead of node names then the
                GUID's have to be included between '{' and '}' characters.
                For GUID's the delimiter parameter is ignored.
```

```
   <DelimChar>     Delimiter character for node names.

   -{fa|fu}        Use the parameter FA or FU if the expression which qualifies
                   the messages to be changed is stored in a file. This makes
                   sense for long and complex expressions. FA specifies that
                   the file uses WINANSI character encoding, FU specifies that
                   the file uses UNICODE character encoding.

   <SqlFile>       Name of the file which contains the qualifying expression.

   -help           Displays this help text.
```

Examples

```
ovowmsgutil -ack -exp "(NodeName='#hpspi011#' OR NodeName='#jupiter#') AND Severity = 2" -noded

            Acknowledge messages originating from the nodes hpspi011 or jupiter
            with severity normal. Use the # character as delimiter for node names.

ovowmsgutil -sev 32 -exp "NodeName='@sap002@' AND Application='R3' AND Type = 'Error'"

            Change the severity of messages to critical which originate from node sap002 and fr
            application R3 and are of type Error.

ovowmsgutil -del -exp "State = 4 AND TimeOfStateChangeTimeStamp < '2000-01-31 13:45:00'"

            Delete all messages which have been acknowledged before january 31st 2000 1:45:00PM
            Note: time stamps have the format "YYYY-MM-DD HH:MM:SS" or "YYYY-MM-DD".

ovowmsgutil -disown -fa "c:\Program Files\Hewlett-Packard\OvEnterprise\ExpFiles\Disown1"

            Disown messages. The expression which qualifies the messages to be disowned is
            stored in file "c:\Program Files\Hewlett-Packard\OvEnterprise\ExpFiles\Disown1"
            The file is WINANSI encoded.

ovowmsgutil -unack -exp *

            Unacknowledge all messages regardless of their state.

ovowmsgutil -dnl c:\temp\test -exp "State <> 4 AND Severity = 32" -cp UNICODE -sep ;

            Download all messages (and their annotations) with critical severity and state diff
            than "Acknowledged". The resulting files (c:\temp\test_MSG.csv and c:\temp\test_ANN
            will be saved in Unicode encoding and the CSV field separator will be semicolon ";"

ovowmsgutil -upl c:\temp\test -replace

            Upload messages and annotations from c:\temp\test_MSG.csv and c:\temp\test_ANN.csv
            any existing records in the DB with records from the files.
```

ovowreqcheck

The tool ovowreqcheck enables remote HP Operations agent prerequisite checking from command line on HPOM for Windows server and console. The tool can be used to:

- Check HP Operations agent prerequisites for remote nodes (Windows or UNIX)
- Get the list of all prerequisites for a specific system
- Get the list of all supported systems

Command Synopsis

```
ovowreqcheck -srv <HP Operations server> (-det | -sum)
             (( -node <node>
                [-agt_comm_type<agent_comm_type>] [-agt_bin_format <agent_bin_format>]
                [-plat (WINDOWS | UNIX)] [-u <user> [-p <password>]] ) |
              -file <file>)

ovowreqcheck -srv <HP Operations server> -req [<system>]
             [-agt_comm_type<agent_comm_type>] [-agt_bin_format <agent_bin_format>]

ovowreqcheck -srv <HP Operations server> -allsystems

ovowreqcheck [-? | -help ]
```

`-help`	Prints tool usage and description.
`-srv <HP Operations server>`	Defines HPOM management server.
`-det`	Prints detailed result of requirements and recommendations validation. All prerequisites are displayed regardless if they have passed or failed.
`-sum`	Prints summary result of requirements and recommendations validation. Only failed prerequisites are displayed.
`-node <node>`	Name of the node to be checked.
`-agt_comm_type <agent_comm_type>`	*Optional.* Specifies the agent communication type of the node. If you specify `-det` or `-sum` , and <node> is not set up as managed node on the HPOM management server, then this parameter is required. (An error message

	displays asking you to provide this parameter to get a meaningful result.) Valid values for `<agent_comm_type>` are `DCE` and `HTTPS` .
`-agt_bin_format <agent_bin_format>`	*Optional.* Defines the agent binary format of the node. If you specify `-det` or `-sum` , and `<node>` is not set up as managed node on the HPOM management server, then this parameter is required. (An error message displays asking you to provide this parameter to get a meaningful result.) Valid values for `<agent_bin_format>` are as follows: ■ `Alpha` ■ `IPF32` ■ `IPF64` ■ `PA-RISC` ■ `PowerPC` ■ `SPARC` ■ `x64` ■ `x86`
`-plat (WINDOWS|UNIX)`	*Optional.* Defines platform of the node (`WINDOWS` or `UNIX`). The default is `WINDOWS` .
`-u <user> -p <password>`	*Optional.* Login information. Not required for Windows nodes where an interactive user has administrative privileges. This is mainly used for checking UNIX nodes.
`-file <file>`	Reads node names to be checked from a file. `<file>` must contain each node in a separate line (with or without corresponding optional parameters `-u` , `-p` , `-plat`).
`-req [<system>]`	Shows requirements and recommendations for the specified operating system. If the system is not given, it shows prerequisites for all supported systems.
`-allsystems`	Prints the list of all supported systems.

Exit Values

Exit value indicates prerequisite checking status:

0 - All checked prerequisites (requirements and recommendations) are OK.

1 - All requirements are OK; at least one recommendation has failed.

2 - All requirements are OK; at least one recommendation could not be checked.

3 - At least one requirement has failed.

55 - No prerequisites specified for <system> system.

101 - Client does not have administrative privileges on the node. Prerequisites can be checked only with administrative rights.

102 - At least one requirement could not be checked (it is unknown whether requirement is OK or not); all other successfully checked requirements are OK.

103 - Checked node is not available; either there is no network connection or firewall ports are not opened.

104 - Node (name) cannot be resolved

105 - The platform/OS version on node xxx may not yet be supported - consult the latest support matrix ; if platform is supported ignore prerequisite check and check prerequisites manually.

106 - Node's platform properties (System Type, Operating System, Version) are not set.

107 - Cannot discover platform.

Restrictions

This command can only be issued by HPOM administrators.

Examples

- Check prerequisites for Windows node `mynode1.your.com` . Display all prerequisites with associated status (`PASS` , `FAIL`):

  ```
  ovowreqcheck -srv myovoserver.your.com -det –node mynode1.your.com -agt_comm_type
  HTTPS -agt_bin_format x86
  ```

- Check prerequisites for UNIX node `mynode2.your.com` . Display all prerequisites with associated status (`PASS` , `FAIL`):

  ```
  ovowreqcheck -srv myovoserver.your.com -det –node mynode2.your.com -agt_comm_type
  HTTPS -agt_bin_format PA-RISC -plat UNIX -u myusername -p mypassword
  ```

- Check prerequisites for several nodes specified in a file:

a. Create an ASCII text file `nodes.txt` with content similar to the following:

   ```
   mynode1.your.com
   mynode2.your.com -plat UNIX -u myusername -p mypassword
   ```

b. Run: `ovowreqcheck -srv myovoserver.your.com -det -file nodes.txt`

- Check prerequisites for node `mynode.your.com` . Display only failed prerequisites:

  ```
  ovowreqcheck -srv myovoserver.your.com -sum –node mynode1.your.com -agt_comm_type
  HTTPS -agt_bin_format x86
  ```

- Display prerequisites for Windows Vista system with HTTPS agent. Do not check the prerequisites:

  ```
  ovowreqcheck -srv myovoserver.your.com -req WindowsVista_HTTPS
  ```

- Display all supported systems:

  ```
  ovowreqcheck -srv myovoserver.your.com -allsystems
  ```

ovowserviceutil

The ovowserviceutil tool allows command-line management of services. Specific services and service sub-trees are referred to in several different ways.

Services

Services are referred to by service name and ID:

- Service name (-service_name)

  Represents the Caption property of the OV_Service object that uniquely identifies a service. The parameter should also include the full hierarchy path. The path must always start with a single backslash (\ ), which denotes the Root.

  NOTE:
  To access the root, use a single slash (\ ) only.

- Service ID (-service_id)

  Represents the Name property of the OV_Service object that uniquely identifies a service. It is usually a GUID, but can also be any other unique string.

Service sub-trees

Service sub-trees are referred to by service name and ID:

- Service name (-service_hierarchy_name)

  Same as -service_name , except a whole tree of services gets managed.

- Service ID (-service_hierarchy_id)

  Same as -service_id , except a whole tree of services gets managed.

Command synopsis

```
ovowserviceutil
   [ -help ]
   [ -add_service -service_name <service path>
               [ -service_id <service id>]
               -service_type <service type>
               [ -virtual_service | -hosted_on_node <host node> |
               -hosted_on_nodegroup <host node group> ] ]
   [ -delete_service (-service_name <service path> |
               -service_id <service id>) ]
   [ -list_sub_services (-service_name <service path> |
               -service_id <service id>) [ -recursive]
               [-by_name | -by_id | -detailed] ]
   [ -list_antecedent_services (-service_name <service path> |
               -service_id <service id>)
               [-by_name | -by_id | -detailed] ]
   [ -rename_service (-service_name <service path> |
               -service_id <service id>)
               -new_name <new service caption> ]
   [ -set_service_calc_rule (-service_name <service path> |
               -service_id <service id>)
               -calc_rule <service calculation rule> ]
   [ -set_service_msg_prop_rule (-service_name <service path> |
               -service_id <service id>)
               -prop_rule <service propagation rule> ]
   [ -set_service_comp_prop_rule (-service_name <service path> |
               -service_id <service id>)
               -prop_rule <service propagation rule>
               (-child_service_name <child service path> |
               -child_service_id <child service id>) ]
   [ -set_service_depend_prop_rule (-service_name <service path> |
               -service_id <service id>)
               -prop_rule <service propagation rule>
               (-antecedent_service_name <dependant service path> |
               -antecedent_service_id <dependant service id>) ]
   [ -outage_service [-scheduled | -unplanned]
               [-service_name {<service name>} |
               -service_id {<service id> }]
               [-service_hierarchy_name {<service hierarchy name>} |
               -service_hierarchy_id {<service hierarchy id>}]
               -get [-single_line | -detailed | -show_outage_only] ]
   [ -outage_service [-scheduled | -unplanned ]
               [-service_name {<service name>} |
               -service_id {<service id> }]
               [-service_hierarchy_name {<service hierarchy name>} |
               -service_hierarchy_id {<service hierarchy id>}]
               (-on | -off | -toggle)
               [ -delete_msgs ] ]
   [ -add_dependency (-service_name <service path> |
               -service_id <service id>)
               (-antecedent_service_name <antecedent service path> |
               -antecedent_service_id <antecedent service id>) ]
   [ -delete_dependency (-service_name <service path> |
               -service_id <service id>)
               (-antecedent_service_name <antecedent service path> |
               -antecedent_service_id <antecedent service id>) ]
   [ -list_sub_service_types -service_type_name <service type path>
               [ -recursive] ]
   [ -list_antecedent_service_types
               -service_type_name <service type path> ]
   [ -set_service_type_calc_rule -service_type_name <service type path>
               -calc_rule <calculation rule>
   [ -set_service_type_msg_prop_rule -service_type_name <service type path>
               -prop_rule <propagation rule>
   [ -set_service_type_comp_prop_rule
               -service_type_name <service type path>
               -prop_rule <propagation rule>
               -child_service_type_name <child service type path>
   [ -set_service_type_depend_prop_rule
               -service_type_name <service type path>
               -prop_rule <propagation rule>
               -antecedent_service_type_name <antecedent service type path>
   [ -add_calc_rule -rule_name <new calculation rule name>
               [ -rule_id <propagation rule id>]
               [ -most_critical ] ]
   [ -add_calc_rule -rule_name <new calculation rule name>
               [ -rule_id <propagation rule id>]
               -single_threshold [ -by_percentage | -by_value ]
               [ -threshold <threshold value>]
               [ -set_to_value <severity>] ]
   [ -add_calc_rule -rule_name <new calculation rule name>
               [ -rule_id <propagation rule id>]
               -multi_threshold [ -by_percentage | -by_value ]
               [ -normal_threshold <threshold value>]
               [ -normal_set_to_value <severity>]
               [ -warning_threshold <threshold value>]
               [ -warning_set_to_value <severity>]
               [ -minor_threshold <threshold value>]
               [ -minor_set_to_value <severity>]
               [ -major_threshold <threshold value>]
               [ -major_set_to_value <severity>]
               [ -critical_threshold <threshold value>]
               [ -critical_set_to_value <severity>]
   [ -delete_calc_rule -rule_name <calculation rule name> ]
   [ -change_calc_rule -rule_name <new calculation rule name>
               [ -most_critical ] ]
   [ -change_calc_rule -rule_name <new calculation rule name>
               -single_threshold [ -by_percentage | -by_value ]
               [ -threshold <threshold value>]
               [ -set_to_value <severity>] ]
   [ -change_calc_rule -rule_name <new calculation rule name>
               -multi_threshold [ -by_percentage | -by_value ]
               [ -normal_threshold <threshold value>]
               [ -normal_set_to_value <severity>]
               [ -warning_threshold <threshold value>]
               [ -warning_set_to_value <severity>]
               [ -minor_threshold <threshold value>]
               [ -minor_set_to_value <severity>]
               [ -major_threshold <threshold value>]
               [ -major_set_to_value <severity>]
               [ -critical_threshold <threshold value>]
               [ -critical_set_to_value <severity>]
   [ -add_prop_rule -rule_name <new propagation rule name> [ -simple]
               [ -unchanged | -ignored | -fixed_as [<serverity>] |
               -relative [<relative>] ] ]
   [ -add_prop_rule -rule_name <new propagation rule name>
               -severity_based
               [ -unchanged | -ignored | -fixed_as [<serverity>] |
               -relative [<relative>] ]
               [ -prop_normal <propagation>]
               [ -prop_warning <propagation>]
               [ -prop_minor <propagation>]
               [ -prop_major <propagation>]
               [ -prop_critical <propagation>] ]
   [ -delete_prop_rule -rule_name <new propagation rule name> ]
   [ -change_prop_rule -rule_name <new propagation rule name> [ -simple]
               [ -unchanged | -ignored | -fixed_as [<serverity>] |
               -relative [<relative>] ] ]
   [ -change_prop_rule -rule_name <new propagation rule name>
               -severity_based
               [ -unchanged | -ignored | -fixed_as [<serverity>] |
               -relative [<relative>] ]
               [ -prop_normal <propagation>]
               [ -prop_warning <propagation>]
               [ -prop_minor <propagation>]
               [ -prop_major <propagation>]
               [ -prop_critical <propagation>] ]
```

Tool options

`-help`

Show tool usage and description.

`-add_service`

Adds a service to this management server.

`-service_name <service path>`
Specifies a service by its Caption property.

`-service_id <service id>`
Specifies a service by its Name property.

`-service_type <service type>`
Specifies a service type by its Name property.

`-virtual_service`
Specifies a virtual service with no node on which to reside.

`-hosted_on <host node>`
Specifies a service host node by its PrimaryNodeName property.

`-hosted_on_nodegroup <host node group>`
Specifies a service host node group by its Name property.

`-delete_service`
Deletes a service from this management server.

`-service_name <service path>`
Specifies a service by its Caption property.

`-service_id <service id>`
Specifies a service by its Name property.

`-list_sub_services`
Lists subservices.

`-service_name <service path>`
Specifies a service by its Caption property.

`-service_id <service id>`
Specifies a service by its Name property.

`-recursive`
Specifies a listing of all subservices. Without this parameter, only child services are listed.

`-by_name`
Prints services by name. (This option is used if `-by_name` , `-by_id` , and `-detailed` are not specified.)

`-by_id`
Prints services by ID.

`-detailed`
Prints service details (by name and ID).

`-list_antecedent_services`
Lists antecedent services.

`-service_name <service path>`
Specifies a service by its Caption property.

`-service_id <service id>`
Specifies a service by its Name property.

`-by_name`
Prints services by name. (This option is used if `-by_name` , `-by_id` , and `-detailed` are not specified.)

`-by_id`
Prints services by ID.

`-detailed`
Prints service details (by name and ID).

`-rename_service`
Renames a service.

`-service_name <service path>`
Specifies a service by its Caption property.

`-service_id <service id>`
Specifies a service by its Name property.

`-new_name <new service name>`
Specifies a new name for the service Caption property. (The `<new service name>` does not have a hierarchy path.) The Caption may not begin or end with a backslash (`\`). If Captions contain backslashes, they should be escaped with additional backslashes (for example, `\\`).

`-set_service_calc_rule`
Sets a service calculation rule.

`-service_name <service path>`
Specifies a service by its Caption property.

`-service_id <service id>`
Specifies a service by its Name property.

`-calc_rule <service calculation rule>`
Specifies a calculation rule by its Caption property.

`-set_service_msg_prop_rule`
Sets a service message propagation rule.

`-service_name <service path>`
Specifies a service by its Caption property.

`-service_id <service id>`
Specifies a service by its Name property.

`-prop_rule <service propagation rule>`
Specifies a propagation rule by its Caption property.

`-set_service_comp_prop_rule`

Sets a service composition propagation rule.

`-service_name <service path>`

Specifies a service by its Caption property.

`-service_id <service id>`

Specifies a service by its Name property.

`-prop_rule <service propagation rule>`

Specifies a propagation rule by its Caption property.

`-child_service_name <child service path>`

Specifies a child service by its Caption property.

`-child_service_id <child service id>`

Specifies a child service by its Name property.

`-set_service_depend_prop_rule`

Sets a service propagation rule.

`-service_name <service path>`

Specifies a service by its Caption property.

`-service_id <service id>`

Specifies a service by its Name property.

`-prop_rule <service propagation rule>`

Specifies a propagation rule by its Caption property.

`-antecedent_service_name <dependant service path>`

Specifies a dependant service by its Caption property.

`-antecedent_service_id <dependant service id>`

Specifies a dependant service by its Name property.

`-outage_service`

Sets or gets the outage state of services.

`-scheduled`

Sets scheduled outage mode. (Used for scheduled outages.)

`-unplanned`

Sets unplanned outage mode. (Used for maintenance.)

`-service_name {<service name>}`

Specifies services by their Caption properties.

`-service_id <service id>`

Specifies services by their Name properties.

`-service_hierarchy_name <service hierarchy name>`

Specifies service trees by their Caption properties.

`-service_hierarchy_id <service hierarchy id>`
Specifies service trees by their Name properties.

`-on`
Enables outage for a specified service.

`-off`
Disables outage for a specified service.

`-toggle`
Switches the service outage state to the opposite state (on to off, off to on)

`-get`
Returns the outage state (on or off) for a specified service. If neither switch (-scheduled or -unplanned) is defined, the outage states of both modes are returned.

`-delete_msgs`
Specifies messages to be deleted during and outage. Otherwise, messages are automatically acknowledged.

`-service_name, -service_id, -service_hierarchy_name, -service_hierarchy_id`
Specifies at least one service. To specify more than one, you can use multiple switches. For each switch, you can use one or more parameters.

`-detailed`
Specifies the detailed format of output.

`-single_line`
Specifies the compressed format of output that is printed in one line.

`-show_outage_only`
Specifies the detailed format of output (same as detailed format) for services that are in at least one of the outage modes.

`-add_dependency`
Adds a service dependency.

`-service_name <service path>`
Specifies a service by its Caption property.

`-service_id <service id>`
Specifies a service by its Name property.

`-antecedent_service_name <dependant service path>`
Specifies an antecedent service by its Caption property.

`-antecedent_service_id <dependant service id>`
Specifies an antecedent service by its Name property.

`-delete_dependency`
Deletes a service dependency.

`-service_name <service path>`

Specifies a service by its Caption property.

`-service_id <service id>`

Specifies a service by its Name property.

`-antecedent_service_name <antecedent service path>`

Specifies an antecedent service by its Caption property.

`-antecedent_service_id <antecedent service id>`

Specifies an antecedent service by its Name property.

`-list_sub_service_types`

List subservice types.

`-service_type_name <service type path>`

Specifies a service type by its Caption property.

`-recursive`

Lists all subservice types. Without this parameter, only child services types are listed.

`-list_antecedent_service_types`

List dependant service types.

`-service_type_name <service type path>`

Specifies a service type by its Caption property.

`-set_service_type_calc_rule`

Sets a service type calculation rule.

`-service_type_name <service type path>`

Specifies a service type by its Caption property.

`-calc_rule <calculation rule>`

Specifies a calculation rule by its Caption property.

`-set_service_type_msg_prop_rule`

Sets a service type propagation rule.

`-service_type_name <service type path>`

Specifies a service type by its Caption property.

`-prop_rule <service propagation rule>`

Specifies a propagation rule by its Caption property.

`-set_service_type_comp_prop_rule`

Sets a child service type propagation rule.

`-service_type_name <service type path>`

Specifies a service type by its Caption property.

`-prop_rule <service propagation rule>`

Specifies a propagation rule by its Caption property.

`-child_service_type_name <child service type path>`

Specifies a child service type by its Caption property.

`-set_service_type_depend_prop_rule`

Sets a dependent service type propagation rule.

`-service_type_name <service type path>`

Specifies a service type by its Caption property.

`-prop_rule <service propagation rule>`

Specifies a propagation rule by its Caption property.

`-antecedent_service_type_name <antecedent service type path>`

Specifies an antecedent service type by its Caption property.

`-add_calc_rule`

Adds a calculation rule.

`-rule_name <new calculation rule name>`

Specifies a calculation rule name by its Caption property.

`-rule_id <propagation rule id>`

Specifies a calculation rule by its SettingID property.

`-most_critical`

Specifies a calculation rule that resets the most critical severity as a new value.

`-single_threshold`

Specifies a single-threshold calculation rule.

`-multi_threshold`

Specifies a multiple-threshold calculation rule.

`-by_percentage`

Specifies the evaluation of threshold values by percentage.

`-by_value`

Specifies the evaluation of threshold values by value.

`-threshold <threshold value>`

Specifies a threshold value for a single-threshold calculation rule.

`-set_to_value <severity>`

Specifies a severity value to be set when the threshold value of a single-threshold calculation rule is reached.

`-normal_threshold <threshold value>`

Specifies a normal threshold value for a multiple-threshold calculation rule.

`-normal_set_to_value <severity>`

Specifies a severity value to be set when the normal threshold value of a multiple-threshold calculation rule is reached.

`-warning_threshold <threshold value>`

Specifies a warning threshold value for a multiple-threshold calculation rule.

`-warning_set_to_value <severity>`

Specifies a severity value to be set when the warning threshold value of a multiple-threshold calculation rule is reached.

`-minor_threshold <threshold value>`

Specifies a minor threshold value for a multiple-threshold calculation rule.

`-minor_set_to_value <severity>`

Specifies a severity value to be set when the minor threshold value of a multiple-threshold calculation rule is reached.

`-major_threshold <threshold value>`

Specifies a major threshold value for a multiple-threshold calculation rule.

`-major_set_to_value <severity>`

Specifies a severity value to be set when the major threshold value of a multiple-threshold calculation rule is reached.

`-critical_threshold <threshold value>`

Specifies a critical threshold value for a multiple-threshold calculation rule.

`-critical_set_to_value <severity>`

Specifies a severity value to be set when the critical threshold value of a multiple-threshold calculation rule is reached.

`-delete_calc_rule`

Deletes a calculation rule.

`-rule_name <new calculation rule name>`

Specifies a calculation rule name by its Caption property.

`-change_calc_rule`

Changes a calculation rule.

`-rule_name <calculation rule name>`

Specifies a calculation rule name by its Caption property.

`-most_critical`

Specifies a calculation rule that resets the most critical severity as a new value.

`-single_threshold`

Specifies a single-threshold calculation rule.

`-multi_threshold`

Specifies a multiple-threshold calculation rule.

`-by_percentage`

Specifies the evaluation of threshold values by percentage.

`-by_value`

Specifies the evaluation of threshold values by value.

`-threshold <threshold value>`
    Specifies a threshold value for single-threshold calculation rule.

`-set_to_value <severity>`
    Specifies a severity value to be set when the threshold value of a single-threshold calculation rule is reached.

`-normal_threshold <threshold value>`
    Specifies a normal threshold value for a multiple-threshold calculation rule.

`-normal_set_to_value <severity>`
    Specifies a severity value to be set when the normal threshold value of multiple-threshold calculation rule is reached.

`-warning_threshold <threshold value>`
    Specifies a warning threshold value for a multiple-threshold calculation rule.

`-warning_set_to_value <severity>`
    Specifies a severity value to be set when the warning threshold value of a multiple-threshold calculation rule is reached.

`-minor_threshold <threshold value>`
    Specifies a minor threshold value for a multiple-threshold calculation rule.

`-minor_set_to_value <severity>`
    Specifies a severity value to be set when the minor threshold value of a multiple-threshold calculation rule is reached.

`-major_threshold <threshold value>`
    Specifies a major threshold value for a multiple-threshold calculation rule.

`-major_set_to_value <severity>`
    Specifies a severity value to be set when the major threshold value of a multiple-threshold calculation rule is reached.

`-critical_threshold <threshold value>`
    Specifies a critical threshold value for a multiple-threshold calculation rule.

`-critical_set_to_value <severity>`
    Specifies a severity value to be set when the critical threshold value of a multiple-threshold calculation rule is reached.

`-add_prop_rule`
  Adds a propagation rule.

`-rule_name <new propagation rule name>`
    Specifies a propagation rule name by its Caption property.

`-simple`
    Specifies a simple propagation rule.

`-severity_based`

Specifies a severity-based propagation rule.

`-unchanged`

Specifies a propagation rule that does not change the severity state that propagates.

`-ignored`

Specifies a propagation rule that ignores the severity state.

`-fixed_as <severity>`

Specifies a rule that propagates a fixed severity.

`-relative <relative>`

Specifies a rule that propagates a relative severity.

`-prop_normal <propagation>`

Specifies relative or absolute severity propagation for the normal severity of a severity-based propagation rule.

`-prop_warning <propagation>`

Specifies relative or absolute severity propagation for the warning severity of a severity-based propagation rule.

`-prop_minor <propagation>`

Specifies relative or absolute severity propagation for the minor severity of a severity-based propagation rule.

`-prop_major <propagation>`

Specifies relative or absolute severity propagation for the major severity of a severity-based propagation rule.

`-prop_critical <propagation>`

Specifies relative or absolute severity propagation for the critical severity of a severity-based propagation rule.

`-delete_prop_rule`

Deletes a propagation rule.

`-rule_name`

Specifies a propagation rule name by its Caption property.

`-change_prop_rule`

Changes a propagation rule.

`-rule_name <propagation rule name>`

Specifies a propagation rule name by its Caption property.

`-simple`

Specifies a simple propagation rule.

`-severity_based`

Specifies a severity-based propagation rule.

`-unchanged`

Specifies a propagation rule that does not change the severity state that propagates.

`-ignored`
: Specifies a propagation rule that ignores the severity state.

`-fixed_as <severity>`
: Specifies a rule that propagates a fixed severity.

`-relative <relative>`
: Specifies a rule that propagates a relative severity.

`-prop_normal <propagation>`
: Specifies relative or absolute severity propagation for the normal severity of a severity-based propagation rule.

`-prop_warning <propagation>`
: Specifies relative or absolute severity propagation for the warning severity of a severity-based propagation rule.

`-prop_minor <propagation>`
: Specifies relative or absolute severity propagation for the minor severity of a severity-based propagation rule.

`-prop_major <propagation>`
: Specifies relative or absolute severity propagation for the major severity of a severity-based propagation rule.

`-prop_critical <propagation>`
: Specifies relative or absolute severity propagation for the critical severity of a severity-based propagation rule.

Examples

Add a service to the HPOM for Windows management server WMI:

```
ovowserviceutil -add_service -service_name \Applications\Service1 -service_type
"\Application Services" -virtual_service
ovowserviceutil -add_service -service_name \Applications\Service1 -service_id
Service1_ID -service_type "\Application Services" -hosted_on_node \node1
ovowserviceutil -add_service -service_name \Applications\Service1 -service_id
Service1_ID -service_type "\Application Services" -hosted_on_nodegroup \nodegroup1
```

Delete a service from the HPOM for Windows management server WMI:

```
ovowserviceutil -delete_service -service_name \Applications\Service1
ovowserviceutil -delete_service -service_id Service1_ID
```

List child services:

```
ovowserviceutil -list_sub_services -service_name \Applications\Service1
ovowserviceutil -list_sub_services -service_id Service1_ID -recursive
```

List antecedent services:

```
ovowserviceutil -list_antecedent_services -service_name \Applications\Service1
ovowserviceutil -list_antecedent_services -service_id Service1_ID
```

Rename a service:

```
ovowserviceutil -rename_service -service_name \Applications\Service1 -new_name
Service2
ovowserviceutil -rename_service -service_id Service1_ID -new_name Service2
```

Set a service calculation rule:

```
ovowserviceutil -set_service_calc_rule -service_name \Applications\Service1 -calc_rule
rule_name1
ovowserviceutil -set_service_calc_rule -service_id Service1_ID -calc_rule rule_name1
```

Set a service message propagation rule:

```
ovowserviceutil -set_service_msg_prop_rule -service_name \Applications\Service1 -
prop_rule rule_name1
ovowserviceutil -set_service_msg_prop_rule -service_id Service1_ID -prop_rule
rule_name1
```

Set a service composition propagation rule:

```
ovowserviceutil -set_service_comp_prop_rule -service_name \Applications\Service1 -
prop_rule rule_name1 -child_service_name \Applications\Service1\SubService1
ovowserviceutil -set_service_comp_prop_rule -service_id Service1_ID -calc_rule
rule_name1 -child_service_id SubService1_ID
```

Set a service dependency propagation rule:

```
ovowserviceutil -set_service_depend_prop_rule -service_name \Applications\Service1 -
prop_rule rule_name1 -antecedent_service_name \Applications\Service2
ovowserviceutil -set_service_depend_prop_rule -service_id Service1_ID -prop_rule
rule_name1 -antecedent_service_name \Applications\Service2
```

Set a service outage:

```
ovowserviceutil -outage_service -service_name \Applications\Service1
\Applications\Service2 -scheduled -off -delete_msgs
ovowserviceutil -outage_service -service_hierarchy_name \Applications\Service1 -
unplanned -on -delete_msgs
```

Get the state of a service outage:

```
ovowserviceutil -outage_service -service_name \Applications\Service1 -scheduled -get
ovowserviceutil -outage_service -service_hierarchy_id Service1_ID -unplanned -get
```

Add a service dependency:

```
ovowserviceutil -add_dependency -service_name \Applications\Service1 -
antecedent_service_name \Service2
ovowserviceutil -add_dependency -service_ID Service1_ID -antecedent_service_id
Service2_ID
```

Delete a service dependency:

```
ovowserviceutil -delete_dependency -service_name \Applications\Service1 -
```

```
antecedent_service_name \Service2
ovowserviceutil -delete_dependency -service_ID Service1_ID -antecedent_service_id
Service2_ID
```

Add a calculation rule:

```
ovowserviceutil -add_calc_rule -rule_name CalcRule1 -most_critical
ovowserviceutil -add_calc_rule -rule_name CalcRule2 -single_threshold -by_percentage -
threshold 0.66 -set_to_value Minor
ovowserviceutil -add_calc_rule -rule_name CalcRule3 -multi_threshold -by_percentage -
warning_threshold 0.3 -major_threshold 0.66 -major_set_to_value Major
```

Delete calculation rule:

```
ovowserviceutil -delete_calc_rule -rule_name CalcRule1
```

Change a calculation rule:

```
ovowserviceutil -change_calc_rule -rule_name CalcRule1 -most_critical
ovowserviceutil -change_calc_rule -rule_name CalcRule2 -single_threshold -
by_percentage -threshold 0.66 -set_to_value Minor
ovowserviceutil -change_calc_rule -rule_name CalcRule3 -multi_threshold -by_percentage
-warning_threshold 0.3 -major_threshold 0.66 -major_set_to_value Major
```

Add a propagation rule:

```
ovowserviceutil -add_prop_rule –rule_name PropRule1 -simple -fixed_as Minor
ovowserviceutil -add_prop_rule –rule_name PropRule1 -severity_based -unchanged -
prop_minor Unchanged –prop_major Incr1
```

Delete a propagation rule:

```
ovowserviceutil -delete_prop_rule -rule_name PropRule1
```

Change a propagation rule:

```
ovowserviceutil -change_prop_rule -rule_name PropRule1 -simple -fixed_as Minor
ovowserviceutil -change_prop_rule -rule_name PropRule1 -severity_based -unchanged -
prop_minor Unchanged –prop_major Incr1
```

List child services types:

```
ovowserviceutil -list_sub_service_types -service_type_name "\System
Infrastructure\SystemServices"
ovowserviceutil -list_sub_service_types -service_type_name "\System
Infrastructure\SystemServices" -include_SPI_service_types -recursive
```

List child services types:

```
ovowserviceutil -list_antecedent_service_types -service_type_name "\System
Infrastructure\SystemServices"
ovowserviceutil -list_antecedent_service_types -service_type_name "\System
Infrastructure\SystemServices" -recursive
```

Rename service type:

```
ovowserviceutil -rename_service_type -service_type_name "\System
Infrastructure\ServiceType1" -new_name ServiceType2
ovowserviceutil -rename_service_type -service_type_name "\System
```

```
Infrastructure\ServiceType1" - new_id ServiceType2_ID
```

Set a service type calculation rule:

```
ovowserviceutil -set_service_type_calc_rule -service_type_name "\System
Infrastructure\ServiceType1" -calc_rule rule_name1
```

Set a service type message propagation rule:

```
ovowserviceutil -set_service_type_msg_prop_rule -service_type_name "\System
Infrastructure\ServiceType1" -prop_rule rule_name1
```

Set a service type composition propagation rule:

```
ovowserviceutil -set_service_type_comp_prop_rule -service_type_name "\System
Infrastructure\ServiceType1" -prop_rule rule_name1 -child_service_type_name "\System
Infrastructure\ServiceType1\SubServiceType1"
```

Set a service type dependency propagation rule:

```
ovowserviceutil -set_service_type_depend_ prop_rule -service_type_name "\System
Infrastructure\ServiceType1" -prop_rule rule_name1 antcedent_service_type_name
"\System Infrastructure\ServiceType2"
```

ovowtoolutil

The `ovowtoolutil` tool allows command-line management of tools and tool groups. Specific tools or tool groups are referred to in several different ways:

For tools

- By hierarchical path (-tool_path_parameter)

  A hierarchical path is a directory-like path of tool group names followed by a tool name that uniquely identifies a group. Captions/Display Names are used as names, backslash \ is used as a separator (for example, \rocco\cad\room108). The path must always start with \. A single \ denotes the Root. If a tool or tool group name itself contains a backslash, it must be escaped with an additional backslash (\\).
- By tool ID (-tool_id_parameter)

  Tool ID represents the Name property of the tool that uniquely identifies it. It is usualy a GUID but it can also be any other unique string.

For tool groups

- By hierarchical path (-group_path_parameter)

  A hierarchical path is a directory-like path of tool group names that uniquely identifies a group. Captions/Display Names are used as tool group names, backslash \ is used as a separator (for example, \rocco\cad\room108). The path must always start with \. A single \ denotes the Root. If group name itself contains a backslash, it must be escaped with an additional backslash (\\).
- By tool ID (-group_id_parameter)

  Group ID represents the Name property a tool group that uniquely identifies it. It is usually a GUID but it can also be any other unique string.

Command synopsis

```
ovowtoolutil
   [ -help ]
   [ -add_group -group_path <hierarchy path> ]
   [ -add_tool -type EXE|VBS|JS|WSH|PERL|URL -tool_name <tool name>
               (-cmd <command> | cmd_file <file name>)
               [-tool_id <tool id>] [-descr <descr>]
               [-group_path <hierarchy path> | group_id <group id>]
               [-output yes|no] [-params <params>] [-edit_params yes|no]
               [-req_pass yes|no] [-pass <password>]
               [-exec_as <user>] [-edit_login yes|no]
               [-exec_on MGMTSRV|NODE|NODELIST|CONSOLE]
               [-node_list_id <list of ids> |
                -node_list_pnn <list of prim. node names>] ]
   [ -delete_group (-group_path <hierarchy path> | -group_id <group id>) ]
   [ -delete_tool (-tool_path <tool hierarchy path> | tool_id <tool id>) ]
   [ -rename_group (-group_path <hierarchy path> | -group_id <group id>)
                   (-new_id <new tool id> -new_name <new tool name> |
                   (-new_id <new tool id> -new_name <tool name>) ]
   [ -rename_tool (-tool_path <tool hierarchy path> | tool_id <tool id>=>)
                  (-new_id <new tool id> -new_name <new tool name> |
                  (-new_id <new tool id> | -new_name <new tool name>) ]
```

Options

```
-help         Show tool usage and description.

-add_group    Add a tool group to this management server.
   -group_path <hierarchy path>
      Specifies a tool group by hierarchical path. All non-existing groups
      in the path will be created.

-add_tool     Add a tool to this management server.
   -type EXE|VBS|JS|WSH|PERL|URL
      Specifies the type of tool: EXE=executable, VBS=Visual Basic script,
      JS=JScript, WSH=Windows Scripting Host script, PERL=Perl script
      URL=internet URL
   -tool_name <tool name>
      Specifies the name (caption) of the tool.
   -cmd <command> | cmd_file <file name>
      Specifies the command to be executed. It can be an executable path or
      a script (given directly on command line). To read script from a file
      use -cmd_file to specify a file path.
   -tool_id <tool id>]
      Specifies the id of the tool. If not given, new GUID will be generated.
   -descr <descr>
      Specifies a description of the tool.
   -group_path <hierarchy path> | group_id <group id>
      Specifies a parent tool group for the new tool, either as hierarchy
      path or as id.
   -output yes|no
      Specifies whether tool has output or not (defaults to "no").
   -params <params>
      Specifies additional parameters of the tool
      (applicable only for executables).
   -edit_params yes|no
      Specifies whether operator will be allowed to edit parameters or not
      (defaults to "no").
   -req_pass yes|no
      Specifies whether password is required to run the tool (defaults to "no"").
   -pass <password>
      Specifies the password for the tool.
   -exec_as <user>
      Specifies the user account under which the tool will run.
   -edit_login yes|no
      Specifies whether operator will be allowed to edit user information
      (defaults to "no").
   -exec_on MGMTSRV|NODE|NODELIST|CONSOLE
      Specifies the target to run the tool on: MGMTSRV=management server,
      NODE=selected node, NODELIST=list of nodes, CONSOLE=console.
   -node_list_id <list of ids> | -node_list_pnn <list of prim. node names>
      Specifies a list of nodes (to run the tool on), either as list of ids or
      a list of primary node names. Items must be separated by commas (,).

-delete_group     Delete a tool group from this management server.
   -group_path <hierarchy path>
      Specifies a tool group to be deleted by hierarchical path.
   -group_id <group id>
      Specifies a tool group to be deleted by its ID.

-delete_tool      Delete a tool from this management server.
   -tool_path <hierarchy path>
      Specifies a tool to be deleted by hierarchical path.
   -tool_id <group id>
      Specifies a tool to be deleted by its ID.

-rename_group     Rename a tool group (its name, id or both).
   -group_path <hierarchy path>
      Specifies a tool group to be renamed by hierarchical path.
   -group_id <group id>
      Specifies a tool group to be renamed by its Id.
   -new_id <new group id>
      Specifies the new Id of the group.
   -new_name <new group name>
      Specifies the new name of the group.

-rename_tool      Rename a tool (its name, id or both).
   -tool_path <hierarchy path>
      Specifies a tool to be renamed by hierarchical path.
   -tool_id <group id>
      Specifies a tool to be renamed by its ID.
   -new_id <new tool id>
      Specifies the new Id of the tool.
   -new_name <new tool name>
      Specifies the new name of the tool.
```

Examples

```
ovowtoolutil -help
  outputs help and usage of the tool

ovowtoolutil -add_tool -type EXE -tool_name MY_TOOL -cmd mytool.exe
  creates a new tool named MY_TOOL that will be started through the mytool.exe
  executable; the tool is assigned to the root tool group as no specific tool group
  was given

ovowtoolutil -rename_tool -tool_path \test_tools\beep -new_name my_beep
  renames the tool caption from beep from tool group \test_tools to my_beep
```

ovtrap2opc

The ovtrap2opc utility is a command line support tool, used for converting an HP NNM `trapd.conf` file into an HPOM for Windows SNMP Interceptor policy. The tool is part of the HPOM management server.

The ovtrap2opc command line tool supports several parameters with arguments, as shown:

```
ovtrap2opc -trap <trapd_conf_file>
                  -pol <policy_name>
                  [-out <policy_files_folder>]
                  [-msgtype]
                  [-c ASCII|-c UTF8|-c ISO81|-c ISO82|-c ISO85|-c ROMAN8|
                   -c SJIS|-c EUCJP|-c GB2312|-c BIG5|-c EUCTW|-c EUCKR]
```

Options

- -trap <trapd_conf_file> HP NNM trap file (trapd.conf)
- -pol <policy_name> SNMP Interceptor policy name
- -out <policy_files_folder> Output directory for the generated policy files; If not specified, the current directory is used.
- -msgtype Sets the "Message Type" message property value to the event name.
- -c code set of <trapd_conf_files>; the default is ASCII.

In case the code set of the configuration file is other then ASCII, the –c parameter is required. This parameter specifies the code set of the input trap configuration file.

Using the ovtrap2opc tool, you can create an SNMP Interceptor policy based on the HP NNM trap file `trapd.conf` . The SNMP interceptor policy is created in binary and text versions (.data and .header files). Policy files are created and written to the current directory if the ovtrap2opc tool parameter "-out" is not specified.

To upload a policy

You can automatically upload a newly-created policy to HPOM. When the SNMP policy is created, you are asked if you want to upload the newly-created SNMP policy into HPOM. If you answer yes, the policy is uploaded.

The created policy (either binary or text version) can also be manually uploaded with the `ovpmutil` utility.

The binary policy version can be uploaded with this command:

`ovpmutil.exe reg pol "<policy_name >"`

The text policy version can be uploaded with this command:

`ovpmutil pcv /c "<policy_name>"` (convert text to binary policy)

`ovpmutil reg pol "<policy_name> "` (upload newly-created policy)

Examples

Generate the SNMP Interceptor policy on the English system (English HP NNM):

```
ovtrap2opc -trap "C:\Program Files\HP\HP BTO Software\conf\C\trapd.conf" -pol
"TestSNMPPolicy"
```

Generate the SNMP Interceptor policy on the Japanese system (Japanese HP NNM)

```
ovtrap2opc -trap "C:\Program Files\HP\HP BTO
Software\NNM\conf\Japanese_Japan.932/trapd.conf" -pol "TestSNMPPol2" -c SJIS
```

Command-Line Programs

HP Operations provides several command-line programs that allow you to interact directly with the agent, instead of using a policy. The commands are:

- opcmsg

  The command opcmsg submits a message for HP Operations Manager.

- ovpmutil

  The command ovpmutil.exe allows you to perform some configuration tasks from the command line.

- opcagt

  The command opcagt administers the agent processes running on an HP Operations managed node.

- opcmon

  The command opcmon is the commandline interface to the monitor agent.

- opcntprocs

  The command opcntprocs is used in conjunction with a measurement threshold policy to check if a particular process is running on a managed node.

- ovpmpwutil

  The command ovpmpwutil is used to change the user name or password for measurement threshold policies, scheduled task policies, and Windows Management Interface policies on the management server where the command runs.

- opctemplate

  The command opctemplate is used to enable and disable policies on managed nodes.

WMI Methods

This section describes WMI methods, including the following:

- Properties
- Class methods
- Instance methods
- Extended status codes

NOTE:
The HP Operations agent API includes support for C/C++ and Java, as well as for every language that supports DCOM automation (for example, VB, VBScript, JScript, and so on). However, the agent message stream interface supports C APIs only. All of the APIs are built using Microsoft Visual Studio 2005.

Extended Status Codes

WMI methods (calling with IWbemServices::ExecMethod) error handling: If a call to IWbemServices::ExecMethod() returns a failed HRESULT, then extended status information may be obtained by calling GerErrorInfo() and calling IErrorInfo::QueryInterface() to get the IWbemClassObject that is an instance of __ExtendedStatus, which has StatusCode and Description properties that can provide extended error information.

The __ExtendedStatus::StatusCodes are listed in the following table:

Symbol Hex/Dec Values	Description
MDLAPI_E_INVALID_PARAMETER 0xC1FF0001 / -1040252927	Parameter is not valid.
MDLAPI_E_INVALID_OBJECT_PATH 0xC1FF0002 / -1040252926	Not a valid object path syntax. A syntax error occurred at the specified offset.
MDLAPI_E_INVALID_CLASS 0xC1FF0003 / -1040252925	Methods of the class are not implemented.
MDLAPI_E_METHOD_NOT_IMPLEMENTED 0xC1FF0004 / -1040252924	Method of the class is not implemented.
MDLAPI_E_NOT_CLASS_METHOD 0xC1FF0005 / -1040252923	Method is not a class method of the class. You should specify the instance of the class.
MDLAPI_E_GET_OUT_PARAMETERS 0xC1FF0006 / -1040252922	Out parameters for the method could not be retrieved.
MDLAPI_E_PARAMETER_MISSING 0xC1FF0007 / -1040252921	Mandatory parameter is not specified.
MDLAPI_E_INVALID_PARAMETER_TYPE 0xC1FF0008 / -1040252920	Parameter has an invalid type.
MDLAPI_E_EXECQUERY_INTERNAL 0xC1FF0009 / -1040252919	Internal error when executing the query.
MDLAPI_E_PROPERTY_MISSING 0xC1FF000A / -1040252918	Property is not set.
MDLAPI_E_INVALID_PROPERTY 0xC1FF000B / -1040252917	Property is not valid.
MDLAPI_E_UNEXPECTED_ERROR 0xC1FF000C / -1040252916	Unexpected error occurred. Check trace file to get more information.
MDLAPI_E_LICENSE_MISSING 0xC1FF000D / -1040252915	Specified operation cannot be executed because of missing licenses.

MDLAPI_E_INVALID_CAPTION 0xC1FF000E / -1040252914	Specified object Caption parameter is not valid.
MDLAPI_E_INVALID_NODE_PROPERTY 0xC1FF000F / -1040252913	Property is not valid for specified node.
MDLAPI_E_INVALID_CONFIG_VALUE 0xC1FF0010 / -1040252912	Configuration value is not valid.
MDLAPI_E_NODE_NOT_EXIST 0xC1FF0011 / -1040252911	Node does not exist.
MDLAPI_E_NODE_NAME_EXISTS 0xC1FF0012 / -1040252910	Node with the same Name already exists.
MDLAPI_E_NODE_PNNAME_EXISTS 0xC1FF0013 / -1040252909	Node with the same Primary Node Name already exists.
MDLAPI_E_NODE_PNNAME_NOT_UNIQUE 0xC1FF0014 / -1040252908	More than one node with the specified Primary Node Name is found.
MDLAPI_E_NODE_DISCOVERY_FAILED 0xC1FF0015 / -1040252907	Discovery of the node type failed.
MDLAPI_E_NODE_IS_MANAGEMENT_SERVER 0xC1FF0016 / -1040252906	Management server cannot be removed.
MDLAPI_E_NODE_HIERPATH_EXISTS 0xC1FF0017 / -1040252905	Node with the same hierarchy path (Caption) already exists.
MDLAPI_E_NODE_PLATFORM_MATCHING 0xC1FF0018 / -1040252904	Node platform matching failed.
MDLAPI_E_NODE_OSBITS_NOT_EXIST 0xC1FF0019 / -1040252903	You must specify the OSBits parameter when the OS type cannot be discovered.
MDLAPI_E_NODE_OSBITS_NOT_MATCH 0xC1FF001A / -1040252902	OSBits parameter does not match any supported OS types.
MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST 0xC1FF0020 / -1040252896	Parent node group does not exist.
MDLAPI_E_NODEGROUP_NOT_EXIST 0xC1FF0021 / -1040252895	Node Group does not exist.
MDLAPI_E_NODEGROUP_NAME_EXISTS 0xC1FF0022 / -1040252894	Node Group with the same Name already exists.
MDLAPI_E_NODEGROUP_HIERPATH_EXISTS 0xC1FF0023 / -1040252893	Node Group with the same hierarchy path (Caption) already exists.
MDLAPI_E_NODEGROUP_IS_ROOT 0xC1FF0024 / -1040252892	Root Node Group cannot be removed from or added to another node group.
MDLAPI_E_NODEGROUP_IS_SPECIAL_GROUP 0xC1FF0025 / -1040252891	Special Node Group cannot be removed from or added to another node group.

MDLAPI_E_NODEGROUP_SPECIAL_ADD 0xC1FF0026 / -1040252890	Child cannot be added to a special node group.
MDLAPI_E_NODEGROUP_SPECIAL_REMOVE 0xC1FF0027 / -1040252889	Child cannot be removed from a special node group.
MDLAPI_E_NODEGROUP_IS_ALREADY_PARENT 0xC1FF0028 / -1040252888	Node group already has a parent with the same name.
MDLAPI_E_ADPOLGROUP_NOT_EXIST 0xC1FF0031 / -1040252879	Auto Deploy Policy Group does not exist.
MDLAPI_E_ADPOLGROUP_NAME_EXISTS 0xC1FF0032 / -1040252878	Auto Deploy Policy Group with the same Name already exists.
MDLAPI_E_ADPOLGROUP_HIERPATH_EXISTS 0xC1FF0033 / -1040252877	Auto Deploy Policy Group with the same hierarchy path already exists.
MDLAPI_E_REPORT_NOT_EXIST 0xC1FF0041 / -1040252863	Report does not exist.
MDLAPI_E_REPORT_PROP_NOT_VALID 0xC1FF0042 / -1040252862	Report properties (ReportFamily and ReportCategory) are not valid.
MDLAPI_E_NO_REPORT_INTEGRATION 0xC1FF0043 / -1040252861	Reporter integration tool is not installed.
MDLAPI_E_GRAPH_NOT_EXIST 0xC1FF0051 / -1040252847	Graph does not exist.
MDLAPI_E_GRAPH_PROP_NOT_VALID 0xC1FF0052 / -1040252846	Graph properties (GraphFamily and GraphCategory) are not valid.
MDLAPI_E_NO_GRAPH_INTEGRATION 0xC1FF0053 / -1040252845	Performance Manager integration tool is not installed.
MDLAPI_E_TOOL_NOT_EXIST 0xC1FF0061 / -1040252831	Tool does not exist.
MDLAPI_E_TOOL_NAME_EXISTS 0xC1FF0062 / -1040252830	Tool with the same Name already exists.
MDLAPI_E_TOOL_HIERPATH_EXISTS 0xC1FF0063 / -1040252829	Tool with the same Caption is already a child of the parent tool group.
MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST 0xC1FF0070 / -1040252816	Parent Tool Group does not exist.
MDLAPI_E_TOOLGROUP_NOT_EXIST 0xC1FF0071 / -1040252815	Tool Group does not exist.
MDLAPI_E_TOOLGROUP_NAME_EXISTS 0xC1FF0072 / -1040252814	Tool Group with the same Name already exists.
MDLAPI_E_TOOLGROUP_HIERPATH_EXISTS 0xC1FF0073 / -1040252813	Tool Group with the same hierarchy path (Caption) already exists.

MDLAPI_E_TOOLGROUP_IS_ROOT 0xC1FF0074 / -1040252812	Root Tool Group cannot be removed from or added to another tool group.
MDLAPI_E_TOOLGROUP_IS_ALREADY_PARENT 0xC1FF0075 / -1040252811	Tool group already has a parent with the same name.
MDLAPI_E_TRANSACTION_NOT_EXIST 0xC1FF0081 / -1040252799	Transaction with the specified ID does not exist.
MDLAPI_E_SERVICE_NOT_EXIST 0xC1FF0091 / -1040252783	Service does not exist.
MDLAPI_E_SERVICE_HIERPATH_EXISTS 0xC1FF0092 / -1040252782	Service with the same Caption is already a child of the parent service.
MDLAPI_E_PARENT_SERVICE_NOT_EXIST 0xC1FF0093 / -1040252781	Parent Service does not exist.
MDLAPI_E_SERVICE_NAME_EXISTS 0xC1FF0094 / -1040252780	Service with the same Name already exists.
MDLAPI_E_SERVICE_IS_ROOT 0xC1FF0095 / -1040252779	Specified operation cannot be executed for the root service.
MDLAPI_E_SERVICE_HOSTED_ON_NODEGROUP 0xC1FF0096 / -1040252778	Service is hosted on a node group. The operation cannot be executed.
MDLAPI_E_SERVICETYPE_NOT_VALID 0xC1FF0097 / -1040252777	Service type is not valid.
MDLAPI_E_SERVICE_IS_ALREADY_PARENT 0xC1FF0098 / -1040252776	Service already has a parent with the same Name.
MDLAPI_E_SERVICETYPE_NOT_EXIST 0xC1FF00A1 / -1040252767	Service type does not exist.
MDLAPI_E_SERVICETYPE_HIERPATH_EXISTS 0xC1FF00A2 / -1040252766	Service type with the same Caption is already a child of the parent service type.
MDLAPI_E_PARENT_SERVICETYPE_NOT_EXIST 0xC1FF00A3 / -1040252765	Parent Service type does not exist.
MDLAPI_E_SERVICETYPE_GUID_EXISTS 0xC1FF00A4 / -1040252764	Service type with the same GUID already exists.
MDLAPI_E_SERVICETYPE_IS_ROOT 0xC1FF00A5 / -1040252763	Specified operation cannot be executed for the root service type.
MDLAPI_E_SERVICETYPE_CANNOT_REMOVE 0xC1FF00A6 / -1040252762	Service type cannot be removed because another object or association uses it.
MDLAPI_E_CALCRULE_NOT_EXIST 0xC1FF00B1 / -1040252751	Calculation rule does not exist.
MDLAPI_E_CALCRULE_CAPTION_EXIST 0xC1FF00B2 / -1040252750	Calculation rule with the same Caption already exists.

MDLAPI_E_CALCRULE_SETTINGID_EXISTS 0xC1FF00B3 / -1040252749	Calculation rule with the same SettingId already exists.
MDLAPI_E_CALCRULE_CANNOT_REMOVE 0xC1FF00B4 / -1040252748	Calculation rule cannot be removed because another object or association uses it.
MDLAPI_E_CALCRULE_CAPTION_NOT_UNIQUE 0xC1FF00B5 / -1040252747	More than one calculation rule with the specified Caption is found.
MDLAPI_E_PROPRULE_NOT_EXIST 0xC1FF00C1 / -1040252735	Message propagation rule does not exist.
MDLAPI_E_PROPRULE_SETTINGID_EXISTS 0xC1FF00C2 / -1040252734	Propagation rule with the same SettingId already exists.
MDLAPI_E_PROPRULE_CAPTION_EXIST 0xC1FF00C3 / -1040252733	Propagation rule with the same Caption already exists.
MDLAPI_E_PROPRULE_CANNOT_REMOVE 0xC1FF00C4 / -1040252732	Propagation rule cannot be removed because another object or association uses it.
MDLAPI_E_PROPRULE_CAPTION_NOT_UNIQUE 0xC1FF00C5 / -1040252731	More than one propagation rule with the specified Caption is found.
MDLAPI_E_ADPACKAGE_NOT_EXIST 0xC1FF00D1 / -1040252719	Auto Deploy Package does not exist.
MDLAPI_E_EXT_NODE_NOT_EXIST 0xC1FF00E1 / -1040252703	External node does not exist.
MDLAPI_E_EXT_NODE_HIERPATH_EXISTS 0xC1FF00E2 / -1040252702	External node with the same hierarchy path (Caption) already exists.
MDLAPI_E_EXT_NODE_NAME_EXISTS 0xC1FF00E3 / -1040252701	External node with the same Name already exists.
MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST 0xC1FF00F1 / -1040252687	Service composition does not exist.
MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST 0xC1FF00F2 / -1040252686	Service dependency does not exist.
MDLAPI_E_SVC_TYPE_COMP_ASSOC_NOT_EXIST 0xC1FF00F3 / -1040252685	Service type composition does not exist.
MDLAPI_E_SVC_TYPE_DEP_ASSOC_NOT_EXIST 0xC1FF00F4 / -1040252684	Service type dependency does not exist.
MDLAPI_E_SVC_DEP_CYCLE_DETECTED 0xC1FF00F5 / -1040252683	Specified operation cannot be executed because a service dependency cycle has been detected.
MDLAPI_E_SVC_TYPE_DEP_CYCLE_DETECTED 0xC1FF00F6 / -1040252682	Specified operation cannot be executed because a service type dependency cycle has been detected.
MDLAPI_E_TRANS_INDICATE_FAILED 0xC1FF0101 / -1040252671	An error occurred when trying to return output parameters.

MDLAPI_E_TRANS_INDICATE_FAILED1 0xC1FF0102 / -1040252670	Error WBEM_E_SERVER_TOO_BUSY occurred when trying to return output parameters.
MDLAPI_E_TRANS_INDICATE_FAILED2 0xC1FF0103 / -1040252669	Error WBEM_E_FAILED occurred when trying to return output parameters.
MDLAPI_E_TRANS_INDICATE_FAILED3 0xC1FF0104 / -1040252668	Error WBEM_E_CALL_CANCELLED occurred when trying to return output parameters.
MDLAPI_E_TRANS_INDICATE_FAILED4 0xC1FF0105 / -1040252667	Error WBEM_E_INVALID_PARAMETER occurred when trying to return output parameters.
MDLAPI_E_TRANS_INIT 0xC1FF0106 / -1040252666	Transaction cannot be initialized.
MDLAPI_E_TRANS_METHOD_NOTSUPPORTED 0xC1FF0107 / -1040252665	Specified method is not yet supported.
MDLAPI_E_TRANS_INVALID_CLASS 0xC1FF0108 / -1040252664	Specified class is not supported.
MDLAPI_E_INVALID_KEY_PROP 0xC1FF0109 / -1040252663	Specified class does not have the key property %2.
MDLAPI_E_TOO_MANY_KEY_PROP 0xC1FF010A / -1040252662	Class can have only key property(s).
MDLAPI_E_NOT_INSTANCE_OBJECT_PATH 0xC1FF010B / -1040252661	Object path is not an instance of the WMI class.
MDLAPI_E_CONFIGFILEVAR_APP_NOT_EXIST 0xC1FF0121 / -1040252639	Specified Application does not exist.
MDLAPI_E_CONFIGFILEVAR_SUBGRP_NOT_EXIST 0xC1FF0122 / -1040252638	Specified Application does not have such a SubGroup.
MDLAPI_E_CONFIGFILEVAR_FNAME_NOT_EXIST 0xC1FF0123 / -1040252637	Specified SubGroup does not have such a Filename.
MDLAPI_E_CONFIGFILEVAR_SUBGRP_NOT_EMPTY 0xC1FF0124 / -1040252636	Specified SubGroup has one or more Filenames.
MDLAPI_E_CONFIGFILEVAR_TEMPL_NOT_EXIST 0xC1FF0125 / -1040252635	Variety does not have the associated template.
MDLAPI_E_CONFIGFILEVAR_TEMPL_NOT_ALLOW 0xC1FF0126 / -1040252634	Template is not allowed for this variety.
MDLAPI_E_CONFIGFILEVAR_TEMPL_FILENAME 0xC1FF0127 / -1040252633	String '<*>' cannot be used as the Filename.
MDLAPI_E_NO_OPERATION_RIGHTS 0xC1FF0131 / -1040252623	User has no rights to execute this operation.

OV_Action

Description

Identifies the actions that can be executed by an operator. These actions can be associated with LogicalElements.

```
class OV_Action
{
        Properties:
        string Name;
        string Icon;
        string Command;
        sint32 IOType;
        sint32 CallType;
        uint16 ExecuteOn;
        string ExecuteAsUser;
        string Parameters;
        boolean EditParameters;
        boolean EditLogin;
        string Password;
        boolean RequirePassword;
        string PredefinedList[];
        boolean ShowInContextMenu;

        Class Methods:

        OV_Action Create(
                [in] string Command,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] sint32 CallType,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] string Parameters,
                [in, optional] boolean EditParameters,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action Create_Trans(
                [in] string TransId,
                [in] string Command,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] sint32 CallType,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] string Parameters,
                [in, optional] boolean EditParameters,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsExecutable(
                [in] string Command,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] string Parameters,
                [in, optional] boolean EditParameters,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsExecutable_Trans(
                [in] string TransId,
                [in] string Command,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] string Parameters,
                [in, optional] boolean EditParameters,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsVBScript(
                [in] string Script,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsVBScript_Trans(
                [in] string TransId,
                [in] string Script,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsJScript(
                [in] string Script,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsJScript_Trans(
                [in] string TransId,
                [in] string Script,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsWScript(
                [in] string Script,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsWScript_Trans(
                [in] string TransId,
                [in] string Script,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsPerlScript(
                [in] string Script,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsPerlScript_Trans(
                [in] string TransId,
                [in] string Script,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] uint16 ExecuteOn,
                [in, optional] sint32 IOType,
                [in, optional] boolean EditLogin,
                [in, optional] string ExecuteAsUser,
                [in, optional] boolean RequirePassword,
                [in, optional] string Password,
                [in, optional] string PredefinedList[]
                );

        OV_Action CreateAsURL(
                [in] string Url,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description
                );

        OV_Action CreateAsURL_Trans(
                [in] string TransId,
                [in] string Url,
                [in] string Caption,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description
                );

        void Remove(
                [in] string Name
                );

        void Remove_Trans(
                [in] string TransId,
                [in] string Name
                );

        OV_Action GetByName(
                [in] string Name
                );

        OV_Action GetByName_Trans(
                [in] string TransId,
                [in] string Name
                );

        OV_Action GetByHierarchicalPath(
                [in] string Path
                );

        OV_Action GetByHierarchicalPath_Trans(
                [in] string TransId,
                [in] string Path
                );

        void ChangeName(
                [in] OV_Action Action,
                [in, optional] string NewName
                );

        void ChangeName_Trans(
                [in] string TransId,
                [in] OV_Action Action,
                [in, optional] string NewName
                );
```

Instance Methods:

```
void Modify(
        [in] OV_Action Action
        );

void Modify_Trans(
        [in] string TransId,
        [in] OV_Action Action
        );

sint32 GetParents(
        [out] OV_Tools ToolGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_Tools ToolGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

boolean IsChildOf(
        [in] string ParentName
        );

boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName
        );

sint32 GetNodeGroups(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeInherited
        );

sint32 GetNodeGroups_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeInherited
        );

boolean IsAssociatedWithNodeGroup(
        [in] string NodeGroupName,
        [in, optional] boolean IncludeInherited
        );

boolean IsAssociatedWithNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName,
        [in, optional] boolean IncludeInherited
        );
        sint32 GetNodes(
                [out] OV_ManagedNode Nodes[],
                [in, optional] boolean IncludeInherited
                );

        sint32 GetNodes_Trans(
                [in] string TransId,
                [out] OV_ManagedNode Nodes[],
                [in, optional] boolean IncludeInherited
                );

        boolean IsAssociatedWithNode(
                [in] string NodeName,
                [in, optional] boolean IncludeInherited
                );

        boolean IsAssociatedWithNode_Trans(
                [in] string TransId,
                [in] string NodeName,
                [in, optional] boolean IncludeInherited
                );

        sint32 GetServices(
                [out] OV_Service Services[]
                );

        sint32 GetServices_Trans(
                [in] string TransId,
                [out] OV_Service Services[]
                );

        sint32 GetServiceTypes(
                [out] OV_ServiceTypeDefinition ServiceTypes[]
                );

        sint32 GetServiceTypes_Trans(
                [in] string TransId,
                [out] OV_ServiceTypeDefinition ServiceTypes[]
                );
    };
```

OV_Action-Properties

Name

Signature
string Name

Description
The name identifying the Action.

Icon

Signature
string Icon

Description
The icon to use for this tool. If blank, the default icon defined for the Actions is used.

Command

Signature
string Command

Description
The command string that is executed.

IOType

Signature
sint32 IOType

Description
Indicates whether the command is expected to give an output stream.

Values

ValueMap	Values
0	= NoOutput
1	= Output

CallType

Signature

sint32 CallType

Description

Identifies the type of the command.

Values

ValueMap	Values
0	= Executable
1	= VB script
2	= JScript
3	= Windows Scripting Host
4	= Perl script
100	= URL

ExecuteOn

Signature

uint16 ExecuteOn

Description

Identifies the location from which the command runs.

Values

ValueMap	Values
0	= Management Server
1	= Selected Node
2	= Node List
3	= Console

ExecuteAsUser

Signature

string ExecuteAsUser

Description

The user who execute the command. If this string is blank or empty, the default user is used.

Parameters

Signature

string Parameters

Description

The parameters to be added to the command.

EditParameters

Signature

boolean EditParameters

Description

Indicates whether the parameters can be edited by the user before the command is executed.

EditLogin

Signature

boolean EditLogin

Description

Indicates whether the logon can be edited by the user before the command is executed.

Password

Signature

string Password

Description

The encrypted password for the user identified in ExecuteAsUser.

RequirePassword

Signature

boolean RequirePassword

Description

If true, the user is required to type a password.

PredefinedList

Signature

string PredefinedList[]

Description

A predefined list of nodes or services on which the action is executed. This list is valid only if the ExecuteOn property is set to NodeList. This is an array of strings in which each element in the array is a node or a service.

ShowInContextMenu

Signature

boolean ShowInContextMenu

Description

If true, the tool is shown in the context menu when clicking on a message.

OV_Action Class Methods

This section contains the information on Class methods for OV_Action.

OV_Action::ChangeName()

OV_Action::ChangeName_Trans()

```
void ChangeName(
        [in] OV_Action Action,
        [in, optional] string NewName)
```

```
void ChangeName_Trans(
        [in] string TransId,
        [in] OV_Action Action,
        [in, optional] string NewName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Action
: Instance of OV_Action in which the Name property is to be changed.

NewName
: The New Name property for this instance of OV_Action. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, the Name property is created as a new GUID. If a tool with the same Name exists, nothing happens, and the method fails with MDLAPI_E_TOOL_NAME_EXISTS.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Changes the Name property of this tool.

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOL_NAME_EXISTS
: Tool with the same Name already exists.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action::Create()

OV_Action::Create_Trans()

```
OV_Action Create(
        [in] string Command,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] sint32 CallType,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] string Parameters,
        [in, optional] boolean EditParameters,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

```
OV_Action Create_Trans(
        [in] string TransId,
        [in] string Command,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] sint32 CallType,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] string Parameters,
        [in, optional] boolean EditParameters,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Command
: If the command type (parameter callType) is executable, it is an executable path. If the command type is one of the script types, it contains a script. If the command type is URL, it contains an URL.

Caption
: Caption (Display Name). If a tool with the same Caption/Display is already a child of the parent tool group, nothing happens, and the method fails with MDLAPI_E_TOOL_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more than 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName
: The Name property of the instance of OV_Tools where the newly created tool is added. Optional parameter. If the parameter is not specified, or if it is equal to an empty string, the root tool group is used. If the specified tool group does not exist, the method fails with MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST.

CallType
: Copied to CallType property. Optional parameter. Default is 0 ("Executable").

Name
: The Name property of the instance of OV_Action to be created. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, it is created as a new GUID. If a tool with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOL_NAME_EXISTS.

Description
: Copied to the Description property. Optional parameter.

Parameters
: Copied to the Parameters property. This parameter is applicable only when the CallType is equal to 0 ("Executable"). Optional parameter.

EditParameters
: Copied to the EditParameters property. This parameter is applicable only when the CallType is equal to 0 ("Executable"). Optional parameter. Default is false.

ExecuteOn
: Copied to the ExecuteOn property. When the CallType is equal to one of the script types, this parameter may not equal 3 ("Console"). When the CallType is equal to 100 ("URL"), this parameter must equal 3 ("Console"). Optional parameter. Default is 0 ("Management Server").

IOType
: Copied to IOType property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"). Optional parameter. Default is 0 ("NoOutput").

EditLogin
: Copied to EditLogin property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"). Optional parameter. Default is false.

ExecuteAsUser
: Copied to ExecuteAsUser property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"). Optional parameter.

RequirePassword
: Copied to RequirePassword property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"). Optional parameter. Default is false.

Password
: Copied to Password property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"), and only when RequirePassword is equal to true. Optional parameter.

PredefinedList
: Copied to PredefinedList property. This parameter is applicable only when ExecuteOn is equal to 2 ("Node List"). Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a tool (new instance of OV_Action), and adds it to the existing tool group (instance of OV_Tools).

All parameters are validated before the tool is actually generated. If any parameter does not match, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_Action.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_TOOL_NAME_EXISTS
: Tool with the same Name already exists.

MDLAPI_E_TOOL_HIERPATH_EXISTS
: Tool with the same Caption is already a child of the parent tool group.

MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST
: Parent Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action::CreateAsExecutable()

OV_Action::CreateAsExecutable_Trans()

```
OV_Action CreateAsExecutable(
        [in] string Command,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] string Parameters,
        [in, optional] boolean EditParameters,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

```
OV_Action CreateAsExecutable_Trans(
        [in] string TransId,
        [in] string Command,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] string Parameters,
        [in, optional] boolean EditParameters,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Command
: Executable path.

Caption
: Caption (Display Name). If a tool with the same Caption/Display is already a child of the parent tool group, nothing happens, and the method fails with MDLAPI_E_TOOL_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more than 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName
: The Name property of the instance of OV_Tools to which the newly created tool is added. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, the root tool group is used. If the specified tool group does not exist, the method fails with MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST.

Name
: The Name property of the instance of OV_Action to be created. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, it is created as a new GUID. If a tool with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOL_NAME_EXISTS.

Description
: Copied to the Description property. Optional parameter.

Parameters
: Copied to the Parameters property. Optional parameter.

EditParameters
: Copied to the EditParameters property. Optional parameter. Default is false.

ExecuteOn
: Copied to the ExecuteOn property. Optional parameter. Default is 0 ("Management Server").

IOType
: Copied to the IOType property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"). Optional parameter. Default is 0 ("NoOutput").

EditLogin
: Copied to the EditLogin property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"). Optional parameter. Default is false.

ExecuteAsUser

Copied to the ExecuteAsUser property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"). Optional parameter.

RequirePassword
: Copied to the RequirePassword property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"). Optional parameter. Default is false.

Password
: Copied to the Password property. This parameter is applicable only when ExecuteOn is not equal to 3 ("Console"), and only when RequirePassword is equal to true. Optional parameter.

PredefinedList
: Copied to the PredefinedList property. This parameter is applicable only when ExecuteOn is equal to 2 ("Node List"). Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a tool (new instance of OV_Action) as an executable, and adds it to the existing tool group (instance of OV_Tools).

All parameters are validated before the tool is actually generated. If any parameter does not match, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_Action.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_TOOL_NAME_EXISTS
: Tool with the same Name already exists.

MDLAPI_E_TOOL_HIERPATH_EXISTS
: Tool with the same Caption is already a child of the parent tool group.

MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST
: Parent Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Action::CreateAsJScript()
OV_Action::CreateAsJScript_Trans()

```
OV_Action CreateAsJScript(
        [in] string Script,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

```
OV_Action CreateAsJScript_Trans(
        [in] string TransId,
        [in] string Script,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Script

Contains a JScript.

Caption
: Caption (Display Name). If a tool with the same Caption/Display is already a child of the parent tool group, nothing happens, and the method fails with MDLAPI_E_TOOL_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more than 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName
: The Name property of the instance of OV_Tools to which the newly created tool is added. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, the root tool group is used. If the specified tool group does not exist, the method fails with MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST.

Name
: The Name property of an instance of OV_Action to be created. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, it is created as a new GUID. If a tool with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOL_NAME_EXISTS.

Description
: Copied to the Description property. Optional parameter.

ExecuteOn
: Copied to the ExecuteOn property. Optional parameter. Default is 0 ("Management Server").

IOType
: Copied to the IOType property. Optional parameter. Default is 0 ("NoOutput").

EditLogin
: Copied to the EditLogin property. Optional parameter. Default is false.

ExecuteAsUser
: Copied to the ExecuteAsUser property. Optional parameter.

RequirePassword
: Copied to the RequirePassword property. Optional parameter. Default is false.

Password
: Copied to the Password property. This parameter is applicable only when RequirePassword is equal to true. Optional parameter.

PredefinedList
: Copied to PredefinedList property. This parameter is applicable only when ExecuteOn is equal to 2

("Node List"). Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a tool (new instance of OV_Action) as a JScript, and adds it to an existing tool group (instance of OV_Tools).

All parameters are validated before the tool is actually generated. If any parameter does not match, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_Action.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_TOOL_NAME_EXISTS
: Tool with the same Name already exists.

MDLAPI_E_TOOL_HIERPATH_EXISTS
: Tool with the same Caption is already a child of the parent tool group.

MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST
: Parent Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action::CreateAsPerlScript()
OV_Action::CreateAsPerlScript_Trans()

```
OV_Action CreateAsPerlScript(
        [in] string Script,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

```
OV_Action CreateAsPerlScript_Trans(
        [in] string TransId,
        [in] string Script,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Script

Contains a Perl script.

Caption

Caption (Display Name). If a tool with the same Caption/Display is already a child of the parent tool group, nothing happens, and the method fails with MDLAPI_E_TOOL_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more than 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName

The Name property of the instance of OV_Tools to which the newly created tool is added. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, the root tool group is used. If the specified tool group does not exist, the method fails with MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST.

Name

The Name property of the instance of OV_Action to be created. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, it is created as a new GUID. If a tool with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOL_NAME_EXISTS.

Description

Copied to the Description property. Optional parameter.

ExecuteOn

Copied to the ExecuteOn property. Optional parameter. Default is 0 ("Management Server").

IOType

Copied to the IOType property. Optional parameter. Default is 0 ("NoOutput").

EditLogin

Copied to the EditLogin property. Optional parameter. Default is false.

ExecuteAsUser

Copied to the ExecuteAsUser property. Optional parameter.

RequirePassword

Copied to the RequirePassword property. Optional parameter. Default is false.

Password

Copied to the Password property. This parameter is applicable only when RequirePassword is equal to true. Optional parameter.

PredefinedList

Copied to the PredefinedList property. This parameter is applicable only when ExecuteOn is equal to 2

("Node List"). Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a tool (new instance of OV_Action) as a Perl script, and adds it to an existing tool group (instance of OV_Tools).

All parameters are validated before the tool is actually generated. If any parameter does not match, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_Action.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_TOOL_NAME_EXISTS
: Tool with the same Name already exists.

MDLAPI_E_TOOL_HIERPATH_EXISTS
: Tool with the same Caption is already a child of the parent tool group.

MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST
: Parent Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action::CreateAsURL()

OV_Action::CreateAsURL_Trans()

```
OV_Action CreateAsURL(
        [in] string Url,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description)
```

```
OV_Action CreateAsURL_Trans(
        [in] string TransId,
        [in] string Url,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Url

Contains an URL.

Caption

Caption (Display Name). If a tool with the same Caption/Display is already a child of the parent tool group, nothing happens, and the method fails with MDLAPI_E_TOOL_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more than 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName

The Name property of the instance of OV_Tools to which the newly created tool is added. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, the root tool group is used. If the specified tool group does not exist, the method fails with MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST.

Name

The Name property of an instance of OV_Action to be created. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, it is created as a new GUID. If a tool with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOL_NAME_EXISTS.

Description
: Copied to the Description property. Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a tool (new instance of OV_Action) as an URL, and adds it to an existing tool group (instance of OV_Tools).

All parameters are validated before the tool is actually generated. If any parameter does not match, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_Action.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_TOOL_NAME_EXISTS
: Tool with the same Name already exists.

MDLAPI_E_TOOL_HIERPATH_EXISTS
: Tool with the same Caption is already a child of the parent tool group.

MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST
: Parent Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action::CreateAsVBScript() OV_Action::CreateAsVBScript_Trans()

```
OV_Action CreateAsVBScript(
        [in] string Script,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )

OV_Action CreateAsVBScript_Trans(
        [in] string TransId,
        [in] string Script,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Script
: Contains a VBScript.

Caption
: Caption (Display Name). If a tool with the same Caption/Display is already a child of the parent tool group, nothing happens, and the method fails with MDLAPI_E_TOOL_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more than 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName
: The Name property of an instance of OV_Tools to which the newly created tool is added. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, the root tool group is used. If the specified tool group does not exist, the method fails with MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST.

Name
: The Name property of an instance of OV_Action to be created. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, it is created as a new GUID. If a tool with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOL_NAME_EXISTS.

Description
: Copied to the Description property. Optional parameter.

ExecuteOn
: Copied to the ExecuteOn property. Optional parameter. Default is 0 ("Management Server").

IOType
: Copied to the IOType property. Optional parameter. Default is 0 ("NoOutput").

EditLogin
: Copied to the EditLogin property. Optional parameter. Default is false.

ExecuteAsUser
: Copied to the ExecuteAsUser property. Optional parameter.

RequirePassword
: Copied to the RequirePassword property. Optional parameter. Default is false.

Password
: Copied to the Password property. This parameter is applicable only when RequirePassword is equal to true. Optional parameter.

PredefinedList
: Copied to the PredefinedList property. This parameter is applicable only when ExecuteOn is equal to 2

("Node List"). Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a tool (new instance of OV_Action) as a VBScript, and adds it to an existing tool group (instance of OV_Tools).

All parameters are validated before the tool is actually generated. If any parameters do not match, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_Action.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
  Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
  Specified object Caption parameter is not valid.

MDLAPI_E_TOOL_NAME_EXISTS
  Tool with the same Name already exists.

MDLAPI_E_TOOL_HIERPATH_EXISTS
  Tool with the same Caption is already a child of the parent tool group.

MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST
  Parent Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
  Transaction with the specified ID does not exist.

OV_Action::CreateAsWScript()

OV_Action::CreateAsWScript_Trans()

```
OV_Action CreateAsWScript(
        [in] string Script,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

```
OV_Action CreateAsWScript_Trans(
        [in] string TransId,
        [in] string Script,
        [in] string Caption,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] uint16 ExecuteOn,
        [in, optional] sint32 IOType,
        [in, optional] boolean EditLogin,
        [in, optional] string ExecuteAsUser,
        [in, optional] boolean RequirePassword,
        [in, optional] string Password,
        [in, optional] string PredefinedList[] )
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Script

Contains a WScript.

Caption
: Caption (Display Name). If a tool with the same Caption/Display is already a child of the parent tool group, nothing happens, and the method fails with MDLAPI_E_TOOL_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more than 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName
: The Name property of an instance of OV_Tools to which the newly created tool is added. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, the root tool group is used. If the specified tool group does not exist, the method fails with MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST.

Name
: The Name property of an instance of OV_Action to be created. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, it is created as a new GUID. If a tool with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOL_NAME_EXISTS.

Description
: Copied to the Description property. Optional parameter.

ExecuteOn
: Copied to the ExecuteOn property. Optional parameter. Default is 0 ("Management Server").

IOType
: Copied to the IOType property. Optional parameter. Default is 0 ("NoOutput").

EditLogin
: Copied to the EditLogin property. Optional parameter. Default is false.

ExecuteAsUser
: Copied to the ExecuteAsUser property. Optional parameter.

RequirePassword
: Copied to the RequirePassword property. Optional parameter. Default is false.

Password
: Copied to the Password property. This parameter is applicable only when RequirePassword is equal to true. Optional parameter.

PredefinedList
: Copied to PredefinedList property. This parameter is applicable only when ExecuteOn is equal to 2

("Node List"). Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a tool (new instance of OV_Action) as a WScript, and adds it to an existing tool group (instance of OV_Tools).

All parameters are validated before the tool is actually generated. If any parameter does not match, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_Action.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER	Parameter is not valid.
MDLAPI_E_INVALID_CAPTION	Specified object Caption parameter is not valid.
MDLAPI_E_TOOL_NAME_EXISTS	Tool with the same Name already exists.
MDLAPI_E_TOOL_HIERPATH_EXISTS	Tool with the same Caption is already a child of the parent tool group.
MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST	Parent Tool Group does not exist.
MDLAPI_E_TRANSACTION_NOT_EXIST	Transaction with the specified ID does not exist.

OV_Action::GetByHierarchicalPath()
OV_Action::GetByHierarchicalPath_Trans()

```
OV_Action GetByHierarchicalPath(
        [in] string Path)
```

```
OV_Action GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Path
: The tool specified by the hierarchy Path, which consists of Caption/Display Names of the parent Tool Groups and a single backslash as a separator. The Path starts with a single backslash. If the Caption contains additional backslashes, they should be escaped with backslashes (for example, "\\").

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the tool (instance of OV_Action) specified by the hierarchical path.

Return Value

Instance of OV_Action specified by the hierarchical path.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
: Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action::GetByName()
OV_Action::GetByName_Trans()

```
OV_Action GetByName(
        [in] string Name)
```

```
OV_Action GetByName_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Name
    The Name property of the instance of OV_Action to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the tool (instance of OV_Action) specified by the Name property.

Return Value

Instance of OV_Action specified by the Name property.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
    Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_Action::Remove()
OV_Action::Remove_Trans()

```
void Remove(
        [in] string Name)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
  Transaction ID returned from OV_Transaction::Start().

Name
  The Name property of the instance of OV_Action to be removed.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified tool (instance of OV_Action).

Objects that are associated with this tool are not removed. Only the associations are removed.

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
  Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
  Transaction with the specified ID does not exist.

OV_Action Instance Methods

This section contains information for Instance methods for OV_Action.

OV_Action::GetNodeGroups()
OV_Action::GetNodeGroups_Trans()

```
sint32 GetNodeGroups(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeInherited)
```

```
sint32 GetNodeGroups_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeInherited)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

NodeGroups

Instances of OV_NodeGroup to which this node group is added. An instance is valid as a returned out parameter only if the method does not fail.

IncludeInherited

Defines whether the NodeGroup also contains child node groups (recursive) of the node groups where this tool was associated. (Tools are inherited from the node group parent.) Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of node groups (instances of OV_NodeGroup) with which this tool is associated.

Return Value

The Number of node groups in out parameter NodeGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Action::GetNodes()
OV_Action::GetNodes_Trans()

```
sint32 GetNodes(
        [out] OV_ManagedNode Nodes[],
        [in, optional] boolean IncludeInherited)
```

```
sint32 GetNodes_Trans(
        [in] string TransId,
        [out] OV_ManagedNode Nodes[],
        [in, optional] boolean IncludeInherited)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Nodes
    Instances of OV_ManagedNode to which this tool is added. An instance is valid as a returned out parameter only if the method does not fail.

IncludeInherited
    Indicates whether Nodes also contains nodes to which this tool is not directly associated. The nodes are inherited from a parent node group. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of nodes (instances of OV_ManagedNode) with which this tool is associated.

Return Value

Number of nodes in out parameter Nodes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Action::GetParents()
OV_Action::GetParents_Trans()

```
sint32 GetParents(
        [out] OV_Tools ToolGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

```
sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_Tools ToolGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ToolGroups
: Instances of OV_Tools that are the direct parents of this node. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllHierarchicalParents
: Defines whether ToolGroups also include instances of OV_Tools that are hierarchical parents (recursive until the root tool group), rather than direct parents. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of tool groups for which this tool is a child.

Return Value

Number of tool groups (parents) in out parameter ToolGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Action::GetServices() OV_Action::GetServices_Trans()

```
sint32 GetServices(
        [out] OV_Service Services[] )
```

```
sint32 GetServices_Trans(
        [in] string TransId,
        [out] OV_Service Services[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Services
: Instances of OV_Service that are associated with this tool. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) where this tool is added.

Return Value

Number of services in the out parameter Services.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action::GetServiceTypes()

OV_Action::GetServiceTypes_Trans()

```
sint32 GetServiceTypes(
        [out] OV_ServiceTypeDefinition ServiceTypes[] )
```

```
sint32 GetServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ServiceTypes[] )
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ServiceTypes

Instances of OV_ServiceTypeDefinition that are associated with this tool. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) where this tool is added.

Return Value

Number of service types in the out parameter ServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Action::IsAssociatedWithNode()

OV_Action::IsAssociatedWithNode_Trans()

```
boolean IsAssociatedWithNode(
        [in] string NodeName,
        [in, optional] boolean IncludeInherited)
```

```
boolean IsAssociatedWithNode_Trans(
        [in] string TransId,
        [in] string NodeName,
        [in, optional] boolean IncludeInherited)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeName
: The Name property of a node (instance of OV_ManagedNode) to check if this tool is associated with the node.

IncludeInherited
: Indicates whether true should be returned also when this tool is not associated directly to a node with the parameter NodeName, but is inherited from a parent node group. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if this tool is associated with the specified node (instance of OV_ManagedNode).

Return Value

True if this tool is associated with the node (instance of OV_ManagedNode) specified by the NodeName parameter.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Action::IsChildOf()
OV_Action::IsChildOf_Trans()

```
boolean IsChildOf(
        [in] string ParentName)
```

```
boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentName
: The Name property of the tool group (instance of OV_Tools) to check if the tool group is the parent of this tool.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified tool group (instance of OV_Tools) is a parent of this tool.

Return Value

True if the tool group (instance of OV_Tools) specified by the ParentName parameter is a parent of this tool.

Extended Status Codes

MDLAPI_E_TOOLGROUP_NOT_EXIST
: Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action::Modify()

OV_Action::Modify_Trans()

```
void Modify(
        [in] OV_Action Action)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_Action Action)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Action
: Modifies an instance of OV_Action to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes made to one or more properties.

It is not possible to get an instance of OV_Action to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, an instance of OV_Action is specified as the Action parameter, even though this method is already called in the context of this instance.

As with the OV_Action::Create method, the following checking is performed:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If a property has an invalid value, the method fails with MDLAPI_E_INVALID_PROPERTY.
- If the Caption property is an empty string, contains invalid characters, or has more than 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.
- If a service with the same hierarchical path already exists, the method fails with MDLAPI_E_TOOL_HIERPATH_EXISTS.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING
: Property is not set.

MDLAPI_E_INVALID_PROPERTY
: Property is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_TOOL_HIERPATH_EXISTS
: Tool with the same Caption is already a child of the parent tool group.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Action_ActionRequest

Description

An action is executed on a node. The management server sends an action request event to the node where the action is to be executed. When the action is finished, the node sends an action response event with the action result back to the management server.

OV_Action_ActionRequest is the new action request format.

```
class OV_Action_ActionRequest
{
        Properties:
        string Id;
        string Ctx;
        string NodeName;
        OV_Action ref ToolToStart;
        string Parameters;
        sint32 Encryption;
        string User;
        string Password;
        string Display;
        datetime StartTime;
        string SessionId;
        string paraArray[];
        string valueArray[];

        Instance Methods:

        sint32 Cancel(
                );
};
```

OV_Action_ActionRequest-Properties

Id

Signature

string Id

Description

Id of the action request. If you do not specify the value, the provider generates it.

Ctx

Signature

string Ctx

Description

Context information. If the Id is unique, you can use the value NULL.

NodeName

Signature

string NodeName

Description

Name of the node where the action is executed. If the value is NULL, the provider uses the OV_Action node name.

ToolToStart

Signature

OV_Action ref ToolToStart

Description

Reference used to start the tool.

Parameters

Signature

string Parameters

Description

String of action parameters.

Encryption

Signature

sint32 Encryption

Description

Encryption type used to encrypt data (password and parameters).

Values

ValueMap	Values
0	= LocalPutInstance
1	= 1

User

Signature

string User

Description

User who starts the action. If the value is NULL, the provider uses the OV_Action user.

Password

Signature

string Password

Description

Password that legitimates action execution.

If you do not specify the user, the provider ignores the attribute.

Display

Signature

string Display

Description

Where the X application or NT application should open the display or GUI.

StartTime

Signature

datetime StartTime

Description

Action start time. The provider sets this attribute automatically.

SessionId

Signature

string SessionId

Description

Session or caller ID that can be used to filter action responses, based on the issuer.

paraArray

Signature

string paraArray[]

Description

Dynamic array of parameter names.

valueArray

Signature

string valueArray[]

Description

Dynamic array of corresponding parameter values.

OV_Action_ActionRequest::Cancel()
OV_Action_ActionRequest::Cancel_Trans()

sint32 Cancel()

Parameters

None.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Cancels the action.

Return Value

Values

ValueMap		Values
0	=	Succeeded
-1	=	Error

Extended Status Codes

None.

OV_ActionRequest

Description

An action is executed on a node. The management server sends an action request event to the node where the action is to be executed. When the action is finished, the node sends an action response event with the action result back to the management server.

OV_ActionRequest is the old action request format. OV_Action_ActionRequest replaces it.

```
class OV_ActionRequest
{
        Properties:
        string Id;
        string Ctx;
        string NodeName;
        string Call;
        sint32 CallType;
        sint32 Encryption;
        string User;
        string Password;
        string Display;
        boolean SendOutput;
        datetime StartTime;
        string SessionId;

        Instance Methods:

        sint32 Cancel(
                );
};
```

OV_ActionRequest-Properties

Id

Signature

string Id

Description

If you do not specify the value, the provider generates it.

Ctx

Signature

string Ctx

Description

Context information. If the Id is unique, you can use the value NULL.

NodeName

Signature

string NodeName

Description

Name of the node where the action is executed.

Call

Signature

string Call

Description

Specifies the action command (see CallType).

CallType

Signature

sint32 CallType

Description

Action type, which is currently one of the following:

"Executable": 'Call' = Name of program or script to be started on the node (default)

"JScript": 'Call' = JavaScript code

"VB script": 'Call' = Visual Basic code

"WSHScript": 'Call' = Windows Scripting Host script code

Values

ValueMap	Values
0	= Executable
1	= VBScript
2	= JScript
3	= WSHScript

Encryption

Signature

sint32 Encryption

Description

Encryption type used to encrypt data (password and parameters).

Values

ValueMap	Values
0	= LocalPutInstance
1	= 1

User

Signature

string User

Description

If the value is NULL, the provider uses the OV_Action user.

Password

Signature

string Password

Description

Password that legitimates action execution.

If you do not specify the user, the provider ignores the attribute.

Display

Signature

string Display

Description

Where the X application or NT application should open the display or GUI.

SendOutput

Signature

boolean SendOutput

Description

If true, the output is sent to Stdout.

StartTime

Signature

datetime StartTime

Description

Action start time. The provider sets this value automatically.

SessionId

Signature

string SessionId

Description

Session or caller ID that can be used to filter action responses, based on the issuer.

OV_ActionRequest::Cancel()
OV_ActionRequest::Cancel_Trans()

sint32 Cancel()

Parameters

None.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Cancel the action.

Return Value

Values

ValueMap		Values
0	=	Succeeded
-1	=	Error

Extended Status Codes

None.

OV_ActionResponseEvent

Description

An action is executed on a node. The management server sends an action request event to the node where the action is to be executed. When the action is finished, the node sends an action response event with the action result back to the management server.

```
class OV_ActionResponseEvent
{
        Properties:
        string Id;
        string Ctx;
        string NodeName;
        datetime FinishTime;
        sint32 Result;
        string StdOutput;
        string SessionId;
        sint32 Status;
};
```

OV_ActionResponseEvent-Properties

Id

Signature

string Id

Description

Id of the action request.

Ctx

Signature

string Ctx

Description

Context information of the action request.

NodeName

Signature

string NodeName

Description

Name of the node where the action was executed.

FinishTime

Signature

datetime FinishTime

Description

Time when the action was finished on the node.

Result

Signature

sint32 Result

Description

Result of the action.

StdOutput

Signature

string StdOutput

Description

Output device of the action.

SessionId

Signature

string SessionId

Description

Session or caller ID that can be used to filter action responses, based on the issuer.

Status

Signature

sint32 Status

Description

State of the action response event.

Values

ValueMap	Values
0	= Succeeded
1	= Failed

OV_AutoDeployPolicyGroup

Description

The policy group that gets deployed when a managed node is placed in a node group associated with this object, or when a service is instantiated from a service type definition associated with this object.

```
class OV_AutoDeployPolicyGroup
{
        Properties:
        string Name;
        string Caption;
        string Description;
        string PolicyGroup;
        boolean EnableDeployment;

        Class Methods:

        OV_AutoDeployPolicyGroup Create(
                [in] string PolicyGroupPath,
                [in, optional] boolean EnableDeployment,
                [in, optional] string Caption,
                [in, optional] string Name,
                [in, optional] string Description
                );

        OV_AutoDeployPolicyGroup Create_Trans(
                [in] string TransId,
                [in] string PolicyGroupPath,
                [in, optional] boolean EnableDeployment,
                [in, optional] string Caption,
                [in, optional] string Name,
                [in, optional] string Description
                );

        void Remove(
                [in] string Name
                );

        void Remove_Trans(
                [in] string TransId,
                [in] string Name
                );
```

```
OV_AutoDeployPolicyGroup GetByName(
        [in] string Name
        );

OV_AutoDeployPolicyGroup GetByName_Trans(
        [in] string TransId,
        [in] string Name
        );

sint32 GetByPolicyPath(
        [out] OV_AutoDeployPolicyGroup PolicyGroups[],
        [in] string PolicyGroupPath
        );

sint32 GetByPolicyPath_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPolicyGroup PolicyGroups[],
        [in] string PolicyGroupPath
        );
```

Instance Methods:

```
void Modify(
        [in] OV_AutoDeployPolicyGroup ADPolicyGroup
        );

void Modify_Trans(
        [in] string TransId,
        [in] OV_AutoDeployPolicyGroup ADPolicyGroup
        );

sint32 GetNodeGroups(
        [out] OV_NodeGroup NodeGroups[]
        );

sint32 GetNodeGroups_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[]
        );

boolean IsAssociatedWithNodeGroup(
        [in] string NodeGroupName
        );

boolean IsAssociatedWithNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName
        );

sint32 GetServiceTypes(
                [out] OV_ServiceTypeDefinition ServiceTypes[]
                );

        sint32 GetServiceTypes_Trans(
                [in] string TransId,
                [out] OV_ServiceTypeDefinition ServiceTypes[]
                );
};
```

OV_AutoDeployPolicyGroup-Properties

Name

Signature

string Name

Description

Unique name/identifier of this Auto-Deployment Policy Group.

Caption

Signature

string Caption

Description

The name used to represent this Auto-Deployment Policy Group when it displays in a console or in an editor.

Description

Signature

string Description

Description

A description of this particular Auto-Deployment Policy Group.

PolicyGroup

Signature

string PolicyGroup

Description

The name of the Policy Group deployed when the associated trigger (adding a node to a node group, or creating a service of a particular service type) occurs.

EnableDeployment

Signature

boolean EnableDeployment

Description

If false, triggers are ignored, and no deployment takes place. If true, deployment occurs when triggers occur, unless a more global mechanism is disabling auto-deployment.

OV_AutoDeployPolicyGroup Class Methods

This section contains the information on Class methods for OV_AutoDeployPolicyGroup.

OV_AutoDeployPolicyGroup::Create() OV_AutoDeployPolicyGroup::Create_Trans()

```
OV_AutoDeployPolicyGroup Create(
        [in] string PolicyGroupPath,
        [in, optional] boolean EnableDeployment,
        [in, optional] string Caption,
        [in, optional] string Name,
        [in, optional] string Description)
```

```
OV_AutoDeployPolicyGroup Create_Trans(
        [in] string TransId,
        [in] string PolicyGroupPath,
        [in, optional] boolean EnableDeployment,
        [in, optional] string Caption,
        [in, optional] string Name,
        [in, optional] string Description)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

PolicyGroupPath

The PolicyGroup property of the instance of OV_AutoDeployPolicyGroup to be created. This property should consist of Caption/Display Names and a single backslash as a separator. It should also start with a single backslash. Before the auto-deploy policy group is actually added, its existence is verified (PMAD). If the policy group does not exist, the method fails with MDLAPI_E_ADPOLGROUP_NOT_EXIST.

EnableDeployment

The EnableDeployment property of the instance of OV_AutoDeployPolicyGroup to be created. Optional parameter. If you do not specify the parameter, the property is set to true.

Caption

The Caption property of the instance of OV_AutoDeployPolicyGroup to be created. If the Caption parameter contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION. Optional parameter.

Name
: The Name property of the instance of OV_AutoDeployPolicyGroup to be created. Optional parameter. If you do not specify the parameter, or if it is equal to an empty string, it is created as a new GUID. If a Node Group with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_ADPOLGROUP_NAME_EXISTS.

Description
: The Description property of the instance of OV_AutoDeployPolicyGroup to be created. Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates an auto-deploy policy group (new instance of OV_AutoDeployPolicyGroup).

Return Value

Instance of the newly created OV_AutoDeployPolicyGroup.

Extended Status Codes

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_ADPOLGROUP_NOT_EXIST
: Auto Deploy Policy Group does not exist.

MDLAPI_E_ADPOLGROUP_NAME_EXISTS
: Auto Deploy Policy Group with the same Name already exists.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_AutoDeployPolicyGroup::GetByName() OV_AutoDeployPolicyGroup::GetByName_Trans()

```
OV_AutoDeployPolicyGroup GetByName(
        [in] string Name)
```

```
OV_AutoDeployPolicyGroup GetByName_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Name
The Name property of the instance of OV_AutoDeployPolicyGroup to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) specified by the Name property.

Return Value

Instance of OV_AutoDeployPolicyGroup specified by the Name property.

Extended Status Codes

MDLAPI_E_ADPOLGROUP_NOT_EXIST
Auto Deploy Policy Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_AutoDeployPolicyGroup::GetByPolicyPath()
OV_AutoDeployPolicyGroup::GetByPolicyPath_Trans()

```
sint32 GetByPolicyPath(
        [out] OV_AutoDeployPolicyGroup PolicyGroups[],
        [in] string PolicyGroupPath)
```

```
sint32 GetByPolicyPath_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPolicyGroup PolicyGroups[],
        [in] string PolicyGroupPath)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

PolicyGroups

Instances of OV_AutoDeployPolicyGroup where the property PolicyGroup is equal to the parameter PolicyGroupPath. An instance is valid as a returned out parameter only if the method does not fail.

PolicyGroupPath

The policy group path, which consists of Caption/Display Names of the policy and its parent policy groups in PMAD, as well as a single backslash as a separator. The path starts with a single backslash. If the Caption contains additional backslashes, you should escape them with backslashes (for example, "\\").

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns a list of the auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) with the specified policy path.

Return Value

The number of instances of OV_AutoDeployPolicyGroup in out parameter PolicyGroups.

Extended Status Codes

MDLAPI_E_ADPOLGROUP_NOT_EXIST
: Auto Deploy Policy Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_AutoDeployPolicyGroup::Remove() OV_AutoDeployPolicyGroup::Remove_Trans()

```
void Remove(
        [in] string Name)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Name
: The Name property of the instance of OV_AutoDeployPolicyGroup to be removed.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified auto-deploy policy group (instance of OV_AutoDeployPolicyGroup).

Return Value

None.

Extended Status Codes

MDLAPI_E_ADPOLGROUP_NOT_EXIST
: Auto Deploy Policy Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_AutoDeployPolicyGroup Instance Methods

This section contains the information on Instance methods for OV_AutoDeployPolicyGroup.

OV_AutoDeployPolicyGroup::GetNodeGroups() OV_AutoDeployPolicyGroup::GetNodeGroups_Trans()

```
sint32 GetNodeGroups(
        [out] OV_NodeGroup NodeGroups[] )
```

```
sint32 GetNodeGroups_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[] )
```

Parameters

TransId
  Transaction ID returned from OV_Transaction::Start().

NodeGroups
  Instances of OV_NodeGroup to which this auto-deploy policy group is added. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of node groups (instances of OV_NodeGroup) to which this auto-deploy policy group is added.

Return Value

Number of node groups in out parameter NodeGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
  Transaction with the specified ID does not exist.

OV_AutoDeployPolicyGroup::GetServiceTypes()
OV_AutoDeployPolicyGroup::GetServiceTypes_Trans()

```
sint32 GetServiceTypes(
        [out] OV_ServiceTypeDefinition ServiceTypes[] )
```

```
sint32 GetServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ServiceTypes[] )
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ServiceTypes
    Instances of OV_ServiceTypeDefinition that are associated with this auto-deploy policy group. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) where this auto-deploy policy group is added.

Return Value

Number of service types in the out parameter ServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_AutoDeployPolicyGroup::IsAssociatedWithNodeGroup() OV_AutoDeployPolicyGroup::IsAssociatedWithNodeGroup_Trans()

```
boolean IsAssociatedWithNodeGroup(
        [in] string NodeGroupName)

boolean IsAssociatedWithNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

NodeGroupName

The Name property of the node group (instance of OV_NodeGroup), which is used to verify that this auto-deploy policy group is associated with the node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if this auto-deploy policy group is associated with a specified node group (instance of OV_NodeGroup).

Return Value

True if this auto-deploy policy group is associated with a node group (instance of OV_NodeGroup) specified by the NodeGroupName parameter.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_AutoDeployPolicyGroup::Modify() OV_AutoDeployPolicyGroup::Modify_Trans()

```
void Modify(
        [in] OV_AutoDeployPolicyGroup ADPolicyGroup)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_AutoDeployPolicyGroup ADPolicyGroup)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ADPolicyGroup
: The modified instance of OV_AutoDeployPolicyGroup to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes performed on one or more properties.

It is not possible to get an instance of OV_AutoDeployPolicyGroup to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, an instance of OV_AutoDeployPolicyGroup is specified as the ADPolicyGroup parameter, even though this method is already called in the context of this instance.

As with the OV_AutoDeployPolicyGroup::Create method, the following checking is performed:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If the Caption property contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING
: Property is not set.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_CalculationRule

Description

If the CalculationThresholdType is set to threshold_by_number, the values indicate the absolute number of children that have to pass a certain severity. Because the Threshold properties are of the type real, you must use 1.0, 2.0, 3.0, and so on.

If the CalculationThresholdType is set to threshold_by_percentage, the values are between 0.0 and 1.0. For example, a single threshold of 25% translates into a value of 0.25 (not 25.0).

```
class OV_CalculationRule
{
    Properties:
    string SettingID;
    uint16 CalculationType;
    uint16 CalculationThresholdType;
    real32 CriticalThreshold;
    real32 MajorThreshold;
    real32 MinorThreshold;
    real32 WarningThreshold;
    real32 NormalThreshold;
    real32 SingleThreshold;
    uint16 CriticalSetTo;
    uint16 MajorSetTo;
    uint16 MinorSetTo;
    uint16 WarningSetTo;
    uint16 NormalSetTo;
    uint16 SingleSetTo;

    Class Methods:

    OV_CalculationRule Create(
        [in] string Caption,
        [in] uint16 CalculationType,
        [in] uint16 CalculationThresholdType,
        [in, optional] uint16 SingleSetTo,
        [in, optional] real32 SingleThreshold,
        [in, optional] uint16 CriticalSetTo,
        [in, optional] real32 CriticalThreshold,
        [in, optional] uint16 MajorSetTo,
        [in, optional] real32 MajorThreshold,
        [in, optional] uint16 MinorSetTo,
        [in, optional] real32 MinorThreshold,
        [in, optional] uint16 WarningSetTo,
        [in, optional] real32 WarningThreshold,
        [in, optional] uint16 NormalSetTo,
        [in, optional] real32 NormalThreshold,
        [in, optional] string SettingID,
        [in, optional] string Description
        );

OV_CalculationRule Create_Trans(
        [in] string TransId,
        [in] string Caption,
        [in] uint16 CalculationType,
        [in] uint16 CalculationThresholdType,
        [in, optional] uint16 SingleSetTo,
        [in, optional] real32 SingleThreshold,
        [in, optional] uint16 CriticalSetTo,
        [in, optional] real32 CriticalThreshold,
        [in, optional] uint16 MajorSetTo,
        [in, optional] real32 MajorThreshold,
        [in, optional] uint16 MinorSetTo,
        [in, optional] real32 MinorThreshold,
        [in, optional] uint16 WarningSetTo,
        [in, optional] real32 WarningThreshold,
        [in, optional] uint16 NormalSetTo,
        [in, optional] real32 NormalThreshold,
        [in, optional] string SettingID,
        [in, optional] string Description
        );

void Remove(
        [in] string SettingId
        );

void Remove_Trans(
        [in] string TransId,
        [in] string SettingId
        );

OV_CalculationRule GetById(
        [in] string SettingId
        );

OV_CalculationRule GetById_Trans(
        [in] string TransId,
        [in] string SettingId
        );
OV_CalculationRule GetByCaption(
        [in] string Caption
        );

OV_CalculationRule GetByCaption_Trans(
        [in] string TransId,
        [in] string Caption
        );
```

Instance Methods:

```
        void Modify(
                [in] OV_CalculationRule CalculationRule
                );

        void Modify_Trans(
                [in] string TransId,
                [in] OV_CalculationRule CalculationRule
                );

        sint32 GetServices(
                [out] OV_Service Services[]
                );

        sint32 GetServices_Trans(
                [in] string TransId,
                [out] OV_Service Services[]
                );

        sint32 GetServiceTypes(
                [out] OV_ServiceTypeDefinition ServiceTypes[]
                );

        sint32 GetServiceTypes_Trans(
                [in] string TransId,
                [out] OV_ServiceTypeDefinition ServiceTypes[]
                );
    };
```

OV_CalculationRule-Properties

SettingID

Signature

string SettingID

Description

The identifier to uniquely identify the setting.

CalculationType

Signature

uint16 CalculationType

Description

Values

ValueMap	Values
0	= calculation_none
1	= calculation_most_critical
2	= calculation_single_threshold
3	= calculation_multi_threshold

CalculationThresholdType

Signature

uint16 CalculationThresholdType

Description

If the CalculationType is 'none' or 'most_critical', the threshold type must be 'not_applicable'.

If the threshold type is 'by_percentage', the values for single and multiple thresholds must be between 0 and 100.

If the threshold type is 'not_applicable', the values for single and multiple thresholds are undefined.

Values

ValueMap	Values
0 =	threshold_not_applicable
1 =	threshold_by_number
2 =	threshold_by_percentage

CriticalThreshold

Signature
real32 CriticalThreshold

Description
Value for the critical threshold.

MajorThreshold

Signature
real32 MajorThreshold

Description
Value for the major threshold.

MinorThreshold

Signature
real32 MinorThreshold

Description
Value for the minor threshold.

WarningThreshold

Signature
real32 WarningThreshold

Description
Value for the warning threshold.

NormalThreshold

Signature
real32 NormalThreshold

Description
Value for the normal threshold.

SingleThreshold

Signature

real32 SingleThreshold

Description

Value for a single threshold.

CriticalSetTo

Signature

uint16 CriticalSetTo

Description

Values

ValueMap	Values
0 =	No SetTo Value
1 =	Unknown
2 =	Normal
4 =	Warning
8 =	Minor
16 =	Major
32 =	Critical

MajorSetTo

Signature

uint16 MajorSetTo

Description

Values

ValueMap	Values
0 =	No SetTo Value
1 =	Unknown
2 =	Normal
4 =	Warning
8 =	Minor
16 =	Major
32 =	Critical

MinorSetTo

Signature

uint16 MinorSetTo

Description

Values

ValueMap	Values
0 =	No SetTo Value
1 =	Unknown
2 =	Normal
4 =	Warning
8 =	Minor
16 =	Major
32 =	Critical

WarningSetTo

Signature

uint16 WarningSetTo

Description

Values

ValueMap	Values
0	No SetTo Value
1	Unknown
2	Normal
4	Warning
8	Minor
16	Major
32	Critical

NormalSetTo

Signature

uint16 NormalSetTo

Description

Values

ValueMap	Values
0	No SetTo Value
1	Unknown
2	Normal
4	Warning
8	Minor
16	Major
32	Critical

SingleSetTo

Signature

uint16 SingleSetTo

Description

Values

ValueMap		Values
0	=	No SetTo Value
1	=	Unknown
2	=	Normal
4	=	Warning
8	=	Minor
16	=	Major
32	=	Critical

OV_CalculationRule Class Methods

This section contains the information on Class methods for OV_CalculationRule.

OV_CalculationRule::Create()
OV_CalculationRule::Create_Trans()

```
OV_CalculationRule Create(
        [in] string Caption,
        [in] uint16 CalculationType,
        [in] uint16 CalculationThresholdType,
        [in, optional] uint16 SingleSetTo,
        [in, optional] real32 SingleThreshold,
        [in, optional] uint16 CriticalSetTo,
        [in, optional] real32 CriticalThreshold,
        [in, optional] uint16 MajorSetTo,
        [in, optional] real32 MajorThreshold,
        [in, optional] uint16 MinorSetTo,
        [in, optional] real32 MinorThreshold,
        [in, optional] uint16 WarningSetTo,
        [in, optional] real32 WarningThreshold,
        [in, optional] uint16 NormalSetTo,
        [in, optional] real32 NormalThreshold,
        [in, optional] string SettingID,
        [in, optional] string Description)

OV_CalculationRule Create_Trans(
        [in] string TransId,
        [in] string Caption,
        [in] uint16 CalculationType,
        [in] uint16 CalculationThresholdType,
        [in, optional] uint16 SingleSetTo,
        [in, optional] real32 SingleThreshold,
        [in, optional] uint16 CriticalSetTo,
        [in, optional] real32 CriticalThreshold,
        [in, optional] uint16 MajorSetTo,
        [in, optional] real32 MajorThreshold,
        [in, optional] uint16 MinorSetTo,
        [in, optional] real32 MinorThreshold,
        [in, optional] uint16 WarningSetTo,
        [in, optional] real32 WarningThreshold,
        [in, optional] uint16 NormalSetTo,
        [in, optional] real32 NormalThreshold,
        [in, optional] string SettingID,
        [in, optional] string Description)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Caption
: The Caption property of the instance of OV_CalculationRule to be created. If a calculation rule with the same Caption already exists, nothing happens, and the method fails with MDLAPI_E_CALCRULE_CAPTION_EXIST. If the Caption parameter is an empty string, or contains invalid characters or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

CalculationType
: The CalculationType property of the instance of OV_CalculationRule to be created. Possible values for calculation type are: 0 (calculation_none) 1 (calculation_most_critical) 2 (calculation_single_threshold) 3 (calculation_multi_threshold)

CalculationThresholdType
: The CalculationThresholdType property of the instance of OV_CalculationRule to be created. Possible values for the calculation threshold type are: 0 (threshold_not_applicable) 1 (threshold_by_number) 2 (threshold_by_percentage) If the CalculationType is 'none' or 'most_critical', the threshold type must be 'not_applicable'. If the threshold type is 'not_applicable', the values for single and multiple thresholds are undefined. If the threshold type is 'threshold_by_number', the values for single and multiple thresholds must be between the 1 and 1000. If the threshold type is 'by_percentage', the values for single and multiple thresholds must be between 0.1 and 1.0

SingleSetTo
: The SingleSetTo property of the instance of OV_CalculationRule to be created. Possible values are: 0 (No SetTo Value) 1 (Unknown) 2 (Normal) 4 (Warning) 8 (Minor) 16 (Major) 32 (Critical) Optional parameter. Default is 0 - 'No ToSet Value'.

SingleThreshold
: The SingleThreshold property of the instance of OV_CalculationRule to be created. Optional parameter. If the threshold type is 'threshold_by_number', the values for single and multiple thresholds must be between 1 and 1000. Default is 1. If the threshold type is 'by_percentage', the values for single and multiple thresholds must be between 0.1 and 1.0. Default is 0.66.

CriticalSetTo
: The CriticalSetTo property of the instance of OV_CalculationRule to be created. Possible values are: 0 (No SetTo Value) 1 (Unknown) 2 (Normal) 4 (Warning) 8 (Minor) 16 (Major) 32 (Critical) Optional parameter. Default is 32 - 'Critical'.

CriticalThreshold

The CriticalThreshold property of the instance of OV_CalculationRule to be created. Optional parameter. If the threshold type is 'threshold_by_number', the values for single and multiple thresholds must be between the 1 and 1000. Default is 5. If the threshold type is 'by_percentage', the values for single and multiple thresholds must be between the 0.1 and 1.0. Default is 0.8.

MajorSetTo

The MajorSetTo property of the instance of OV_CalculationRule to be created. Possible values are: 0 (No SetTo Value) 1 (Unknown) 2 (Normal) 4 (Warning) 8 (Minor) 16 (Major) 32 (Critical) Optional parameter. Default is 16 - 'Major'.

MajorThreshold

The MajorThreshold property of the instance of OV_CalculationRule to be created. Optional parameter. If the threshold type is 'threshold_by_number', the values for single and multiple thresholds must be between the 1 and 1000. Default is 4. If the threshold type is 'by_percentage', the values for single and multiple thresholds must be between 0.1 and 1.0. Default is 0.66.

MinorSetTo

The MinorSetTo property of the instance of OV_CalculationRule to be created. Possible values are: 0 (No SetTo Value) 1 (Unknown) 2 (Normal) 4 (Warning) 8 (Minor) 16 (Major) 32 (Critical) Optional parameter. Default is 8 - 'Minor'.

MinorThreshold

The MinorThreshold property of the instance of OV_CalculationRule to be created. Optional parameter. If the threshold type is 'threshold_by_number', the values for single and multiple thresholds must be between 1 and 1000. Default is 3. If the threshold type is 'by_percentage', the values for single and multiple thresholds must be between 0.1 and 1.0. Default is 0.5.

WarningSetTo

The WarningSetTo property of the instance of OV_CalculationRule to be created. Possible values are: 0 (No SetTo Value) 1 (Unknown) 2 (Normal) 4 (Warning) 8 (Minor) 16 (Major) 32 (Critical) Optional parameter. Default is 4 - 'Warning'.

WarningThreshold

The WarningThreshold property of the instance of OV_CalculationRule to be created. Optional parameter. If the threshold type is 'threshold_by_number', the values for single and multiple thresholds must be between 1 and 1000. Default is 2. If the threshold type is 'by_percentage', the values for single and multiple thresholds must be between 0.1 and 1.0. Default is 0.4.

NormalSetTo

The NormalSetTo property of the instance of OV_CalculationRule to be created. Possible values are: 0 (No SetTo Value) 1 (Unknown) 2 (Normal) 4 (Warning) 8 (Minor) 16 (Major) 32 (Critical) Optional parameter. Default is 2 - 'Normal'.

NormalThreshold
: The NormalThreshold property of the instance of OV_CalculationRule to be created. Optional parameter. If the threshold type is 'threshold_by_number', the values for single and multiple thresholds must be between 1 and 1000. Default is 1. If the threshold type is 'by_percentage', the values for single and multiple thresholds must be between 0.1 and 1.0. Default is 0.2.

SettingID
: The SettingID property of an instance of the OV_CalculationRule to be created. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is created as a new GUID. If a calculation rule with the same SettingID already exists, nothing happens, and the method fails with MDLAPI_E_CALCRULE_SETTINGID_EXISTS.

Description
: The Description property of the instance of OV_CalculationRule to be created. Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a calculation rule (new instance of OV_CalculationRule).

If any calculation rule parameter is not valid, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_CalculationRule.

Extended Status Codes

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_CALCRULE_CAPTION_EXIST
: Calculation rule with the same Caption already exists.

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_CALCRULE_SETTINGID_EXISTS
: Calculation rule with the same SettingId already exists.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_CalculationRule::GetByCaption()
OV_CalculationRule::GetByCaption_Trans()

```
OV_CalculationRule GetByCaption(
        [in] string Caption)
```

```
OV_CalculationRule GetByCaption_Trans(
        [in] string TransId,
        [in] string Caption)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Caption
    The Caption property of an instance of OV_CalculationRule to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the calculation rule (instance of OV_CalculationRule) specified by the Caption property.

Return Value

Instance of OV_CalculationRule specified by a Caption property.

Extended Status Codes

MDLAPI_E_CALCRULE_NOT_EXIST
    Calculation rule does not exist.

MDLAPI_E_CALCRULE_CAPTION_NOT_UNIQUE
    More than one calculation rule with the specified Caption is found.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_CalculationRule::GetById()
OV_CalculationRule::GetById_Trans()

```
OV_CalculationRule GetById(
        [in] string SettingId)
```

```
OV_CalculationRule GetById_Trans(
        [in] string TransId,
        [in] string SettingId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

SettingId
: The SettingId property of an instance of OV_CalculationRule to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the calculation rule (instance of OV_CalculationRule) specified by the SettingId property.

Return Value

Instance of OV_CalculationRule specified by a SettingId property.

Extended Status Codes

MDLAPI_E_CALCRULE_NOT_EXIST
: Calculation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_CalculationRule::Remove()
OV_CalculationRule::Remove_Trans()

```
void Remove(
        [in] string SettingId)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string SettingId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

SettingId
: The SettingId property of an instance of OV_CalculationRule to be removed. If the calculation rule with the SettingId does not exist, the method fails with MDLAPI_E_CALCRULE_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified calculation rule (instance of OV_CalculationRule).

If the specified calculation rule is used by any another object or association, it is not removed, and the method fails with MDLAPI_E_CALCRULE_CANNOT_REMOVE

Return Value

None.

Extended Status Codes

MDLAPI_E_CALCRULE_CANNOT_REMOVE
: Calculation rule cannot be removed because another object or association uses it.

MDLAPI_E_CALCRULE_NOT_EXIST
: Calculation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_CalculationRule Instance Methods

This section contains the information on Instance methods for OV_CalculationRule.

OV_CalculationRule::GetServices()
OV_CalculationRule::GetServices_Trans()

```
sint32 GetServices(
        [out] OV_Service Services[] )
```

```
sint32 GetServices_Trans(
        [in] string TransId,
        [out] OV_Service Services[] )
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Services
    Instances of OV_Service where this calculation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) where this calculation rule is used.

Return Value

Number of services in the out parameter Services.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_CalculationRule::GetServiceTypes()
OV_CalculationRule::GetServiceTypes_Trans()

```
sint32 GetServiceTypes(
        [out] OV_ServiceTypeDefinition ServiceTypes[] )
```

```
sint32 GetServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ServiceTypes[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ServiceTypes
: Instances of OV_ServiceTypeDefinition where this calculation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) where this calculation rule is used.

Return Value

Number of service types in the out parameter ServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_CalculationRule::Modify()
OV_CalculationRule::Modify_Trans()

```
void Modify(
        [in] OV_CalculationRule CalculationRule)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_CalculationRule CalculationRule)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

CalculationRule
: A modified instance of OV_CalculationRule to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes performed to one or more properties.

It is not possible to get an instance of OV_CalculationRule to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, an instance of OV_CalculationRule is specified as the calculation rule parameter, even though this method is already called in the context of this instance.

As with the OV_CalculationRule::Create method, the following checking is performed:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If the Caption property is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.
- If a property has an invalid value, the method fails with MDLAPI_E_INVALID_PROPERTY.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING
Property is not set.

MDLAPI_E_INVALID_CAPTION
Specified object Caption parameter is not valid.

MDLAPI_E_INVALID_PROPERTY
Property is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_ConfigFileVariety

Description

ConfigFile Variety provider.

```
class OV_ConfigFileVariety
{
        Properties:
        string Application;
        string SubGroup;
        string Filename;
        string AboutTemplate;
        string CallFunctionClassID;
        string ValidateClassID;
        string ModificationClassID;

        Class Methods:

        sint32 GetApplicationStrings(
                [out] string Applications[]
                );

        sint32 GetSubGroupStrings(
                [in] string Application,
                [out] string SubGroups[]
                );

        sint32 GetFilenameStrings(
                [in] string Application,
                [in] string SubGroup,
                [out] string Filenames[]
                );

        boolean VarietyIsExisting(
                [in] string Application,
                [in] string SubGroup,
                [in] string Filename
                );

        OV_Transaction GetVariety(
                [in] string Application,
                [in] string SubGroup,
                [in] string Filename
```

```
            );

        Instance Methods:

        boolean HasTemplate(
            [in, optional] string Filename
            );

        boolean HasTemplateFolder(
            [in, optional] string Filename
            );

        string GetTemplate(
            [in, optional] string Filename
            );

        void SaveAsTemplate(
            [in] string Content,
            [in, optional] string Filename
            );
    };
```

OV_ConfigFileVariety-Properties

Application

Signature

string Application

Description

Application part of a ConfigFile Variety.

SubGroup

Signature

string SubGroup

Description

SubGroup part of a ConfigFile Variety.

Filename

Signature

string Filename

Description

Filename part of a ConfigFile Variety.

AboutTemplate

Signature

string AboutTemplate

Description

Command to execute when the policy editor button 'Help on ConfigFile' is pressed. Usually starts the SPI online help with a context- sensitive jump to the description of the current ConfigFile.

If the property is not set, the editor button 'Help on ConfigFile' is disabled.

CallFunctionClassID

Signature

string CallFunctionClassID

Description

COM ClassID of the optional deployment hook routines for deployment commands and content modification (encryption). If the property is not set, the method performs no custom deployment action. It deploys the ConfigFile policy as-is to the node.

It is obsolete as of HPOM for Windows 8.00.

ValidateClassID

Signature

string ValidateClassID

Description

COM ClassID of an optional policy editor validation routine.

If the property is not set, the editor button 'Check Syntax' is disabled, and the method performs no validation when the editor is closed.

ModificationClassID

Signature

string ModificationClassID

Description

COM ClassID of an optional policy editor contents modification routine. If the property is not set, the method performs no modification.

OV_ConfigFileVariety Class Methods

This section contains the information on Class methods for OV_ConfigFileVariety.

OV_ConfigFileVariety::GetApplicationStrings()
OV_ConfigFileVariety::GetApplicationStrings_Trans()

```
sint32 GetApplicationStrings(
        [out] string Applications[] )
```

Parameters

Applications
: Strings representing the existing variety Applications.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns a list of strings representing the existing variety Applications.

Return Value

Number of strings in the out parameter Applications.

Extended Status Codes

None.

OV_ConfigFileVariety::GetSubGroupStrings() OV_ConfigFileVariety::GetSubGroupStrings_Trans()

```
sint32 GetSubGroupStrings(
        [in] string Application,
        [out] string SubGroups[] )
```

Parameters

Application
: Application part of the existing variety.

SubGroups
: Strings representing the SubGroups of the specified Application.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns a list of strings representing the SubGroups of the specified Application.

Return Value

Number of strings in the out parameter SubGroups.

Extended Status Codes

MDLAPI_E_CONFIGFILEVAR_APP_NOT_EXIST
: Specified Application does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ConfigFileVariety::GetFilenameStrings()
OV_ConfigFileVariety::GetFilenameStrings_Trans()

```
sint32 GetFilenameStrings(
        [in] string Application,
        [in] string SubGroup,
        [out] string Filenames[] )
```

Parameters

Application
: Application part of the existing variety.

SubGroup
: SubGroup part of the existing variety.

Filenames
: Strings representing the Filenames of the specified Application and SubGroup.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns a list of strings representing the Filenames of the specified Application and SubGroup.

Return Value

Number of strings in the out parameter Filenames.

Extended Status Codes

MDLAPI_E_CONFIGFILEVAR_APP_NOT_EXIST
: Specified Application does not exist.

MDLAPI_E_CONFIGFILEVAR_SUBGRP_NOT_EXIST
: Specified Application does not have such a SubGroup.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ConfigFileVariety::VarietyIsExisting()
OV_ConfigFileVariety::VarietyIsExisting_Trans()

```
boolean VarietyIsExisting(
        [in] string Application,
        [in] string SubGroup,
        [in] string Filename)
```

Parameters

Application
: Application part of the existing variety.

SubGroup
: SubGroup part of the existing variety.

Filename
: Filename part of the existing variety.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns true if the variety exists.

If the parameter Filename equals "<*>" (without quotes), there must be no Filename under the SubGroup. Otherwise, the method returns false.

Return Value

True if the variety exists.

Extended Status Codes

None.

OV_ConfigFileVariety::GetVariety()
OV_ConfigFileVariety::GetVariety_Trans()

```
OV_Transaction GetVariety(
        [in] string Application,
        [in] string SubGroup,
        [in] string Filename)
```

Parameters

Application
: Application part of the existing variety.

SubGroup
: SubGroup part of the existing variety.

Filename
: Filename part of the existing variety.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the node group (instance of OV_Transaction) specified with the properties Application, SubGroup, and Filename.

If the specified Application does not exist, the method fails with MDLAPI_E_CONFIGFILEVAR_APP_NOT_EXIST.

If the specified SubGroup does not exist under Application, the method fails with MDLAPI_E_CONFIGFILEVAR_SUBGRP_NOT_EXIST.

If the parameter Filename equals "<*>" (without quotes), there must be no Filename under the SubGroup. Otherwise, the method fails with MDLAPI_E_CONFIGFILEVAR_SUBGRP_NOT_EMPTY.

If the specified FileName does not exist under SubGroup, the method fails with MDLAPI_E_CONFIGFILEVAR_FNAME_NOT_EXIST.

Return Value

Instance of OV_Transaction specified with the properties Application, SubGroup, and Filename.

Extended Status Codes

MDLAPI_E_CONFIGFILEVAR_APP_NOT_EXIST
: Specified Application does not exist.

MDLAPI_E_CONFIGFILEVAR_SUBGRP_NOT_EXIST
: Specified Application does not have such a SubGroup.

MDLAPI_E_CONFIGFILEVAR_FNAME_NOT_EXIST
: Specified SubGroup does not have such a Filename.

MDLAPI_E_CONFIGFILEVAR_SUBGRP_NOT_EMPTY
: Specified SubGroup has one or more Filenames.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ConfigFileVariety Instance Methods

This section contains the information on Instance methods for OV_ConfigFileVariety.

OV_ConfigFileVariety::HasTemplate() OV_ConfigFileVariety::HasTemplate_Trans()

```
boolean HasTemplate(
        [in, optional] string Filename)
```

Parameters

Filename
: Name of a file to use as a template. If you do not specify this parameter, the method uses the Filename property.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the current variety has the template associated.

If the current variety does not have the template associated, the policy editor button 'Load Template' is disabled.

If the name of the file (either the Filename parameter or the Filename property) is '<*>', the method fails with MDLAPI_E_CONFIGFILEVAR_TEMPL_FILENAME.

Return Value

True if the current variety has the template associated.

Extended Status Codes

MDLAPI_E_CONFIGFILEVAR_TEMPL_FILENAME
: String '<*>' cannot be used as the Filename.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ConfigFileVariety::GetTemplate()
OV_ConfigFileVariety::GetTemplate_Trans()

```
string GetTemplate(
        [in, optional] string Filename)
```

Parameters

Filename
: Name of a file to use as a template. If you do not specify this parameter, the method uses the Filename property.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the content of the template associated.

If the current variety does not have the template associated (methods HasTemplate, HasTemplateFolder, or both return false), the method fails with MDLAPI_E_CONFIGFILEVAR_TEMPL_NOT_EXIST.

If name of file (either the Filename parameter or the Filename property) is '<*>', the method fails with MDLAPI_E_CONFIGFILEVAR_TEMPL_FILENAME.

Return Value

The content of the template associated.

Extended Status Codes

MDLAPI_E_CONFIGFILEVAR_TEMPL_NOT_EXIST
: Variety does not have the associated template.

MDLAPI_E_CONFIGFILEVAR_TEMPL_FILENAME
: String '<*>' cannot be used as the Filename.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ConfigFileVariety::SaveAsTemplate()
OV_ConfigFileVariety::SaveAsTemplate_Trans()

```
void SaveAsTemplate(
        [in] string Content,
        [in, optional] string Filename)
```

Parameters

Content
: The content of a file that should be saved as the template.

Filename
: Name of a file to use as a template. If you do not specify this parameter, the method uses the Filename property.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Saves the specified content of a file as the template.

If, for the current variety, save-as-template is not allowed (method HasTemplateFolder returns false), the method fails with MDLAPI_E_CONFIGFILEVAR_TEMPL_NOT_ALLOW.

If the name of file (either the Filename parameter or the Filename property) is '<*>', the method fails with MDLAPI_E_CONFIGFILEVAR_TEMPL_FILENAME.

Return Value

None.

Extended Status Codes

MDLAPI_E_CONFIGFILEVAR_TEMPL_NOT_ALLOW
: Template is not allowed for this variety.

MDLAPI_E_CONFIGFILEVAR_TEMPL_FILENAME
: String '<*>' cannot be used as the Filename.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ConfigItem

Description

This class handles configuration items. A configuration item is a value stored in Registry or XPLConfig (location). If earlier versions of HPOM for Windows stored it at another location, it may also have legacy locations, which the provider migrates to the current location.

```
class OV_ConfigItem
{
        Properties:
        string Id;
        string NamespaceId;
        string Caption;
        string Description;
        boolean RequiresServiceRestart;
        boolean InverseLogic;
        string ValueType;
        string DefaultValue;
        string ValueRange;
        string LocationStoreType[];
        string LocationPath[];
        string LocationName[];
        string LocationValueType[];
        boolean ReadOnly;
        string GUIMode;
        sint32 OrderNumber;
        string Dependency;
        boolean Migrate;
        string OnlineHelp;
};
```

OV_ConfigItem-Properties

Id

Signature

string Id

Description

Identifier, PRIMARY KEY. Unique Id, which should contain the component name (for example, "ovow.MsgActSrv.HBChecker.DisableHealthCheck").

NamespaceId

Signature

string NamespaceId

Description

Namespace to which this OV_ConfigItem belongs.

Caption

Signature

string Caption

Description

Caption to display in the Generic Config GUI. This string is localized into all languages supported by HPOM for Windows.

Description

Signature

string Description

Description

Description to display in the Generic Config GUI. This string is localized into all languages supported by HPOM for Windows.

RequiresServiceRestart

Signature

boolean RequiresServiceRestart

Description

The idea of the OvOWConfigProvider is that configuration changes in Registry or XPLConfig become immediately active. There may still be exceptions where a configuration change needs a restart of some HPOM components. For these cases this property must be set to TRUE to make the user aware of the necessary restart actions. Also the Description property must end wi th a "Note:" specifying exactly which services have to be restarted in order to make changes to this ConfigItem become effective. For ReadOnly ConfigItems this property should not be set because it may confuse the user when he gets instruction which services to restart when the ConfigItem was changed for a ConfigItem that he cannot change. The default for this property is FALSE.

InverseLogic

Signature

boolean InverseLogic

Description

In the Config UI we have some boolean ConfigItems that use negative logic, for example "ovow.Auditing.TurnOffAuditingInGeneral". From the usability aspect this is not so easy to use (You need to set the ConfigItem to FALSE to enable auditing which is rather confusing). To work around this usability problem, the property "InverseLogic" was introduced. Setting this property to "TRUE" tells the Config UI to show the opposite of the actual value given by OV_ConfigValue, , i.e. if the registry key for "ovow.Auditing.TurnOffAuditingInGeneral" is set to "FALSE", then the Config UI would show "TRUE" be cause it is an InverseLogic ConfigItem. To make this workaround work, the Caption and the Description also need to be adapted, for example the saption for "ovow.Auditing.TurnOffAuditingInGeneral" was changed from "Turn off in general" to "Turn on in general". The default for this property is "FALSE", i.e. this is not an InverseLogic ConfigItem. NOTE: Setting this property to "TRUE" does not change the behaviour of the ConfigProvider, and it does not change the value that you get from the ConfigProvider via C API or via WMI (OV_ConfigItem); it only advises the Config UI to negate the value it shows. So be careful if you use such a ConfigItem programmatically, because the Caption and the Description actually tells the opposite of what the ConfigItem means. NOTE: This property only makes sense for ConfigItems of the type "Boolean" and "NumericBoolean". For all other ConfigItems it is just ignored.

ValueType

Signature

string ValueType

Description

What the value represents (String, Integer, Boolean, NumericBoolean, or Enumeration). If the value does not match this type, then the provider resets it to DefaultValue.

Values

ValueMap

String

Integer

Boolean

NumericBoolean

Enumeration

DefaultValue

Signature

string DefaultValue

Description

Value to use when the configuration item does not exist in Registry or XPLConfig, or if it contains an invalid value.

ValueRange

Signature

string ValueRange

Description

Defines the acceptable value range for this configuration item. For Boolean values, this attribute is disregarded: Boolean values must always be either "FALSE" or "TRUE". For NumericBoolean values, this attribute is disregarded: NumericBoolean values must always be either "0" or "1". For Integer values, this attribute defines the minimum and maximum value of the Integer value. It must be in the form ".." (for example, "-10..10"). If the Integer value is beyond that range, the provider corrects it to the nearest boundary (min or max). For String values, the provider uses this attribute as an OvXplMatch::Pattern_t, and performs a pattern match of m_currentValue against m_valueRange. If the pattern match fails, then the value is set to m_defaultValue. For Enumeration values, this attribute defines all acceptable Strings as a comma-separated list (for example, "Karl,Gustav,Erwin,*"). If an asterisk (*) is part of this list, the provider also accepts free text input. If the value is not one of the Strings in the list, and if the list does not contain an asterisk (*) item, the provider sets the value to m_defaultValue.

LocationStoreType

Signature

string LocationStoreType[]

Description

LocationStoreType, LocationPath, LocationName, and LocationValueType are String arrays that are strongly interconnected. LocationStoreType[x], LocationPath[x], LocationName[x], and LocationValueType[x] are a quadruple that describes the physical location of this configuration item.

The first quadruple (x=0) describes the current physical location of this configuration item. The following quadruples (x>0) describe legacy locations of this configuration item. If a value is set at a legacy location, the provider deletes the legacy location value, and stores the value in the current location. LocationStoreType[x] determines whether this configuration item is stored in a Registry key or in a XPLConfig namespace.

Values

ValueMap

XPLConfig

Registry

LocationPath

Signature

string LocationPath[]

Description

-- see general comments on LocationStoreType, LocationPath, LocationName and LocationValueType under LocationStoreType attribute. If the LocationStoreType is "XPLConfig", the provider treats the LocationPath attribute as an XPLConfig namespace name. If the LocationStoreType is "Registry", the provider treats the LocationPath attribute as a Registry key name. For Registry keys, the path can start with a key handle to specify the registry hive to use. Key handles may be one of the following: HKLM (for HKEY_LOCAL_MACHINE) HKCR (for HKEY_CLASSES_ROOT) HKCU (for HKEY_CURRENT_USER) HKUS (for HKEY_USERS) HKCC (for HKEY_CURRENT_CONFIG) HKDD (for HKEY_DYN_DATA) HKPD (for HKEY_PERFORMANCE_DATA) HKPT (for HKEY_PERFORMANCE_TEXT) HKPN (for HKEY_PERFORMANCE_NLSTEXT) If you do not specify a key handle, the provider assumes HKLM. Example for a LocationPath: "HKLM\\Software\\Hewlett-Packard\\OVEnterprise\\Management Server\\MsgActSrv"

LocationName

Signature

string LocationName[]

Description

-- see general comments on LocationStoreType, LocationPath, LocationName and LocationValueType under LocationStoreType attribute. LocationName[x] contains the value name within the OV_ConfigNamespace (Registry key or XPLConfig namespace) where this configuration item is stored.

LocationValueType

Signature

string LocationValueType[]

Description

-- see general comments on LocationStoreType, LocationPath, LocationName and LocationValueType under LocationStoreType attribute. For Registry values, LocationValueType[x] determines whether this configuration item is physically stored as an Integer or as a String.

Values

ValueMap
REG_SZ_Value
REG_DWORD_Value

ReadOnly

Signature

boolean ReadOnly

Description

If "TRUE", this configuration item is read-only. That is, it does not allow writing. NOTE: Read-only ConfigItems cannot be migrated. If "ReadOnly" is "TRUE", "Migrate" must be "FALSE".

GUIMode

Signature

string GUIMode

Description

Determines how this configuration item is displayed in the Generic Config GUI. It can be hidden (for product-internal values), visible only in Expert Mode, or always visible.

Values

ValueMap
StandardMode
ExpertMode
Hide

OrderNumber

Signature

sint32 OrderNumber

Description

Number that defines in which sequence the OV_ConfigItem instances are sorted in the Generic Config GUI. Sorting is done in ascending order, which means that an OV_ConfigItem with an OrderNumber of 1 is shown before one with an OrderNumber of 2. Two OV_ConfigItem instances with the same OrderNumber are sorted by their Caption in ascending order. The default for the OrderNumber (if not set) is 0.

Dependency

Signature

string Dependency

Description

Allows defining a dependency to another OV_ConfigItem (or a list of other OV_ConfigItem instance). It means that this OV_ConfigItem (THIS) is dependant on another OV_ConfigItem (DEP). The OV_ConfigItem DEP that this one is dependant on must be of "Boolean" or "NumericBoolean" type. If the OV_ConfigItem DEP is shown as "FALSE" in the Generic Config GUI (this includes the consideration of the "InverseLogic" property) then the OV_ConfigItem THIS is grayed out in the Generic Config GUI and cannot be modified. If there are multiple OV_ConfigItems DEP1, DEP2, ... defined, then OV_ConfigItem THIS is only modifiable if all OV_ConfigItems DEP1, DEP2, ... are shown as "TRUE". A use case for this is for example that OV_ConfigItem THIS is about the logging detail level and OV_ConfigItem DEP is about turning logging on or off; if logging is turned off, then it makes no sense configuring the detail level. This property contains a list of OV_ConfigItem "Id"s of the OV_ConfigItem DEP; this list is separated by comma or semicolon.

Migrate

Signature

boolean Migrate

Description

If "TRUE", the system exports this configuration item during HPOM for Windows migration. NOTE: To be migrated, ConfigItems may not be read-only. If "Migrate" is "TRUE", "ReadOnly" must be "FALSE".

OnlineHelp

Signature

string OnlineHelp

Description

Links to the online help topic that gives more detailed information about this configuration item.

OV_ConfigNamespace

Description

This class handles namespace information. Namespaces are used to group OV_ConfigItems for display in the Generic Config GUI.

```
class OV_ConfigNamespace
{
        Properties:
        string Id;
        string Caption;
        string Description;
        sint32 OrderNumber;
        string OnlineHelp;
};
```

OV_ConfigNamespace-Properties

Id

Signature

string Id

Description

Identifier, PRIMARY KEY. The Id must be unique. It should contain the component name (for example, "ovow.MsgActSrv.HBChecker").

Caption

Signature

string Caption

Description

Caption to display in the Generic Config GUI. The caption string is localized into all languages supported by HPOM for Windows.

Description

Signature

string Description

Description

Description to display in the Generic Config GUI. The description string is localized into all languages supported by HPOM for Windows.

OrderNumber

Signature

sint32 OrderNumber

Description

Number that defines in which sequence the OV_ConfigNamespace instances are sorted in the Generic Config GUI. Sorting is done in ascending order, which means that an OV_ConfigNamespace with an OrderNumber of 1 is shown before one with an OrderNumber of 2. Two OV_ConfigNamespace instances with the same OrderNumber are sorted by their Caption in ascending order. The default for the OrderNumber (if not set) is 0.

OnlineHelp

Signature

string OnlineHelp

Description

Link to the online help topic that provides more detailed information about the OV_ConfigItems in this namespace. This online help topic is the default for all OV_ConfigItems that belong to this namespace.

OV_ConfigValue

Description

This class is used to provide actual values to the configuration items defined in the OV_ConfigItem class. Using this class, you can query and set values for OV_ConfigItems.

```
class OV_ConfigValue
{
        Properties:
        string ItemId;
        string Value;
};
```

OV_ConfigValue-Properties

ItemId

Signature

string ItemId

Description

Id of the OV_ConfigItem to which this OV_ConfigValue belongs.

Value

Signature

string Value

Description

Current value of the OV_ConfigItem.

OV_ConfigValue_ChangeEvent

Description

The OV_ConfigValue_ChangeEvent indicates that an OV_ConfigValue has changed.

```
class OV_ConfigValue_ChangeEvent
{
        Properties:
        string ItemId;
        string NewValue;
};
```

OV_ConfigValue_ChangeEvent-Properties

ItemId

Signature

string ItemId

Description

Id of the OV_ConfigItem of which the value has changed.

NewValue

Signature

string NewValue

Description

New value of the OV_ConfigItem.

OV_ExternalMessage

Description

Users and external applications connected to HPOM for Windows can create an OV_ExternalMessage, which is transformed into an internal message, and processed by the HPOM for Windows message action server.

```
class OV_ExternalMessage
{
        Properties:
        string Id;
        string PrimaryNodeName;
        datetime TimeCreated = "20060101000000.000000-000";
        string Text;
        string OriginalText;
        string MessageGroup;
        string Object;
        string Application;
        string Type;
        string ServiceId;
        boolean DoNotification = FALSE;
        boolean LogOnly = FALSE;
        boolean CreateTroubleTicketInterface = FALSE;
        boolean AcknowledgeAfterTroubleTicket = FALSE;
        sint32 Severity = 1;
        string Source;
        string MessageKey;
        string MessageKeyRelation;
        OV_Message_CA CMA[];
};
```

OV_ExternalMessage-Properties

Id

Signature
: string Id

Description
: Key to identify the external message.

PrimaryNodeName

Signature
: string PrimaryNodeName

Description
: Managed node on which the event occurred.

TimeCreated

Signature
: datetime TimeCreated = "20060101000000.000000-000"

Description
: Time when the message was created on the agent.

Format: YYYYMMDDHHMMSS.mmmmmmsUUU

YYYY - Four-digit year.

MM - Two-digit month.

DD - Two-digit day (01-31).

HH - Two-digit hour (00-23).

MM - Two-digit minute (00-59).

SS - Two-digit number of seconds (00-59).

mmmmmm - Six-digit number of microseconds (000000-999999).

s - Plus sign (+) or minus sign (-) to indicate a positive or negative offset from universal time coordinates (UTC).

UUU - Three-digit offset indicating the number of minutes that the originating time zone deviates from UTC. Initial value is "20060101000000.000000-000".

Text

Signature
: string Text

Description
: The message text.

OriginalText

Signature
: string OriginalText

Description
: The original message text.

MessageGroup

Signature
: string MessageGroup

Description
: Logical grouping of messages by type.

Object

Signature
: string Object

Description
: Specifies the object that was affected by, that was detected, or that caused the message. For example, this could be a printer that sent a message when it stopped accepting requests, or a backup device that sent a message when a backup stopped.

Application

Signature
: string Application

Description
: Specifies the application that was affected by the message or that detected the message.

Type

Signature

string Type

Description

Defines filters for the message stream interface.

ServiceId

Signature

string ServiceId

Description

Service that is affected by the message.

DoNotification

Signature

boolean DoNotification = FALSE

Description

If true, the system sends a notification for the message (for example, by using email or pager).

LogOnly

Signature

boolean LogOnly = FALSE

Description

If true, the system stores the message as a history message.

CreateTroubleTicketInterface

Signature

boolean CreateTroubleTicketInterface = FALSE

Description

If true, the system creates a trouble ticket for the message.

AcknowledgeAfterTroubleTicket

Signature

boolean AcknowledgeAfterTroubleTicket = FALSE

Description

If true, the system acknowledges the message after creating the trouble ticket.

Severity

Signature

sint32 Severity = 1

Description

Indicates how urgent the message is.

Values

ValueMap	Values
1 =	Unknown
2 =	Normal
4 =	Warning
8 =	Minor
16 =	Major
32 =	Critical

Source

Signature

string Source

Description

Name and version of the policy that created the message.

MessageKey

Signature

string MessageKey

Description

Identifies a certain type of message.

MessageKeyRelation

Signature

string MessageKeyRelation

Description

Messages with message keys that match the MessageKeyRelation are acknowledged by this message.

CMA

Signature

OV_Message_CA CMA[]

Description

List of custom message attributes (CMAs).

OV_ExternalNode

Description

Nodes for external events are used in HPOM to handle the following types of messages:

- Messages from systems without the need to configure each system in HPOM as a separate managed node. This is useful for managers in an environment that gets messages forwarded from lower-level managers, from another network, or from system management products such as HP Network Node Manager.
- Messages from systems without an installed agent (for example, SNMP devices).
- Messages from new systems in a certain IP subnet (created from SNMP traps), making it possible to get messages immediately without the need to set them up.

```
class OV_ExternalNode
{
        Properties:
        string Name;
        uint16 Type;
        string Pattern;
        uint32 Order;
        boolean CheckBeforeManagedNodes = FALSE;
        boolean IsInMaintMode = FALSE;
        boolean IsInSchedOutage = FALSE;
        boolean DeleteMsgInMaintMode = FALSE;
        boolean DeleteMsgInSchedOutage = FALSE;

        Class Methods:

        OV_ExternalNode Create(
                [in] string Caption,
                [in] uint16 Type,
                [in] string Pattern,
                [in, optional] uint32 Order,
                [in, optional] boolean CheckBeforeManagedNodes,
                [in, optional] string Name,
                [in, optional] string ParentName,
                [in, optional] string Description
                );

        OV_ExternalNode Create_Trans(
                [in] string TransId,
                [in] string Caption,
                [in] uint16 Type,
                [in] string Pattern,
                [in, optional] uint32 Order,
                [in, optional] boolean CheckBeforeManagedNodes,
                [in, optional] string Name,
                [in, optional] string ParentName,
                [in, optional] string Description
                );

        void Remove(
                [in] string Name
                );

        void Remove_Trans(
                [in] string TransId,
                [in] string Name
                );

        OV_ExternalNode GetByName(
                [in] string Name
                );

        OV_ExternalNode GetByName_Trans(
                [in] string TransId,
                [in] string Name
                );

        OV_ExternalNode GetByHierarchicalPath(
                [in] string Path
                );

        OV_ExternalNode GetByHierarchicalPath_Trans(
                [in] string TransId,
                [in] string Path
                );
```

Instance Methods:

```
void Modify(
        [in] OV_ExternalNode ExternalNode
        );

void Modify_Trans(
        [in] string TransId,
        [in] OV_ExternalNode ExternalNode
        );

sint32 GetParents(
        [out] OV_NodeGroup NodeGroups[],
                [in, optional] boolean IncludeAllHierarchicalParents
                );

        sint32 GetParents_Trans(
                [in] string TransId,
                [out] OV_NodeGroup NodeGroups[],
                [in, optional] boolean IncludeAllHierarchicalParents
                );

        boolean IsChildOf(
                [in] string ParentName
                );

        boolean IsChildOf_Trans(
                [in] string TransId,
                [in] string ParentName
                );

        void SetOutage(
                [in] sint32 IsInOutage,
                [in] boolean IsScheduled,
                [in, optional] sint32 DeleteMessageInOutage
                );

        void SetOutage_Trans(
                [in] string TransId,
                [in] sint32 IsInOutage,
                [in] boolean IsScheduled,
                [in, optional] sint32 DeleteMessageInOutage
                );
};
```

OV_ExternalNode-Properties

Name

Signature

string Name

Description

The inherited Name serves as a key of an external node instance in an enterprise environment.

Type

Signature

uint16 Type

Description

An integer representing the external node type.

Values

ValueMap	Values
0 =	IP Name
1 =	IP Address
2 =	Other

Pattern

Signature

string Pattern

Description

Pattern definition of an external node.

Order

Signature

uint32 Order

Description

Order number for the evaluation.

CheckBeforeManagedNodes

Signature

boolean CheckBeforeManagedNodes = FALSE

Description

If true, the method checks the external node before it checks the managed nodes. If false, it checks the external node after it checks the managed nodes.

IsInMaintMode

Signature

boolean IsInMaintMode = FALSE

Description

If true, the external node is in maintenance mode.

IsInSchedOutage

Signature

boolean IsInSchedOutage = FALSE

Description

If true, the external node is in maintenance mode.

DeleteMsgInMaintMode

Signature

boolean DeleteMsgInMaintMode = FALSE

Description

If true, the method deletes messages for an external node in maintenance.

DeleteMsgInSchedOutage

Signature

boolean DeleteMsgInSchedOutage = FALSE

Description

If true, the method deletes messages for an external node in a scheduled outage.

OV_ExternalNode Class Methods

This section contains the information on Class methods for OV_ExternalNode.

OV_ExternalNode::Create()
OV_ExternalNode::Create_Trans()

```
OV_ExternalNode Create(
        [in] string Caption,
        [in] uint16 Type,
        [in] string Pattern,
        [in, optional] uint32 Order,
        [in, optional] boolean CheckBeforeManagedNodes,
        [in, optional] string Name,
        [in, optional] string ParentName,
        [in, optional] string Description)
```

```
OV_ExternalNode Create_Trans(
        [in] string TransId,
        [in] string Caption,
        [in] uint16 Type,
        [in] string Pattern,
        [in, optional] uint32 Order,
        [in, optional] boolean CheckBeforeManagedNodes,
        [in, optional] string Name,
        [in, optional] string ParentName,
        [in, optional] string Description)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Caption

Caption (Display Name). If an external node with the same Caption already exists on the same hierarchy path level (that is, has the same parent), nothing happens, and the method fails with MDLAPI_E_EXT_NODE_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

Type

Defines the type of external node.

Pattern
: Defines the string pattern that is used to evaluate whether an incoming message belongs to an external node.

Order
: Defines the order in which the external nodes are evaluated. Optional parameter. Default is 0.

CheckBeforeManagedNodes
: Defines whether the external node is checked before the managed nodes. Optional property. Default is false.

Name
: A Name property of an instance of OV_ExternalNode to be created. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, the method creates it as a new GUID. If an external node with same Name already exists, nothing happens, and the method fails with MDLAPI_E_EXT_NODE_NAME_EXISTS.

ParentName
: Name property of an instance of OV_NodeGroup to which a newly created external node is added. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, the method uses the root node group. If the specified node group does not exist, the method fails with MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST.

Description
: The Description property of the instance of OV_ExternalNode to be created. Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates an external node (instance of OV_ExternalNode).

Return Value

Instance of the newly created OV_ExternalNode.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_EXT_NODE_NAME_EXISTS
: External node with the same Name already exists.

MDLAPI_E_EXT_NODE_HIERPATH_EXISTS
: External node with the same hierarchy path (Caption) already exists.

MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST
: Parent node group does not exist.

MDLAPI_E_NODEGROUP_SPECIAL_ADD
: Child cannot be added to a special node group.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ExternalNode::GetByHierarchicalPath() OV_ExternalNode::GetByHierarchicalPath_Trans()

```
OV_ExternalNode GetByHierarchicalPath(
        [in] string Path)
```

```
OV_ExternalNode GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Path
: The external node specified by the hierarchy Path. The Path consists of the Caption/Display Names of parent Node Groups and a single backslash as a separator. The Path starts with a single backslash. If some of the Captions contain backslashes, you should escape them with additional backslashes (for example, "\\").

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the external node (instance of OV_ExternalNode) specified by a hierarchical path.

Return Value

Instance of OV_ExternalNode specified by a hierarchical path.

Extended Status Codes

MDLAPI_E_EXT_NODE_NOT_EXIST
: External node does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ExternalNode::GetByName()
OV_ExternalNode::GetByName_Trans()

```
OV_ExternalNode GetByName(
        [in] string Name)
```

```
OV_ExternalNode GetByName_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Name
    The Name property of an instance of OV_ExternalNode to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the external node (instance of OV_ExternalNode) specified by the Name property.

Return Value

Instance of OV_ExternalNode specified by a Name property.

Extended Status Codes

MDLAPI_E_EXT_NODE_NOT_EXIST
    External node does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_ExternalNode::Remove()
OV_ExternalNode::Remove_Trans()

```
void Remove(
        [in] string Name)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Name
    The Name property of an instance of OV_ExternalNode to be removed.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified external node (instance of OV_ExternalNode).

The method does not remove objects that are associated with this node. It removes associations only.

Return Value

None.

Extended Status Codes

MDLAPI_E_EXT_NODE_NOT_EXIST
    External node does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_ExternalNode Instance Methods

This section contains the information on Instance methods for OV_ExternalNode.

OV_ExternalNode::GetParents()
OV_ExternalNode::GetParents_Trans()

```
sint32 GetParents(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

```
sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

NodeGroups

Instances of OV_NodeGroup that are the direct parents of this external node. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllHierarchicalParents

Defines whether NodeGroups also include instances of OV_NodeGroup that are hierarchical parents (recursive until the root node group), rather than direct parents. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of the node groups for which this external node is a child.

Return Value

Number of node groups (parents) in out parameter NodeGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ExternalNode::IsChildOf() OV_ExternalNode::IsChildOf_Trans()

```
boolean IsChildOf(
        [in] string ParentName)
```

```
boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentName
: The Name property of the node group (instance of OV_NodeGroup) to check if the node group is the parent of this external node.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified node group (instance of OV_NodeGroup) is the parent of this external node.

Return Value

True if the node group (instance of OV_NodeGroup) specified by a parameter of ParentName is the parent of this external node.

Extended Status Codes

MDLAPI_E_NODEGROUP_NOT_EXIST
: Node Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ExternalNode::Modify()
OV_ExternalNode::Modify_Trans()

```
void Modify(
        [in] OV_ExternalNode ExternalNode)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_ExternalNode ExternalNode)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ExternalNode
: A modified instance of OV_ExternalNode to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes performed to one or more properties.

It is not possible to get an instance of OV_ExternalNode to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, the method specifies an instance of OV_ExternalNode as the Node parameter, even though this method is already called in the context of this instance.

As with the OV_ExternalNode::Create method, the method performs the following checking:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If the Caption property is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.
- If a property has an invalid value, the method fails with MDLAPI_E_INVALID_PROPERTY.
- If an external node with the same hierarchical path already exists, the method fails with MDLAPI_E_EXT_NODE_HIERPATH_EXISTS.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING
: Property is not set.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_INVALID_PROPERTY
: Property is not valid.

MDLAPI_E_EXT_NODE_HIERPATH_EXISTS
: External node with the same hierarchy path (Caption) already exists.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ExternalNode::SetOutage()
OV_ExternalNode::SetOutage_Trans()

```
void SetOutage(
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage)
```

```
void SetOutage_Trans(
        [in] string TransId,
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

IsInOutage

Changes/sets external node mode (scheduled outage or maintenance). Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. KEEP does not change the outage mode, ON enables and OFF disables the outage mode, and TOGGLE changes the outage mode from ON to OFF, or from OFF to ON. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

IsScheduled

Flag for scheduled outage or maintenance mode. If this parameter is set to true, the external node is in scheduled outage mode. Otherwise (false), it is in unplanned maintenance mode.

DeleteMessageInOutage

Flag that indicates whether the node message is removed. Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. Default is KEEP. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets scheduled outage/maintenance mode properties of this external node.

Only administrators and operators with special rights can execute this method. Otherwise, the method fails with MDLAPI_E_NO_OPERATION_RIGHTS

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
    Parameter is not valid.

MDLAPI_E_NO_OPERATION_RIGHTS
    User has no rights to execute this operation.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_ManagedNode

Description

A node being managed by HP Operations. The Name property contains a GUID to uniquely identify the node, and to function as the key. The PrimaryNodeName property contains a friendly name (fully qualified DNS name or WINS name) that you can use to identify the node outside of NextGen (for example, ITO/UX or HP NNM). The CommunicationPath property contains a DNS name, a WINS name, or an IP address. This property may be different from the PrimaryNodeName property.

```
class OV_ManagedNode
{
        Properties:
        string Name;
        string AgentId;
        string Domain;
        string Manufacturer;
        string Model;
        uint16 SystemTypeId;
        uint16 OsTypeId;
        uint32 OsVersionId;
        uint16 AgentCommTypeId;
        uint16 AgentBinaryFormatId;
        sint16 CurrentTimeZone;
        string CommunicationPath;
        string PrimaryNodeName;
        boolean AutoUpdateCommunicationPath;
        boolean AlwaysResolveCommunicationAddresses;
        string Alias[];
        string NameFormat;
        boolean ManuallyInstalled = FALSE;
        boolean PrerequisitesOkay = FALSE;
        boolean DisableAutoDeployment = FALSE;
        uint8 NumOfCPUs;
        uint8 CertificateState;
        uint16 HeartBeatInterval;
        uint8 HeartBeatMode;
        boolean IsAgentInstalled;
        boolean IsInMaintMode;
        boolean IsInSchedOutage;
        boolean DeleteMsgInMaintMode;
        boolean DeleteMsgInSchedOutage;
        boolean DisableHeartBeatInMaintMode;
        boolean DisableHeartBeatInSchedOutage;
        boolean UseDefaultHeartBeatInterval;
        boolean RemovePolicyInMaintMode;
        boolean RemovePolicyInSchedOutage;
        boolean AllowCertAutoGranting;
        uint8 CredentialsMode;
```

Class Methods:

```
OV_ManagedNode Create(
        [in] string PrimaryNodeName,
        [in, optional] string ParentName,
        [in, optional] string Caption,
        [in, optional] string CommPath,
        [in, optional] boolean AlwaysResolveCommAddr,
        [in, optional] boolean AutoUpdateCommPath,
        [in, optional] string Domain,
        [in, optional] boolean DisableAutoDeployment,
        [in, optional] string Name,
        [in, optional] uint16 AgentCommTypeId,
        [in, optional] uint16 AgentBinaryFormatId,
        [in, optional] uint16 OsBits
        );

OV_ManagedNode Create_Trans(
        [in] string TransId,
        [in] string PrimaryNodeName,
        [in, optional] string ParentName,
        [in, optional] string Caption,
        [in, optional] string CommPath,
        [in, optional] boolean AlwaysResolveCommAddr,
        [in, optional] boolean AutoUpdateCommPath,
        [in, optional] string Domain,
        [in, optional] boolean DisableAutoDeployment,
        [in, optional] string Name,
        [in, optional] uint16 AgentCommTypeId,
        [in, optional] uint16 AgentBinaryFormatId,
        [in, optional] uint16 OsBits
        );

OV_ManagedNode CreateWithNodeType(
        [in] string PrimaryNodeName,
        [in] uint16 SystemTypeId,
        [in] uint16 OsTypeId,
        [in] uint32 OsVersionId,
        [in, optional] string ParentName,
        [in, optional] string Caption,
        [in, optional] string CommPath,
        [in, optional] boolean AlwaysResolveCommAddr,
        [in, optional] boolean AutoUpdateCommPath,
        [in, optional] string Domain,
        [in, optional] boolean DisableAutoDeployment,
        [in, optional] string Name,
        [in, optional] uint16 AgentCommTypeId,
        [in, optional] uint16 AgentBinaryFormatId,
        [in, optional] uint16 OsBits
        );

    OV_ManagedNode CreateWithNodeType_Trans(
        [in] string TransId,
        [in] string PrimaryNodeName,
        [in] uint16 SystemTypeId,
        [in] uint16 OsTypeId,
        [in] uint32 OsVersionId,
        [in, optional] string ParentName,
        [in, optional] string Caption,
        [in, optional] string CommPath,
        [in, optional] boolean AlwaysResolveCommAddr,
        [in, optional] boolean AutoUpdateCommPath,
        [in, optional] string Domain,
        [in, optional] boolean DisableAutoDeployment,
        [in, optional] string Name,
        [in, optional] uint16 AgentCommTypeId,
        [in, optional] uint16 AgentBinaryFormatId,
        [in, optional] uint16 OsBits
        );

    void Remove(
        [in] string Name
        );

    void Remove_Trans(
        [in] string TransId,
        [in] string Name
        );

    OV_ManagedNode GetByName(
        [in] string Name
        );

    OV_ManagedNode GetByName_Trans(
        [in] string TransId,
        [in] string Name
        );
```

```
OV_ManagedNode GetByPrimaryNodeName(
        [in] string PrimaryNodeName
        );

OV_ManagedNode GetByPrimaryNodeName_Trans(
        [in] string TransId,
        [in] string PrimaryNodeName
        );

OV_ManagedNode GetByHierarchicalPath(
        [in] string Path
        );

OV_ManagedNode GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path
        );

void AssignToOpenViewDefinedGroups(
        );

void AssignToOpenViewDefinedGroups_Trans(
        [in] string TransId
        );
```

Instance Methods:

```
void Modify(
        [in] OV_ManagedNode Node
        );

void Modify_Trans(
        [in] string TransId,
        [in] OV_ManagedNode Node
        );

sint32 GetParents(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

boolean IsChildOf(
        [in] string ParentName
        );
```

```
boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName
        );

boolean AddAction(
        [in] string ActionName
        );

boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

void RemoveAction(
        [in] string ActionName
        );

void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

sint32 GetActions(
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeInherited
        );

sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeInherited
        );

boolean HasAction(
        [in] string ActionName,
        [in, optional] boolean IncludeInherited
        );

boolean HasAction_Trans(
        [in] string TransId,
        [in] string ActionName,
        [in, optional] boolean IncludeInherited
        );

sint32 GetServices(
        [out] OV_Service Services[]
        );
        sint32 GetServices_Trans(
                [in] string TransId,
                [out] OV_Service Services[]
                );

        void SetOutage(
                [in] sint32 IsInOutage,
                [in] boolean IsScheduled,
                [in, optional] sint32 DeleteMessageInOutage,
                [in, optional] sint32 DisableHeartBeatPolingInOutage,
                [in, optional] sint32 RemovePolicyInMaintenance
                );

        void SetOutage_Trans(
                [in] string TransId,
                [in] sint32 IsInOutage,
                [in] boolean IsScheduled,
                [in, optional] sint32 DeleteMessageInOutage,
                [in, optional] sint32 DisableHeartBeatPolingInOutage,
                [in, optional] sint32 RemovePolicyInMaintenance
                );
    };
```

OV_ManagedNode-Properties

Name

Signature

string Name

Description

The inherited Name serves as a key of a System instance in an enterprise environment.

AgentId

Signature

string AgentId

Description

A value that uniquely identifies the managed node. This value is assigned to the managed node when it first comes under management. The value is guaranteed to be the same on all management servers that manage the node.

Domain

Signature

string Domain

Description

The Domain property indicates the domain name of the computer system.

Manufacturer

Signature

string Manufacturer

Description

The Manufacturer property indicates the name of the computer manufacturer (for example, Seagate).

Model

Signature

string Model

Description

The Model property indicates the model name of the computer system.

SystemTypeId

Signature
uint16 SystemTypeId

Description
An integer representing the System processor family type.

It represents the instance of OV_SystemType identified with the key property OV_SystemType::Id.

Together with properties OsTypeId, OsVersionId, AgentCommTypeId, and AgentBinaryFormatId, it identifies the valid platform (instance of OV_NodePlatform).

Part of the HPOM for Windows Platform Model.

OsTypeId

Signature
uint16 OsTypeId

Description
An integer representing the Operating System type.

It represents the instance of OV_OsType identified with the key property OV_OsType::Id.

Together with the properties SystemTypeId, OsVersionId, AgentCommTypeId, and AgentBinaryFormatId, it identifies the valid platform (instance of OV_NodePlatform).

Part of the HPOM for Windows Platform Model.

OsVersionId

Signature
uint32 OsVersionId

Description
An integer describing the Operating System version.

It represents the instance of OV_OsType identified with the key property OV_SystemType::Id.

Together with the properties SystemTypeId, OsTypeId, AgentCommTypeId, and AgentBinaryFormatId, it identifies the valid platform (instance of OV_NodePlatform).

Part of the HPOM for Windows Platform Model.

AgentCommTypeId

Signature

uint16 AgentCommTypeId

Description

An integer representing the type of communication to contact the agent.

It represents the instance of OV_SystemType identified with the key property OV_SystemType::Id.

Together with the properties SystemTypeId, OsTypeId, OsVersionId, and AgentBinaryFormatId, it identifies the valid platform (instance of OV_NodePlatform).

Part of the HPOM for Windows Platform Model.

AgentBinaryFormatId

Signature

uint16 AgentBinaryFormatId

Description

An integer representing the binary format of the agent.

It represents the instance of OV_SystemType identified with the key property OV_SystemType::Id.

Together with the properties SystemTypeId, OsTypeId, OsVersionId, and AgentCommTypeId, it identifies the valid platform (instance of OV_NodePlatform). When the node is created, it can be optional. Other HPOM for Windows Platform Model properties should identify one or more instance(s) of OV_NodePlatform.

Part of the HPOM for Windows Platform Model.

CurrentTimeZone

Signature

sint16 CurrentTimeZone

Description

A CIM-derived attribute that indicates the number of minutes the OperatingSystem is offset from Greenwich Mean Time. The number can be positive, negative, or zero.

CommunicationPath

Signature

string CommunicationPath

Description

The communication path used to address the computer system if the primary name cannot or should not be used to address the system. The path can be a DNS name, the unique WINS name, or an IP address. This path enables users to specify an alternative communication path for computer systems with multiple IP addresses, or to specify multiple ways to contact the system.

PrimaryNodeName

Signature

string PrimaryNodeName

Description

The primary name for this node. The default should be either the fully qualified DNS name or the unique WINS name. You can use this name for reporting purposes, or for linking to other management systems (for example, ITO/UX, HP NNM, and so on).

AutoUpdateCommunicationPath

Signature

boolean AutoUpdateCommunicationPath

Description

If true, allows the CommunicationPath to be automatically updated when the agent sends the server a "Update Communication Information" message.

AlwaysResolveCommunicationAddresses

Signature

boolean AlwaysResolveCommunicationAddresses

Description

If true, communication to the object must always be resolved before contacting the agent.

Alias

Signature

string Alias[]

Description

The Alias property is used to identify the additional names by which a node may be known. This property is used by the Event server so that a managed node can generate a message using one of several names associated with the node, and attach the message to the correct node. Each string in the array represents an alternate name for the node. These names can be DNS names or unique WINS names.

The Alias property can be used to solve problems that can occur if the node and the management system use a different name resolution and return different node names for the same computer system.

NameFormat

Signature

string NameFormat

Description

The ComputerSystem object and its derivatives are Top Level Objects of CIM. They provide the scope for numerous components. Having unique System keys is required. A heuristic is defined to create the ComputerSystem Name in such a way that the same Name is always generated, regardless of discovery protocol. This heuristic prevents inventory and management problems in which the same asset or entity is discovered multiple times, but cannot be resolved to a single object. Use of the heuristic is optional but recommended.

The NameFormat property indicates how the ComputerSystem Name is generated, using a heuristic. The heuristic is outlined, in detail, in the CIM V2 System Model specification. It assumes that the documented rules are traversed in order, to determine and assign a Name. The NameFormat Values list defines the precedence order for assigning the Computer System Name. Several rules map to the same Value.

NOTE: The ComputerSystem Name calculated using the heuristic is the System key value. Other names can be assigned and used for the ComputerSystem that better suit a business, using Aliases.

Values

ValueMap
Other
IP
Dial
HID
NWA
HWA
X25
ISDN
IPX
DCC
ICD
E.164
SNA
OID/OSI

ManuallyInstalled

Signature

boolean ManuallyInstalled = FALSE

Description

If true, this node is brought under management from the unmanaged node listing in the node editor. This property is used by PMAD to determine the correct error message to write when there is a policy deployment failure.

PrerequisitesOkay

Signature

boolean PrerequisitesOkay = FALSE

Description

If true, the prerequisites for this node have been checked, and the node is deemed manageable. This property is set by the node editor.

DisableAutoDeployment

Signature

boolean DisableAutoDeployment = FALSE

Description

If true, no policies or packages are automatically deployed to this node. This property is primarily set when nodes are imported from another management server. The primary mechanism for such imports is ovpmutil, which wraps OvOWMofComp. OvOWMofComp has a -noautodeploy option.

NumOfCPUs

Signature

uint8 NumOfCPUs

Description

The number of CPUs in the managed node. Used by the HPOM Embedded Licensing.

CertificateState

Signature

uint8 CertificateState

Description

An integer representing the Certificate State. Used by the Certificate Server Adapter.

Values

ValueMap	Values
0	= Undefined
1	= Pending
2	= Granted
3	= Denied
4	= Failed
5	= Installed

HeartBeatInterval

Signature

uint16 HeartBeatInterval

Description

Interval between heart beat polling checks in seconds. Part of HTTPS agent integration.

HeartBeatMode

Signature

uint8 HeartBeatMode

Description

Heart beat mode. Part of HTTPS agent integration.

Values

ValueMap	Values
0	= HeartBeatDisabled
1	= AgentPollingOnly
2	= ICMPPollingOnly
3	= HeartBeatEnabled
4	= UseSystemDefault

IsAgentInstalled

Signature

boolean IsAgentInstalled

Description

True if the managed node has an agent installed on it. Part of HTTPS agent integration.

IsInMaintMode

Signature

boolean IsInMaintMode

Description

True if the managed node is in maintenance mode.

IsInSchedOutage

Signature

boolean IsInSchedOutage

Description

True if the managed node is in a scheduled outage.

DeleteMsgInMaintMode

Signature

boolean DeleteMsgInMaintMode

Description

True if the messages to the managed node should be deleted while it is in maintenance mode.

DeleteMsgInSchedOutage

Signature

boolean DeleteMsgInSchedOutage

Description

True if the messages to the managed node should be deleted during a scheduled outage.

DisableHeartBeatInMaintMode

Signature

boolean DisableHeartBeatInMaintMode

Description

True if heart beat polling should be disabled while the managed node is in maintenance mode.

DisableHeartBeatInSchedOutage

Signature

boolean DisableHeartBeatInSchedOutage

Description

True if heart beat polling should be disabled during a scheduled outage.

UseDefaultHeartBeatInterval

Signature

boolean UseDefaultHeartBeatInterval

Description

True if heart beat polling should use the default heart beat interval.

RemovePolicyInMaintMode

Signature

boolean RemovePolicyInMaintMode

Description

True if policies should be removed while the managed node is in maintenance mode.

RemovePolicyInSchedOutage

Signature

boolean RemovePolicyInSchedOutage

Description

True if policies should be removed during a scheduled outage.

AllowCertAutoGranting

Signature

boolean AllowCertAutoGranting

Description

True if certificate requests sent from this node should be granted automatically.

CredentialsMode

Signature

uint8 CredentialsMode

Description

Sets the authentication mechanism used to communicate with the managed node.

Values

ValueMap	Values
0 =	PMAD User
1 =	Client Impersonation
2 =	Username/Password

OV_ManagedNode Class Methods

This section contains the information on Class methods for OV_ManagedNode.

OV_ManagedNode::Create()
OV_ManagedNode::Create_Trans()

```
OV_ManagedNode Create(
        [in] string PrimaryNodeName,
        [in, optional] string ParentName,
        [in, optional] string Caption,
        [in, optional] string CommPath,
        [in, optional] boolean AlwaysResolveCommAddr,
        [in, optional] boolean AutoUpdateCommPath,
        [in, optional] string Domain,
        [in, optional] boolean DisableAutoDeployment,
        [in, optional] string Name,
        [in, optional] uint16 AgentCommTypeId,
        [in, optional] uint16 AgentBinaryFormatId,
        [in, optional] uint16 OsBits)
```

```
OV_ManagedNode Create_Trans(
        [in] string TransId,
        [in] string PrimaryNodeName,
        [in, optional] string ParentName,
        [in, optional] string Caption,
        [in, optional] string CommPath,
        [in, optional] boolean AlwaysResolveCommAddr,
        [in, optional] boolean AutoUpdateCommPath,
        [in, optional] string Domain,
        [in, optional] boolean DisableAutoDeployment,
        [in, optional] string Name,
        [in, optional] uint16 AgentCommTypeId,
        [in, optional] uint16 AgentBinaryFormatId,
        [in, optional] uint16 OsBits)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

PrimaryNodeName

Primary Node Name. Cannot use the Primary Node Name of an existing node. If you use an existing name, nothing happens, and the method fails with MDLAPI_E_NODE_PNNAME_EXISTS.

ParentName
: Name property of an instance of OV_NodeGroup to which a newly created node is added. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, the root node group is used. If the specified node group does not exist, the method fails with MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST.

Caption
: Caption (Display Name). Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is the same as a short name (without a domain, before the first '.') in the PrimaryNodeName. If a node with the same Caption already exists on the same hierarchy path level (that is, has the same parent), nothing happens, and the method fails with MDLAPI_E_NODE_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

CommPath
: Communication Path. Optional parameter. If you do not specify this parameter, an empty string is used.

AlwaysResolveCommAddr
: Copied to the AlwaysResolveCommunicationAddresses property. Optional parameter.

AutoUpdateCommPath
: Copied to the AutoUpdateCommunicationPath property. Optional parameter.

Domain
: Copied to the Domain property. Optional parameter. If you do not specify this parameter, the domain (text after the first '.') in PrimaryNodeName is used. If PrimaryNodeName contains a short name, the domain is empty.

DisableAutoDeployment
: Copied to the DisableAutoDeployment property. Optional parameter.

Name
: A Name property of an instance of OV_ManagedNode to be created. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is created as a new GUID. If a node with same Name already exists, nothing happens, and the method fails with MDLAPI_E_NODE_NAME_EXISTS.

AgentCommTypeId
: The Id property of the instance of OV_AgentCommType that represents agent comunication type (HTTPS or DCE). Optional parameter. Default is 0 (HTTPS) if the IsDefault property on OV_AgentCommType is set to true. Otherwise, it is empty.

AgentBinaryFormatId

The Id property of the instance of OV_AgentBinaryFormat that represents the agent binary format (x86, x64, IA64, and so on). Optional parameter. You can specify the default value if there is exactly one OV_AgentBinaryFormat instance for this node (based on the node discovered system and agent communication type).

OsBits

Bit length of the default OS type when the OS type could not be discovered. Optional parameter. It is matched with the BitLen property of all supported instances of OV_OsType. It is not used when OS type can be discovered. If the OS type cannot be discovered, and this parameter is not specified, the method fails with MDLAPI_E_NODE_OSBITS_NOT_EXIST. If the OS type cannot be discovered, and this parameter does not match the BitLen property of any of the supported instances of OV_OsType, the method fails with MDLAPI_E_NODE_OSBITS_NOT_MATCH.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a node (a new instance of OV_ManagedNode) with the node type auto-discovery, and adds it to an existing node group (instance of OV_NodeGroup).

This method consists of several operations:

- Attempts to get OS values directly from the node, using DNS discovery (system type, OS type, and OS version). If this fails, defaults are used (Unknown/Other).
- Adds a node to HPOM for Windows.
- Adds a node to the HPOM for Windows pre-defined node group. If the OS detection fails, the node is set to Unknown.
- Adds a node to the node group specified by NodeGroupPath (default is root). If this node group does not exist, the method fails with MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST.
- If any node parameter is not valid, the method fails with MDLAPI_E_INVALID_PARAMETER.
- If node platform checking fails, the method fails with MDLAPI_E_NODE_PLATFORM_MATCHING.
- If license status is critical, node creation is refused, and the method fails with MDLAPI_E_LICENSE_MISSING

If an operation fails, a "rollback" is performed, and an error is returned.

Return Value

Instance of the newly created OV_ManagedNode.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_NODE_NAME_EXISTS
: Node with the same Name already exists.

MDLAPI_E_NODE_PNNAME_EXISTS
: Node with the same Primary Node Name already exists.

MDLAPI_E_NODE_DISCOVERY_FAILED
: Discovery of the node type failed.

MDLAPI_E_NODE_HIERPATH_EXISTS
: Node with the same hierarchy path (Caption) already exists.

MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST
: Parent node group does not exist.

MDLAPI_E_NODE_PLATFORM_MATCHING
: Node platform matching failed.

MDLAPI_E_LICENSE_MISSING
: Specified operation cannot be executed because of missing licenses.

MDLAPI_E_NODE_OSBITS_NOT_EXIST
: You must specify the OSBits parameter when the OS type cannot be discovered.

MDLAPI_E_NODE_OSBITS_NOT_MATCH
: OSBits parameter does not match any supported OS types.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ManagedNode::CreateWithNodeType()
OV_ManagedNode::CreateWithNodeType_Trans()

```
OV_ManagedNode CreateWithNodeType(
        [in] string PrimaryNodeName,
        [in] uint16 SystemTypeId,
        [in] uint16 OsTypeId,
        [in] uint32 OsVersionId,
        [in, optional] string ParentName,
        [in, optional] string Caption,
        [in, optional] string CommPath,
        [in, optional] boolean AlwaysResolveCommAddr,
        [in, optional] boolean AutoUpdateCommPath,
        [in, optional] string Domain,
        [in, optional] boolean DisableAutoDeployment,
        [in, optional] string Name,
        [in, optional] uint16 AgentCommTypeId,
        [in, optional] uint16 AgentBinaryFormatId,
        [in, optional] uint16 OsBits)

OV_ManagedNode CreateWithNodeType_Trans(
        [in] string TransId,
        [in] string PrimaryNodeName,
        [in] uint16 SystemTypeId,
        [in] uint16 OsTypeId,
        [in] uint32 OsVersionId,
        [in, optional] string ParentName,
        [in, optional] string Caption,
        [in, optional] string CommPath,
        [in, optional] boolean AlwaysResolveCommAddr,
        [in, optional] boolean AutoUpdateCommPath,
        [in, optional] string Domain,
        [in, optional] boolean DisableAutoDeployment,
        [in, optional] string Name,
        [in, optional] uint16 AgentCommTypeId,
        [in, optional] uint16 AgentBinaryFormatId,
        [in, optional] uint16 OsBits)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

PrimaryNodeName
: Primary Node Name. If a node with the same Primary Node Name already exists, nothing happens, and the method fails with MDLAPI_E_NODE_PNNAME_EXISTS.

SystemTypeId
: The Id property of the instance of OV_SystemType that represents the processor family type (Pentium, Itanium, Power PC, and so on).

OsTypeId
: The Id property of the instance of OV_OsType that represents the operating system (Windows, HP-UX, Solaris, and so on).

OsVersionId
: The Id property of the instance of OV_OsVersion that represents the operating system version number (2000 (5.0), SuSE 9.X (2.4), and so on).

ParentName
: The Name property of an instance of OV_NodeGroup to which the newly created node is added. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, the root node group is used. If the specified node group does not exist, the method fails with MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST.

Caption
: Caption (Display Name). Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is the same as the short name (without a domain, until the first '.') in PrimaryNodeName. If a node with the same Caption already exists on the same hierarchy path level (that is, has the same parent), nothing happens, and the method fails with MDLAPI_E_NODE_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

CommPath
: Communication Path. Optional parameter. If you do not specify this parameter, an empty string is used.

AlwaysResolveCommAddr
: Copied to the AlwaysResolveCommunicationAddresses property. Optional parameter.

AutoUpdateCommPath
: Copied to the AutoUpdateCommunicationPath property. Optional parameter.

Domain

Copied to the Domain property. Optional parameter. If you do not specify this parameter, the domain (the text after the first '.') in PrimaryNodeName is used. If PrimaryNodeName contains a short name, the string is empty.

DisableAutoDeployment

Copied to the DisableAutoDeployment property. Optional parameter.

Name

The Name property of an instance of the OV_ManagedNode to be created. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is created as a new GUID. If a node with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_NODE_NAME_EXISTS.

AgentCommTypeId

The Id property of the instance of OV_AgentCommType that represents the agent communication type (HTTPS or DCE). Optional parameter. Default is 0 (HTTPS) if the IsDefault property on OV_AgentCommType is set to true. Otherwise, is it is empty.

AgentBinaryFormatId

The Id property of the instance of OV_AgentBinaryFormat that represents the agent binary format (x86, x64, IA64, and so on). Optional parameter. You can specify the default value if there is exactly one OV_AgentBinaryFormat instance for this node (based on the node system input parameters).

OsBits

Bit length of the default OS type when the OS type could not be discovered. Optional parameter. It is matched with the BitLen property of all supported instances of OV_OsType. It is not used when OS type can be discovered. If the OS type cannot be discovered, and this parameter is not specified, the method fails with MDLAPI_E_NODE_OSBITS_NOT_EXIST. If the OS type cannot be discovered, and this parameter does not match the BitLen property of any of the supported instances of OV_OsType, the method fails with MDLAPI_E_NODE_OSBITS_NOT_MATCH.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a node (a new instance of OV_ManagedNode) with a specified type (without auto-discovery), and adds it to an existing node group (an instance of OV_NodeGroup).

This method consists of several operations:

- Adds a node to HPOM for Windows.

- Adds a node to the HPOM for Windows pre-defined node group.
- Adds a node to the node group specified by NodeGroupPath (the default is root). If this node group does not exist, the method fails with MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST.
- If any node type parameter is not valid, the method fails with MDLAPI_E_INVALID_PARAMETER.
- If node platform checking fails, the method fails with MDLAPI_E_NODE_PLATFORM_MATCHING.
- If license status is critical, node creation is refused, and the method fails with MDLAPI_E_LICENSE_MISSING

If some of the operations fail, a "rollback" is performed, and an error is returned.

Return Value

Instance of the newly created OV_ManagedNode.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
    Parameter is not valid.

MDLAPI_E_INVALID_CAPTION
    Specified object Caption parameter is not valid.

MDLAPI_E_NODE_NAME_EXISTS
    Node with the same Name already exists.

MDLAPI_E_NODE_PNNAME_EXISTS
    Node with the same Primary Node Name already exists.

MDLAPI_E_NODE_HIERPATH_EXISTS
    Node with the same hierarchy path (Caption) already exists.

MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST
    Parent node group does not exist.

MDLAPI_E_NODE_PLATFORM_MATCHING
    Node platform matching failed.

MDLAPI_E_LICENSE_MISSING
    Specified operation cannot be executed because of missing licenses.

MDLAPI_E_NODE_OSBITS_NOT_EXIST
    You must specify the OSBits parameter when the OS type cannot be discovered.

MDLAPI_E_NODE_OSBITS_NOT_MATCH
    OSBits parameter does not match any supported OS types.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_ManagedNode::GetByHierarchicalPath() OV_ManagedNode::GetByHierarchicalPath_Trans()

```
OV_ManagedNode GetByHierarchicalPath(
        [in] string Path)
```

```
OV_ManagedNode GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Path
: The node specified by the hierarchy Path. The Path consists of the Caption/Display Names of the parent Node Groups and a single backslash as a separator. The Path starts with a single backslash. If some of the Captions contain backslashes, you should escape them with additional backslashes (for example, "\\").

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the node (instance of OV_ManagedNode) specified by a hierarchical path.

Return Value

Instance of OV_ManagedNode specified by a hierarchical path.

Extended Status Codes

MDLAPI_E_NODE_NOT_EXIST
: Node does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ManagedNode::GetByName() OV_ManagedNode::GetByName_Trans()

```
OV_ManagedNode GetByName(
        [in] string Name)
```

```
OV_ManagedNode GetByName_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Name
The Name property of an instance of OV_ManagedNode to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the node (instance of OV_ManagedNode) specified by the Name property.

Return Value

Instance of OV_ManagedNode specified by a Name property.

Extended Status Codes

MDLAPI_E_NODE_NOT_EXIST
Node does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_ManagedNode::GetByPrimaryNodeName()
OV_ManagedNode::GetByPrimaryNodeName_Trans()

```
OV_ManagedNode GetByPrimaryNodeName(
        [in] string PrimaryNodeName)
```

```
OV_ManagedNode GetByPrimaryNodeName_Trans(
        [in] string TransId,
        [in] string PrimaryNodeName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

PrimaryNodeName
: The PrimaryNodeName property of an instance of OV_ManagedNode to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the node (instance of OV_ManagedNode) specified by the PrimaryNodeName property.

Return Value

Instance of OV_ManagedNode specified by the PrimaryNodeName property.

Extended Status Codes

MDLAPI_E_NODE_NOT_EXIST
: Node does not exist.

MDLAPI_E_NODE_PNNAME_NOT_UNIQUE
: More than one node with the specified Primary Node Name is found.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ManagedNode::Remove()
OV_ManagedNode::Remove_Trans()

```
void Remove(
        [in] string Name)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Name
The Name property of an instance of OV_ManagedNode to be removed.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified node (instance of OV_ManagedNode).

Objects that are associated with this node are not removed. Only associations are removed. The management server cannot be removed! Any attempt to remove the management server fails with MDLAPI_E_NODE_IS_MANAGEMENT_SERVER.

⚠ CAUTION:
This method removes the specified node, and the services hosted on it, from the management server inventory. Policies and packages on the specified managed node are not removed automatically. But inventory information about the node is removed from the management server database. Always remove policies and packages before removing the node.

Return Value

None.

Extended Status Codes

MDLAPI_E_NODE_NOT_EXIST
: Node does not exist.

MDLAPI_E_NODE_IS_MANAGEMENT_SERVER
: Management server cannot be removed.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ManagedNode Instance Methods

This section contains the information on Instance methods for OV_ManagedNode.

OV_ManagedNode::AddAction()
OV_ManagedNode::AddAction_Trans()

```
boolean AddAction(
        [in] string ActionName)
```

```
boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ActionName
: The Name property of the instance of OV_Action to be added to this node.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds the tool (instance of OV_Action) to this node.

Return Value

False if the tool is already added to this node.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
: Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ManagedNode::GetActions()
OV_ManagedNode::GetActions_Trans()

```
sint32 GetActions(
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeInherited)
```

```
sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeInherited)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Actions

Instances of OV_Action that are added to this node. An instance is valid as a returned out parameter only if the method does not fail.

IncludeInherited

Defines whether Actions also contains instances of OV_Action that are inherited from the parents of this node (and their parents, recursively). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of tools (instances of OV_Action) added to this node.

Return Value

Number of tools in the out parameter Actions.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ManagedNode::GetParents()
OV_ManagedNode::GetParents_Trans()

```
sint32 GetParents(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

```
sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

NodeGroups

Instances of OV_NodeGroup that are the direct parents of this node. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllHierarchicalParents

Defines whether NodeGroups also include instances of OV_NodeGroup that are hierarchical parents (recursive until the root node group), rather than direct parents. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of node groups for which this node is a child.

Return Value

Number of node groups (parents) in out parameter NodeGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ManagedNode::GetServices()
OV_ManagedNode::GetServices_Trans()

```
sint32 GetServices(
        [out] OV_Service Services[] )
```

```
sint32 GetServices_Trans(
        [in] string TransId,
        [out] OV_Service Services[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Services
: Instances of OV_Service that are added to this node. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) hosted on this node.

Return Value

Number of services in the out parameter Services.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ManagedNode::HasAction()

OV_ManagedNode::HasAction_Trans()

```
boolean HasAction(
        [in] string ActionName,
        [in, optional] boolean IncludeInherited)
```

```
boolean HasAction_Trans(
        [in] string TransId,
        [in] string ActionName,
        [in, optional] boolean IncludeInherited)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ActionName
: The Name property of the tool (instance of OV_Action), which is used to verify that the tool is associated with this node.

IncludeInherited
: Defines whether the tool to find can also be inherited from these node parents (and their parents, recursively). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified tool (instance of OV_Action) is associated with this node.

Return Value

False if the tool is already added to this node.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

```
Transaction with the specified ID does not exist.
```

OV_ManagedNode::IsChildOf() OV_ManagedNode::IsChildOf_Trans()

```
boolean IsChildOf(
        [in] string ParentName)
```

```
boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentName
: The Name property of the node group (instance of OV_NodeGroup) to check if the node group is the parent of this node.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified node group (instance of OV_NodeGroup) is the parent of this node.

Return Value

True if the node group (instance of OV_NodeGroup) specified by a parameter of ParentName is the parent of this node.

Extended Status Codes

MDLAPI_E_NODEGROUP_NOT_EXIST
: Node Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ManagedNode::Modify()
OV_ManagedNode::Modify_Trans()

```
void Modify(
        [in] OV_ManagedNode Node)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_ManagedNode Node)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Node
: A modified instance of OV_ManagedNode to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes performed to one or more properties.

It is not possible to get an instance of OV_ManagedNode to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, an instance of OV_ManagedNode is specified as the Node parameter, even though this method is already called in the context of this instance.

As with the OV_ManagedNode::Create method, this method checks the following:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If the Caption property is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.
- If a property has an invalid value, the method fails with MDLAPI_E_INVALID_PROPERTY.
- If the OS properties has been changed before calling this method, the node is also moved to the appropriate special node group.

- If a node with the same Primary Node Name already exists, the method fails with MDLAPI_E_NODE_PNNAME_EXISTS.
- If a node with the same hierarchical path already exists, the method fails with MDLAPI_E_NODE_HIERPATH_EXISTS.
- If node platform checking fails, the method fails with MDLAPI_E_NODE_PLATFORM_MATCHING.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING	Property is not set.
MDLAPI_E_INVALID_CAPTION	Specified object Caption parameter is not valid.
MDLAPI_E_INVALID_PROPERTY	Property is not valid.
MDLAPI_E_NODE_PNNAME_EXISTS	Node with the same Primary Node Name already exists.
MDLAPI_E_NODE_HIERPATH_EXISTS	Node with the same hierarchy path (Caption) already exists.
MDLAPI_E_NODE_PLATFORM_MATCHING	Node platform matching failed.
MDLAPI_E_TRANSACTION_NOT_EXIST	Transaction with the specified ID does not exist.

OV_ManagedNode::RemoveAction()
OV_ManagedNode::RemoveAction_Trans()

```
void RemoveAction(
        [in] string ActionName)
```

```
void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ActionName
    The Name property of the instance of OV_Action to be removed from this node.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes the tool (instance of OV_Action) from this node.

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
    Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_ManagedNode::SetOutage()
OV_ManagedNode::SetOutage_Trans()

```
void SetOutage(
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage,
        [in, optional] sint32 DisableHeartBeatPolingInOutage,
        [in, optional] sint32 RemovePolicyInMaintenance)
```

```
void SetOutage_Trans(
        [in] string TransId,
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage,
        [in, optional] sint32 DisableHeartBeatPolingInOutage,
        [in, optional] sint32 RemovePolicyInMaintenance)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

IsInOutage
: Changes/sets the node mode (scheduled outage or maintenance). Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. KEEP does not change the node outage mode, ON enables and OFF disables the node outage mode, and TOGGLE changes the node outage mode from ON to OFF, or from OFF to ON. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

IsScheduled
: Flag for a scheduled outage or maintenance mode. If this parameter is set to true, the node is in scheduled outage mode. Otherwise (false), the node is in unplanned maintenance mode.

DeleteMessageInOutage
: Flag that indicates whether node messages are removed. Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. Default is KEEP. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

DisableHeartBeatPolingInOutage
: Flag that indicates whether node heart beat polling is disabled. Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. Default is KEEP. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

RemovePolicyInMaintenance
: Flag that indicates whether policies are removed from the node. Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. Default is KEEP. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets scheduled outage/maintenance mode properties of this node.

Also services hosted on this node are affected (according to new node mode). Only administrators and operators with special rights can execute this method. Otherwise the method fails with MDLAPI_E_NO_OPERATION_RIGHTS

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_NO_OPERATION_RIGHTS
: User has no rights to execute this operation.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Message

Description

Messages are received from managed nodes that have been generated as a result of many different types of events.

```
class OV_Message
{
	Properties:
	string Id;
	string OriginalId;
	string ConditionId;
	string NodeName;
	string AgentId;
	string PrimaryNodeName;
	datetime TimeCreated;
	datetime TimeReceived;
	string Text;
	string OriginalText;
	string MessageGroup;
	string Object;
	string Application;
	string Type;
	string ServiceId;
	string OriginalServiceId;
	sint32 NumberOfAnnotations = 0;
	string UsedNotificationInterfaces;
	boolean DoNotification = FALSE;
	boolean Unmatched = TRUE;
	boolean LogOnly = FALSE;
	boolean IsProxied = FALSE;
	boolean CreateTroubleTicketInterface = FALSE;
	boolean AcknowledgeAfterTroubleTicket = FALSE;
	boolean IsExternalNode = FALSE;
	sint32 Severity = 1;
	sint32 State = 1;
	datetime TimeOfStateChange;
	string UserOfStateChange;
	sint32 SourceType = 0x0;
	string Source;
	string MessageKey;
	string MessageKeyRelation;
```

```
sint32 NumberOfDuplicates = 0;
datetime TimeFirstReceived;
sint32 NumberOfStoredDuplicates = 0;
OV_MessageAction AutomaticAction;
OV_MessageAction OperatorAction;
boolean InstructionAvailable = FALSE;
string InstructionID;
sint32 InstructionType = 0;
string InstructionParameters;
OV_Message_CA CMAs[];
string Origin;
string Sender;
```

Class Methods:

```
sint32 AcknowledgeMessages(
        [in] string MessageIDs[],
        [out] OV_Message_MassOperationResult MassOperationResults[]
        );

sint32 UnacknowledgeMessages(
        [in] string MessageIDs[],
        [out] OV_Message_MassOperationResult MassOperationResults[]
        );

sint32 OwnMessages(
        [in] string MessageIDs[],
        [in] string NewOwner,
        [out] OV_Message_MassOperationResult MassOperationResults[]
        );

sint32 DisownMessages(
        [in] string MessageIDs[],
        [out] OV_Message_MassOperationResult MassOperationResults[]
        );

sint32 CountMessages(
        [in] string WhereClause,
        [out] sint32 Count
        );

sint32 GetDistinctMessageAttributes(
        [in] string MessageAttribute,
        [in] string WhereClause,
        [in] sint32 Options,
        [out] string DistinctMessageAttributes[],
        [out] sint32 DistinctCount[]
        );
```

```
sint32 GetMessageEventCounters(
        [out] sint32 TimeOfEventProcStartOnServer,
        [out] sint32 CurrentTimeOnServer,
        [out] sint32 MaxTimeDiff,
        [out] sint32 MaxCountDiff,
        [out] sint32 DuplMsgMaxBufferDelay,
        [out] sint32 DuplMsgMaxBufferCount,
        [out] sint32 MaxOutOfSyncCount,
        [out] sint32 Mode,
        [out] OV_MessageEventCounter MessageEventCounterArray[]
        );
```

Instance Methods:

```
sint32 GetInstruction(
        [out] string Instruction
        );

sint32 ChangeText(
        [in] string NewText
        );

sint32 ChangeSeverity(
        [in] sint32 NewSeverity
        );

sint32 Acknowledge(
        );

sint32 Unacknowledge(
        );

sint32 Own(
        [in] string NewOwner
        );

sint32 Disown(
        );

sint32 GetAnnotation(
        [in] sint32 AnnotationNumber,
        [out] OV_MessageAnnotation Annotation
        );

sint32 GetAnnotationById(
        [in] string AnnoId,
        [out] OV_MessageAnnotation Annotation
        );
```

```
sint32 ModifyAnnotation(
        [in] sint32 AnnotationNumber,
        [in] string NewText
        );

sint32 ModifyAnnotationById(
        [in] string AnnoId,
        [in] string NewText
        );

sint32 DeleteAnnotation(
        [in] sint32 AnnotationNumber
        );

sint32 DeleteAnnotationById(
        [in] string AnnoId
        );

sint32 AddAnnotation(
        [in] string Text
        );

sint32 GetAnnotationArray(
        [in] sint32 StartAnnotationNumber,
        [in] sint32 NumberOfAnnotations,
        [in] sint32 Mode,
        [out] OV_MessageAnnotation AnnotationArray[]
        );

sint32 GetOriginalText(
        [out] string OriginalText
        );

sint32 GetStoredDuplicate(
        [in] sint32 StoredDuplicateNumber,
        [out] OV_MessageStoredDuplicate StoredDuplicate
        );

sint32 GetStoredDuplicateArray(
        [in] sint32 StartStoredDuplicateNumber,
        [in] sint32 NumberOfStoredDuplicates,
        [in] sint32 Mode,
        [out] OV_MessageStoredDuplicate StoredDuplicateArray[]
        );

sint32 SetCMAs(
        [in] OV_Message_CA cma[]
        );

sint32 GetServerList(
                [out] string serverName[]
                );
};
```

OV_Message-Properties

Id

Signature

string Id

Description

Key to identify the external message.

OriginalId

Signature

string OriginalId

Description

Contains the original message ID if the property "Id" was changed (for example, by the message stream interface).

ConditionId

Signature

string ConditionId

Description

Identifies the condition that matches the message.

NodeName

Signature

string NodeName

Description

UUID of the managed node on which the event occurred.

AgentId

Signature

string AgentId

Description

Contains the ID of the agent.

PrimaryNodeName

Signature

string PrimaryNodeName

Description

"Readable" name of the managed node on which the event occurred.

TimeCreated

Signature

datetime TimeCreated

Description

Time when the message was created on the agent.

Format: YYYYMMDDHHMMSS.mmmmmmsUUU

YYYY - Four-digit year.

MM - Two-digit month.

DD - Two-digit day (01-31).

HH - Two-digit hour (00-23).

MM - Two-digit minute (00-59).

SS - Two-digit number of seconds (00-59).

mmmmmm - Six-digit number of microseconds (000000-999999).

s - Plus sign (+) or minus sign (-) to indicate a positive or negative offset from universal time coordinates (UTC).

UUU - Three-digit offset indicating the number of minutes that the originating time zone deviates from UTC.

TimeReceived

Signature

datetime TimeReceived

Description

The time when the message was received on the management server. In case of duplicates, it is the time when the last duplicate arrived on the server (if duplicate detection is enabled).

Format: YYYYMMDDHHMMSS.mmmmmmsUUU

YYYY - Four-digit year.

MM - Two-digit month.

DD - Two-digit day (01-31).

HH - Two-digit hour (00-23).

MM - Two-digit minute (00-59).

SS - Two-digit number of seconds (00-59).

mmmmmm - Six-digit number of microseconds (000000-999999).

s - Plus sign (+) or minus sign (-) to indicate a positive or negative offset from universal time coordinates (UTC).

UUU - Three-digit offset indicating the number of minutes that the originating time zone deviates from UTC.

Text

Signature
: string Text

Description
: Message text.

OriginalText

Signature
: string OriginalText

Description
: Contains the original text received from the agent.

MessageGroup

Signature
: string MessageGroup

Description
: Logical grouping of messages by type.

Object

Signature

string Object

Description

Specifies the object that was affected by, that was detected, or that caused the message. For example, this could be a printer that sent a message when it stopped accepting requests, or a backup device that sent a message when a backup stopped.

Application

Signature

string Application

Description

Specifies the application that was affected by or detected the message.

Type

Signature

string Type

Description

Defines filters for the message stream interface.

ServiceId

Signature

string ServiceId

Description

Service that is affected by the message.

OriginalServiceId

Signature

string OriginalServiceId

Description

Contains the service ID without parameter replacement.

NumberOfAnnotations

Signature

sint32 NumberOfAnnotations = 0

Description

Number of annotations attached to this message.

UsedNotificationInterfaces

Signature

string UsedNotificationInterfaces

Description

Identifies the interfaces used for notifications (for example, email).

DoNotification

Signature

boolean DoNotification = FALSE

Description

If true, the system sends a notification for the message (for example, by using email or pager).

Unmatched

Signature

boolean Unmatched = TRUE

Description

If true, the message does not match any condition.

LogOnly

Signature

boolean LogOnly = FALSE

Description

If true, the system stores the message as a history message.

IsProxied

Signature

boolean IsProxied = FALSE

Description

If true, the agent node is different from the node on which the event occurred.

CreateTroubleTicketInterface

Signature

boolean CreateTroubleTicketInterface = FALSE

Description

If true, the system creates a trouble ticket for the message.

AcknowledgeAfterTroubleTicket

Signature

boolean AcknowledgeAfterTroubleTicket = FALSE

Description

If true, the system acknowledges the message after it creates the trouble ticket.

IsExternalNode

Signature

boolean IsExternalNode = FALSE

Description

If true, the message arrived from an external node. An external node is not managed by HPOM for Windows. (for example, a node without an HP Operations agent).

Severity

Signature

sint32 Severity = 1

Description

Indicates how urgent the message is.

Values

ValueMap	Values
1	= Unknown
2	= Normal
4	= Warning
8	= Minor
16	= Major
32	= Critical

State

Signature

sint32 State = 1

Description

The current state of the message.

Values

ValueMap	Values
1	= Undefined
2	= Unowned
3	= Owned
4	= Acknowledged
5	= Node Deleted
6	= Deleted

TimeOfStateChange

Signature

datetime TimeOfStateChange

Description

Time when the last state change occurred.

Format: YYYYMMDDHHMMSS.mmmmmmsUUU

YYYY - Four-digit year.

MM - Two-digit month.

DD - Two-digit day (01-31).

HH - Two-digit hour (00-23).

MM - Two-digit minute (00-59).

SS - Two-digit number of seconds (00-59).

mmmmmm - Six-digit number of microseconds (000000-999999).

s - Plus sign (+) or minus sign (-) to indicate a positive or negative offset from universal time coordinates (UTC).

UUU - Three-digit offset indicating the number of minutes that the originating time zone deviates from UTC.

UserOfStateChange

Signature

string UserOfStateChange

Description

User responsible for the last state change.

SourceType

Signature

sint32 SourceType = 0x0

Description

Message source type (for example, console, logfile, or SNMP, and so on).

Values

ValueMap	Values
0x0 =	Unknown
0x1 =	MPE Console
0x2 =	Open Message Interface
0x4 =	Logfile Entry | Windows Event Log
0x8 =	Measurement Threshold | Service/Process Monitoring
0x10 =	SNMP Interceptor
0x20 =	Server Message Stream Interface
0x40 =	Agent Message Stream Interface
0x80 =	Reserved
0x100 =	Scheduled Command
0x200 =	Measurement Threshold | Service/Process Monitoring
0x400 =	Windows Management Interface
0x1000 =	Internal
0x2000 =	Unknown
0x2001 =	MPE Console
0x2002 =	Open Message Interface
0x2004 =	Logfile Entry | Windows Event Log
0x2008 =	Measurement Threshold | Service/Process Monitoring
0x2010 =	SNMP Interceptor
0x2020 =	Server Message Stream Interface
0x2040 =	Agent Message Stream Interface
0x2080 =	Reserved
0x2100 =	Scheduled Command
0x2200 =	Measurement Threshold | Service/Process Monitoring
0x2400 =	Windows Management Interface

Source

Signature

string Source

Description

Name and version of the policy that created the message.

MessageKey

Signature

string MessageKey

Description

Identifies a certain type of message.

MessageKeyRelation

Signature

string MessageKeyRelation

Description

Messages with message keys that match the MessageKeyRelation are acknowledged by this message.

NumberOfDuplicates

Signature

sint32 NumberOfDuplicates = 0

Description

If duplicate detection is enabled, the number of duplicates of the current message. Otherwise, the value is 0.

TimeFirstReceived

Signature

datetime TimeFirstReceived

Description

If duplicate detection is enabled, the time when the message was first received on the management server. Otherwise, the value is equal to TimeReceived.

Format: YYYYMMDDHHMMSS.mmmmmmsUUU

YYYY - Four-digit year.

MM - Two-digit month.

DD - Two-digit day (01-31).

HH - Two-digit hour (00-23).

MM - Two-digit minute (00-59).

SS - Two-digit number of seconds (00-59).

mmmmmm - Six-digit number of microseconds (000000-999999).

s - Plus sign (+) or minus sign (-) to indicate a positive or negative offset from universal time coordinates (UTC).

UUU - Three-digit offset indicating the number of minutes that the originating time zone deviates from UTC.

NumberOfStoredDuplicates

Signature
sint32 NumberOfStoredDuplicates = 0

Description
Number of duplicate annotations stored with the message.

AutomaticAction

Signature
OV_MessageAction AutomaticAction

Description
Action that is automatically executed when the message arrived on the management server (see OV_MessageAction).

OperatorAction

Signature
OV_MessageAction OperatorAction

Description
Action that can be executed by an operator when the message arrived on the management server (see OV_MessageAction).

InstructionAvailable

Signature
boolean InstructionAvailable = FALSE

Description

If true, an instruction is attached to the message.

InstructionID

Signature

string InstructionID

Description

Identifies the instruction attached to the message.

InstructionType

Signature

sint32 InstructionType = 0

Description

Type of the instruction (for example, text).

Values

ValueMap	Values
0	= CSM_C_INSTR_NO
1	= CSM_C_INSTR_TEXT
2	= CSM_C_INSTR_INTERF
3	= CSM_C_INSTR_INTERNAL

InstructionParameters

Signature

string InstructionParameters

Description

Instruction parameters passed to the instruction text interface.

CMAs

Signature

OV_Message_CA CMAs[]

Description

List of custom message attributes (CMAs).

Origin

Signature

string Origin

Description

In a MOM environment, identifies the management server that received the message first.

Sender

Signature

string Sender

Description

In a MOM environment, identifies the management server from which the message was received.

OV_Message Class Methods

This section contains the information on Class methods for OV_Message.

OV_Message::AcknowledgeMessages()
OV_Message::AcknowledgeMessages_Trans()

```
sint32 AcknowledgeMessages(
        [in] string MessageIDs[],
        [out] OV_Message_MassOperationResult MassOperationResults[] )
```

Parameters

MessageIDs
    Message GUIDs that identify the messages to be acknowledged.

MassOperationResults
    Contains detailed result information for each message.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Acknowledges all messages identified by MessageIDs.

Return Value

Values

ValueMap	Values
0 =	Succeeded
-1 =	Errors Occurred

Extended Status Codes

None.

OV_Message::CountMessages()
OV_Message::CountMessages_Trans()

```
sint32 CountMessages(
        [in] string WhereClause,
        [out] sint32 Count)
```

Parameters

WhereClause
: Condition that identifies messages to be counted.

Count
: Number of messages that match the condition in WhereClause.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Counts messages that match the condition in WhereClause.

Return Value

Values

ValueMap	Values
0 =	Succeeded
-1 =	Errors Occurred

Extended Status Codes

None.

OV_Message::DisownMessages()
OV_Message::DisownMessages_Trans()

```
sint32 DisownMessages(
        [in] string MessageIDs[],
        [out] OV_Message_MassOperationResult MassOperationResults[] )
```

Parameters

MessageIDs
: Message GUIDs that identify the messages to be disowned.

MassOperationResults
: Contains detailed result information for each message.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Disowns all messages identified by MessageIDs.

Return Value

Values

ValueMap		Values
0	=	Succeeded
-1	=	Errors Occurred

Extended Status Codes

None.

OV_Message::GetDistinctMessageAttributes()
OV_Message::GetDistinctMessageAttributes_Trans()

```
sint32 GetDistinctMessageAttributes(
        [in] string MessageAttribute,
        [in] string WhereClause,
        [in] sint32 Options,
        [out] string DistinctMessageAttributes[],
        [out] sint32 DistinctCount[] )
```

Parameters

MessageAttribute
: Message attribute for which distinct values are sought.

WhereClause
: Condition to restrict the search space. Example: "... where severity = critical"

Options
: 0 = Returns only the DistinctMessageAttributes without counting messages.

  1 = Returns the DistinctMessageAttributes and counts the number of messages per distinct attribute value in DistinctCount.

DistinctMessageAttributes
: Distinct values of the message attribute.

DistinctCount
: For each distinct attribute value, contains the number of messages with this attribute value.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Gets the distinct values of a certain message attribute.

Return Value

Values

ValueMap		Values
0	=	Succeeded
-1	=	Errors Occurred

Extended Status Codes

None.

OV_Message::GetMessageEventCounters()
OV_Message::GetMessageEventCounters_Trans()

```
sint32 GetMessageEventCounters(
        [out] sint32 TimeOfEventProcStartOnServer,
        [out] sint32 CurrentTimeOnServer,
        [out] sint32 MaxTimeDiff,
        [out] sint32 MaxCountDiff,
        [out] sint32 DuplMsgMaxBufferDelay,
        [out] sint32 DuplMsgMaxBufferCount,
        [out] sint32 MaxOutOfSyncCount,
        [out] sint32 Mode,
        [out] OV_MessageEventCounter MessageEventCounterArray[] )
```

Parameters

TimeOfEventProcStartOnServer
: Time the event processing was started on the message server

CurrentTimeOnServer
: Current time on the message server

MaxTimeDiff
: Maximum time difference allowed. This value is set in the registry.

MaxCountDiff
: Maximum count difference allowed. This value is set in the registry.

DuplMsgMaxBufferDelay
: Maximum allowed buffer delay. This value is set in the registry.

DuplMsgMaxBufferCount
: Maximum allowed buffer count. This value is set in the registry.

MaxOutOfSyncCount
: Maximum allowed out of sync count. This value is set in the registry.

Mode

Mode. This value is set in the registry.

MessageEventCounterArray
Retrieved array of message event counter information

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Gets some information about the WMI events sent from the message server to the registered WMI clients

Return Value

Values

ValueMap		Values
0	=	Succeeded
-1	=	Errors Occurred

Extended Status Codes

None.

OV_Message::OwnMessages()
OV_Message::OwnMessages_Trans()

```
sint32 OwnMessages(
        [in] string MessageIDs[],
        [in] string NewOwner,
        [out] OV_Message_MassOperationResult MassOperationResults[] )
```

Parameters

MessageIDs
    Message GUIDs that identify the messages to be owned.

NewOwner
    New message owner.

MassOperationResults
    Contains detailed result information for each message.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Owns all messages identified by MessageIDs.

Return Value

Values

ValueMap	Values
0 =	Succeeded
-1 =	Errors Occurred

Extended Status Codes

None.

OV_Message::UnacknowledgeMessages()
OV_Message::UnacknowledgeMessages_Trans()

```
sint32 UnacknowledgeMessages(
        [in] string MessageIDs[],
        [out] OV_Message_MassOperationResult MassOperationResults[] )
```

Parameters

MessageIDs
: Message GUIDs that identify the messages to be unacknowledged.

MassOperationResults
: Contains detailed result information for each message.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Unacknowledges all messages identified by MessageIDs.

Return Value

Values

ValueMap	Values
0 =	Succeeded
-1 =	Errors Occurred

Extended Status Codes

None.

OV_Message Instance Methods

This section contains the information on Instance methods for OV_Message.

OV_Message::Acknowledge() OV_Message::Acknowledge_Trans()

sint32 Acknowledge()

Parameters

None.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Acknowledges the selected message.

Return Value

Values

ValueMap	Values
0 =	(MS900) Succeeded
-1 =	(MS908) Error
-10 =	(MS902) Incompatible Message State
-11 =	(MS903) User Not Owner of Message
-12 =	(MS904) Message is Acknowledged
-20 =	(MS905) Invalid Severity Value
-21 =	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::AddAnnotation()

OV_Message::AddAnnotation_Trans()

```
sint32 AddAnnotation(
        [in] string Text)
```

Parameters

Text
Text of the annotation to be added.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds an annotation to the selected message.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::ChangeSeverity()
OV_Message::ChangeSeverity_Trans()

```
sint32 ChangeSeverity(
        [in] sint32 NewSeverity)
```

Parameters

NewSeverity
: New message severity.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Changes the message severity.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::ChangeText()
OV_Message::ChangeText_Trans()

```
sint32 ChangeText(
        [in] string NewText)
```

Parameters

NewText
    New message text.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Changes the message text.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::DeleteAnnotation()
OV_Message::DeleteAnnotation_Trans()

```
sint32 DeleteAnnotation(
        [in] sint32 AnnotationNumber)
```

Parameters

AnnotationNumber
Number of the wanted annotation.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Deletes the annotation identified by the AnnotationNumber of the selected message.

Return Value

Values

ValueMap	Values
0 =	(MS900) Succeeded
-1 =	(MS908) Error
-10 =	(MS902) Incompatible Message State
-11 =	(MS903) User Not Owner of Message
-12 =	(MS904) Message is Acknowledged
-20 =	(MS905) Invalid Severity Value
-21 =	(MS906) Invalid Annotation Number
-22 =	(MS909) User Not Creator of Annotation

Extended Status Codes

None.

OV_Message::DeleteAnnotationById()
OV_Message::DeleteAnnotationById_Trans()

```
sint32 DeleteAnnotationById(
        [in] string AnnoId)
```

Parameters

AnnoId
    Annotation ID of the wanted annotation.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Deletes the annotation identified by the Annotation Id of the selected message.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number
-22	=	(MS909) User Not Creator of Annotation

Extended Status Codes

None.

OV_Message::Disown()
OV_Message::Disown_Trans()

sint32 Disown()

Parameters

None.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Disowns the selected message.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::GetAnnotation()

OV_Message::GetAnnotation_Trans()

```
sint32 GetAnnotation(
        [in] sint32 AnnotationNumber,
        [out] OV_MessageAnnotation Annotation)
```

Parameters

AnnotationNumber

Number of the wanted annotation.

Annotation

Annotation identified by AnnotationNumber.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Gets the annotation specified by the AnnotationNumber of the selected message.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::GetAnnotationArray()
OV_Message::GetAnnotationArray_Trans()

```
sint32 GetAnnotationArray(
        [in] sint32 StartAnnotationNumber,
        [in] sint32 NumberOfAnnotations,
        [in] sint32 Mode,
        [out] OV_MessageAnnotation AnnotationArray[] )
```

Parameters

StartAnnotationNumber
: Number of the first annotation to be retrieved.

NumberOfAnnotations
: Number of annotations to be retrieved.

Mode
: 0 = Gets all annotations with annotation text.

: 1 = Gets all annotations without annotation text.

AnnotationArray
: Retrieved array of annotations.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Gets the NumberOfAnnotations annotations of the selected message, starting with StartAnnotationNumber.

Return Value

Values

ValueMap	Values
0 =	Succeeded
-1 =	Errors Occurred

Extended Status Codes

None.

OV_Message::GetAnnotationById()
OV_Message::GetAnnotationById_Trans()

```
sint32 GetAnnotationById(
        [in] string AnnoId,
        [out] OV_MessageAnnotation Annotation)
```

Parameters

AnnoId
Annotation ID of the wanted annotation.

Annotation
Annotation identified by AnnotationNumber.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Gets the annotation specified by the Annotation ID of the selected message.

Return Value

Values

ValueMap	Values
0 =	(MS900) Succeeded
-1 =	(MS908) Error
-10 =	(MS902) Incompatible Message State
-11 =	(MS903) User Not Owner of Message
-12 =	(MS904) Message is Acknowledged
-20 =	(MS905) Invalid Severity Value
-21 =	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::GetInstruction()
OV_Message::GetInstruction_Trans()

```
sint32 GetInstruction(
        [out] string Instruction)
```

Parameters

Instruction
: Message instruction.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Gets the instruction of the selected message.

Return Value

Values

ValueMap	Values
0 =	(MS900) Succeeded
-1 =	(MS908) Error
-10 =	(MS902) Incompatible Message State
-11 =	(MS903) User Not Owner of Message
-12 =	(MS904) Message is Acknowledged
-20 =	(MS905) Invalid Severity Value
-21 =	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::GetOriginalText()
OV_Message::GetOriginalText_Trans()

```
sint32 GetOriginalText(
        [out] string OriginalText)
```

Parameters

OriginalText
    Original message text.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Gets the original text of the selected message.

Return Value

Values

ValueMap	Values
0 =	Succeeded
-1 =	Errors Occurred

Extended Status Codes

None.

OV_Message::GetServerList()
OV_Message::GetServerList_Trans()

```
sint32 GetServerList(
        [out] string serverName[] )
```

Parameters

serverName
    List of server names, from which the message has been forwarded.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Gets the list of server names from which the message has been forwarded.

Return Value

None.

Extended Status Codes

None.

OV_Message::GetStoredDuplicate()
OV_Message::GetStoredDuplicate_Trans()

```
sint32 GetStoredDuplicate(
        [in] sint32 StoredDuplicateNumber,
        [out] OV_MessageStoredDuplicate StoredDuplicate)
```

Parameters

StoredDuplicateNumber
Number of the wanted duplicate of the selected message.

StoredDuplicate
Duplicate of the selected message, specified by StoredDuplicateNumber.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Gets the duplicate of the selected message, specified by StoredDuplicateNumber.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::GetStoredDuplicateArray()
OV_Message::GetStoredDuplicateArray_Trans()

```
sint32 GetStoredDuplicateArray(
        [in] sint32 StartStoredDuplicateNumber,
        [in] sint32 NumberOfStoredDuplicates,
        [in] sint32 Mode,
        [out] OV_MessageStoredDuplicate StoredDuplicateArray[] )
```

Parameters

StartStoredDuplicateNumber
: Number of the first duplicate to be retrieved.

NumberOfStoredDuplicates
: Number of duplicates to be retrieved.

Mode
: 0 = Gets all duplicates with duplicate text.

: 1 = Gets all duplicates without duplicate text.

StoredDuplicateArray
: Retrieved array of message duplicates.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Gets the NumberOfStoredDuplicates duplicates of the selected message, starting with StartStoredDuplicateNumber.

Return Value

Values

ValueMap	Values
0 =	Succeeded
-1 =	Errors Occurred

Extended Status Codes

None.

OV_Message::ModifyAnnotation() OV_Message::ModifyAnnotation_Trans()

```
sint32 ModifyAnnotation(
        [in] sint32 AnnotationNumber,
        [in] string NewText)
```

Parameters

AnnotationNumber
    Number of the wanted annotation.

NewText
    New annotation text.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Modifies the text of the annotation specified by the AnnotationNumber of the selected message.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number
-22	=	(MS909) User Not Creator of Annotation

Extended Status Codes

None.

OV_Message::ModifyAnnotationById()
OV_Message::ModifyAnnotationById_Trans()

```
sint32 ModifyAnnotationById(
        [in] string AnnoId,
        [in] string NewText)
```

Parameters

AnnoId
  Annotation ID of the wanted annotation.

NewText
  New annotation text.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Modifies the text of the annotation specified by the Annotation Id of the selected message.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number
-22	=	(MS909) User Not Creator of Annotation

Extended Status Codes

None.

OV_Message::Own()

OV_Message::Own_Trans()

```
sint32 Own(
        [in] string NewOwner)
```

Parameters

NewOwner
New message owner.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Owns the selected message.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message::SetCMAs()
OV_Message::SetCMAs_Trans()

```
sint32 SetCMAs(
        [in] OV_Message_CA cma[] )
```

Parameters

cma
: cma-array to be attached to the message.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Attach custom message attribute (CMA) array to the selected message.

Return Value

None.

Extended Status Codes

None.

OV_Message::Unacknowledge() OV_Message::Unacknowledge_Trans()

sint32 Unacknowledge()

Parameters

None.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Unacknowledges the selected message.

Return Value

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS908) Error
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number

Extended Status Codes

None.

OV_Message_AnnotationTextChangeEvent

Description

The OV_Message_AnnotationTextChangeEvent indicates the annotationtext of the message changed.

```
class OV_Message_AnnotationTextChangeEvent
{
        Properties:
        string OldText;
        string NewText;
};
```

OV_Message_AnnotationTextChangeEvent-Properties

OldText

Signature

string OldText

Description

Old text of message annotation.

NewText

Signature

string NewText

Description

New text of message annotation.

OV_Message_AttributeChangeEvent

Description

The OV_Message_AttributeChangeEvent indicates that the custom message attributes (CMAs) of the message changed.

```
class OV_Message_AttributeChangeEvent
{
    Properties:
    sint32 EventType;
    OV_Message_CA oldCMAs[];
    OV_Message_CA newCMAs[];
};
```

OV_Message_AttributeChangeEvent-Properties

EventType

Signature

sint32 EventType

Description

Type of the event (for example, CMA).

Values

ValueMap	Values
0	= Undefined
1	= CMA

oldCMAs

Signature

OV_Message_CA oldCMAs[]

Description

Optional list of old custom message attributes.

newCMAs

Signature

OV_Message_CA newCMAs[]

Description

List of new custom message attributes.

OV_Message_CA

Description

A custom message attribute (CMA) is a name-value pair. The set of standard message attributes (for example, text, severity, and so on) can be enhanced by CMAs.

```
class OV_Message_CA
{
        Properties:
        string name;
        string value;
};
```

OV_Message_CA-Properties

name

Signature

string name

Description

Name of the CMA.

value

Signature

string value

Description

Value of the CMA.

OV_Message_ChangeEvent

Description

The OV_Message_ChangeEvent is the base message change event class. It contains the message Id that specifies the message affected by a change event.

```
class OV_Message_ChangeEvent
{
        Properties:
        string MessageId;
};
```

OV_Message_ChangeEvent-Properties

MessageId

Signature

string MessageId

Description

Id specifying the message affected by a change event.

OV_Message_CounterChangeEvent

Description

The OV_Message_CounterChangeEvent indicates that the number of duplicates of the message changed.

```
class OV_Message_CounterChangeEvent
{
        Properties:
        sint32 NumberOfDuplicates;
        datetime TimeReceived;
        sint32 SourceType;
        string Source;
        string ConditionId;
        sint32 BufferCount;
};
```

OV_Message_CounterChangeEvent-Properties

NumberOfDuplicates

Signature

sint32 NumberOfDuplicates

Description

Number of duplicates of the message.

TimeReceived

Signature

datetime TimeReceived

Description

Time when the duplicate arrived at the management server.

SourceType

Signature

sint32 SourceType

Description

Message source type (for example, console, logfile, or SNMP, and so on).

Source

Signature

string Source

Description

Name and version of the policy that created the message.

ConditionId

Signature

string ConditionId

Description

Identifies the condition that matches the message.

BufferCount

Signature

    sint32 BufferCount

Description

    Buffer Count

OV_Message_MassOperationResult

Description

The result of a message mass operation describes success or failure in detail.

```
class OV_Message_MassOperationResult
{
        Properties:
        string MessageId;
        sint32 Result;
        string AdditionalErrorInformation;
};
```

OV_Message_MassOperationResult-Properties

MessageId

Signature

string MessageId

Description

Message GUID that identifies the message.

Result

Signature

sint32 Result

Description

Result of the message operation.

Values

ValueMap		Values
0	=	(MS900) Succeeded
-1	=	(MS901) General Failure
-10	=	(MS902) Incompatible Message State
-11	=	(MS903) User Not Owner of Message
-12	=	(MS904) Message is Acknowledged
-20	=	(MS905) Invalid Severity Value
-21	=	(MS906) Invalid Annotation Number
-30	=	(MS907) Message Not Found

AdditionalErrorInformation

Signature

string AdditionalErrorInformation

Description

Additional error information that is not covered by the result.

OV_Message_NumberOfAnnotationsChangeEvent

Description

The OV_Message_NumberOfAnnotationsChangeEvent indicates that the number of annotations of the message changed.

```
class OV_Message_NumberOfAnnotationsChangeEvent
{
	Properties:
	sint32 OldNumber;
	sint32 NewNumber;
};
```

OV_Message_NumberOfAnnotationsChangeEvent-Properties

OldNumber

Signature

    sint32 OldNumber

Description

    Old number of message annotations.

NewNumber

Signature

    sint32 NewNumber

Description

    New number of message annotations.

OV_Message_SeverityChangeEvent

Description

The OV_Message_SeverityChangeEvent indicates that the severity of the message changed.

```
class OV_Message_SeverityChangeEvent
{
        Properties:
        sint32 OldSeverity;
        sint32 NewSeverity;
        datetime TimeOfSeverityChange;
        string ServiceId;
        string NodeName;
        sint32 State;
        string MessageGroup;
};
```

OV_Message_SeverityChangeEvent-Properties

OldSeverity

Signature

sint32 OldSeverity

Description

Old severity of the message.

Values

ValueMap	Values
1	= Unknown
2	= Normal
4	= Warning
8	= Minor
16	= Major
32	= Critical

NewSeverity

Signature

sint32 NewSeverity

Description

New severity of the message.

Values

ValueMap	Values
1	= Unknown
2	= Normal
4	= Warning
8	= Minor
16	= Major
32	= Critical

TimeOfSeverityChange

Signature

datetime TimeOfSeverityChange

Description

Time when the severity of the message changed.

ServiceId

Signature

string ServiceId

Description

Id of the service to which the message belongs.

NodeName

Signature

string NodeName

Description

Name of the node from which the message comes.

State

Signature

sint32 State

Description

State of the message.

Values

ValueMap	Values
1	= Undefined
2	= Unowned
3	= Owned
4	= Acknowledged

MessageGroup

Signature

string MessageGroup

Description

Name of the group the message belongs to.

OV_Message_StateChangeEvent

Description

The OV_Message_StateChangeEvent indicates that the state of the message changed.

```
class OV_Message_StateChangeEvent
{
        Properties:
        sint32 OldState;
        sint32 NewState;
        string UserOfStateChange;
        datetime TimeOfStateChange;
        sint32 Severity;
        string ServiceId;
        string NodeName;
        string MessageGroup;
};
```

OV_Message_StateChangeEvent-Properties

OldState

Signature

sint32 OldState

Description

Old state of the message.

Values

ValueMap	Values
1	Undefined
2	Unowned
3	Owned
4	Acknowledged

NewState

Signature

sint32 NewState

Description

New state of the message.

Values

ValueMap	Values
1	Undefined
2	Unowned
3	Owned
4	Acknowledged

UserOfStateChange

Signature

string UserOfStateChange

Description

User who changed the state of the message.

TimeOfStateChange

Signature

    datetime TimeOfStateChange

Description

    Time when the state of the message changed.

Severity

Signature

    sint32 Severity

Description

    Severity of the message.

ServiceId

Signature

    string ServiceId

Description

    Id of the service to which the message belongs.

NodeName

Signature

    string NodeName

Description

    Name of the node from which the message comes.

MessageGroup

Signature

    string MessageGroup

Description

    Name of the group to which the message belongs.

OV_Message_TextChangeEvent

Description

The OV_Message_TextChangeEvent indicates that the text of the message changed.

```
class OV_Message_TextChangeEvent
{
	Properties:
	string NewText;
};
```

OV_Message_TextChangeEvent-Properties

NewText

Signature

string NewText

Description

New text of the message.

OV_MessageAction

Description

An action attached to the message. The system can trigger an action execution when the server receives the message (automatic action) or when an operator manually starts the action execution (operator action).

```
class OV_MessageAction
{
        Properties:
        string MessageId;
        sint32 Type;
        string NodeName;
        string Call;
        sint32 State;
        boolean CreateAnnotation = FALSE;
        boolean DoAcknowledge = FALSE;

        Instance Methods:

        sint32 Execute(
                );

        sint32 Stop(
                );
};
```

OV_MessageAction-Properties

MessageId

Signature

string MessageId

Description

Message GUID that identifies the message.

Type

Signature

sint32 Type

Description

The action type.

Values

ValueMap	Values
1 =	operator initiated action
2 =	automatic action

NodeName

Signature

string NodeName

Description

Node on which the action is executed.

Call

Signature

string Call

Description

Action command to be executed.

State

Signature

sint32 State

Description

State of the action.

Values

ValueMap	Values
0	Succeeded
1	Failed
2	Started
3	Not Started
4	Discarded
5	Discarded (Security)

CreateAnnotation

Signature

boolean CreateAnnotation = FALSE

Description

If true, it appends the command output as an annotation to the message.

DoAcknowledge

Signature

boolean DoAcknowledge = FALSE

Description

If true, it acknowledges the message when the command is successful.

OV_MessageAction::Execute()
OV_MessageAction::Execute_Trans()

sint32 Execute()

Parameters

None.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Starts the action.

Return Value

Values

ValueMap		Values
0	=	Succeeded
-1	=	Error
-11	=	(MS903) User Not Owner of Message

Extended Status Codes

None.

OV_MessageAction::Stop()
OV_MessageAction::Stop_Trans()

sint32 Stop()

Parameters

None.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stops the action.

Return Value

Values

ValueMap		Values
0	=	Succeeded
-1	=	Error

Extended Status Codes

None.

OV_MessageAction_StateChangeEvent

Description

The OV_MessageAction_StateChangeEvent indicates that the state of the message action changed.

```
class OV_MessageAction_StateChangeEvent
{
        Properties:
        sint32 Type;
        sint32 NewState;
};
```

OV_MessageAction_StateChangeEvent-Properties

Type

Signature

sint32 Type

Description

Type of the message action (operator initiated, automatic, or scheduled action).

Values

ValueMap	Values
1	= Operator Action
2	= Automatic Action
3	= Schedule Action

NewState

Signature

sint32 NewState

Description

New state of the message action.

Values

ValueMap	Values
0	= Succeeded
1	= Failed
2	= Started
3	= Not Started
4	= Discarded

OV_MessageAnnotation

Description

The message contains an additional note. This note can be a command output (succeeded or failed) or operator information about how the message has been handled.

```
class OV_MessageAnnotation
{
        Properties:
        string MessageId;
        sint32 OrderNumber;
        string AnnoId;
        string Text;
        string User;
        datetime TimeCreated;
};
```

OV_MessageAnnotation-Properties

MessageId

Signature

string MessageId

Description

Message GUID that identifies the message.

OrderNumber

Signature

sint32 OrderNumber

Description

A message can have several annotations. The OrderNumber identifies an annotation within the message.

AnnoId

Signature

string AnnoId

Description

Annotation GUID that identifies the annotation.

Text

Signature

string Text

Description

Annotation text.

User

Signature

string User

Description

User who created the annotation.

TimeCreated

Signature

datetime TimeCreated

Description

Time when the annotation was created.

Format: YYYYMMDDHHMMSS.mmmmmmsUUU

YYYY - Four-digit year.

MM - Two-digit month.

DD - Two-digit day (01-31).

HH - Two-digit hour (00-23).

MM - Two-digit minute (00-59).

SS - Two-digit number of seconds (00-59).

mmmmmm - Six-digit number of microseconds (000000-999999).

s - Plus sign (+) or minus sign (-) to indicate a positive or negative offset from universal time coordinates (UTC).

UUU - Three-digit offset indicating the number of minutes that the originating time zone deviates from UTC.

OV_MessageEventCounter

Description

The OV_MessageEventCounter is used to get some information about the last WMI event sent from the message action server to the registered clients.

```
class OV_MessageEventCounter
{
        Properties:
        string EventName;
        sint32 EventType;
        uint32 CounterLowVal;
        sint32 CounterHighVal;
        sint32 TimeLastEventTriggered;
        sint32 Flags;
};
```

OV_MessageEventCounter-Properties

EventName

Signature

    string EventName

Description

    Name of the WMI event

EventType

Signature

    sint32 EventType

Description

    Type of the WMI event

CounterLowVal

Signature

    uint32 CounterLowVal

Description

    Low part of the 64 bit event counter

CounterHighVal

Signature

    sint32 CounterHighVal

Description

    High part of the 64 bit event counter

TimeLastEventTriggered

Signature

    sint32 TimeLastEventTriggered

Description

    Time the last event was sent

Flags

Signature

sint32 Flags

Description

Additional flags for the event

OV_MessageStoredDuplicate

Description

There are duplicate messages.

```
class OV_MessageStoredDuplicate
{
        Properties:
        string MessageId;
        sint32 OrderNumber;
        string Text;
        string User;
        datetime TimeCreated;
};
```

OV_MessageStoredDuplicate-Properties

MessageId

Signature

string MessageId

Description

Message GUID that identifies the message.

OrderNumber

Signature

sint32 OrderNumber

Description

A message can have several duplicates. The OrderNumber identifies a duplicate within the message.

Text

Signature

string Text

Description

Duplicate text.

User

Signature

string User

Description

User who created the duplicate.

TimeCreated

Signature

datetime TimeCreated

Description

Time when the stored duplicate was created.

Format: YYYYMMDDHHMMSS.mmmmmmsUUU

YYYY - Four-digit year.

MM - Two-digit month.

DD - Two-digit day (01-31).

HH - Two-digit hour (00-23).

MM - Two-digit minute (00-59).

SS - Two-digit number of seconds (00-59).

mmmmmm - Six-digit number of microseconds (000000-999999).

s - Plus sign (+) or minus sign (-) to indicate a positive or negative offset from universal time coordinates (UTC).

UUU - Three-digit offset indicating the number of minutes that the originating time zone deviates from UTC.

OV_NodeGroup

Description

A class used as a grouping mechanism for ManagedNodes.

```
class OV_NodeGroup
{
        Properties:
        string Name;
        boolean CannotModifyContents;
        string ReportGroup;
        string GraphFamily;
        string GraphCategory;

        Class Methods:

        OV_NodeGroup Create(
                [in] string Caption,
                [in] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description
                );

        OV_NodeGroup Create_Trans(
                [in] string TransId,
                [in] string Caption,
                [in] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description
                );

        void Remove(
                [in] string Name
                );

        void Remove_Trans(
                [in] string TransId,
                [in] string Name
                );

        OV_NodeGroup GetRoot(
                );

        OV_NodeGroup GetRoot_Trans(
                [in] string TransId
                );

        OV_NodeGroup GetByName(
                [in] string Name
                );

        OV_NodeGroup GetByName_Trans(
                [in] string TransId,
                [in] string Name
                );

        OV_NodeGroup GetByHierarchicalPath(
                [in] string Path
                );

        OV_NodeGroup GetByHierarchicalPath_Trans(
                [in] string TransId,
                [in] string Path
                );

        sint32 GetNodeSet(
                [out] OV_ManagedNode Nodes[],
                [in, optional] string PrimaryNodeNames[],
                [in, optional] string NodeNames[],
                [in, optional] string NodeGroupPaths[],
                [in, optional] string NodeGroupNames[]
                );

        sint32 GetNodeSet_Trans(
                [in] string TransId,
                [out] OV_ManagedNode Nodes[],
                [in, optional] string PrimaryNodeNames[],
                [in, optional] string NodeNames[],
                [in, optional] string NodeGroupPaths[],
                [in, optional] string NodeGroupNames[]
                );

        sint32 GetExternalNodeSet(
                [out] OV_ExternalNode Nodes[],
                [in, optional] string NodeNames[],
                [in, optional] string NodeGroupPaths[],
                [in, optional] string NodeGroupNames[]
                );

        sint32 GetExternalNodeSet_Trans(
                [in] string TransId,
                [out] OV_ExternalNode Nodes[],
                [in, optional] string NodeNames[],
                [in, optional] string NodeGroupPaths[],
                [in, optional] string NodeGroupNames[]
                );
```

Instance Methods:

```
void Modify(
        [in] OV_NodeGroup NodeGroup
        );

void Modify_Trans(
        [in] string TransId,
        [in] OV_NodeGroup NodeGroup
        );

sint32 GetParents(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

boolean IsChildOf(
        [in] string ParentName
        );

boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName
        );

boolean AddNode(
        [in] string NodeName
        );

boolean AddNode_Trans(
        [in] string TransId,
        [in] string NodeName
        );

void RemoveNode(
        [in] string NodeName
        );

void RemoveNode_Trans(
        [in] string TransId,
        [in] string NodeName
        );

sint32 GetNodes(
        [out] OV_ManagedNode Nodes[],
        [in, optional] boolean IncludeSubGroups
        );

sint32 GetNodes_Trans(
        [in] string TransId,
        [out] OV_ManagedNode Nodes[],
        [in, optional] boolean IncludeSubGroups
        );

boolean HasChildNode(
        [in] string NodeName,
        [in, optional] boolean IncludeSubGroups
        );

boolean HasChildNode_Trans(
        [in] string TransId,
        [in] string NodeName,
        [in, optional] boolean IncludeSubGroups
        );

boolean AddNodeGroup(
        [in] string NodeGroupName
        );

boolean AddNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName
        );

void RemoveNodeGroup(
        [in] string NodeGroupName
        );

void RemoveNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName
        );

sint32 GetChildNodeGroups(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeSubGroups
        );

sint32 GetChildNodeGroups_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeSubGroups
        );

boolean HasChildNodeGroup(
        [in] string NodeGroupName,
        [in, optional] boolean IncludeSubGroups
        );

boolean HasChildNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName,
        [in, optional] boolean IncludeSubGroups
        );

boolean AddAction(
        [in] string ActionName
        );

boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

void RemoveAction(
        [in] string ActionName
        );

void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

sint32 GetActions(
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeInherited
        );

sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeInherited
        );

boolean HasAction(
        [in] string ActionName,
        [in, optional] boolean IncludeInherited
        );

boolean HasAction_Trans(
        [in] string TransId,
        [in] string ActionName,
        [in, optional] boolean IncludeInherited
        );

boolean AddAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName
        );

boolean AddAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName
        );

void RemoveAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName
        );

void RemoveAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName
        );

sint32 GetAutoDeployPolicyGroups(
        [out] OV_AutoDeployPolicyGroup AutoDeployPolicyGroups[]
        );

sint32 GetAutoDeployPolicyGroups_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPolicyGroup AutoDeployPolicyGroups[]
        );

boolean HasAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName
        );

boolean HasAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName
        );

boolean AddReport(
        [in] string ReportName
        );

boolean AddReport_Trans(
        [in] string TransId,
        [in] string ReportName
        );
```

```
void RemoveReport(
        [in] string ReportName
        );

void RemoveReport_Trans(
        [in] string TransId,
        [in] string ReportName
        );

sint32 GetReports(
        [out] string Reports[]
        );

sint32 GetReports_Trans(
        [in] string TransId,
        [out] string Reports[]
        );

boolean HasReport(
        [in] string ReportName
        );

boolean HasReport_Trans(
        [in] string TransId,
        [in] string ReportName
        );

sint32 GetServices(
        [out] OV_Service Services[]
        );

sint32 GetServices_Trans(
        [in] string TransId,
        [out] OV_Service Services[]
        );

void SetGraph(
        [in, optional] string GraphFamily,
        [in] string GraphCategory
        );

void SetGraph_Trans(
        [in] string TransId,
        [in, optional] string GraphFamily,
        [in] string GraphCategory
        );

void DeleteGraph(
        );
```

```
void DeleteGraph_Trans(
        [in] string TransId
        );

void SetOutage(
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage,
        [in, optional] sint32 DisableHeartBeatPolingInOutage,
        [in, optional] sint32 RemovePolicyInMaintenance,
        [in, optional] boolean IncludeAllSubordinate
        );

void SetOutage_Trans(
        [in] string TransId,
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage,
        [in, optional] sint32 DisableHeartBeatPolingInOutage,
        [in, optional] sint32 RemovePolicyInMaintenance,
        [in, optional] boolean IncludeAllSubordinate
        );

boolean AddExternalNode(
        [in] string NodeName
        );

boolean AddExternalNode_Trans(
        [in] string TransId,
        [in] string NodeName
        );

void RemoveExternalNode(
        [in] string NodeName
        );

void RemoveExternalNode_Trans(
        [in] string TransId,
        [in] string NodeName
        );

sint32 GetExternalNodes(
        [out] OV_ExternalNode Nodes[],
        [in, optional] boolean IncludeSubGroups
        );

sint32 GetExternalNodes_Trans(
        [in] string TransId,
        [out] OV_ExternalNode Nodes[],
                [in, optional] boolean IncludeSubGroups
                );

        boolean HasChildExternalNode(
                [in] string NodeName,
                [in, optional] boolean IncludeSubGroups
                );

        boolean HasChildExternalNode_Trans(
                [in] string TransId,
                [in] string NodeName,
                [in, optional] boolean IncludeSubGroups
                );

        boolean AddAutoDeployPackage(
                [in] string AutoDeployPackageName
                );

        boolean AddAutoDeployPackage_Trans(
                [in] string TransId,
                [in] string AutoDeployPackageName
                );

        void RemoveAutoDeployPackage(
                [in] string AutoDeployPackageName
                );

        void RemoveAutoDeployPackage_Trans(
                [in] string TransId,
                [in] string AutoDeployPackageName
                );

        sint32 GetAutoDeployPackages(
                [out] OV_AutoDeployPackages AutoDeployPackages[]
                );

        sint32 GetAutoDeployPackages_Trans(
                [in] string TransId,
                [out] OV_AutoDeployPackages AutoDeployPackages[]
                );

        boolean HasAutoDeployPackage(
                [in] string AutoDeployPackageName
                );

        boolean HasAutoDeployPackage_Trans(
                [in] string TransId,
                [in] string AutoDeployPackageName
                );
};
```

OV_NodeGroup-Properties

Name

Signature

string Name

Description

The inherited Name serves as a key of a System instance in an enterprise environment.

CannotModifyContents

Signature

boolean CannotModifyContents

Description

If true, it disallows nodes from being added to or deleted from this node group manually. Also, it disallows the node group itself from being deleted manually.

ReportGroup

Signature

string ReportGroup

Description

The Report group to display when this service instance is the context for displaying the list of available reports. If this property is blank, no report displays. This property is updated by Reporter, based on its internal name for this node group.

GraphFamily

Signature

string GraphFamily

Description

The Graph family to display when this node group instance is the context for displaying the list of available graphs. If this property is blank, no graphs are displayed.

GraphCategory

Signature

string GraphCategory

Description

The Graph Category within the Graph Family to display when this node group instance is the context for displaying the list of available graphs. If this property is blank, the graphs defined in the Graph family are displayed.

OV_NodeGroup Class Methods

This section contains the information on Class methods for OV_NodeGroup.

OV_NodeGroup::Create()
OV_NodeGroup::Create_Trans()

```
OV_NodeGroup Create(
        [in] string Caption,
        [in] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description)
```

```
OV_NodeGroup Create_Trans(
        [in] string TransId,
        [in] string Caption,
        [in] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Caption

The Caption property of the instance of OV_NodeGroup to be created. If a node group with the same Caption already exists on the same hierarchy path level (as a child of the same parent), nothing happens, and the method fails with MDLAPI_E_NODEGROUP_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName

The Name property of the instance of OV_NodeGroup to which the newly created node group is added. If the specified node group does not exist, the method fails with MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST.

Name

The Name property of the instance of OV_NodeGroup to be created. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is created as a new GUID. If a Node Group with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_NODEGROUP_NAME_EXISTS.

Description
: The Description property of the instance of OV_NodeGroup to be created. Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a node group (new instance of OV_NodeGroup).

Return Value

Instance of the newly created OV_NodeGroup.

Extended Status Codes

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_PARENT_NODEGROUP_NOT_EXIST
: Parent node group does not exist.

MDLAPI_E_NODEGROUP_NAME_EXISTS
: Node Group with the same Name already exists.

MDLAPI_E_NODEGROUP_HIERPATH_EXISTS
: Node Group with the same hierarchy path (Caption) already exists.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::GetByHierarchicalPath()
OV_NodeGroup::GetByHierarchicalPath_Trans()

```
OV_NodeGroup GetByHierarchicalPath(
        [in] string Path)
```

```
OV_NodeGroup GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Path
    The node group specified by the hierarchy Path, which consists of the Caption/Display Names of the parent Node Groups and a single backslash as a separator. It also starts with a single backslash. If it contains only a single backslash (\), it denotes the Root. If a Caption contains a backslash, you should escape it with an additional backslash (for example, "\\").

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the node group (instance of OV_NodeGroup) specified by the hierarchical path.

Return Value

Instance of OV_NodeGroup specified by the hierarchical path.

Extended Status Codes

MDLAPI_E_NODEGROUP_NOT_EXIST
    Node Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_NodeGroup::GetByName()

OV_NodeGroup::GetByName_Trans()

```
OV_NodeGroup GetByName(
        [in] string Name)
```

```
OV_NodeGroup GetByName_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Name
    The Name property of the instance of OV_NodeGroup to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the node group (instance of OV_NodeGroup) specified with the Name property.

Return Value

Instance of OV_NodeGroup specified with the Name property.

Extended Status Codes

MDLAPI_E_NODEGROUP_NOT_EXIST
    Node Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_NodeGroup::GetNodeSet()
OV_NodeGroup::GetNodeSet_Trans()

```
sint32 GetNodeSet(
        [out] OV_ManagedNode Nodes[],
        [in, optional] string PrimaryNodeNames[],
        [in, optional] string NodeNames[],
        [in, optional] string NodeGroupPaths[],
        [in, optional] string NodeGroupNames[] )
```

```
sint32 GetNodeSet_Trans(
        [in] string TransId,
        [out] OV_ManagedNode Nodes[],
        [in, optional] string PrimaryNodeNames[],
        [in, optional] string NodeNames[],
        [in, optional] string NodeGroupPaths[],
        [in, optional] string NodeGroupNames[] )
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Nodes

Set of nodes (instances of OV_ManagedNode) without duplicates.

PrimaryNodeNames

List of primary node names (PrimaryNodeName property of the OV_ManagedNode instance). Default is an empty list. If one of the nodes from this list does not exist, nothing happens, and the method fails with MDLAPI_E_NODE_NOT_EXIST.

NodeNames

List of node names (Name property of the OV_ManagedNode instance). Default is an empty list. If one of the nodes from this list does not exist, nothing happens, and the method fails with MDLAPI_E_NODE_NOT_EXIST.

NodeGroupPaths

List of node groups specified with a hierarchical path. Default is an empty list. If one of the node groups from this list does not exist, nothing happens, and the method fails with

MDLAPI_E_NODEGROUP_NOT_EXIST.

NodeGroupNames
: List of node group names (Name property of OV_NodeGroup instance). Default is an empty list. If one of the node groups from this list does not exist, nothing happens, and the method fails with MDLAPI_E_NODEGROUP_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns a set of nodes (instances of OV_ManagedNode).

This method accepts different arrays (lists of nodes or node groups represented with IDs, primary node names, or hierarchical paths), and arranges them into a node set (node list without duplicate instances).

Return Value

Number of nodes in out parameter Nodes.

Extended Status Codes

MDLAPI_E_NODE_NOT_EXIST
: Node does not exist.

MDLAPI_E_NODE_PNNAME_NOT_UNIQUE
: More than one node with the specified Primary Node Name is found.

MDLAPI_E_NODEGROUP_NOT_EXIST
: Node Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::GetRoot()
OV_NodeGroup::GetRoot_Trans()

```
OV_NodeGroup GetRoot()
```

```
OV_NodeGroup GetRoot_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the root node group (instance of OV_NodeGroup).

Return Value

Root instance of OV_NodeGroup.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::Remove()
OV_NodeGroup::Remove_Trans()

```
void Remove(
        [in] string Name)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Name
    The Name property of the instance of OV_NodeGroup to be removed.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified node group (instance of OV_NodeGroup).

CAUTION:
This method also removes all child node groups that are not contained in any other node group (except special node groups). If nodes that are contained in this node group are not contained in any other node group (except special node groups), they are also removed. (To find out what else is removed, see the description of the Remove class method of OV_ManagedNode.) If the management server is also removed, the method fails with MDLAPI_E_NODE_IS_MANAGEMENT_SERVER. In addition, associated reports and associations to Auto-Deploy policies and tools are removed. (Only associations, not objects, are removed.)

NOTE:
You cannot remove root or special node groups. Attempts fail with MDLAPI_E_NODEGROUP_IS_ROOT /

MDLAPI_E_NODEGROUP_IS_SPECIAL_GROUP.

Return Value

None.

Extended Status Codes

MDLAPI_E_NODE_IS_MANAGEMENT_SERVER
: Management server cannot be removed.

MDLAPI_E_NODEGROUP_NOT_EXIST
: Node Group does not exist.

MDLAPI_E_NODEGROUP_IS_ROOT
: Root Node Group cannot be removed from or added to another node group.

MDLAPI_E_NODEGROUP_IS_SPECIAL_GROUP
: Special Node Group cannot be removed from or added to another node group.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup Instance Methods

This section contains the information on Instance methods for OV_NodeGroup.

OV_NodeGroup::AddAction()
OV_NodeGroup::AddAction_Trans()

```
boolean AddAction(
        [in] string ActionName)
```

```
boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ActionName
    The Name property of the instance of OV_Action to be added to this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds a tool (instance of OV_Action) to this node group.

Return Value

False if the tool has already been added to this node group.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
    Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_NodeGroup::AddAutoDeployPackage()
OV_NodeGroup::AddAutoDeployPackage_Trans()

```
boolean AddAutoDeployPackage(
        [in] string AutoDeployPackageName)
```

```
boolean AddAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

AutoDeployPackageName
    The Name property of the instance of OV_AutoDeployPackage to be added to this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds an auto-deploy package (instance of OV_AutoDeployPackage) to this node group.

Return Value

False if the auto-deploy package has already been added to this node group.

Extended Status Codes

MDLAPI_E_ADPACKAGE_NOT_EXIST
    Auto Deploy Package does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_NodeGroup::AddAutoDeployPolicyGroup() OV_NodeGroup::AddAutoDeployPolicyGroup_Trans()

```
boolean AddAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName)
```

```
boolean AddAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPolicyGroupName
: The Name property of the instance of OV_AutoDeployPolicyGroup to be added to this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds an auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) to this node group.

Before you can add a policy group as an auto-deployment policy group on a node group, you must create an instance of OV_AutoDeployPolicyGroup by calling the OV_AutoDeployPolicyGroup::Create method, where the path (DisplayName in the HPOM for Windows console) of the policy group is specified as the PolicyGroupPath parameter. Then the Name property of the returned instance of OV_AutoDeployPolicyGroup is used as the AutoDeployPolicyGroupName parameter of this method.

Return Value

False if the auto-deploy policy group has already been added to this node group.

Extended Status Codes

MDLAPI_E_ADPOLGROUP_NOT_EXIST
: Auto Deploy Policy Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::AddExternalNode()
OV_NodeGroup::AddExternalNode_Trans()

```
boolean AddExternalNode(
        [in] string NodeName)
```

```
boolean AddExternalNode_Trans(
        [in] string TransId,
        [in] string NodeName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

NodeName
    The Name property of the instance of OV_ExternalNode to be added to this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds an external node (instance of OV_ExternalNode) to this node group as a child node.

You cannot add an external node to a special node group (the method fails with MDLAPI_E_NODEGROUP_SPECIAL_ADD).

Return Value

False if the external node is already a child of this node group.

Extended Status Codes

MDLAPI_E_EXT_NODE_NOT_EXIST
    External node does not exist.

MDLAPI_E_EXT_NODE_HIERPATH_EXISTS
    External node with the same hierarchy path (Caption) already exists.

MDLAPI_E_NODEGROUP_SPECIAL_ADD

Child cannot be added to a special node group.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::AddNode()
OV_NodeGroup::AddNode_Trans()

```
boolean AddNode(
        [in] string NodeName)
```

```
boolean AddNode_Trans(
        [in] string TransId,
        [in] string NodeName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeName
: The Name property of the instance of OV_ManagedNode to be added to this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds a node (instance of OV_ManagedNode) to this node group as a child node.

A node cannot be added to a special node group (the method fails with MDLAPI_E_NODEGROUP_SPECIAL_ADD).

Return Value

False if the node is already a child of this node group.

Extended Status Codes

MDLAPI_E_NODE_NOT_EXIST
: Node does not exist.

MDLAPI_E_NODE_HIERPATH_EXISTS
: Node with the same hierarchy path (Caption) already exists.

MDLAPI_E_NODEGROUP_SPECIAL_ADD

Child cannot be added to a special node group.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_NodeGroup::AddNodeGroup()
OV_NodeGroup::AddNodeGroup_Trans()

```
boolean AddNodeGroup(
        [in] string NodeGroupName)
```

```
boolean AddNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

NodeGroupName
    The Name property of the instance of OV_NodeGroup to be added to this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds a node group (instance of OV_NodeGroup) to this node group as a child node group.

You cannot add node groups to a special node group (the method fails with MDLAPI_E_NODEGROUP_SPECIAL_ADD). You cannot add root or special node groups to this node (the method fails with MDLAPI_E_NODEGROUP_IS_ROOT / MDLAPI_E_NODEGROUP_IS_SPECIAL_GROUP).

If a node group with the name NodeGroupName is the same as this node group, or if it is one of its parents (recursive until the Root node group), the method fails with MDLAPI_E_NODEGROUP_IS_ALREADY_PARENT.

Return Value

False if the node group is already a child of this node group.

Extended Status Codes

MDLAPI_E_NODEGROUP_NOT_EXIST
    Node Group does not exist.

MDLAPI_E_NODEGROUP_HIERPATH_EXISTS
: Node Group with the same hierarchy path (Caption) already exists.

MDLAPI_E_NODEGROUP_IS_ROOT
: Root Node Group cannot be removed from or added to another node group.

MDLAPI_E_NODEGROUP_IS_SPECIAL_GROUP
: Special Node Group cannot be removed from or added to another node group.

MDLAPI_E_NODEGROUP_SPECIAL_ADD
: Child cannot be added to a special node group.

MDLAPI_E_NODEGROUP_IS_ALREADY_PARENT
: Node group already has a parent with the same name.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::AddReport()

OV_NodeGroup::AddReport_Trans()

```
boolean AddReport(
        [in] string ReportName)
```

```
boolean AddReport_Trans(
        [in] string TransId,
        [in] string ReportName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ReportName
: The Name of the report to be added to this node group. If the report name does not exist, the method fails with MDLAPI_E_REPORT_NOT_EXIST.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds a report to this node group.

If the Reporter integration tool is not installed, the method returns MDLAPI_E_NO_REPORT_INTEGRATION.

Return Value

False if the report has already been added to this node group.

Extended Status Codes

MDLAPI_E_INVALID_CONFIG_VALUE
: Configuration value is not valid.

MDLAPI_E_NO_REPORT_INTEGRATION
: Reporter integration tool is not installed.

MDLAPI_E_REPORT_NOT_EXIST

Report does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::DeleteGraph()
OV_NodeGroup::DeleteGraph_Trans()

```
void DeleteGraph()
```

```
void DeleteGraph_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Deletes graph properties of this node group.

Return Value

None.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::GetActions()

OV_NodeGroup::GetActions_Trans()

```
sint32 GetActions(
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeInherited)
```

```
sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeInherited)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Actions

Instances of OV_Action that are added to this node group. An instance is valid as a returned out parameter only if the method does not fail.

IncludeInherited

Indicates whether Actions also contains instances of OV_Action that are inherited from these node group parents (and their parents, recursively). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of tools (instances of OV_Action) added to this node group.

Return Value

Number of tools in the out parameter Actions.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

```
Transaction with the specified ID does not exist.
```

OV_NodeGroup::GetAutoDeployPackages()
OV_NodeGroup::GetAutoDeployPackages_Trans()

```
sint32 GetAutoDeployPackages(
        [out] OV_AutoDeployPackages AutoDeployPackages[] )
```

```
sint32 GetAutoDeployPackages_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPackages AutoDeployPackages[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPackages
: Instances of OV_AutoDeployPackage that are added to this node group. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of auto-deploy packages (instances of OV_AutoDeployPackage) added to this node group.

Return Value

Number of auto-deploy packages in the out parameter AutoDeployPackages.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::GetAutoDeployPolicyGroups() OV_NodeGroup::GetAutoDeployPolicyGroups_Trans()

```
sint32 GetAutoDeployPolicyGroups(
        [out] OV_AutoDeployPolicyGroup AutoDeployPolicyGroups[] )
```

```
sint32 GetAutoDeployPolicyGroups_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPolicyGroup AutoDeployPolicyGroups[] )
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

AutoDeployPolicyGroups

Instances of OV_AutoDeployPolicyGroup that are added to this node group. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of auto-deploy policy groups (instances of OV_AutoDeployPolicyGroup) added to this node group.

Return Value

Number of auto-deploy policy groups in the out parameter AutoDeployPolicyGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::GetChildNodeGroups()
OV_NodeGroup::GetChildNodeGroups_Trans()

```
sint32 GetChildNodeGroups(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeSubGroups)
```

```
sint32 GetChildNodeGroups_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

NodeGroups

Instances of OV_NodeGroup that are children (directly or indirectly, if IncludeSubGroups is equal to true) of this node group. An instance is valid as a returned out parameter only if the method does not fail.

IncludeSubGroups

Indicates whether NodeGroups also contains instances of OV_NodeGroup that are child node groups of all child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of node groups (instances of OV_NodeGroup) that are children of this node group.

Return Value

Number of node groups (children) in the out parameter NodeGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::GetExternalNodes() OV_NodeGroup::GetExternalNodes_Trans()

```
sint32 GetExternalNodes(
        [out] OV_ExternalNode Nodes[],
        [in, optional] boolean IncludeSubGroups)
```

```
sint32 GetExternalNodes_Trans(
        [in] string TransId,
        [out] OV_ExternalNode Nodes[],
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Nodes

Instances of OV_ExternalNode that are children (directly or indirectly if IncludeSubGroups is equal to true) of this node group. An instance is valid as a returned out parameter only if the method does not fail.

IncludeSubGroups

Indicates whether Nodes also contain instances of OV_ExternalNode that are child nodes of all child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of external nodes (instances of OV_ExternalNode) that are children of this node group.

Return Value

Number of external nodes (children) in out parameter Nodes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::GetNodes()
OV_NodeGroup::GetNodes_Trans()

```
sint32 GetNodes(
        [out] OV_ManagedNode Nodes[],
        [in, optional] boolean IncludeSubGroups)
```

```
sint32 GetNodes_Trans(
        [in] string TransId,
        [out] OV_ManagedNode Nodes[],
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Nodes

Instances of OV_ManagedNode that are children (directly or indirectly, if IncludeSubGroups is equal to true) of this node group. An instance is valid as a returned out parameter only if the method does not fail.

IncludeSubGroups

Indicates whether Nodes also contain instances of OV_ManagedNode that are child nodes of all child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of nodes (instances of OV_ManagedNode) that are children of this node group.

Return Value

Number of nodes (children) in out parameter Nodes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::GetParents() OV_NodeGroup::GetParents_Trans()

```
sint32 GetParents(
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

```
sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

NodeGroups

Instances of OV_NodeGroup that are parents of this node group. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllHierarchicalParents

Indicates whether NodeGroups also include instances of OV_NodeGroup that are hierarchical parents (recursive until the root node group), rather than direct parents. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of node groups for which this node group is a child.

Return Value

Number of node groups (parents) in the out parameter NodeGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::GetReports()
OV_NodeGroup::GetReports_Trans()

```
sint32 GetReports(
        [out] string Reports[] )
```

```
sint32 GetReports_Trans(
        [in] string TransId,
        [out] string Reports[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Reports
: The Names of reports that are added to this node group. The list is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of names of reports added to this node group.

Return Value

Number of report names in the out parameter Reports.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::GetServices() OV_NodeGroup::GetServices_Trans()

```
sint32 GetServices(
        [out] OV_Service Services[] )
```

```
sint32 GetServices_Trans(
        [in] string TransId,
        [out] OV_Service Services[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Services
: Instances of OV_Service that are added to this node group. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) hosted on this node group.

Return Value

Number of services in the out parameter Services.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::HasAction()
OV_NodeGroup::HasAction_Trans()

```
boolean HasAction(
        [in] string ActionName,
        [in, optional] boolean IncludeInherited)
```

```
boolean HasAction_Trans(
        [in] string TransId,
        [in] string ActionName,
        [in, optional] boolean IncludeInherited)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ActionName
: The Name property of the tool (instance of OV_Action) used to verify that the tool is associated with this node group.

IncludeInherited
: Indicates whether the tool to find can also be inherited from these node group parents (and their parents, recursively). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified tool (instance of OV_Action) is associated with this node group.

Return Value

True if the tool (instance of OV_Action) specified with the ActionName parameter is associated with this node group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::HasAutoDeployPackage()
OV_NodeGroup::HasAutoDeployPackage_Trans()

```
boolean HasAutoDeployPackage(
        [in] string AutoDeployPackageName)
```

```
boolean HasAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPackageName
: The Name property of the auto-deploy package (instance of OV_AutoDeployPackage) used to verify that the package is associated with this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified auto-deploy package (instance of OV_AutoDeployPackage) is associated to this node group.

Return Value

True if the auto-deploy package (instance of OV_AutoDeployPackage) specified with the AutoDeployPackageName parameter is associated with this node group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::HasAutoDeployPolicyGroup() OV_NodeGroup::HasAutoDeployPolicyGroup_Trans()

```
boolean HasAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName)
```

```
boolean HasAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPolicyGroupName
: The Name property of the auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) used to verify that the policy group is associated with this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) is associated with this node group.

Return Value

True if the auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) specified with the AutoDeployPolicyGroupName parameter is associated with this node group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::HasChildExternalNode()

OV_NodeGroup::HasChildExternalNode_Trans()

```
boolean HasChildExternalNode(
        [in] string NodeName,
        [in, optional] boolean IncludeSubGroups)
```

```
boolean HasChildExternalNode_Trans(
        [in] string TransId,
        [in] string NodeName,
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeName
: The Name property of the external node (instance of OV_ExternalNode) used to verify that the external node is a child of this node group.

IncludeSubGroups
: Indicates whether the external node to find is also searched in child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified external node (instance of OV_ExternalNode) is a child of this node group.

Return Value

True if the external node (instance of OV_ExternalNode) specified by the NodeName parameter is a child of this node group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::HasChildNode()
OV_NodeGroup::HasChildNode_Trans()

```
boolean HasChildNode(
        [in] string NodeName,
        [in, optional] boolean IncludeSubGroups)
```

```
boolean HasChildNode_Trans(
        [in] string TransId,
        [in] string NodeName,
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeName
: The Name property of the node (instance of OV_ManagedNode) used to verify that the node is a child of this node group.

IncludeSubGroups
: Indicates whether the node to find is also searched in child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified node (instance of OV_ManagedNode) is a child of this node group.

Return Value

True if the node (instance of OV_ManagedNode) specified by the NodeName parameter is a child of this node group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::HasChildNodeGroup()
OV_NodeGroup::HasChildNodeGroup_Trans()

```
boolean HasChildNodeGroup(
        [in] string NodeGroupName,
        [in, optional] boolean IncludeSubGroups)
```

```
boolean HasChildNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName,
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeGroupName
: The Name property of the node group (instance of OV_NodeGroup) used to verify that the node group is a child of this node group.

IncludeSubGroups
: Indicates whether the node group to find is also searched in child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified node group (instance of OV_NodeGroup) is a child of this node group.

Return Value

True if the node group (instance of OV_NodeGroup) specified by the NodeGroupName parameter is a child of this node group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::HasReport()
OV_NodeGroup::HasReport_Trans()

```
boolean HasReport(
        [in] string ReportName)
```

```
boolean HasReport_Trans(
        [in] string TransId,
        [in] string ReportName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ReportName
    The Name of the report used to verify that it is associated with this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified report is associated with this node group.

Return Value

True if the report with the name ReportName is associated with this node group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_NodeGroup::IsChildOf()

OV_NodeGroup::IsChildOf_Trans()

```
boolean IsChildOf(
        [in] string ParentName)
```

```
boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

ParentName
The Name property of the node group (instance of OV_NodeGroup) that is used to verify whether it is a parent of this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified node group (instance of OV_NodeGroup) is a parent of this node group.

Return Value

True if the node group (instance of OV_NodeGroup) specified with the ParentName parameter is a parent of this node group.

Extended Status Codes

MDLAPI_E_NODEGROUP_NOT_EXIST
Node Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_NodeGroup::Modify()
OV_NodeGroup::Modify_Trans()

```
void Modify(
        [in] OV_NodeGroup NodeGroup)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_NodeGroup NodeGroup)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

NodeGroup
    Modified instance of OV_NodeGroup to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes performed to one or more properties.

It is not possible to get an instance of OV_Nodegroup to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, an instance of OV_Nodegroup is specified as the NodeGroup parameter, even though this method is already called in the context of this instance.

As with the OV_NodeGroup::Create method, this method checks the following:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If a property has an invalid value, the method fails with MDLAPI_E_INVALID_PROPERTY.
- If the Caption property is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.
- If node group with the same hierarchical path already exists, the method fails with MDLAPI_E_NODEGROUP_HIERPATH_EXISTS.

- If graph properties validation fails, the method fails with MDLAPI_E_GRAPH_PROP_NOT_VALID.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING
: Property is not set.

MDLAPI_E_INVALID_PROPERTY
: Property is not valid.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_NODEGROUP_HIERPATH_EXISTS
: Node Group with the same hierarchy path (Caption) already exists.

MDLAPI_E_INVALID_CONFIG_VALUE
: Configuration value is not valid.

MDLAPI_E_NO_GRAPH_INTEGRATION
: Performance Manager integration tool is not installed.

MDLAPI_E_GRAPH_PROP_NOT_VALID
: Graph properties (GraphFamily and GraphCategory) are not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::RemoveAction() OV_NodeGroup::RemoveAction_Trans()

```
void RemoveAction(
        [in] string ActionName)
```

```
void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ActionName
: The Name property of the instance of OV_Action to be removed from this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes a tool (instance of OV_Action) from this node group.

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
: Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::RemoveAutoDeployPackage()
OV_NodeGroup::RemoveAutoDeployPackage_Trans()

```
void RemoveAutoDeployPackage(
        [in] string AutoDeployPackageName)
```

```
void RemoveAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPackageName
: The Name property of the instance of OV_AutoDeployPackage to be removed from this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes an auto-deploy package (instance of OV_AutoDeployPackage) from this node group.

Return Value

None.

Extended Status Codes

MDLAPI_E_ADPACKAGE_NOT_EXIST
: Auto Deploy Package does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::RemoveAutoDeployPolicyGroup() OV_NodeGroup::RemoveAutoDeployPolicyGroup_Trans()

```
void RemoveAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName)
```

```
void RemoveAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

AutoDeployPolicyGroupName
    The Name property of the instance of OV_AutoDeployPolicyGroup to be removed from this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes an auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) from this node group.

This method removes only the association to the instance of OV_AutoDeployPolicyGroup. To remove the actual instance of OV_AutoDeployPolicyGroup, you call the OV_AutoDeployPolicyGroup::Remove method.

Return Value

None.

Extended Status Codes

MDLAPI_E_ADPOLGROUP_NOT_EXIST
    Auto Deploy Policy Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_NodeGroup::RemoveExternalNode()
OV_NodeGroup::RemoveExternalNode_Trans()

```
void RemoveExternalNode(
        [in] string NodeName)
```

```
void RemoveExternalNode_Trans(
        [in] string TransId,
        [in] string NodeName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeName
: The Name property of the instance of OV_ExternalNode to be removed from this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes an external node (instance of OV_ExternalNode) from this node group.

CAUTION:
If the external node no longer belongs to any node group after this operation, it is removed from STORE.

Return Value

None.

Extended Status Codes

MDLAPI_E_EXT_NODE_NOT_EXIST
: External node does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_NodeGroup::RemoveNode() OV_NodeGroup::RemoveNode_Trans()

```
void RemoveNode(
        [in] string NodeName)
```

```
void RemoveNode_Trans(
        [in] string TransId,
        [in] string NodeName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeName
: The Name property of the instance of OV_ManagedNode to be removed from this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes a node (instance of OV_ManagedNode) from this node group.

CAUTION:
If a node no longer belongs to any node group after this operation, it is removed from STORE. To find out what else is removed when the node is removed, see the description of the Remove class method of OV_ManagedNode.

Return Value

None.

Extended Status Codes

MDLAPI_E_NODE_NOT_EXIST
: Node does not exist.

MDLAPI_E_NODE_IS_MANAGEMENT_SERVER
: Management server cannot be removed.

MDLAPI_E_NODEGROUP_SPECIAL_REMOVE
: Child cannot be removed from a special node group.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::RemoveNodeGroup()
OV_NodeGroup::RemoveNodeGroup_Trans()

```
void RemoveNodeGroup(
        [in] string NodeGroupName)
```

```
void RemoveNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

NodeGroupName
    The Name property of the instance of OV_NodeGroup to be removed from this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes a child node group (instance of OV_NodeGroup) from this node group.

You cannot remove special node groups (the method fails with MDLAPI_E_NODEGROUP_IS_SPECIAL_GROUP). You cannot remove a node group from a special node group (the method fails with MDLAPI_E_NODEGROUP_SPECIAL_REMOVE).

⚠ CAUTION:
If a node group no longer belongs to any node group after this operation, it is removed from STORE. To find out what else is removed when the node group is removed, see the description of the Remove class method.

Return Value

None.

Extended Status Codes

MDLAPI_E_NODE_IS_MANAGEMENT_SERVER	Management server cannot be removed.
MDLAPI_E_NODEGROUP_NOT_EXIST	Node Group does not exist.
MDLAPI_E_NODEGROUP_IS_SPECIAL_GROUP	Special Node Group cannot be removed from or added to another node group.
MDLAPI_E_NODEGROUP_SPECIAL_REMOVE	Child cannot be removed from a special node group.
MDLAPI_E_TRANSACTION_NOT_EXIST	Transaction with the specified ID does not exist.

OV_NodeGroup::RemoveReport()
OV_NodeGroup::RemoveReport_Trans()

```
void RemoveReport(
        [in] string ReportName)
```

```
void RemoveReport_Trans(
        [in] string TransId,
        [in] string ReportName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ReportName
: The Name of the report to be removed from this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes a report from this node group.

Return Value

None.

Extended Status Codes

MDLAPI_E_REPORT_NOT_EXIST
: Report does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_NodeGroup::SetGraph()
OV_NodeGroup::SetGraph_Trans()

```
void SetGraph(
        [in, optional] string GraphFamily,
        [in] string GraphCategory)
```

```
void SetGraph_Trans(
        [in] string TransId,
        [in, optional] string GraphFamily,
        [in] string GraphCategory)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

GraphFamily
: New GraphFamily property value of this node group. Optional parameter.

GraphCategory
: New GraphCategory property value of this node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds graph properties to this node group.

If the Performance Manager integration tool is not installed, the method returns MDLAPI_E_NO_GRAPH_INTEGRATION. Graph family and graph category must be valid. Otherwise, the method fails with MDLAPI_E_GRAPH_NOT_EXIST. If only graph category is selected, the graph family is selected automatically.

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_CONFIG_VALUE
Configuration value is not valid.

MDLAPI_E_NO_GRAPH_INTEGRATION
Performance Manager integration tool is not installed.

MDLAPI_E_GRAPH_NOT_EXIST
Graph does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_NodeGroup::SetOutage()

OV_NodeGroup::SetOutage_Trans()

```
void SetOutage(
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage,
        [in, optional] sint32 DisableHeartBeatPolingInOutage,
        [in, optional] sint32 RemovePolicyInMaintenance,
        [in, optional] boolean IncludeAllSubordinate)
```

```
void SetOutage_Trans(
        [in] string TransId,
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage,
        [in, optional] sint32 DisableHeartBeatPolingInOutage,
        [in, optional] sint32 RemovePolicyInMaintenance,
        [in, optional] boolean IncludeAllSubordinate)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

IsInOutage

Changes/sets the node mode (scheduled outage or maintenance). Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. KEEP does not change the node outage mode, ON enables and OFF disables the node outage mode, and TOGGLE changes the node outage mode from ON to OFF, or from OFF to ON. If this parameter has invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

IsScheduled

Flag for scheduled outage or maintenance mode. If this parameter is set to true, the node is in scheduled outage mode. Otherwise (false), the node is in unplanned maintenance mode.

DeleteMessageInOutage

Flag that indicates if the node messages is removed or not. Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. Default is KEEP. If this parameter has an invalid value, the method fails with

MDLAPI_E_INVALID_PARAMETER.

DisableHeartBeatPolingInOutage
: Flag that indicates whether node heart beat polling is disabled. Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. Default is KEEP. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

RemovePolicyInMaintenance
: Flag that indicates whether policies are removed from the node. Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. Default is KEEP. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

IncludeAllSubordinate
: Indicates whether the node to set outage is also searched in child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets scheduled outage/maintenance mode properties of nodes belonging to this node group.

Also services hosted on these nodes are affected (according to new node mode). If node subgroups are included, you need to remove duplicate nodes from the node list. Only administrators and operators with special rights can execute this method. Otherwise, the method fails with MDLAPI_E_NO_OPERATION_RIGHTS

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_NO_OPERATION_RIGHTS
: User has no rights to execute this operation.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule

Description

The set of rules to apply to a component object, dependent object, or messages related to the object. These rules are used to determine the state of an object or association.

```
class OV_PropagationRule
{
        Properties:
        string SettingID;
        sint16 DefaultRule;
        sint16 NormalRule;
        sint16 WarningRule;
        sint16 MinorRule;
        sint16 MajorRule;
        sint16 CriticalRule;

        Class Methods:

        OV_PropagationRule Create(
                [in] string Caption,
                [in, optional] sint16 DefaultRule,
                [in, optional] sint16 NormalRule,
                [in, optional] sint16 WarningRule,
                [in, optional] sint16 MinorRule,
                [in, optional] sint16 MajorRule,
                [in, optional] sint16 CriticalRule,
                [in, optional] string SettingID,
                [in, optional] string Description
                );

        OV_PropagationRule Create_Trans(
                [in] string TransId,
                [in] string Caption,
                [in, optional] sint16 DefaultRule,
                [in, optional] sint16 NormalRule,
                [in, optional] sint16 WarningRule,
                [in, optional] sint16 MinorRule,
                [in, optional] sint16 MajorRule,
                [in, optional] sint16 CriticalRule,
                [in, optional] string SettingID,
                [in, optional] string Description
                );

        void Remove(
                [in] string SettingId
                );

        void Remove_Trans(
                [in] string TransId,
                [in] string SettingId
                );

        OV_PropagationRule GetById(
                [in] string SettingId
                );

        OV_PropagationRule GetById_Trans(
                [in] string TransId,
                [in] string SettingId
                );

        OV_PropagationRule GetByCaption(
                [in] string Caption
                );

        OV_PropagationRule GetByCaption_Trans(
                [in] string TransId,
                [in] string Caption
                );

        OV_PropagationRule GetFromServiceComposition(
                [in] string ParentService,
                [in] string ChildService
                );

        OV_PropagationRule GetFromServiceComposition_Trans(
                [in] string TransId,
                [in] string ParentService,
                [in] string ChildService
                );

        OV_PropagationRule GetFromServiceDependency(
                [in] string AntecedentService,
                [in] string DependentService
                );

        OV_PropagationRule GetFromServiceDependency_Trans(
                [in] string TransId,
                [in] string AntecedentService,
                [in] string DependentService
                );
```

```
OV_PropagationRule GetFromServiceTypeComposition(
        [in] string ParentServiceType,
        [in] string ChildServiceType
        );

OV_PropagationRule GetFromServiceTypeComposition_Trans(
        [in] string TransId,
        [in] string ParentServiceType,
        [in] string ChildServiceType
        );

OV_PropagationRule GetFromServiceTypeDependency(
        [in] string AntecedentServiceType,
        [in] string DependentServiceType
        );

OV_PropagationRule GetFromServiceTypeDependency_Trans(
        [in] string TransId,
        [in] string AntecedentServiceType,
        [in] string DependentServiceType
        );
```

Instance Methods:

```
void Modify(
        [in] OV_PropagationRule PropagationRule
        );

void Modify_Trans(
        [in] string TransId,
        [in] OV_PropagationRule PropagationRule
        );

sint32 GetServicesWithMsgPropRule(
        [out] OV_Service Services[]
        );

sint32 GetServicesWithMsgPropRule_Trans(
        [in] string TransId,
        [out] OV_Service Services[]
        );

sint32 GetParentServices(
        [out] OV_Service ParentServices[]
        );

sint32 GetParentServices_Trans(
        [in] string TransId,
        [out] OV_Service ParentServices[]
```

```
          );

     sint32 GetChildServices(
          [out] OV_Service ChildServices[]
          );

     sint32 GetChildServices_Trans(
          [in] string TransId,
          [out] OV_Service ChildServices[]
          );

     sint32 GetAntecedentServices(
          [out] OV_Service AntecedentServices[]
          );

     sint32 GetAntecedentServices_Trans(
          [in] string TransId,
          [out] OV_Service AntecedentServices[]
          );

     sint32 GetDependentServices(
          [out] OV_Service DependentServices[]
          );

     sint32 GetDependentServices_Trans(
          [in] string TransId,
          [out] OV_Service DependentServices[]
          );

     sint32 GetServiceTypesWithMsgPropRule(
          [out] OV_ServiceTypeDefinition ServiceTypes[]
          );

     sint32 GetServiceTypesWithMsgPropRule_Trans(
          [in] string TransId,
          [out] OV_ServiceTypeDefinition ServiceTypes[]
          );

     sint32 GetParentServiceTypes(
          [out] OV_ServiceTypeDefinition ParentServiceTypes[]
          );

     sint32 GetParentServiceTypes_Trans(
          [in] string TransId,
          [out] OV_ServiceTypeDefinition ParentServiceTypes[]
          );

     sint32 GetChildServiceTypes(
          [out] OV_ServiceTypeDefinition ChildServiceTypes[]
          );
```

```
        sint32 GetChildServiceTypes_Trans(
                [in] string TransId,
                [out] OV_ServiceTypeDefinition ChildServiceTypes[]
                );

        sint32 GetAntecedentServiceTypes(
                [out] OV_ServiceTypeDefinition AntecedentServiceTypes[]
                );

        sint32 GetAntecedentServiceTypes_Trans(
                [in] string TransId,
                [out] OV_ServiceTypeDefinition AntecedentServiceTypes[]
                );

        sint32 GetDependentServiceTypes(
                [out] OV_ServiceTypeDefinition DependentServiceTypes[]
                );

        sint32 GetDependentServiceTypes_Trans(
                [in] string TransId,
                [out] OV_ServiceTypeDefinition DependentServiceTypes[]
                );
};
```

OV_PropagationRule-Properties

SettingID

Signature

string SettingID

Description

The identifier to uniquely identify the setting.

DefaultRule

Signature

sint16 DefaultRule

Description

-99 = ignore, -98 = Use Default Rule, -4..3 = increment/decrement, 1026 = SetTo Normal, 1028 = SetTo Warning, 1032 = SetTo Minor, 1040 = SetTo Major, 1056 = SetTo Critical.

Values

ValueMap		Values
-99	=	Ignore
-98	=	UseDefault
-4	=	Decrement4
-3	=	Decrement3
-2	=	Decrement2
-1	=	Decrement1
0	=	NoChange
1	=	Increment1
2	=	Increment2
3	=	Increment3
4	=	Increment4
1026	=	SetToNormal
1028	=	SetToWarning
1032	=	SetToMinor
1040	=	SetToMajor
1056	=	SetToCritical

NormalRule

Signature

sint16 NormalRule

Description

-99 = ignore, -98 = Use Default Rule, -4..3 = increment/decrement, 1026 = SetTo Normal, 1028 = SetTo Warning, 1032 = SetTo Minor, 1040 = SetTo Major, 1056 = SetTo Critical.

Values

ValueMap	Values
-99 =	Ignore
-98 =	UseDefault
-4 =	Decrement4
-3 =	Decrement3
-2 =	Decrement2
-1 =	Decrement1
0 =	NoChange
1 =	Increment1
2 =	Increment2
3 =	Increment3
4 =	Increment4
1026 =	SetToNormal
1028 =	SetToWarning
1032 =	SetToMinor
1040 =	SetToMajor
1056 =	SetToCritical

WarningRule

Signature

sint16 WarningRule

Description

-99 = ignore, -98 = Use Default Rule, -4..3 = increment/decrement, 1026 = SetTo Normal, 1028 = SetTo Warning, 1032 = SetTo Minor, 1040 = SetTo Major, 1056 = SetTo Critical.

Values

ValueMap	Values
-99	= Ignore
-98	= UseDefault
-4	= Decrement4
-3	= Decrement3
-2	= Decrement2
-1	= Decrement1
0	= NoChange
1	= Increment1
2	= Increment2
3	= Increment3
4	= Increment4
1026	= SetToNormal
1028	= SetToWarning
1032	= SetToMinor
1040	= SetToMajor
1056	= SetToCritical

MinorRule

Signature

sint16 MinorRule

Description

-99 = ignore, -98 = Use Default Rule, -4..3 = increment/decrement, 1026 = SetTo Normal, 1028 = SetTo Warning, 1032 = SetTo Minor, 1040 = SetTo Major, 1056 = SetTo Critical.

Values

ValueMap	Values
-99	= Ignore
-98	= UseDefault
-4	= Decrement4
-3	= Decrement3
-2	= Decrement2
-1	= Decrement1
0	= NoChange
1	= Increment1
2	= Increment2
3	= Increment3
4	= Increment4
1026	= SetToNormal
1028	= SetToWarning
1032	= SetToMinor
1040	= SetToMajor
1056	= SetToCritical

MajorRule

Signature

sint16 MajorRule

Description

-99 = ignore, -98 = Use Default Rule, -4..3 = increment/decrement, 1026 = SetTo Normal, 1028 = SetTo Warning, 1032 = SetTo Minor, 1040 = SetTo Major, 1056 = SetTo Critical.

Values

ValueMap	Values
-99	= Ignore
-98	= UseDefault
-4	= Decrement4
-3	= Decrement3
-2	= Decrement2
-1	= Decrement1
0	= NoChange
1	= Increment1
2	= Increment2
3	= Increment3
4	= Increment4
1026	= SetToNormal
1028	= SetToWarning
1032	= SetToMinor
1040	= SetToMajor
1056	= SetToCritical

CriticalRule

Signature

sint16 CriticalRule

Description

-99 = ignore, -98 = Use Default Rule, -4..3 = increment/decrement, 1026 = SetTo Normal, 1028 = SetTo Warning, 1032 = SetTo Minor, 1040 = SetTo Major, 1056 = SetTo Critical.

Values

ValueMap	Values
-99	= Ignore
-98	= UseDefault
-4	= Decrement4
-3	= Decrement3
-2	= Decrement2
-1	= Decrement1
0	= NoChange
1	= Increment1
2	= Increment2
3	= Increment3
4	= Increment4
1026	= SetToNormal
1028	= SetToWarning
1032	= SetToMinor
1040	= SetToMajor
1056	= SetToCritical

OV_PropagationRule Class Methods

This section contains the information on Class methods for OV_PropagationRule.

OV_PropagationRule::Create()
OV_PropagationRule::Create_Trans()

```
OV_PropagationRule Create(
        [in] string Caption,
        [in, optional] sint16 DefaultRule,
        [in, optional] sint16 NormalRule,
        [in, optional] sint16 WarningRule,
        [in, optional] sint16 MinorRule,
        [in, optional] sint16 MajorRule,
        [in, optional] sint16 CriticalRule,
        [in, optional] string SettingID,
        [in, optional] string Description)
```

```
OV_PropagationRule Create_Trans(
        [in] string TransId,
        [in] string Caption,
        [in, optional] sint16 DefaultRule,
        [in, optional] sint16 NormalRule,
        [in, optional] sint16 WarningRule,
        [in, optional] sint16 MinorRule,
        [in, optional] sint16 MajorRule,
        [in, optional] sint16 CriticalRule,
        [in, optional] string SettingID,
        [in, optional] string Description)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Caption

The Caption property of the instance of OV_PropagationRule to be created. If a propagation rule with the same Caption already exists, nothing happens, and the method fails with MDLAPI_E_PROPRULE_CAPTION_EXIST. If the Caption parameter is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

DefaultRule

The DefaultRule property value of this service. Optional parameter. Default is 0.

NormalRule
: The NormalRule property value of this service. Optional parameter. Default is -98.

WarningRule
: The WarningRule property value of this service. Optional parameter. Default is -98.

MinorRule
: The MinorRule property value of this service. Optional parameter. Default is -98.

MajorRule
: The MajorRule property value of this service. Optional parameter. Default is -98.

CriticalRule
: The CriticalRule property value of this service. Optional parameter. Default is -98.

SettingID
: The SettingID property of an instance of the OV_PropagationRule to be created. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is created as a new GUID. If a propagation rule with the same SettingID already exists, nothing happens, and the method fails with MDLAPI_E_PROPRULE_SETTINGID_EXISTS.

Description
: The Description property of the instance of OV_PropagationRule to be created. Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a propagation rule (new instance of OV_PropagationRule).

If any propagation rule parameter is not valid, the method fails with MDLAPI_E_INVALID_PARAMETER.

Return Value

Instance of the newly created OV_PropagationRule.

Extended Status Codes

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_PROPRULE_CAPTION_EXIST
: Propagation rule with the same Caption already exists.

MDLAPI_E_PROPRULE_SETTINGID_EXISTS
: Propagation rule with the same SettingId already exists.

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::GetByCaption() OV_PropagationRule::GetByCaption_Trans()

```
OV_PropagationRule GetByCaption(
        [in] string Caption)
```

```
OV_PropagationRule GetByCaption_Trans(
        [in] string TransId,
        [in] string Caption)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Caption
: The Caption property of an instance of OV_PropagationRule to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the propagation rule (instance of OV_PropagationRule) specified by the Caption property.

Return Value

Instance of OV_PropagationRule specified by a Caption property.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_PROPRULE_CAPTION_NOT_UNIQUE
: More than one propagation rule with the specified Caption is found.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::GetById() OV_PropagationRule::GetById_Trans()

```
OV_PropagationRule GetById(
        [in] string SettingId)
```

```
OV_PropagationRule GetById_Trans(
        [in] string TransId,
        [in] string SettingId)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

SettingId
    The SettingId property of an instance of OV_PropagationRule to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the propagation rule (instance of OV_PropagationRule) specified by the SettingId property.

Return Value

Instance of OV_PropagationRule specified by a SettingId property.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
    Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_PropagationRule::GetFromServiceComposition() OV_PropagationRule::GetFromServiceComposition_Trans()

```
OV_PropagationRule GetFromServiceComposition(
        [in] string ParentService,
        [in] string ChildService)
```

```
OV_PropagationRule GetFromServiceComposition_Trans(
        [in] string TransId,
        [in] string ParentService,
        [in] string ChildService)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentService
: The Name property of the parent service (instance of OV_Service) in the service composition.

ChildService
: The Name property of the child service (instance of OV_Service) in the service composition.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the propagation rule (instance of OV_Propagation) specified with the service composition.

Return Value

Instance of OV_PropagationRule specified with the service composition.

Extended Status Codes

MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST
: Service composition does not exist.

MDLAPI_E_PROPRULE_NOT_EXIST

Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_PropagationRule::GetFromServiceDependency() OV_PropagationRule::GetFromServiceDependency_Trans()

```
OV_PropagationRule GetFromServiceDependency(
        [in] string AntecedentService,
        [in] string DependentService)
```

```
OV_PropagationRule GetFromServiceDependency_Trans(
        [in] string TransId,
        [in] string AntecedentService,
        [in] string DependentService)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentService
: The Name property of the antecedent service (instance of OV_Service) in the service dependency.

DependentService
: The Name property of the dependent service (instance of OV_Service) in the service dependency.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the propagation rule (instance of OV_Propagation) specified with the service dependency.

Return Value

Instance of OV_PropagationRule specified with the service dependency.

Extended Status Codes

MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST
: Service dependency does not exist.

MDLAPI_E_PROPRULE_NOT_EXIST

Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_PropagationRule::GetFromServiceTypeComposition()
OV_PropagationRule::GetFromServiceTypeComposition_Trans()

```
OV_PropagationRule GetFromServiceTypeComposition(
        [in] string ParentServiceType,
        [in] string ChildServiceType)
```

```
OV_PropagationRule GetFromServiceTypeComposition_Trans(
        [in] string TransId,
        [in] string ParentServiceType,
        [in] string ChildServiceType)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentServiceType
: The Name property of the parent service type (instance of OV_ServiceTypeDefinition) in the service type composition.

ChildServiceType
: The Name property of the child service type (instance of OV_ServiceTypedefinition) in the service type composition.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the propagation rule (instance of OV_Propagation) specified with the service type composition.

Return Value

Instance of OV_PropagationRule specified with the service type composition.

Extended Status Codes

MDLAPI_E_SVC_TYPE_COMP_ASSOC_NOT_EXIST

Service type composition does not exist.

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::GetFromServiceTypeDependency() OV_PropagationRule::GetFromServiceTypeDependency_Trans()

```
OV_PropagationRule GetFromServiceTypeDependency(
        [in] string AntecedentServiceType,
        [in] string DependentServiceType)
```

```
OV_PropagationRule GetFromServiceTypeDependency_Trans(
        [in] string TransId,
        [in] string AntecedentServiceType,
        [in] string DependentServiceType)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentServiceType
: The Name property of the antecedent service type (instance of OV_ServiceTypeDefinition) in the service type dependency.

DependentServiceType
: The Name property of the dependent service type (instance of OV_ServiceTypeDefinition) in the service type dependency.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the propagation rule (instance of OV_Propagation) specified with the service type dependency.

Return Value

Instance of OV_PropagationRule specified with the service type dependency.

Extended Status Codes

MDLAPI_E_SVC_TYPE_DEP_ASSOC_NOT_EXIST

Service type dependency does not exist.

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::Remove()
OV_PropagationRule::Remove_Trans()

```
void Remove(
        [in] string SettingId)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string SettingId)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

SettingId
    The SettingId property of an instance of OV_PropagationRule to be removed. If the propagation rule with the SettingId does not exist, the method fails with MDLAPI_E_PROPRULE_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified propagation rule (instance of OV_PropagationRule).

If the specified propagation rule is used by any other object or association, it is not removed, and the method fails with MDLAPI_E_PROPRULE_CANNOT_REMOVE

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPRULE_CANNOT_REMOVE
    Propagation rule cannot be removed because another object or association uses it.

MDLAPI_E_PROPRULE_NOT_EXIST
    Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_PropagationRule Instance Methods

This section contains the information on Instance methods for OV_PropagationRule.

OV_PropagationRule::GetAntecedentServices() OV_PropagationRule::GetAntecedentServices_Trans()

```
sint32 GetAntecedentServices(
        [out] OV_Service AntecedentServices[] )

sint32 GetAntecedentServices_Trans(
        [in] string TransId,
        [out] OV_Service AntecedentServices[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentServices
: Instances of OV_Service where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) where this propagation is used for message propagation from dependent services.

Return Value

Number of services in the out parameter AntecedentServices.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::GetAntecedentServiceTypes()
OV_PropagationRule::GetAntecedentServiceTypes_Trans()

```
sint32 GetAntecedentServiceTypes(
        [out] OV_ServiceTypeDefinition AntecedentServiceTypes[] )
```

```
sint32 GetAntecedentServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition AntecedentServiceTypes[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentServiceTypes
: Instances of OV_ServiceTypeDefinition where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) where this propagation is used for message propagation from dependent service types.

Return Value

Number of service types in the out parameter AntecedentServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::GetChildServices() OV_PropagationRule::GetChildServices_Trans()

```
sint32 GetChildServices(
        [out] OV_Service ChildServices[] )
```

```
sint32 GetChildServices_Trans(
        [in] string TransId,
        [out] OV_Service ChildServices[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ChildServices
: Instances of OV_Service where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) where this propagation is used for message propagation from parent services.

Return Value

Number of services in the out parameter ChildServices.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::GetChildServiceTypes()
OV_PropagationRule::GetChildServiceTypes_Trans()

```
sint32 GetChildServiceTypes(
        [out] OV_ServiceTypeDefinition ChildServiceTypes[] )
```

```
sint32 GetChildServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ChildServiceTypes[] )
```

Parameters

TransId
 Transaction ID returned from OV_Transaction::Start().

ChildServiceTypes
 Instances of OV_ServiceTypeDefinition where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) where this propagation is used for message propagation from parent service types.

Return Value

Number of service types in the out parameter ChildServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
 Transaction with the specified ID does not exist.

OV_PropagationRule::GetDependentServices() OV_PropagationRule::GetDependentServices_Trans()

```
sint32 GetDependentServices(
        [out] OV_Service DependentServices[] )

sint32 GetDependentServices_Trans(
        [in] string TransId,
        [out] OV_Service DependentServices[] )
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

DependentServices
Instances of OV_Service where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) where this propagation is used for message propagation from antecedent services.

Return Value

Number of services in the out parameter DependentServices.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_PropagationRule::GetDependentServiceTypes() OV_PropagationRule::GetDependentServiceTypes_Trans()

```
sint32 GetDependentServiceTypes(
        [out] OV_ServiceTypeDefinition DependentServiceTypes[] )
```

```
sint32 GetDependentServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition DependentServiceTypes[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

DependentServiceTypes
: Instances of OV_ServiceTypeDefinition where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) where this propagation is used for message propagation from antecedent service types.

Return Value

Number of service types in the out parameter DependentServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::GetParentServices()
OV_PropagationRule::GetParentServices_Trans()

```
sint32 GetParentServices(
        [out] OV_Service ParentServices[] )
```

```
sint32 GetParentServices_Trans(
        [in] string TransId,
        [out] OV_Service ParentServices[] )
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ParentServices
    Instances of OV_Service where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) where this propagation is used for message propagation from child services.

Return Value

Number of services in the out parameter ParentServices.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_PropagationRule::GetParentServiceTypes()
OV_PropagationRule::GetParentServiceTypes_Trans()

```
sint32 GetParentServiceTypes(
        [out] OV_ServiceTypeDefinition ParentServiceTypes[] )
```

```
sint32 GetParentServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ParentServiceTypes[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentServiceTypes
: Instances of OV_ServiceTypeDefinition where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) where this propagation is used for message propagation from child service types.

Return Value

Number of service types in the out parameter ParentServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::GetServicesWithMsgPropRule() OV_PropagationRule::GetServicesWithMsgPropRule_Trans()

```
sint32 GetServicesWithMsgPropRule(
        [out] OV_Service Services[] )
```

```
sint32 GetServicesWithMsgPropRule_Trans(
        [in] string TransId,
        [out] OV_Service Services[] )
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Services
Instances of OV_Service where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) where this propagation is used for message propagation.

Return Value

Number of services in the out parameter Services.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_PropagationRule::GetServiceTypesWithMsgPropRule() OV_PropagationRule::GetServiceTypesWithMsgPropRule_Trans()

```
sint32 GetServiceTypesWithMsgPropRule(
        [out] OV_ServiceTypeDefinition ServiceTypes[] )
```

```
sint32 GetServiceTypesWithMsgPropRule_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ServiceTypes[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ServiceTypes
: Instances of OV_ServiceTypeDefinition where this propagation rule is used. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) where this propagation is used for message propagation.

Return Value

Number of service types in the out parameter ServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_PropagationRule::Modify()
OV_PropagationRule::Modify_Trans()

```
void Modify(
        [in] OV_PropagationRule PropagationRule)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_PropagationRule PropagationRule)
```

Parameters

TransId
  Transaction ID returned from OV_Transaction::Start().

PropagationRule
  A modified instance of OV_PropagationRule to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes performed on one or more properties.

It is not possible to get an instance of OV_PropagationRule to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, an instance of OV_PropagationRule is specified as the propagation rule parameter, even though this method is already called in the context of this instance.

As with the OV_PropagationRule::Create method, this method checks the following:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If the Caption property is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.
- If a property has an invalid value, the method fails with MDLAPI_E_INVALID_PROPERTY.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING	Property is not set.
MDLAPI_E_INVALID_CAPTION	Specified object Caption parameter is not valid.
MDLAPI_E_INVALID_PROPERTY	Property is not valid.
MDLAPI_E_TRANSACTION_NOT_EXIST	Transaction with the specified ID does not exist.

OV_Service

Description

This class identifies service objects that represent actual or virtual services in the enterprise. These services may be discovered or entered manually.

```
class OV_Service
{
        Properties:
        string Name;
        string HostPath;
        string Attributes[];
        string CalcRuleId;
        string Icon;
        string MsgPropRuleId;
        real32 MsgWeightFactor;
        string ServiceTypeId;
        string ReportFamily;
        string ReportCategory;
        string GraphFamily;
        string GraphCategory;
        string GraphInstanceID;
        string SDSearchCode;
        string OriginalId;
        string MsgSvcName[];
        boolean IsInMaintMode;
        boolean IsInSchedOutage;
        boolean DeleteMsgInMaintMode;
        boolean DeleteMsgInSchedOutage;

        Class Methods:

        OV_Service Create(
                [in] string Caption,
                [in] string ServiceTypeId,
                [in, optional] sint32 HostingType,
                [in, optional] string HostedOnName,
                [in, optional] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description,
                [in, optional] string Attributes[]
                );
```

```
OV_Service Create_Trans(
        [in] string TransId,
        [in] string Caption,
        [in] string ServiceTypeId,
        [in, optional] sint32 HostingType,
        [in, optional] string HostedOnName,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] string Attributes[]
        );

void Remove(
        [in] string Name
        );

void Remove_Trans(
        [in] string TransId,
        [in] string Name
        );

OV_Service GetRoot(
        );

OV_Service GetRoot_Trans(
        [in] string TransId
        );

OV_Service GetByName(
        [in] string Name
        );

OV_Service GetByName_Trans(
        [in] string TransId,
        [in] string Name
        );

OV_Service GetByHierarchicalPath(
        [in] string Path
        );

OV_Service GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path
        );

sint32 GetServiceSet(
        [out] OV_Service Services[],
        [in, optional] string ServiceNames[],
        [in, optional] string ServicePaths[],
        [in, optional] string ParentServiceNames[],
        [in, optional] string ParentServicePaths[]
        );

sint32 GetServiceSet_Trans(
        [in] string TransId,
        [out] OV_Service Services[],
        [in, optional] string ServiceNames[],
        [in, optional] string ServicePaths[],
        [in, optional] string ParentServiceNames[],
        [in, optional] string ParentServicePaths[]
        );
```

Instance Methods:

```
void Modify(
        [in] OV_Service Service
        );

void Modify_Trans(
        [in] string TransId,
        [in] OV_Service Service
        );

sint32 GetParents(
        [out] OV_Service Services[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_Service Services[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

boolean IsChildOf(
        [in] string ParentName
        );

boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName
        );

sint32 GetChildServices(
        [out] OV_Service Services[],
        [in, optional] boolean IncludeAllSubordinate
        );
```

```
sint32 GetChildServices_Trans(
        [in] string TransId,
        [out] OV_Service Services[],
        [in, optional] boolean IncludeAllSubordinate
        );

boolean HasChildService(
        [in] string ServiceName,
        [in, optional] boolean IncludeAllSubordinate
        );

boolean HasChildService_Trans(
        [in] string TransId,
        [in] string ServiceName,
        [in, optional] boolean IncludeAllSubordinate
        );

sint32 GetAntecedentServices(
        [out] OV_Service AntecedentServices[],
        [in, optional] boolean IncludeAllAntecedent
        );

sint32 GetAntecedentServices_Trans(
        [in] string TransId,
        [out] OV_Service AntecedentServices[],
        [in, optional] boolean IncludeAllAntecedent
        );

boolean HasAntecedentService(
        [in] string AntecedentName,
        [in, optional] boolean IncludeAllAntecedent
        );

boolean HasAntecedentService_Trans(
        [in] string TransId,
        [in] string AntecedentName,
        [in, optional] boolean IncludeAllAntecedent
        );

boolean AddAntecedentService(
        [in] string AntecedentName,
        [in, optional] string PropRuleId,
        [in, optional] real32 WeightFactor,
        [in, optional] sint32 State
        );

boolean AddAntecedentService_Trans(
        [in] string TransId,
        [in] string AntecedentName,
        [in, optional] string PropRuleId,
        [in, optional] real32 WeightFactor,
        [in, optional] sint32 State
        );

void RemoveAntecedentService(
        [in] string AntecedentName
        );

void RemoveAntecedentService_Trans(
        [in] string TransId,
        [in] string AntecedentName
        );

sint32 GetDependentServices(
        [out] OV_Service DependentServices[],
        [in, optional] boolean IncludeAllDependent
        );

sint32 GetDependentServices_Trans(
        [in] string TransId,
        [out] OV_Service DependentServices[],
        [in, optional] boolean IncludeAllDependent
        );

boolean HasDependentService(
        [in] string DependentName,
        [in, optional] boolean IncludeAllDependent
        );

boolean HasDependentService_Trans(
        [in] string TransId,
        [in] string DependentName,
        [in, optional] boolean IncludeAllDependent
        );

boolean AddAction(
        [in] string ActionName
        );

boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

void RemoveAction(
        [in] string ActionName
        );

void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

sint32 GetActions(
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeDefault
        );

sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeDefault
        );

boolean HasAction(
        [in] string ActionName,
        [in, optional] boolean IncludeDefault
        );

boolean HasAction_Trans(
        [in] string TransId,
        [in] string ActionName,
        [in, optional] boolean IncludeDefault
        );

void SetReport(
        [in, optional] string ReportFamily,
        [in] string ReportCategory
        );

void SetReport_Trans(
        [in] string TransId,
        [in, optional] string ReportFamily,
        [in] string ReportCategory
        );

void DeleteReport(
        );

void DeleteReport_Trans(
        [in] string TransId
        );

void SetGraph(
        [in, optional] string GraphFamily,
        [in] string GraphCategory
        );

void SetGraph_Trans(
        [in] string TransId,
        [in, optional] string GraphFamily,
        [in] string GraphCategory
        );

void DeleteGraph(
        );

void DeleteGraph_Trans(
        [in] string TransId
        );

OV_ServiceTypeDefinition GetServiceType(
        );

OV_ServiceTypeDefinition GetServiceType_Trans(
        [in] string TransId
        );

boolean GetHostNode(
        [out] OV_ManagedNode Node
        );

boolean GetHostNode_Trans(
        [in] string TransId,
        [out] OV_ManagedNode Node
        );

boolean IsHostedOnNode(
        [in] string NodeName
        );

boolean IsHostedOnNode_Trans(
        [in] string TransId,
        [in] string NodeName
        );

boolean GetHostNodeGroup(
        [out] OV_NodeGroup NodeGroup
        );

boolean GetHostNodeGroup_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroup
        );

boolean IsHostedOnNodeGroup(
        [in] string NodeGroupName
        );

boolean IsHostedOnNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName
        );

void SetCalculationRule(
        [in] string CalcRuleId
        );

void SetCalculationRule_Trans(
        [in] string TransId,
        [in] string CalcRuleId
        );

OV_CalculationRule GetCalculationRule(
        );

OV_CalculationRule GetCalculationRule_Trans(
        [in] string TransId
        );

void SetMessagePropagationRule(
        [in] string MsgPropRuleId
        );

void SetMessagePropagationRule_Trans(
        [in] string TransId,
        [in] string MsgPropRuleId
        );

OV_PropagationRule GetMessagePropagationRule(
        );

OV_PropagationRule GetMessagePropagationRule_Trans(
        [in] string TransId
        );

void SetMessageWeightFactor(
        [in] real32 MsgWeightFactor
        );

void SetMessageWeightFactor_Trans(
        [in] string TransId,
        [in] real32 MsgWeightFactor
        );

real32 GetMessageWeightFactor(
        );

real32 GetMessageWeightFactor_Trans(
        [in] string TransId
        );
```

```
void SetDependencyPropagationRule(
        [in] string AntecedentService,
        [in] string PropRuleId
        );

void SetDependencyPropagationRule_Trans(
        [in] string TransId,
        [in] string AntecedentService,
        [in] string PropRuleId
        );

OV_PropagationRule GetDependencyPropagationRule(
        [in] string AntecedentService
        );

OV_PropagationRule GetDependencyPropagationRule_Trans(
        [in] string TransId,
        [in] string AntecedentService
        );

void SetDependencyWeightFactor(
        [in] string AntecedentService,
        [in] real32 MsgWeightFactor
        );

void SetDependencyWeightFactor_Trans(
        [in] string TransId,
        [in] string AntecedentService,
        [in] real32 MsgWeightFactor
        );

real32 GetDependencyWeightFactor(
        [in] string AntecedentService
        );

real32 GetDependencyWeightFactor_Trans(
        [in] string TransId,
        [in] string AntecedentService
        );

void SetCompositionPropagationRule(
        [in] string ChildService,
        [in] string PropRuleId
        );

void SetCompositionPropagationRule_Trans(
        [in] string TransId,
        [in] string ChildService,
        [in] string PropRuleId
```

```
                );

        OV_PropagationRule GetCompositionPropagationRule(
                [in] string ChildService
                );

        OV_PropagationRule GetCompositionPropagationRule_Trans(
                [in] string TransId,
                [in] string ChildService
                );

        void SetCompositionWeightFactor(
                [in] string ChildService,
                [in] real32 MsgWeightFactor
                );

        void SetCompositionWeightFactor_Trans(
                [in] string TransId,
                [in] string ChildService,
                [in] real32 MsgWeightFactor
                );

        real32 GetCompositionWeightFactor(
                [in] string ChildService
                );

        real32 GetCompositionWeightFactor_Trans(
                [in] string TransId,
                [in] string ChildService
                );

        void SetOutage(
                [in] sint32 IsInOutage,
                [in] boolean IsScheduled,
                [in, optional] sint32 DeleteMessageInOutage,
                [in, optional] boolean IncludeAllSubordinate
                );

        void SetOutage_Trans(
                [in] string TransId,
                [in] sint32 IsInOutage,
                [in] boolean IsScheduled,
                [in, optional] sint32 DeleteMessageInOutage,
                [in, optional] boolean IncludeAllSubordinate
                );
    };
```

OV_Service-Properties

Name

Signature

string Name

Description

The Name property uniquely identifies the Service and provides an indication of the functionality that is managed. This functionality is described in more detail in the object Description property.

HostPath

Signature

string HostPath

Description

The HostPath matches the Name property value of either an OV_ManagedNode or an OV_NodeGroup. The specified node or node group is the system that is hosting this service.

Attributes

Signature

string Attributes[]

Description

For use by SPIs only. Contains name/value pairs that list the attributes of the instantiated service.

CalcRuleId

Signature

string CalcRuleId

Description

SettingId of the service OV_CalculationRule.

Icon

Signature

string Icon

Description

The icon to use for this service if different from the icon defined for the service type definition. If this property is blank, the icon defined for the service type definition is used.

MsgPropRuleId

Signature
: string MsgPropRuleId

Description
: SettingId of the service message OV_PropagationRule.

MsgWeightFactor

Signature
: real32 MsgWeightFactor

Description
: The weight factor is how the service views the importance of incoming messages in relation to other components, dependencies, or messages.

ServiceTypeId

Signature
: string ServiceTypeId

Description
: GUID of the service associated OV_ServiceTypeDefinition.

ReportFamily

Signature
: string ReportFamily

Description
: The Report family to display when this service instance is the context for displaying the list of available reports. If this property is blank, the value from the service ServiceTypeDefinition is used.

ReportCategory

Signature
: string ReportCategory

Description
: The Report Category within the Report Family to display when this service instance is the context for displaying the list of available reports. If this property is blank, the value from the service

ServiceTypeDefinition is used.

GraphFamily

Signature

string GraphFamily

Description

The Graph family to display when this service instance is the context for displaying the list of available graphs. If this property is blank, the value from the service ServiceTypeDefinition is used.

GraphCategory

Signature

string GraphCategory

Description

The Graph Category within the Graph Family to display when this service instance is the context for displaying the list of available graphs. If this property is blank, the value from the service ServiceTypeDefinition is used.

GraphInstanceID

Signature

string GraphInstanceID

Description

The instance identifier to filter on when retrieving data from the CODA data class. If this property is blank, the instance id defined in the graph is used, or the graph displays all the instances within the data class.

SDSearchCode

Signature

string SDSearchCode

Description

SDSearchCode facilitates the integration of Service Desk and HPOM for Windows. The value of SDSearchCode corresponds to the SearchCode attribute in Service Desk.

OriginalId

Signature

string OriginalId

Description

Used to support Service Navigator Value Pack (SNVP).

MsgSvcName

Signature

```
string MsgSvcName[]
```

Description

An array of service IDs used to support Service Navigator Value Pack (SNVP).

IsInMaintMode

Signature

```
boolean IsInMaintMode
```

Description

True if the service is in maintenance mode.

IsInSchedOutage

Signature

```
boolean IsInSchedOutage
```

Description

True if the service is in a scheduled outage.

DeleteMsgInMaintMode

Signature

```
boolean DeleteMsgInMaintMode
```

Description

True if messages to the service should be deleted while it is in maintenance mode.

DeleteMsgInSchedOutage

Signature

```
boolean DeleteMsgInSchedOutage
```

Description

True if messages to the service should be deleted during a scheduled outage.

OV_Service Class Methods

This section contains the information on Class methods for OV_Service.

OV_Service::Create()
OV_Service::Create_Trans()

```
OV_Service Create(
        [in] string Caption,
        [in] string ServiceTypeId,
        [in, optional] sint32 HostingType,
        [in, optional] string HostedOnName,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] string Attributes[] )
```

```
OV_Service Create_Trans(
        [in] string TransId,
        [in] string Caption,
        [in] string ServiceTypeId,
        [in, optional] sint32 HostingType,
        [in, optional] string HostedOnName,
        [in, optional] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description,
        [in, optional] string Attributes[] )
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Caption

The Caption property of the instance of OV_Service to be created. If a service with the same Caption already exists on the same hierarchy path level (as a child of the same parent), nothing happens, and the method fails with MDLAPI_E_SERVICE_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ServiceTypeId

The GUID property of the instance of OV_ServiceTypeDefinition. Service type can be only one of the service types that have child-parent relationships with the service type of the parent service. If the specified service type cannot be assigned to newly created service, the method fails with

MDLAPI_E_SERVICETYPE_NOT_VALID.

HostingType

Sets the service hosting type. The service can be virtual (0), hosted on a node (1), or hosted on a node group (2). Optional parameter. Default is virtual service (0). If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

HostedOnName

The Name property of the node or node group where this service is hosted. Optional parameter. This parameter is used only if the HostingType parameter is set to 1 or 2. If the specified node or node group does not exist, the method fails with MDLAPI_E_NODE_NOT_EXIST or MDLAPI_E_NODEGROUP_NOT_EXIST.

ParentName

The Name property of the instance of OV_Service to which the newly created service is added. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, the root service is used. If the specified parent service does not exist, the method fails with MDLAPI_E_PARENT_SERVICE_NOT_EXIST. If the specified parent service is the current parent service, the method fails with MDLAPI_E_SERVICE_IS_ALREADY_PARENT.

Name

The Name property of the instance of OV_Service to be created. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is created as a new GUID. If a Service with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_SERVICE_NAME_EXISTS.

Description

The Description property of the instance of OV_Service to be created. Optional parameter.

Attributes

The Attributes array property of the instance of OV_Service to be created. Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a service (new instance of OV_Service).

Calculation and propagation rules are inherited from the service type specified with the ServiceTypeId parameter. If any of these properties are missing, nothing happens, and the method fails with MDLAPI_E_INVALID_PROPERTY. Default value for the message weight factor is 1.

Return Value

Instance of the newly created OV_Service.

Extended Status Codes

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_SERVICE_HIERPATH_EXISTS
: Service with the same Caption is already a child of the parent service.

MDLAPI_E_PARENT_SERVICE_NOT_EXIST
: Parent Service does not exist.

MDLAPI_E_SERVICETYPE_NOT_VALID
: Service type is not valid.

MDLAPI_E_SERVICE_NAME_EXISTS
: Service with the same Name already exists.

MDLAPI_E_SVC_TYPE_COMP_ASSOC_NOT_EXIST
: Service type composition does not exist.

MDLAPI_E_SERVICE_IS_ALREADY_PARENT
: Service already has a parent with the same Name.

MDLAPI_E_NODE_NOT_EXIST
: Node does not exist.

MDLAPI_E_NODEGROUP_NOT_EXIST
: Node Group does not exist.

MDLAPI_E_INVALID_PROPERTY
: Property is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::GetByHierarchicalPath()
OV_Service::GetByHierarchicalPath_Trans()

```
OV_Service GetByHierarchicalPath(
        [in] string Path)
```

```
OV_Service GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Path
: The service specified by the hierarchy Path. The Path consists of the Caption/Display Names of parent Services and a single backslash as a separator. The Path starts with a single backslash. If some of the Captions contain backslashes, you should escape them with additional backslashes (for example, "\\"). If the service with the specified Path does not exist, nothing happens, and the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the service (instance of OV_Service) specified by the hierarchical path.

Return Value

Instance of OV_Service specified by the hierarchical path.

Extended Status Codes

MDLAPI_E_SERVICE_NOT_EXIST
: Service does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::GetByName() OV_Service::GetByName_Trans()

```
OV_Service GetByName(
        [in] string Name)
```

```
OV_Service GetByName_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Name
The Name property of the instance of OV_Service to be returned. If the service with the specified Name does not exist, nothing happens, and the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the service (instance of OV_Service) specified with the Name property.

Return Value

Instance of OV_Service specified with the Name property.

Extended Status Codes

MDLAPI_E_SERVICE_NOT_EXIST
Service does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_Service::GetRoot()
OV_Service::GetRoot_Trans()

```
OV_Service GetRoot()
```

```
OV_Service GetRoot_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the root service (instance of OV_Service).

If the service with the Name property "Root_Services" does not exist, the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

Return Value

Root instance of OV_Service.

Extended Status Codes

MDLAPI_E_SERVICE_NOT_EXIST
: Service does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::GetServiceSet()
OV_Service::GetServiceSet_Trans()

```
sint32 GetServiceSet(
        [out] OV_Service Services[],
        [in, optional] string ServiceNames[],
        [in, optional] string ServicePaths[],
        [in, optional] string ParentServiceNames[],
        [in, optional] string ParentServicePaths[] )
```

```
sint32 GetServiceSet_Trans(
        [in] string TransId,
        [out] OV_Service Services[],
        [in, optional] string ServiceNames[],
        [in, optional] string ServicePaths[],
        [in, optional] string ParentServiceNames[],
        [in, optional] string ParentServicePaths[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Services
: Set of services (instances of OV_Service) without duplicates.

ServiceNames
: List of service names (Name property of OV_Service instance). Default is an empty list. If one of the services in the list does not exist, nothing happens, and the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

ServicePaths
: List of services specified with the hierarchical path. Default is an empty list. If one of the services in the list does not exist, nothing happens, and the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

ParentServiceNames
: List of parent service names (Name property of OV_Service instance). All services from this list and their children are found and included in a result list. Default is an empty list. If one of the services from this list does not exist, nothing happens, and the method fails with

MDLAPI_E_SERVICE_NOT_EXIST.

ParentServicePaths
: List of services specified with a hierarchical path. All services from this list and their children are found and included in a result list. Default is an empty list. If one of the services from this list does not exist, nothing happens, and the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the set of services (instances of OV_Service).

This method accepts different arrays (lists of services represented with IDs or hierarchical paths), and arranges them into service sets (service lists without duplicate instances). The method also verifies that service instances exist. If one of them does not exist, the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

Return Value

Number of services in out parameter Services.

Extended Status Codes

MDLAPI_E_SERVICE_NOT_EXIST
: Service does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::Remove()
OV_Service::Remove_Trans()

```
void Remove(
        [in] string Name)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Name
: The Name property of an instance of OV_Service to be removed. If a service with the specified Name does not exist, the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified service (instance of OV_Service).

Objects that are associated with this service are not removed. Only associations are removed.

CAUTION:
This method also removes all child services and their associations for the service specified with Name parameter.

NOTE:
Root service cannot be removed. Attempts fail with MDLAPI_E_SERVICE_IS_ROOT.

Return Value

None.

Extended Status Codes

MDLAPI_E_SERVICE_NOT_EXIST
: Service does not exist.

MDLAPI_E_SERVICE_IS_ROOT
: Specified operation cannot be executed for the root service.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service Instance Methods

This section contains the information on Instance methods for OV_Service.

OV_Service::AddAction()
OV_Service::AddAction_Trans()

```
boolean AddAction(
        [in] string ActionName)
```

```
boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ActionName
: The Name property of the instance of OV_Action to be added to this service. If tool does not exist, the method fails with MDLAPI_E_TOOL_NOT_EXIST.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds the tool (instance of OV_Action) to this service.

Return Value

False if the tool is already added to this service.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
: Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::AddAntecedentService()

OV_Service::AddAntecedentService_Trans()

```
boolean AddAntecedentService(
        [in] string AntecedentName,
        [in, optional] string PropRuleId,
        [in, optional] real32 WeightFactor,
        [in, optional] sint32 State)
```

```
boolean AddAntecedentService_Trans(
        [in] string TransId,
        [in] string AntecedentName,
        [in, optional] string PropRuleId,
        [in, optional] real32 WeightFactor,
        [in, optional] sint32 State)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

AntecedentName

The Name property of the instance of OV_Service on which this service depends. If the service does not exist, the method fails with MDLAPI_E_SERVICE_NOT_EXIST.

PropRuleId

The MsgPropRuleId property of this service dependency. If the propagation rule does not exist, the method fails with MDLAPI_E_PROPRULE_NOT_EXIST. Optional parameter. If this parameter is not set, the default propagation rule is created (MC365). This rule is created because there is no default rule from a Service Type relationship.

WeightFactor

The WeightFactor property value of this service dependency. Optional parameter. Default is 1.

State

The State property value of this service dependency. Optional parameter. Default is Empty.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds a dependency relationship between this service and the specified service (instance of OV_Service).

Before dependency is created, verify the following: - Service is already dependent on the specified service. - Specified service is the parent (not just immediately) of this service. - Dependency cycle is detected in the service model. If the service dependecy cycle is detected, the method fails with MDLAPI_E_SVC_DEP_CYCLE_DETECTED.

Return Value

False if the relationship already exists (this service is already dependent on the specified service).

Extended Status Codes

MDLAPI_E_SERVICE_NOT_EXIST	Service does not exist.
MDLAPI_E_SVC_DEP_CYCLE_DETECTED	Specified operation cannot be executed because a service dependency cycle has been detected.
MDLAPI_E_PROPRULE_NOT_EXIST	Message propagation rule does not exist.
MDLAPI_E_PROPRULE_CAPTION_EXIST	Propagation rule with the same Caption already exists.
MDLAPI_E_PROPRULE_SETTINGID_EXISTS	Propagation rule with the same SettingId already exists.
MDLAPI_E_TRANSACTION_NOT_EXIST	Transaction with the specified ID does not exist.

OV_Service::DeleteGraph()
OV_Service::DeleteGraph_Trans()

```
void DeleteGraph()

void DeleteGraph_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Deletes the graph properties of this service.

Return Value

None.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::DeleteReport() OV_Service::DeleteReport_Trans()

```
void DeleteReport()

void DeleteReport_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Deletes report properties of this service.

Return Value

None.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::GetActions()

OV_Service::GetActions_Trans()

```
sint32 GetActions(
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeDefault)
```

```
sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeDefault)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Actions
: Instances of OV_Action that are added to this service. An instance is valid as a returned out parameter only if the method does not fail.

IncludeDefault
: Defines whether Actions also contains instances of OV_Action that are included from the service type (default tools). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of tools (instances of OV_Action) added to this service.

Return Value

Number of tools in out parameter Actions.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::GetAntecedentServices()
OV_Service::GetAntecedentServices_Trans()

```
sint32 GetAntecedentServices(
        [out] OV_Service AntecedentServices[],
        [in, optional] boolean IncludeAllAntecedent)
```

```
sint32 GetAntecedentServices_Trans(
        [in] string TransId,
        [out] OV_Service AntecedentServices[],
        [in, optional] boolean IncludeAllAntecedent)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

AntecedentServices

Instances of OV_Service on which the specified service has dependencies. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllAntecedent

This flag indicates which antecedent services to include. Optional parameter. Default value is false (only immediately antecedent services are included).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) on which this service is dependent.

Return Value

Number of services in the out parameter AntecedentServices.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::GetCalculationRule()
OV_Service::GetCalculationRule_Trans()

```
OV_CalculationRule GetCalculationRule()
```

```
OV_CalculationRule GetCalculationRule_Trans(
          [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the calculation rule (instance of OV_CalculationRule) of this service.

Return Value

Instance of OV_CalculationRule of this service.

Extended Status Codes

MDLAPI_E_CALCRULE_NOT_EXIST
: Calculation rule does not exist.

MDLAPI_E_INVALID_PROPERTY
: Property is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::GetChildServices()
OV_Service::GetChildServices_Trans()

```
sint32 GetChildServices(
        [out] OV_Service Services[],
        [in, optional] boolean IncludeAllSubordinate)
```

```
sint32 GetChildServices_Trans(
        [in] string TransId,
        [out] OV_Service Services[],
        [in, optional] boolean IncludeAllSubordinate)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Services
: Instances of OV_Service that are children of this service. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllSubordinate
: This flag indicates which child services to include. Optional parameter. Default value is false (only immediately subordinate services are included).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) that are children of this service.

Return Value

Number of services (children) in out parameter Services.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::GetCompositionPropagationRule()
OV_Service::GetCompositionPropagationRule_Trans()

```
OV_PropagationRule GetCompositionPropagationRule(
        [in] string ChildService)
```

```
OV_PropagationRule GetCompositionPropagationRule_Trans(
        [in] string TransId,
        [in] string ChildService)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ChildService
: The Name property of the service (instance of OV_Service) that is the child of this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the propagation rule (instance of OV_PropagationRule) for the service that is the child of this service.

If the service composition does not exist, the method fails with MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST.
If the new propagation rule does not exist, the method returns MDLAPI_E_PROPRULE_NOT_EXIST.

Return Value

Instance of OV_PropagationRule of this service composition.

Extended Status Codes

MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST
: Service composition does not exist.

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::GetCompositionWeightFactor()
OV_Service::GetCompositionWeightFactor_Trans()

```
real32 GetCompositionWeightFactor(
        [in] string ChildService)
```

```
real32 GetCompositionWeightFactor_Trans(
        [in] string TransId,
        [in] string ChildService)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ChildService
: The Name property of the service (instance of OV_Service) that is the child of this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the weighting factor for the propagation between this service and the child service.

If the service composition does not exist, the method fails with MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST.
If the propagation rule value does not exist, the method returns MDLAPI_E_PROPERTY_MISSING.

Return Value

Message weight factor of this service composition.

Extended Status Codes

MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST
: Service composition does not exist.

MDLAPI_E_PROPERTY_MISSING
: Property is not set.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::GetDependencyPropagationRule()
OV_Service::GetDependencyPropagationRule_Trans()

```
OV_PropagationRule GetDependencyPropagationRule(
        [in] string AntecedentService)
```

```
OV_PropagationRule GetDependencyPropagationRule_Trans(
        [in] string TransId,
        [in] string AntecedentService)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentService
: The Name property of the service (instance of OV_Service) on which this service is dependent.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the propagation rule (instance of OV_PropagationRule) for the service on which this service is dependent.

If the service dependency does not exist, the method fails with MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST. If the new propagation rule does not exist, the method returns MDLAPI_E_PROPRULE_NOT_EXIST.

Return Value

Instance of OV_PropagationRule of this service dependency.

Extended Status Codes

MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST
: Service dependency does not exist.

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::GetDependencyWeightFactor()
OV_Service::GetDependencyWeightFactor_Trans()

```
real32 GetDependencyWeightFactor(
        [in] string AntecedentService)
```

```
real32 GetDependencyWeightFactor_Trans(
        [in] string TransId,
        [in] string AntecedentService)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentService
: The Name property of the service (instance of OV_Service) on which this service is dependent.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the weighting factor for propagation between this service and the antecedent service.

If the service dependency does not exist, the method fails with MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST. If the message weight factor value does not exist, the method returns MDLAPI_E_PROPERTY_MISSING.

Return Value

Message weight factor of this service.

Extended Status Codes

MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST
: Service dependency does not exist.

MDLAPI_E_PROPERTY_MISSING
: Property is not set.

MDLAPI_E_TRANSACTION_NOT_EXIST

```
Transaction with the specified ID does not exist.
```

OV_Service::GetDependentServices()

OV_Service::GetDependentServices_Trans()

```
sint32 GetDependentServices(
        [out] OV_Service DependentServices[],
        [in, optional] boolean IncludeAllDependent)
```

```
sint32 GetDependentServices_Trans(
        [in] string TransId,
        [out] OV_Service DependentServices[],
        [in, optional] boolean IncludeAllDependent)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

DependentServices

Instances of OV_Service that are dependent on this service. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllDependent

This flag indicates which dependent services to include. Optional parameter. Default value is false (only immediately subordinate services are included).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services (instances of OV_Service) that are dependent on this service.

Return Value

Number of services in the out parameter DependentServices.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::GetHostNode()
OV_Service::GetHostNode_Trans()

```
boolean GetHostNode(
        [out] OV_ManagedNode Node)
```

```
boolean GetHostNode_Trans(
        [in] string TransId,
        [out] OV_ManagedNode Node)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Node
Instance of OV_MamagedNode on which this service is hosted.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns false if this service is not hosted on the node, or if it is hosted on another instance (for example, on another node group).

If the node is hosted on a node that cannot be found, the method returns MDLAPI_E_NODE_NOT_EXIST.

Return Value

False if this service is not hosted on a node, or if it is hosted on another instance.

Extended Status Codes

MDLAPI_E_NODE_NOT_EXIST
Node does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_Service::GetHostNodeGroup()
OV_Service::GetHostNodeGroup_Trans()

```
boolean GetHostNodeGroup(
        [out] OV_NodeGroup NodeGroup)
```

```
boolean GetHostNodeGroup_Trans(
        [in] string TransId,
        [out] OV_NodeGroup NodeGroup)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

NodeGroup
    Instance of OV_NodeGroup on which this service is hosted.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns false if this service is not hosted on the specified node group, or if it is hosted on another instance (for example, on another node).

If the node is hosted on a node group that cannot be found, the method returns MDLAPI_E_NODEGROUP_NOT_EXIST.

Return Value

False if this service is not hosted on a node group, or if it is hosted on another instance.

Extended Status Codes

MDLAPI_E_NODEGROUP_NOT_EXIST
    Node Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_Service::GetMessagePropagationRule()
OV_Service::GetMessagePropagationRule_Trans()

```
OV_PropagationRule GetMessagePropagationRule()

OV_PropagationRule GetMessagePropagationRule_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the message propagation rule (instance of OV_PropagationRule) of this service.

Return Value

Instance of OV_PropagationRule of this service.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_INVALID_PROPERTY
: Property is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::GetMessageWeightFactor()
OV_Service::GetMessageWeightFactor_Trans()

```
real32 GetMessageWeightFactor()

real32 GetMessageWeightFactor_Trans(
          [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the message weight factor of this service.

Return Value

Message weight factor of this service.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING
: Property is not set.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::GetParents()

OV_Service::GetParents_Trans()

```
sint32 GetParents(
        [out] OV_Service Services[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

```
sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_Service Services[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Services

Instances of OV_Service that are the parents of this service. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllHierarchicalParents

Defines whether Services also include instances of OV_Service that are hierarchical parents (recursive until the root service), rather than direct parents. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services for which this service is a child.

Return Value

Number of services (parents) in out parameter Services.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

```
Transaction with the specified ID does not exist.
```

OV_Service::GetServiceType()
OV_Service::GetServiceType_Trans()

```
OV_ServiceTypeDefinition GetServiceType()

OV_ServiceTypeDefinition GetServiceType_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the service type (instance of OV_ServiceTypeDefinition) of this service.

Return Value

Instance of OV_ServiceTypeDefinition that have the service type of this service.

Extended Status Codes

MDLAPI_E_SERVICETYPE_NOT_EXIST
: Service type does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::HasAction()
OV_Service::HasAction_Trans()

```
boolean HasAction(
        [in] string ActionName,
        [in, optional] boolean IncludeDefault)
```

```
boolean HasAction_Trans(
        [in] string TransId,
        [in] string ActionName,
        [in, optional] boolean IncludeDefault)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ActionName
    The Name property of the tool (instance of OV_Action), which is used to verify that the tool is associated with this service.

IncludeDefault
    Defines whether the tool to find can also be included from the service type (default tools). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified tool (instance of OV_Action) is associated with this service.

Return Value

False if the tool is already added to this service.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::HasAntecedentService()
OV_Service::HasAntecedentService_Trans()

```
boolean HasAntecedentService(
          [in] string AntecedentName,
          [in, optional] boolean IncludeAllAntecedent)
```

```
boolean HasAntecedentService_Trans(
          [in] string TransId,
          [in] string AntecedentName,
          [in, optional] boolean IncludeAllAntecedent)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentName
: The Name property of the antecedent service (instance of OV_Service) on which this service depends.

IncludeAllAntecedent
: This flag indicates which antecedent services to include. Optional parameter. Default value is false (only immediately antecedent services are included).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if this service is dependent on the specified service (instance of OV_Service).

Return Value

True if the service (instance of OV_Service) specified by a parameter ServiceName is the antecedent service on which this service depends.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

```
Transaction with the specified ID does not exist.
```

OV_Service::HasChildService()
OV_Service::HasChildService_Trans()

```
boolean HasChildService(
        [in] string ServiceName,
        [in, optional] boolean IncludeAllSubordinate)
```

```
boolean HasChildService_Trans(
        [in] string TransId,
        [in] string ServiceName,
        [in, optional] boolean IncludeAllSubordinate)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ServiceName

The Name property of the service (instance of OV_Service) used to verify that the service is a child of this service.

IncludeAllSubordinate

This flag indicates how to search the specified child service. Optional parameter. Default value is false (only immediately subordinate services are searched).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified service (instance of OV_Service) is a child of this service. Only immediately subordinate services are checked.

Return Value

True if the service (instance of OV_Service) specified by the ServiceName parameter is a child of this service.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::HasDependentService()
OV_Service::HasDependentService_Trans()

```
boolean HasDependentService(
        [in] string DependentName,
        [in, optional] boolean IncludeAllDependent)
```

```
boolean HasDependentService_Trans(
        [in] string TransId,
        [in] string DependentName,
        [in, optional] boolean IncludeAllDependent)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

DependentName
: The Name property of the service (instance of OV_Service) used to verify that the specified service is dependent on this service.

IncludeAllDependent
: This flag indicates which dependent services to include. Optional parameter. Default value is false (only the immediately subordinate services are searched).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified service (instance of OV_Service) is dependent on this service.

Return Value

True if the specified service (instance of OV_Service) is dependent on this service.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::IsChildOf()
OV_Service::IsChildOf_Trans()

```
boolean IsChildOf(
        [in] string ParentName)
```

```
boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentName
: The Name property of the service (instance of OV_Service) to check if the service is the parent of this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified service (instance of OV_Service) is the parent of this service. Only immediate child services are checked.

Return Value

True if the service (instance of OV_Service) specified by a parameter of ParentName is the parent of this service.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::IsHostedOnNode() OV_Service::IsHostedOnNode_Trans()

```
boolean IsHostedOnNode(
        [in] string NodeName)
```

```
boolean IsHostedOnNode_Trans(
        [in] string TransId,
        [in] string NodeName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeName
: The Name property of the node (instance of OV_ManagedNode)that is used to verify that the service is hosted on the specified node.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if this service is hosted on the specified node.

Return Value

True if this service is hosted on the specified node.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::IsHostedOnNodeGroup()
OV_Service::IsHostedOnNodeGroup_Trans()

```
boolean IsHostedOnNodeGroup(
        [in] string NodeGroupName)
```

```
boolean IsHostedOnNodeGroup_Trans(
        [in] string TransId,
        [in] string NodeGroupName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

NodeGroupName
: The Name property of the node group (instance of OV_NodeGroup), which is used to verify that this service is hosted on the specified node group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if this service is hosted on the specified node group.

Return Value

True if this service is hosted on the specified node group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::Modify()
OV_Service::Modify_Trans()

```
void Modify(
        [in] OV_Service Service)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_Service Service)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Service
: Modified instance of OV_Service to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes performed to one or more properties.

It is not possible to get an instance of OV_Service to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, an instance of OV_Service is specified as the Service parameter, even though this method is already called in the context of this instance.

As with the OV_ManagedNode::Create method, this method checks the following:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If the Caption property is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.
- If a property has an invalid value, the method fails with MDLAPI_E_INVALID_PROPERTY.
- If a service with the same hierarchical path already exists, the method fails with MDLAPI_E_SERVICE_HIERPATH_EXISTS.

- If graph properties validation fails, the method fails with MDLAPI_E_GRAPH_PROP_NOT_VALID.
- If report properties validation fails, the method fails with MDLAPI_E_REPORT_PROP_NOT_VALID.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING	Property is not set.
MDLAPI_E_INVALID_CAPTION	Specified object Caption parameter is not valid.
MDLAPI_E_INVALID_PROPERTY	Property is not valid.
MDLAPI_E_SERVICE_HIERPATH_EXISTS	Service with the same Caption is already a child of the parent service.
MDLAPI_E_INVALID_CONFIG_VALUE	Configuration value is not valid.
MDLAPI_E_NO_REPORT_INTEGRATION	Reporter integration tool is not installed.
MDLAPI_E_NO_GRAPH_INTEGRATION	Performance Manager integration tool is not installed.
MDLAPI_E_GRAPH_PROP_NOT_VALID	Graph properties (GraphFamily and GraphCategory) are not valid.
MDLAPI_E_REPORT_PROP_NOT_VALID	Report properties (ReportFamily and ReportCategory) are not valid.
MDLAPI_E_TRANSACTION_NOT_EXIST	Transaction with the specified ID does not exist.

OV_Service::RemoveAction()

OV_Service::RemoveAction_Trans()

```
void RemoveAction(
        [in] string ActionName)
```

```
void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ActionName
: The Name property of the instance of OV_Action to be removed from this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes the tool (instance of OV_Action) from this service.

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
: Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::RemoveAntecedentService()
OV_Service::RemoveAntecedentService_Trans()

```
void RemoveAntecedentService(
        [in] string AntecedentName)
```

```
void RemoveAntecedentService_Trans(
        [in] string TransId,
        [in] string AntecedentName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

AntecedentName
    The Name property of the instance of OV_Service on which this service depends.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes a dependency relationship between this service and the specified service (instance of OV_Service).

Return Value

None.

Extended Status Codes

MDLAPI_E_SERVICE_NOT_EXIST
    Service does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_Service::SetCalculationRule()
OV_Service::SetCalculationRule_Trans()

```
void SetCalculationRule(
        [in] string CalcRuleId)
```

```
void SetCalculationRule_Trans(
        [in] string TransId,
        [in] string CalcRuleId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

CalcRuleId
: New CalcRuleId property value of this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the calculation rule (instance of OV_CalculationRule) of this service.

Return Value

None.

Extended Status Codes

MDLAPI_E_CALCRULE_NOT_EXIST
: Calculation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::SetCompositionPropagationRule()
OV_Service::SetCompositionPropagationRule_Trans()

```
void SetCompositionPropagationRule(
        [in] string ChildService,
        [in] string PropRuleId)
```

```
void SetCompositionPropagationRule_Trans(
        [in] string TransId,
        [in] string ChildService,
        [in] string PropRuleId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ChildService
: The Name property of the service (instance of OV_Service) that is the child of this service.

PropRuleId
: New PropRuleId property value of the service composition.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the propagation rule (instance of OV_PropagationRule) for the service that is the child of this service.

If the service composition does not exist, the method fails with MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST.
If the new propagation rule does not exist, the method returns MDLAPI_E_PROPRULE_NOT_EXIST.

Return Value

None.

Extended Status Codes

MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST
: Service composition does not exist.

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::SetCompositionWeightFactor()

OV_Service::SetCompositionWeightFactor_Trans()

```
void SetCompositionWeightFactor(
        [in] string ChildService,
        [in] real32 MsgWeightFactor)
```

```
void SetCompositionWeightFactor_Trans(
        [in] string TransId,
        [in] string ChildService,
        [in] real32 MsgWeightFactor)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ChildService
: The Name property of the service (instance of OV_Service) that is the child of this service.

MsgWeightFactor
: New WeightFactor property value of this service composition.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the weighting factor for the propagation between this service and the child service.

If the new message weight factor value is not valid, the method returns MDLAPI_E_INVALID_PARAMETER. If the service composition does not exist, the method fails with MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST.

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_SVC_COMP_ASSOC_NOT_EXIST
: Service composition does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::SetDependencyPropagationRule()
OV_Service::SetDependencyPropagationRule_Trans()

```
void SetDependencyPropagationRule(
        [in] string AntecedentService,
        [in] string PropRuleId)
```

```
void SetDependencyPropagationRule_Trans(
        [in] string TransId,
        [in] string AntecedentService,
        [in] string PropRuleId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentService
: The Name property of the service (instance of OV_Service) on which this service is dependent.

PropRuleId
: New PropRuleId property value of the service dependency.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the propagation rule (instance of OV_PropagationRule) for the service on which this service is dependent.

If the service dependency does not exist, the method fails with MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST. If the new propagation rule does not exist, the method returns MDLAPI_E_PROPRULE_NOT_EXIST.

Return Value

None.

Extended Status Codes

MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST
: Service dependency does not exist.

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::SetDependencyWeightFactor()
OV_Service::SetDependencyWeightFactor_Trans()

```
void SetDependencyWeightFactor(
        [in] string AntecedentService,
        [in] real32 MsgWeightFactor)
```

```
void SetDependencyWeightFactor_Trans(
        [in] string TransId,
        [in] string AntecedentService,
        [in] real32 MsgWeightFactor)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentService
: The Name property of the service (instance of OV_Service) on which this service is dependent.

MsgWeightFactor
: New WeightFactor property value of this service dependency.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the weighting factor for propagation between this service and the antecedent service.

If the new message weight factor value is not valid, the method returns MDLAPI_E_INVALID_PARAMETER. If the service dependency does not exist, the method fails with MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST.

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_SVC_DEP_ASSOC_NOT_EXIST
: Service dependency does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::SetGraph()
OV_Service::SetGraph_Trans()

```
void SetGraph(
        [in, optional] string GraphFamily,
        [in] string GraphCategory)
```

```
void SetGraph_Trans(
        [in] string TransId,
        [in, optional] string GraphFamily,
        [in] string GraphCategory)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

GraphFamily
: New GraphFamily property value of this service. Optional parameter.

GraphCategory
: New GraphCategory property value of this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds graph properties to this service.

If the Performance Manager integration tool is not installed, the method returns MDLAPI_E_NO_GRAPH_INTEGRATION. Graph family and graph category must be valid. Otherwise, the method fails with MDLAPI_E_GRAPH_PROP_NOT_VALID. If a graph category is selected, the graph family is selected automatically.

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_CONFIG_VALUE

Configuration value is not valid.

MDLAPI_E_NO_GRAPH_INTEGRATION

Performance Manager integration tool is not installed.

MDLAPI_E_GRAPH_PROP_NOT_VALID

Graph properties (GraphFamily and GraphCategory) are not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Service::SetMessagePropagationRule()
OV_Service::SetMessagePropagationRule_Trans()

```
void SetMessagePropagationRule(
        [in] string MsgPropRuleId)
```

```
void SetMessagePropagationRule_Trans(
        [in] string TransId,
        [in] string MsgPropRuleId)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

MsgPropRuleId
    New MsgPropRuleId property value of this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the message propagation rule (instance of OV_PropagationRule) of this service.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
    Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_Service::SetMessageWeightFactor()
OV_Service::SetMessageWeightFactor_Trans()

```
void SetMessageWeightFactor(
        [in] real32 MsgWeightFactor)
```

```
void SetMessageWeightFactor_Trans(
        [in] string TransId,
        [in] real32 MsgWeightFactor)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

MsgWeightFactor
    New MsgWeightFactor property value of this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the message weight factor of this service.

If the new message weight factor value is not valid, the method returns MDLAPI_E_INVALID_PARAMETER.

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
    Parameter is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_Service::SetOutage()
OV_Service::SetOutage_Trans()

```
void SetOutage(
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage,
        [in, optional] boolean IncludeAllSubordinate)
```

```
void SetOutage_Trans(
        [in] string TransId,
        [in] sint32 IsInOutage,
        [in] boolean IsScheduled,
        [in, optional] sint32 DeleteMessageInOutage,
        [in, optional] boolean IncludeAllSubordinate)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

IsInOutage
: Changes or sets the service mode (scheduled outage or maintenance). Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. KEEP does not change the service outage mode, ON enables and OFF disables the service outage mode, and TOGGLE changes the service outage mode from ON to OFF, or from OFF to ON. If this parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

IsScheduled
: Flag for scheduled outage or maintenance mode. If this parameter is set to true, the service is set to scheduled outage mode. Otherwise (false), the service is set to unplanned maintenance mode.

DeleteMessageInOutage
: Flag that indicates whether node messages should be removed. Valid enumeration type values are KEEP, ON, OFF, or TOGGLE. Default is KEEP. If the parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

IncludeAllSubordinate
: This flag indicates that child services are included. Optional parameter. Default value is false. If this

parameter has an invalid value, the method fails with MDLAPI_E_INVALID_PARAMETER.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets scheduled outage/maintenance mode properties of this service.

If the service is hosted on a node group, the scheduled outage and maintenance mode are not set for the service. Only administrators and operators with special rights can execute this method. Otherwise the method fails with MDLAPI_E_NO_OPERATION_RIGHTS

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_PARAMETER
: Parameter is not valid.

MDLAPI_E_NO_OPERATION_RIGHTS
: User has no rights to execute this operation.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Service::SetReport()

OV_Service::SetReport_Trans()

```
void SetReport(
        [in, optional] string ReportFamily,
        [in] string ReportCategory)
```

```
void SetReport_Trans(
        [in] string TransId,
        [in, optional] string ReportFamily,
        [in] string ReportCategory)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ReportFamily
: New ReportFamily property value of this service. Optional parameter.

ReportCategory
: New ReportCategory property value of this service.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds report properties to this service.

If the Reporter integration tool is not installed, the method returns MDLAPI_E_NO_REPORT_INTEGRATION. Report family and report category must be valid. Otherwise, the method fails with MDLAPI_E_REPORT_PROP_NOT_VALID. If a report category is selected, the report family is selected automatically.

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_CONFIG_VALUE
Configuration value is not valid.

MDLAPI_E_NO_REPORT_INTEGRATION
Reporter integration tool is not installed.

MDLAPI_E_REPORT_PROP_NOT_VALID
Report properties (ReportFamily and ReportCategory) are not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition

Description

This is the definitional component of a Service. It describes the format for the name (key) and caption. It is also the anchor point for the default calculation rules and message propagation rules for the service instances created from this type definition. The version qualifier is set by whoever creates an instance of OV_ServiceTypeDefinition. The version provides the discovery agent with the version of the definitional information.

```
class OV_ServiceTypeDefinition
{
        Properties:
        string GUID;
        string CalcRuleId;
        string KeyFormat;
        string CaptionFormat;
        string DescriptionFormat;
        string Icon;
        string MsgPropRuleId;
        string ManagementModuleId;
        string ReportFamily;
        string ReportCategory;
        string GraphFamily;
        string GraphCategory;
        string Version;

        Class Methods:

        OV_ServiceTypeDefinition GetRoot(
                );

        OV_ServiceTypeDefinition GetRoot_Trans(
                [in] string TransId
                );

        OV_ServiceTypeDefinition GetById(
                [in] string GUID
                );

        OV_ServiceTypeDefinition GetById_Trans(
                [in] string TransId,
                [in] string GUID
                );
```

```
OV_ServiceTypeDefinition GetByHierarchicalPath(
        [in] string Path
        );

OV_ServiceTypeDefinition GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path
        );
```

Instance Methods:

```
sint32 GetParents(
        [out] OV_ServiceTypeDefinition ServiceTypes[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ServiceTypes[],
        [in, optional] boolean IncludeAllHierarchicalParents
        );

boolean IsChildOf(
        [in] string ParentGUID
        );

boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentGUID
        );

sint32 GetChildServiceTypes(
        [out] OV_ServiceTypeDefinition ServiceTypes[],
        [in, optional] boolean IncludeAllSubordinate
        );

sint32 GetChildServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ServiceTypes[],
        [in, optional] boolean IncludeAllSubordinate
        );

boolean HasChildServiceType(
        [in] string ChildGUID,
        [in, optional] boolean IncludeAllSubordinate
        );

boolean HasChildServiceType_Trans(
        [in] string TransId,
        [in] string ChildGUID,
        [in, optional] boolean IncludeAllSubordinate
        );

sint32 GetAntecedentServiceTypes(
        [out] OV_ServiceTypeDefinition AntecedentServiceTypes[],
        [in, optional] boolean IncludeAllAntecedent
        );

sint32 GetAntecedentServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition AntecedentServiceTypes[],
        [in, optional] boolean IncludeAllAntecedent
        );

boolean HasAntecedentServiceType(
        [in] string AntecedentGUID,
        [in, optional] boolean IncludeAllAntecedent
        );

boolean HasAntecedentServiceType_Trans(
        [in] string TransId,
        [in] string AntecedentGUID,
        [in, optional] boolean IncludeAllAntecedent
        );

boolean AddAntecedentServiceType(
        [in] string AntecedentGUID,
        [in] string PropRuleId
        );

boolean AddAntecedentServiceType_Trans(
        [in] string TransId,
        [in] string AntecedentGUID,
        [in] string PropRuleId
        );

void RemoveAntecedentServiceType(
        [in] string AntecedentGUID
        );

void RemoveAntecedentServiceType_Trans(
        [in] string TransId,
        [in] string AntecedentGUID
        );

sint32 GetDependentServiceTypes(
        [out] OV_ServiceTypeDefinition DependentServiceTypes[],
        [in, optional] boolean IncludeAllDependent
        );
```

```
sint32 GetDependentServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition DependentServiceTypes[],
        [in, optional] boolean IncludeAllDependent
        );

boolean HasDependentServiceType(
        [in] string DependentGUID,
        [in, optional] boolean IncludeAllDependent
        );

boolean HasDependentServiceType_Trans(
        [in] string TransId,
        [in] string DependentGUID,
        [in, optional] boolean IncludeAllDependent
        );

boolean AddAction(
        [in] string ActionName
        );

boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

void RemoveAction(
        [in] string ActionName
        );

void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

sint32 GetActions(
        [out] OV_Action Actions[]
        );

sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[]
        );

boolean HasAction(
        [in] string ActionName
        );

boolean HasAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

boolean AddAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName
        );

boolean AddAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName
        );

void RemoveAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName
        );

void RemoveAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName
        );

sint32 GetAutoDeployPolicyGroups(
        [out] OV_AutoDeployPolicyGroup AutoDeployPolicyGroups[]
        );

sint32 GetAutoDeployPolicyGroups_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPolicyGroup AutoDeployPolicyGroups[]
        );

boolean HasAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName
        );

boolean HasAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName
        );

boolean AddAutoDeployPackage(
        [in] string AutoDeployPackageName,
        [in, optional] string Attributes[]
        );

boolean AddAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName,
        [in, optional] string Attributes[]
        );
```

```
void RemoveAutoDeployPackage(
        [in] string AutoDeployPackageName
        );

void RemoveAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName
        );

sint32 GetAutoDeployPackages(
        [out] OV_AutoDeployPackages AutoDeployPackages[]
        );

sint32 GetAutoDeployPackages_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPackages AutoDeployPackages[]
        );

boolean HasAutoDeployPackage(
        [in] string AutoDeployPackageName
        );

boolean HasAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName
        );

void SetReport(
        [in, optional] string ReportFamily,
        [in] string ReportCategory
        );

void SetReport_Trans(
        [in] string TransId,
        [in, optional] string ReportFamily,
        [in] string ReportCategory
        );

void DeleteReport(
        );

void DeleteReport_Trans(
        [in] string TransId
        );

void SetGraph(
        [in, optional] string GraphFamily,
        [in] string GraphCategory
        );
```

```
void SetGraph_Trans(
        [in] string TransId,
        [in, optional] string GraphFamily,
        [in] string GraphCategory
        );

void DeleteGraph(
        );

void DeleteGraph_Trans(
        [in] string TransId
        );

void SetCalculationRule(
        [in] string CalcRuleId
        );

void SetCalculationRule_Trans(
        [in] string TransId,
        [in] string CalcRuleId
        );

OV_CalculationRule GetCalculationRule(
        );

OV_CalculationRule GetCalculationRule_Trans(
        [in] string TransId
        );

void SetMessagePropagationRule(
        [in] string MsgPropRuleId
        );

void SetMessagePropagationRule_Trans(
        [in] string TransId,
        [in] string MsgPropRuleId
        );

OV_PropagationRule GetMessagePropagationRule(
        );

OV_PropagationRule GetMessagePropagationRule_Trans(
        [in] string TransId
        );

void SetDependencyPropagationRule(
        [in] string AntecedentServiceType,
        [in] string PropRuleId
        );

void SetDependencyPropagationRule_Trans(
                [in] string TransId,
                [in] string AntecedentServiceType,
                [in] string PropRuleId
                );

        OV_PropagationRule GetDependencyPropagationRule(
                [in] string AntecedentServiceType
                );

        OV_PropagationRule GetDependencyPropagationRule_Trans(
                [in] string TransId,
                [in] string AntecedentServiceType
                );

        void SetCompositionPropagationRule(
                [in] string ChildServiceType,
                [in] string PropRuleId
                );

        void SetCompositionPropagationRule_Trans(
                [in] string TransId,
                [in] string ChildServiceType,
                [in] string PropRuleId
                );

        OV_PropagationRule GetCompositionPropagationRule(
                [in] string ChildServiceType
                );

        OV_PropagationRule GetCompositionPropagationRule_Trans(
                [in] string TransId,
                [in] string ChildServiceType
                );
    };
```

OV_ServiceTypeDefinition-Properties

GUID

Signature

string GUID

Description

A GUID to uniquely identify a service type definition.

CalcRuleId

Signature

string CalcRuleId

Description

SettingId of the service type definition OV_CalculationRule.

KeyFormat

Signature

string KeyFormat

Description

The format map used to build a key for service objects.

CaptionFormat

Signature

string CaptionFormat

Description

The format map used to build the Caption property for Service objects. This string takes the format of a literal string and substitution variables. Variables are delineated by dollar signs ('$'). Example: Oracle_$InstanceID$, where 'Oracle_' is a literal string, and where '$InstanceID$' is substituted with the instance ID discovered by the discovery agent for Oracle.

DescriptionFormat

Signature

string DescriptionFormat

Description

The format map used to build the description property for Service objects. This string takes the format of a literal string and substitution variables. Variables are delineated by dollar signs ('$'). Example: Oracle_$InstanceID$, where 'Oracle_' is a literal string, and where '$InstanceID$' is substituted with the instance ID discovered by the discovery agent for Oracle.

Icon

Signature

string Icon

Description

The default icon for all services instantiated from this service type definition. If the property is blank, the default Service icon is used.

MsgPropRuleId

Signature

string MsgPropRuleId

Description

SettingId of the service type definition message OV_PropagationRule.

ManagementModuleId

Signature

string ManagementModuleId

Description

The Name of the management module with which this service type definition is associated.

ReportFamily

Signature

string ReportFamily

Description

The default Report family to display when a service of this type is the context for displaying the list of available reports.

ReportCategory

Signature

string ReportCategory

Description

The default Report Category within the Report Family to display when a service instance of this type is the context for displaying the list of available reports. If this property is blank, the entire family displays.

GraphFamily

Signature

string GraphFamily

Description

The default Graph family to display when a service of this type is the context for displaying the list of available graphs.

GraphCategory

Signature

string GraphCategory

Description

The default Graph Category within the Graph Family to display when a service instance of this type is the context for displaying the list of available graphs. If this property is blank, the entire family displays.

Version

Signature

string Version

Description

This field is being replaced by the Version qualifier. It is still listed as a property during the transition. DO NOT USE THIS PROPERTY.

OV_ServiceTypeDefinition Class Methods

This section contains the information on Class methods for OV_ServiceTypeDefinition.

OV_ServiceTypeDefinition::GetByHierarchicalPath()
OV_ServiceTypeDefinition::GetByHierarchicalPath_Trans()

```
OV_ServiceTypeDefinition GetByHierarchicalPath(
        [in] string Path)
```

```
OV_ServiceTypeDefinition GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Path
: The service type specified by the hierarchy Path. The Path consists of the Caption/Display Names of the parent Service types and a single backslash as a separator. The Path starts with a single backslash. If some of the Captions contain backslashes, you should escape them with additional backslashes (for example, "\\"). If the service type with the Path does not exist, nothing happens, and the method fails with MDLAPI_E_SERVICETYPE_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the service type (instance of OV_ServiceTypeDefinition) specified by the hierarchical path.

Return Value

Instance of OV_ServiceTypeDefinition specified by the hierarchical path.

Extended Status Codes

MDLAPI_E_SERVICETYPE_NOT_EXIST
: Service type does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetById()

OV_ServiceTypeDefinition::GetById_Trans()

```
OV_ServiceTypeDefinition GetById(
        [in] string GUID)
```

```
OV_ServiceTypeDefinition GetById_Trans(
        [in] string TransId,
        [in] string GUID)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

GUID
: The GUID property of the instance of OV_ServiceTypeDefinition to be returned. If the service type with the GUID does not exist, nothing happens, and the method fails with MDLAPI_E_SERVICETYPE_NOT_EXIST.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the service type (instance of OV_ServiceTypeDefinition) specified by the GUID property.

Return Value

Instance of OV_ServiceTypeDefinition specified with the ServiceTypeGUID property.

Extended Status Codes

MDLAPI_E_SERVICETYPE_NOT_EXIST
: Service type does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetRoot()
OV_ServiceTypeDefinition::GetRoot_Trans()

```
OV_ServiceTypeDefinition GetRoot()

OV_ServiceTypeDefinition GetRoot_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the root service type (instance of OV_ServiceTypeDefinition).

If the service type with the GUID property "root" does not exist, the method fails with MDLAPI_E_SERVICETYPE_NOT_EXIST.

Return Value

Root instance of OV_ServiceTypeDefinition.

Extended Status Codes

MDLAPI_E_SERVICETYPE_NOT_EXIST
: Service type does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition Instance Methods

This section contains the information on Instance methods for OV_ServiceTypeDefinition.

OV_ServiceTypeDefinition::AddAction()
OV_ServiceTypeDefinition::AddAction_Trans()

```
boolean AddAction(
        [in] string ActionName)
```

```
boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ActionName
: The Name property of the instance of OV_Action to be added to this service type. If the tool does not exist, the method fails with MDLAPI_E_TOOL_NOT_EXIST.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds the tool (instance of OV_Action) to this service type.

Return Value

False if the tool is already added to this service type.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
: Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::AddAntecedentServiceType() OV_ServiceTypeDefinition::AddAntecedentServiceType_Trans()

```
boolean AddAntecedentServiceType(
        [in] string AntecedentGUID,
        [in] string PropRuleId)
```

```
boolean AddAntecedentServiceType_Trans(
        [in] string TransId,
        [in] string AntecedentGUID,
        [in] string PropRuleId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentGUID
: The GUID property of the instance of OV_ServiceTypeDefinition on which this service type depends. If the service type does not exist, the method fails with MDLAPI_E_SERVICETYPE_NOT_EXIST.

PropRuleId
: The MsgPropRuleId property of this service type dependency.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds a dependency relationship between this service type and the specified service type (instance of OV_ServiceTypeDefinition).

Before dependency is created, verify the following: - This service type is already dependent on the specified service type. - The specified service type is the parent (not just immediately) of this service type. - The dependency cycle is detected in the service type model. If the service type dependency cycle is detected, the method fails with MDLAPI_E_SVC_TYPE_DEP_CYCLE_DETECTED..

Return Value

False if the relationship already exists (this service type is already dependent on the specified service type).

Extended Status Codes

MDLAPI_E_SERVICETYPE_NOT_EXIST
: Service type does not exist.

MDLAPI_E_SVC_TYPE_DEP_CYCLE_DETECTED
: Specified operation cannot be executed because a service type dependency cycle has been detected.

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::AddAutoDeployPackage()
OV_ServiceTypeDefinition::AddAutoDeployPackage_Trans()

```
boolean AddAutoDeployPackage(
        [in] string AutoDeployPackageName,
        [in, optional] string Attributes[] )
```

```
boolean AddAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName,
        [in, optional] string Attributes[] )
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPackageName
: The Name property of the instance of OV_AutoDeployPackage to be added to this service type.

Attributes
: This parameter is used only by SPIs. It contains name/value pairs that list the attributes. Optional parameter. Default is empty array.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds an auto-deploy package (instance of OV_AutoDeployPackage) to this service type.

Return Value

False if the auto-deploy package has already been added to this service type.

Extended Status Codes

MDLAPI_E_ADPACKAGE_NOT_EXIST
: Auto Deploy Package does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::AddAutoDeployPolicyGroup() OV_ServiceTypeDefinition::AddAutoDeployPolicyGroup_Trans()

```
boolean AddAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName)
```

```
boolean AddAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPolicyGroupName
: The Name property of the instance of OV_AutoDeployPolicyGroup to be added to this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds an auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) to this service type.

Before you can add a policy group as an auto-deployment policy group on a service type, you must create an instance of OV_AutoDeployPolicyGroup by calling the OV_AutoDeployPolicyGroup::Create method, where the path (DisplayName in the HPOM for Windows console) of the policy group is specified as the PolicyGroupPath parameter. Then the Name property of the returned instance of OV_AutoDeployPolicyGroup is used as the AutoDeployPolicyGroupName parameter of this method.

Return Value

False if the auto-deploy policy group has already been added to this service type.

Extended Status Codes

MDLAPI_E_ADPOLGROUP_NOT_EXIST
: Auto Deploy Policy Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::DeleteGraph()
OV_ServiceTypeDefinition::DeleteGraph_Trans()

```
void DeleteGraph()
```

```
void DeleteGraph_Trans(
        [in] string TransId)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Deletes graph properties of this service type.

Return Value

None.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::DeleteReport()
OV_ServiceTypeDefinition::DeleteReport_Trans()

```
void DeleteReport()
```

```
void DeleteReport_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Deletes report properties of this service type.

Return Value

None.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetActions() OV_ServiceTypeDefinition::GetActions_Trans()

```
sint32 GetActions(
        [out] OV_Action Actions[] )
```

```
sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[] )
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Actions
Instances of OV_Action that are added to this service type. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of tools (instances of OV_Action) added to this service type.

Return Value

Number of tools in out parameter Actions.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetAntecedentServiceTypes()
OV_ServiceTypeDefinition::GetAntecedentServiceTypes_Trans()

```
sint32 GetAntecedentServiceTypes(
        [out] OV_ServiceTypeDefinition AntecedentServiceTypes[],
        [in, optional] boolean IncludeAllAntecedent)
```

```
sint32 GetAntecedentServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition AntecedentServiceTypes[],
        [in, optional] boolean IncludeAllAntecedent)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentServiceTypes
: Instances of OV_ServiceTypeDefinition on which the specified service type has dependencies. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllAntecedent
: This flag indicates which antecedent service types to include. Optional parameter. Default value is false (only immediately antecedent service types are included).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of services types (instances of OV_ServiceTypeDefinition) on which this service type is dependent.

Return Value

Number of antecedent service types in the out parameter AntecedentServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetAutoDeployPolicyGroups()
OV_ServiceTypeDefinition::GetAutoDeployPolicyGroups_Trans()

```
sint32 GetAutoDeployPolicyGroups(
        [out] OV_AutoDeployPolicyGroup AutoDeployPolicyGroups[] )
```

```
sint32 GetAutoDeployPolicyGroups_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPolicyGroup AutoDeployPolicyGroups[] )
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

AutoDeployPolicyGroups

Instances of OV_AutoDeployPolicyGroup that are added to this service type. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of auto-deploy policy groups (instances of OV_AutuDeployPolicyGroup) added to this service type.

Return Value

Number of auto-deploy policy groups in the out parameter AutoDeployPolicyGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetAutoDeployPackages()
OV_ServiceTypeDefinition::GetAutoDeployPackages_Trans()

```
sint32 GetAutoDeployPackages(
        [out] OV_AutoDeployPackages AutoDeployPackages[] )
```

```
sint32 GetAutoDeployPackages_Trans(
        [in] string TransId,
        [out] OV_AutoDeployPackages AutoDeployPackages[] )
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

AutoDeployPackages
    Instances of OV_AutoDeployPackage that are added to this service type. An instance is valid as a returned out parameter only if the method does not fail.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of auto-deploy packages (instances of OV_AutoDeployPackage) added to this service type.

Return Value

Number of auto-deploy packages in the out parameter AutoDeployPackages.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetCalculationRule()
OV_ServiceTypeDefinition::GetCalculationRule_Trans()

```
OV_CalculationRule GetCalculationRule()

OV_CalculationRule GetCalculationRule_Trans(
        [in] string TransId)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the calculation rule (instance of OV_CalculationRule) of this service type.

Return Value

Instance of OV_CalculationRule of this service type.

Extended Status Codes

MDLAPI_E_CALCRULE_NOT_EXIST
Calculation rule does not exist.

MDLAPI_E_INVALID_PROPERTY
Property is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetChildServiceTypes() OV_ServiceTypeDefinition::GetChildServiceTypes_Trans()

```
sint32 GetChildServiceTypes(
        [out] OV_ServiceTypeDefinition ServiceTypes[],
        [in, optional] boolean IncludeAllSubordinate)
```

```
sint32 GetChildServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ServiceTypes[],
        [in, optional] boolean IncludeAllSubordinate)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ServiceTypes

Instances of OV_ServiceTypeDefinition that are children of this service type. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllSubordinate

This flag indicates which child service types to include. Optional parameter. Default value is false (only immediately subordinate service types are included).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) that are children of this service type.

Return Value

Number of services types (children) in out parameter ServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

```
Transaction with the specified ID does not exist.
```

OV_ServiceTypeDefinition::GetCompositionPropagationRule() OV_ServiceTypeDefinition::GetCompositionPropagationRule_Trans

```
OV_PropagationRule GetCompositionPropagationRule(
        [in] string ChildServiceType)
```

```
OV_PropagationRule GetCompositionPropagationRule_Trans(
        [in] string TransId,
        [in] string ChildServiceType)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ChildServiceType
: The GUID property of the service type (instance of OV_ServiceTypeDefinition) that is the child of this serv type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the propagation rule (instance of OV_PropagationRule) for the service type that is child of this service type.

If the new propagation rule does not exist, the method returns MDLAPI_E_PROPRULE_NOT_EXIST. If the service type composition does not exist, the method returns MDLAPI_E_SVC_TYPE_COMP_ASSOC_NOT_EXIST.

Return Value

Instance of OV_PropagationRule of this service type composition.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_SVC_TYPE_COMP_ASSOC_NOT_EXIST

Service type composition does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetDependencyPropagationRule() OV_ServiceTypeDefinition::GetDependencyPropagationRule_Trans(

```
OV_PropagationRule GetDependencyPropagationRule(
        [in] string AntecedentServiceType)
```

```
OV_PropagationRule GetDependencyPropagationRule_Trans(
        [in] string TransId,
        [in] string AntecedentServiceType)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentServiceType
: The GUID property of the service type (instance of OV_ServiceTypeDefinition) on which this service type depends.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the propagation rule (instance of OV_PropagationRule) for the service type on which this service type depends.

If the new propagation rule does not exist, the method returns MDLAPI_E_PROPRULE_NOT_EXIST. If the service type dependency does not exist, the method returns MDLAPI_E_SVC_TYPE_DEP_ASSOC_NOT_EXIST.

Return Value

Instance of OV_PropagationRule of this service type dependency.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_SVC_TYPE_DEP_ASSOC_NOT_EXIST

Service type dependency does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetDependentServiceTypes() OV_ServiceTypeDefinition::GetDependentServiceTypes_Trans()

```
sint32 GetDependentServiceTypes(
        [out] OV_ServiceTypeDefinition DependentServiceTypes[],
        [in, optional] boolean IncludeAllDependent)
```

```
sint32 GetDependentServiceTypes_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition DependentServiceTypes[],
        [in, optional] boolean IncludeAllDependent)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

DependentServiceTypes
: Instances of OV_ServiceTypeDefinition that are dependent on this service type. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllDependent
: This flag indicates which dependent service types to include. Optional parameter. Default value is false (only immediately dependent service types are included).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types (instances of OV_ServiceTypeDefinition) that are dependent on this service type.

Return Value

Number of service types in the out parameter DependentServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetMessagePropagationRule() OV_ServiceTypeDefinition::GetMessagePropagationRule_Trans()

```
OV_PropagationRule GetMessagePropagationRule()

OV_PropagationRule GetMessagePropagationRule_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns the message propagation rule (instance of OV_PropagationRule) for this service type.

Return Value

Instance of OV_PropagationRule of this service type.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_INVALID_PROPERTY
: Property is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::GetParents()
OV_ServiceTypeDefinition::GetParents_Trans()

```
sint32 GetParents(
        [out] OV_ServiceTypeDefinition ServiceTypes[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

```
sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_ServiceTypeDefinition ServiceTypes[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ServiceTypes

Instances of OV_ServiceTypeDefinition that are the direct parents of this service type. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllHierarchicalParents

Defines whether ServiceTypes also includes instances of OV_ServiceTypeDefinition that are hierarchical parents (recursive until the root service type), rather than direct parents. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of service types for which this service type is a child.

Return Value

Number of service types (parents) in out parameter ServiceTypes.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::HasAction()
OV_ServiceTypeDefinition::HasAction_Trans()

```
boolean HasAction(
        [in] string ActionName)
```

```
boolean HasAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ActionName
    The Name property of the tool (instance of OV_Action), which is used to verify that the tool is associated with this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified tool (instance of OV_Action) is associated with this service type.

Return Value

False if the tool is already added to this service type.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::HasAntecedentServiceType()
OV_ServiceTypeDefinition::HasAntecedentServiceType_Trans()

```
boolean HasAntecedentServiceType(
        [in] string AntecedentGUID,
        [in, optional] boolean IncludeAllAntecedent)
```

```
boolean HasAntecedentServiceType_Trans(
        [in] string TransId,
        [in] string AntecedentGUID,
        [in, optional] boolean IncludeAllAntecedent)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentGUID
: The GUID property of the antecedent service type (instance of OV_ServiceTypeDefinition) on which this service depends.

IncludeAllAntecedent
: This flag indicates which antecedent service types to include. Optional parameter. Default value is false (only immediately antecedent service types are included).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if this service type is dependent on the specified service type (instance of OV_ServiceTypeDefinition).

Return Value

True if the service type (instance of OV_ServiceTypeDefinition) specified by a parameter ServiceTypeGUID is the antecedent service type on which this service type depends.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::HasAutoDeployPolicyGroup() OV_ServiceTypeDefinition::HasAutoDeployPolicyGroup_Trans()

```
boolean HasAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName)

boolean HasAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPolicyGroupName
: The Name property of the auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) used to verify that the policy group is associated with this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) is associated with this service type.

Return Value

True if the auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) specified with the AutoDeployPolicyGroupName parameter is associated with this service type.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::HasAutoDeployPackage() OV_ServiceTypeDefinition::HasAutoDeployPackage_Trans()

```
boolean HasAutoDeployPackage(
        [in] string AutoDeployPackageName)
```

```
boolean HasAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPackageName
: The Name property of the auto-deploy package (instance of OV_AutoDeployPackage) used to verify that the package is associated with this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified auto-deploy package (instance of OV_AutoDeployPackage) is associated to this service type.

Return Value

True if the auto-deploy package (instance of OV_AutoDeployPackage) specified with the AutoDeployPackageName parameter is associated with this service type.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::HasChildServiceType() OV_ServiceTypeDefinition::HasChildServiceType_Trans()

```
boolean HasChildServiceType(
        [in] string ChildGUID,
        [in, optional] boolean IncludeAllSubordinate)
```

```
boolean HasChildServiceType_Trans(
        [in] string TransId,
        [in] string ChildGUID,
        [in, optional] boolean IncludeAllSubordinate)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ChildGUID

The GUID property of the service (instance of OV_ServiceTypeDefinition) used to verify that the service type is a child of this service type.

IncludeAllSubordinate

This flag indicates how to search the specified child service type. Optional parameter. Default value is false (only immediately subordinate service types are searched).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified service type (instance of OV_ServiceTypeDefinition) is a child of this service type.

Return Value

True if the service type (instance of OV_ServiceTypeDefinition) specified by the ServiceTypeGUID parameter is a child of this service type.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::HasDependentServiceType()
OV_ServiceTypeDefinition::HasDependentServiceType_Trans()

```
boolean HasDependentServiceType(
        [in] string DependentGUID,
        [in, optional] boolean IncludeAllDependent)
```

```
boolean HasDependentServiceType_Trans(
        [in] string TransId,
        [in] string DependentGUID,
        [in, optional] boolean IncludeAllDependent)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

DependentGUID
: The GUID property of the service type (instance of OV_ServiceTypeDefinition) that depends on this service type.

IncludeAllDependent
: This flag indicates which dependent service types to include. Optional parameter. Default value is false (only immediately dependent service types are searched).

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified service type (instance of OV_ServiceTypeDefinition) is dependent on this service type.

Return Value

True if the service type (instance of OV_ServiceTypeDefinition) specified by a parameter ServiceTypeGUID depends on this service type.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::IsChildOf() OV_ServiceTypeDefinition::IsChildOf_Trans()

```
boolean IsChildOf(
        [in] string ParentGUID)
```

```
boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentGUID)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentGUID
: The GUID property of the service type (instance of OV_ServiceTypeDefinition) to check if the service type is the parent of this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified service type (instance of OV_ServiceTypeDefinition) is a parent of this service type. Only immediate child service types are checked.

Return Value

True if the service type (instance of OV_ServiceTypeDefinition) specified by a parameter of ParentGUID is the parent of this service type.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::RemoveAction()

OV_ServiceTypeDefinition::RemoveAction_Trans()

```
void RemoveAction(
        [in] string ActionName)
```

```
void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ActionName
: The Name property of the instance of OV_Action to be removed from this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes the tool (instance of OV_Action) from this service type.

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
: Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::RemoveAntecedentServiceType() OV_ServiceTypeDefinition::RemoveAntecedentServiceType_Trans(

```
void RemoveAntecedentServiceType(
        [in] string AntecedentGUID)
```

```
void RemoveAntecedentServiceType_Trans(
        [in] string TransId,
        [in] string AntecedentGUID)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentGUID
: The GUID property of the instance of OV_ServiceTypeDefinition on which this service type depends.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes a dependency relationship between this service type and the specified service type (instance of OV_ServiceTypeDefinition).

Return Value

None.

Extended Status Codes

MDLAPI_E_SERVICETYPE_NOT_EXIST
: Service type does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::RemoveAutoDeployPolicyGroup()
OV_ServiceTypeDefinition::RemoveAutoDeployPolicyGroup_Trans(

```
void RemoveAutoDeployPolicyGroup(
        [in] string AutoDeployPolicyGroupName)
```

```
void RemoveAutoDeployPolicyGroup_Trans(
        [in] string TransId,
        [in] string AutoDeployPolicyGroupName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPolicyGroupName
: The Name property of the instance of OV_AutoDeployPolicyGroup to be removed from this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes an auto-deploy policy group (instance of OV_AutoDeployPolicyGroup) from this service type.

This method removes only the association to the instance of OV_AutoDeployPolicyGroup. To remove the actual instance of OV_AutoDeployPolicyGroup, you call the OV_AutoDeployPolicyGroup::Remove method.

Return Value

None.

Extended Status Codes

MDLAPI_E_ADPOLGROUP_NOT_EXIST
: Auto Deploy Policy Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::RemoveAutoDeployPackage()
OV_ServiceTypeDefinition::RemoveAutoDeployPackage_Trans()

```
void RemoveAutoDeployPackage(
        [in] string AutoDeployPackageName)
```

```
void RemoveAutoDeployPackage_Trans(
        [in] string TransId,
        [in] string AutoDeployPackageName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AutoDeployPackageName
: The Name property of the instance of OV_AutoDeployPackage to be removed from this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes an auto-deploy package (instance of OV_AutoDeployPackage) from this service type.

Return Value

None.

Extended Status Codes

MDLAPI_E_ADPACKAGE_NOT_EXIST
: Auto Deploy Package does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::SetCalculationRule() OV_ServiceTypeDefinition::SetCalculationRule_Trans()

```
void SetCalculationRule(
        [in] string CalcRuleId)

void SetCalculationRule_Trans(
        [in] string TransId,
        [in] string CalcRuleId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

CalcRuleId
: New CalcRuleId property value of this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the calculation rule (instance of OV_CalculationRule) to this service type.

Return Value

None.

Extended Status Codes

MDLAPI_E_CALCRULE_NOT_EXIST
: Calculation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::SetCompositionPropagationRule()
OV_ServiceTypeDefinition::SetCompositionPropagationRule_Trans(

```
void SetCompositionPropagationRule(
        [in] string ChildServiceType,
        [in] string PropRuleId)
```

```
void SetCompositionPropagationRule_Trans(
        [in] string TransId,
        [in] string ChildServiceType,
        [in] string PropRuleId)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ChildServiceType

The GUID property of the service type (instance of OV_ServiceTypeDefinition) that is the child of this service type.

PropRuleId

New PropRuleId property value of the service type composition.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the propagation rule (instance of OV_PropagationRule) for the service type that is the child of this service type.

If the new propagation rule does not exist, the method returns MDLAPI_E_PROPRULE_NOT_EXIST. If the service type composition does not exist, the method returns MDLAPI_E_SVC_TYPE_COMP_ASSOC_NOT_EXIST.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_SVC_TYPE_COMP_ASSOC_NOT_EXIST
: Service type composition does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::SetDependencyPropagationRule()
OV_ServiceTypeDefinition::SetDependencyPropagationRule_Trans(

```
void SetDependencyPropagationRule(
        [in] string AntecedentServiceType,
        [in] string PropRuleId)
```

```
void SetDependencyPropagationRule_Trans(
        [in] string TransId,
        [in] string AntecedentServiceType,
        [in] string PropRuleId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

AntecedentServiceType
: The GUID property of the service type (instance of OV_ServiceTypeDefinition) on which this service type depends.

PropRuleId
: New PropRuleId property value of the service type dependency.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the propagation rule (instance of OV_PropagationRule) for the service type on which this service type is dependent.

If the new propagation rule does not exist, the method returns MDLAPI_E_PROPRULE_NOT_EXIST. If the service type dependency does not exist, the method returns MDLAPI_E_SVC_TYPE_DEP_ASSOC_NOT_EXIST.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_SVC_TYPE_DEP_ASSOC_NOT_EXIST
: Service type dependency does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::SetGraph() OV_ServiceTypeDefinition::SetGraph_Trans()

```
void SetGraph(
        [in, optional] string GraphFamily,
        [in] string GraphCategory)
```

```
void SetGraph_Trans(
        [in] string TransId,
        [in, optional] string GraphFamily,
        [in] string GraphCategory)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

GraphFamily
: New GraphFamily property value of this service type. Optional parameter.

GraphCategory
: New GraphCategory property value of this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds graph properties to this service type.

If the Performance Manager integration tool is not installed, the method returns MDLAPI_E_NO_GRAPH_INTEGRATION. Graph family and graph category must be valid. Otherwise, the method fails with MDLAPI_E_GRAPH_NOT_EXIST. If a graph category is selected, the graph family is selected automatically.

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_CONFIG_VALUE
Configuration value is not valid.

MDLAPI_E_NO_GRAPH_INTEGRATION
Performance Manager integration tool is not installed.

MDLAPI_E_GRAPH_NOT_EXIST
Graph does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::SetMessagePropagationRule()
OV_ServiceTypeDefinition::SetMessagePropagationRule_Trans()

```
void SetMessagePropagationRule(
        [in] string MsgPropRuleId)
```

```
void SetMessagePropagationRule_Trans(
        [in] string TransId,
        [in] string MsgPropRuleId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

MsgPropRuleId
: New MsgPropRuleId property value of this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Sets the message propagation rule (instance of OV_PropagationRule) for this service type.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPRULE_NOT_EXIST
: Message propagation rule does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_ServiceTypeDefinition::SetReport()
OV_ServiceTypeDefinition::SetReport_Trans()

```
void SetReport(
        [in, optional] string ReportFamily,
        [in] string ReportCategory)
```

```
void SetReport_Trans(
        [in] string TransId,
        [in, optional] string ReportFamily,
        [in] string ReportCategory)
```

Parameters

TransId
  Transaction ID returned from OV_Transaction::Start().

ReportFamily
  New ReportFamily property value of this service type. Optional parameter.

ReportCategory
  New ReportCategory property value of this service type.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds report properties to this service type.

If the Reporter integration tool is not installed, the method returns MDLAPI_E_NO_REPORT_INTEGRATION. Report family and report category must be valid. Otherwise, the method fails with MDLAPI_E_REPORT_NOT_EXIST. If a report category is selected, the report family is selected automatically.

Return Value

None.

Extended Status Codes

MDLAPI_E_INVALID_CONFIG_VALUE
Configuration value is not valid.

MDLAPI_E_NO_REPORT_INTEGRATION
Reporter integration tool is not installed.

MDLAPI_E_REPORT_NOT_EXIST
Report does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_Tools

Description

A container class to group actions into a tool group. This class functions like a folder.

```
class OV_Tools
{
        Properties:
        string Name;

        Class Methods:

        OV_Tools Create(
                [in] string Caption,
                [in] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description
                );

        OV_Tools Create_Trans(
                [in] string TransId,
                [in] string Caption,
                [in] string ParentName,
                [in, optional] string Name,
                [in, optional] string Description
                );

        void Remove(
                [in] string Name
                );

        void Remove_Trans(
                [in] string TransId,
                [in] string Name
                );

        OV_Tools GetRoot(
                );

        OV_Tools GetRoot_Trans(
                [in] string TransId
                );

        OV_Tools GetByName(
                [in] string Name
                );

        OV_Tools GetByName_Trans(
                [in] string TransId,
                [in] string Name
                );

        OV_Tools GetByHierarchicalPath(
                [in] string Path
                );

        OV_Tools GetByHierarchicalPath_Trans(
                [in] string TransId,
                [in] string Path
                );

        void ChangeName(
                [in] OV_Tools ToolGroup,
                [in, optional] string NewName
                );

        void ChangeName_Trans(
                [in] string TransId,
                [in] OV_Tools ToolGroup,
                [in, optional] string NewName
                );
```

Instance Methods:

```
void Modify(
          [in] OV_Tools ToolGroup
          );

void Modify_Trans(
          [in] string TransId,
          [in] OV_Tools ToolGroup
          );

sint32 GetParents(
          [out] OV_Tools ToolGroups[],
          [in, optional] boolean IncludeAllHierarchicalParents
          );

sint32 GetParents_Trans(
          [in] string TransId,
          [out] OV_Tools ToolGroups[],
          [in, optional] boolean IncludeAllHierarchicalParents
          );
```

```
boolean IsChildOf(
        [in] string ParentName
        );

boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName
        );

boolean AddAction(
        [in] string ActionName
        );

boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

void RemoveAction(
        [in] string ActionName
        );

void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName
        );

sint32 GetActions(
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeSubGroups
        );

sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeSubGroups
        );

boolean HasChildAction(
        [in] string ActionName,
        [in, optional] boolean IncludeSubGroups
        );

boolean HasChildAction_Trans(
        [in] string TransId,
        [in] string ActionName,
        [in, optional] boolean IncludeSubGroups
        );

boolean AddToolGroup(
                [in] string ToolGroupName
                );

        boolean AddToolGroup_Trans(
                [in] string TransId,
                [in] string ToolGroupName
                );

        void RemoveToolGroup(
                [in] string ToolGroupName
                );

        void RemoveToolGroup_Trans(
                [in] string TransId,
                [in] string ToolGroupName
                );

        sint32 GetChildToolGroups(
                [out] OV_Tools ToolGroups[],
                [in, optional] boolean IncludeSubGroups
                );

        sint32 GetChildToolGroups_Trans(
                [in] string TransId,
                [out] OV_Tools ToolGroups[],
                [in, optional] boolean IncludeSubGroups
                );

        boolean HasChildToolGroup(
                [in] string ToolGroupName,
                [in, optional] boolean IncludeSubGroups
                );

        boolean HasChildToolGroup_Trans(
                [in] string TransId,
                [in] string ToolGroupName,
                [in, optional] boolean IncludeSubGroups
                );
};
```

OV_Tools-Properties

Name

Signature

string Name

Description

The name used to identify the group containing the actions.

OV_Tools Class Methods

This section contains the information on Class methods for OV_Tools.

OV_Tools::ChangeName()
OV_Tools::ChangeName_Trans()

```
void ChangeName(
        [in] OV_Tools ToolGroup,
        [in, optional] string NewName)
```

```
void ChangeName_Trans(
        [in] string TransId,
        [in] OV_Tools ToolGroup,
        [in, optional] string NewName)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ToolGroup

Instance of OV_Tools to change the Name property.

NewName

The New Name property for this instance of OV_Tools. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, the Name property is created as a new GUID. If a tool group with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOLGROUP_NAME_EXISTS.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Changes the Name property of this tool group.

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOLGROUP_NAME_EXISTS
: Tool Group with the same Name already exists.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools::Create()

OV_Tools::Create_Trans()

```
OV_Tools Create(
        [in] string Caption,
        [in] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description)
```

```
OV_Tools Create_Trans(
        [in] string TransId,
        [in] string Caption,
        [in] string ParentName,
        [in, optional] string Name,
        [in, optional] string Description)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

Caption

The Caption property of the instance of OV_Tools to be created. If a tool group with the same Caption already exists on the same hierarchy path level (as a child of the same parent), nothing happens, and the method fails with MDLAPI_E_TOOLGROUP_HIERPATH_EXISTS. If the Caption parameter is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

ParentName

The Name property of the instance of OV_Tools to which the newly created node group is added. If the property is equal to an empty string, the root node group is used. If the specified node group does not exist, the method fails with MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST.

Name

The Name property of the instance of OV_Tools to be created. Optional parameter. If you do not specify this parameter, or if it is equal to an empty string, it is created as a new GUID. If a Tool Group with the same Name already exists, nothing happens, and the method fails with MDLAPI_E_TOOLGROUP_NAME_EXISTS.

Description
: The Description property of the instance of OV_Tools to be created. Optional parameter.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Creates a tool group (new instance of OV_Tools).

Return Value

Instance of the newly created OV_Tools.

Extended Status Codes

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_PARENT_TOOLGROUP_NOT_EXIST
: Parent Tool Group does not exist.

MDLAPI_E_TOOLGROUP_NAME_EXISTS
: Tool Group with the same Name already exists.

MDLAPI_E_TOOLGROUP_HIERPATH_EXISTS
: Tool Group with the same hierarchy path (Caption) already exists.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools::GetByHierarchicalPath()
OV_Tools::GetByHierarchicalPath_Trans()

```
OV_Tools GetByHierarchicalPath(
        [in] string Path)
```

```
OV_Tools GetByHierarchicalPath_Trans(
        [in] string TransId,
        [in] string Path)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Path
: The tool specified by the hierarchy Path, which consists of Caption/Display Names of the parent Tool Groups and a single backslash as a separator. The Path starts with a single backslash. If the Caption contains additional backslashes, you should escape them with backslashes (for example, "\\").

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the tool group (instance of OV_Tools) specified by the hierarchical path.

Return Value

Instance of OV_Tools specified by the hierarchical path.

Extended Status Codes

MDLAPI_E_TOOLGROUP_NOT_EXIST
: Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools::GetByName()
OV_Tools::GetByName_Trans()

```
OV_Tools GetByName(
        [in] string Name)
```

```
OV_Tools GetByName_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Name
: The Name property of the instance of OV_Tools to be returned.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the tool group (instance of OV_Tools) specified by the Name property.

Return Value

Instance of OV_Tools specified by the Name property.

Extended Status Codes

MDLAPI_E_TOOLGROUP_NOT_EXIST
: Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools::GetRoot()
OV_Tools::GetRoot_Trans()

```
OV_Tools GetRoot()
```

```
OV_Tools GetRoot_Trans(
        [in] string TransId)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the root tool group (instance of OV_Tools).

Return Value

Root instance of OV_Tools.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools::Remove()
OV_Tools::Remove_Trans()

```
void Remove(
        [in] string Name)
```

```
void Remove_Trans(
        [in] string TransId,
        [in] string Name)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

Name
: The Name property of the instance of OV_Tools to be removed.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Removes the specified tool group (instance of OV_Tools).

CAUTION:
This method also removes all child tool groups that are not contained in any other tool group. If tools that are contained in this tool group are not contained in any other tool group, they are also removed.

NOTE:
The root tool group cannot be removed.

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOLGROUP_NOT_EXIST
: Tool Group does not exist.

MDLAPI_E_TOOLGROUP_IS_ROOT
: Root Tool Group cannot be removed from or added to another tool group.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools Instance Methods

This section contains the information on Instance methods for OV_Tools.

OV_Tools::AddAction()
OV_Tools::AddAction_Trans()

```
boolean AddAction(
        [in] string ActionName)
```

```
boolean AddAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ActionName
    The Name property of the instance of OV_Action to be added to this tool group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds a tool (instance of OV_Action) to this tool group as a child tool.

Return Value

False if the tool is already a child of this tool group.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
    Tool does not exist.

MDLAPI_E_TOOL_HIERPATH_EXISTS
    Tool with the same Caption is already a child of the parent tool group.

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

OV_Tools::AddToolGroup()
OV_Tools::AddToolGroup_Trans()

```
boolean AddToolGroup(
        [in] string ToolGroupName)
```

```
boolean AddToolGroup_Trans(
        [in] string TransId,
        [in] string ToolGroupName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ToolGroupName
: The Name property of the instance of OV_Tools to be added to this tool group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Adds a tool group (instance of OV_Tools) to this tool group as a child tool group.

The root tool group cannot be added to this node. (If it is, the method fails with MDLAPI_E_TOOLGROUP_IS_ROOT.) If the tool group with the name ToolGroupName is the same as this tool group or one of its parents (recursive until the Root tool group), the method fails with MDLAPI_E_TOOLGROUP_IS_ALREADY_PARENT.

Return Value

False if the tool group is already a child of this tool group.

Extended Status Codes

MDLAPI_E_TOOLGROUP_NOT_EXIST
: Tool Group does not exist.

MDLAPI_E_TOOLGROUP_HIERPATH_EXISTS

Tool Group with the same hierarchy path (Caption) already exists.

MDLAPI_E_TOOLGROUP_IS_ROOT
: Root Tool Group cannot be removed from or added to another tool group.

MDLAPI_E_TOOLGROUP_IS_ALREADY_PARENT
: Tool group already has a parent with the same name.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools::GetActions()
OV_Tools::GetActions_Trans()

```
sint32 GetActions(
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeSubGroups)
```

```
sint32 GetActions_Trans(
        [in] string TransId,
        [out] OV_Action Actions[],
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

Actions
Instances of OV_Action that are children (directly or indirectly, if IncludeSubGroups is equal to true) of this tool group. An instance is valid as a returned out parameter only if the method does not fail.

IncludeSubGroups
Indicates whether Actions also contain instances of OV_Action that are child tools of all child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of tools (instances of OV_Action) that are children of this tool group.

Return Value

Number of tools (children) in the out parameter Actions.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Tools::GetChildToolGroups()
OV_Tools::GetChildToolGroups_Trans()

```
sint32 GetChildToolGroups(
        [out] OV_Tools ToolGroups[],
        [in, optional] boolean IncludeSubGroups)
```

```
sint32 GetChildToolGroups_Trans(
        [in] string TransId,
        [out] OV_Tools ToolGroups[],
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ToolGroups

Instances of OV_Tools that are children (directly or indirectly, if IncludeSubGroups is equal to true) of this tool group. An instance is valid as a returned out parameter only if the method does not fail.

IncludeSubGroups

Indicates whether ToolGroups also contains instances of OV_Tools that are child tool groups of all child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of tool groups (instances of OV_Tools) that are children of this tool group.

Return Value

Number of tool groups (children) in the out parameter ToolGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Tools::GetParents()

OV_Tools::GetParents_Trans()

```
sint32 GetParents(
        [out] OV_Tools ToolGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

```
sint32 GetParents_Trans(
        [in] string TransId,
        [out] OV_Tools ToolGroups[],
        [in, optional] boolean IncludeAllHierarchicalParents)
```

Parameters

TransId
    Transaction ID returned from OV_Transaction::Start().

ToolGroups
    Instances of OV_Tools that are direct parents of this tool group. An instance is valid as a returned out parameter only if the method does not fail.

IncludeAllHierarchicalParents
    Indicates whether ToolGroups also includes instances of OV_Tools that are hierarchical parents (recursive until the root tool group), rather than direct parents. Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns a list of tool groups for which this tool group is a child.

Return Value

Number of tool groups (parents) in the out parameter ToolGroups.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Tools::HasChildAction()
OV_Tools::HasChildAction_Trans()

```
boolean HasChildAction(
        [in] string ActionName,
        [in, optional] boolean IncludeSubGroups)
```

```
boolean HasChildAction_Trans(
        [in] string TransId,
        [in] string ActionName,
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId

Transaction ID returned from OV_Transaction::Start().

ActionName

The Name property of the tool (instances of OV_Action) that is used to verify that it is a child of this tool group.

IncludeSubGroups

Indicates whether the tool to find is also searched in child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified tool (instances of OV_Action) is a child of this tool group.

Return Value

True if the tool (instance of OV_Action) specified by the ToolName parameter is a child of this tool group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

```
Transaction with the specified ID does not exist.
```

OV_Tools::HasChildToolGroup()
OV_Tools::HasChildToolGroup_Trans()

```
boolean HasChildToolGroup(
        [in] string ToolGroupName,
        [in, optional] boolean IncludeSubGroups)
```

```
boolean HasChildToolGroup_Trans(
        [in] string TransId,
        [in] string ToolGroupName,
        [in, optional] boolean IncludeSubGroups)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ToolGroupName
: The Name property of the tool group (instances of OV_Tools) that is used to verify that it is a child of this tool group.

IncludeSubGroups
: Indicates whether the tool group to find is also searched in child groups (recursive). Optional parameter. Default value is false.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified tool group (instance of OV_Tools) is a child of this tool group.

Return Value

True if the tool group (instance of OV_Tools) specified by the ToolGroupName parameter is a child of this tool group.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Tools::IsChildOf()

OV_Tools::IsChildOf_Trans()

```
boolean IsChildOf(
        [in] string ParentName)
```

```
boolean IsChildOf_Trans(
        [in] string TransId,
        [in] string ParentName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ParentName
: The Name property of the tool group (instance of OV_Tools) used to verify that it is a parent of this tool group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Returns true if the specified tool group (instance of OV_Tools) is a parent of this tool group.

Return Value

True if the tool group (instance of OV_Tools) specified by the ParentName parameter is a parent of this node group.

Extended Status Codes

MDLAPI_E_TOOLGROUP_NOT_EXIST
: Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools::Modify()
OV_Tools::Modify_Trans()

```
void Modify(
        [in] OV_Tools ToolGroup)
```

```
void Modify_Trans(
        [in] string TransId,
        [in] OV_Tools ToolGroup)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ToolGroup
: Modified instance of OV_Tools to store.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Stores changes performed to one or more properties.

It is not possible to get an instance of OV_Tools to reflect changes to its properties from within IWbemService::ExecMethodAsync. For this reason, an instance of OV_Tools is specified as the ToolGroup parameter, even though this method is already called in the context of this instance.

As with the OV_Tools::Create method, this method checks the following:

- If a required property is missing, the method fails with MDLAPI_E_PROPERTY_MISSING.
- If the Caption property is an empty string, contains invalid characters, or has more then 1024 characters, the method fails with MDLAPI_E_INVALID_CAPTION.

Return Value

None.

Extended Status Codes

MDLAPI_E_PROPERTY_MISSING
: Property is not set.

MDLAPI_E_INVALID_CAPTION
: Specified object Caption parameter is not valid.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Tools::RemoveAction()
OV_Tools::RemoveAction_Trans()

```
void RemoveAction(
        [in] string ActionName)
```

```
void RemoveAction_Trans(
        [in] string TransId,
        [in] string ActionName)
```

Parameters

TransId
Transaction ID returned from OV_Transaction::Start().

ActionName
The Name property of the instance of OV_Action to be removed from this tool group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes the child tool (instance of OV_Action) from this tool group.

CAUTION:
If the tool no longer belongs to any tool group after this operation, it is removed from STORE (see the description of the Remove class method of OV_Action).

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOL_NOT_EXIST
Tool does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Tools::RemoveToolGroup()
OV_Tools::RemoveToolGroup_Trans()

```
void RemoveToolGroup(
        [in] string ToolGroupName)
```

```
void RemoveToolGroup_Trans(
        [in] string TransId,
        [in] string ToolGroupName)
```

Parameters

TransId
: Transaction ID returned from OV_Transaction::Start().

ToolGroupName
: The Name property of the instance of OV_Tools to be removed from this tool group.

Calling Convention

These methods can be called only from a WMI instance object.

Description

Removes a child tool group (instance of OV_Tools) from this tool group.

CAUTION:
If the tool group no longer belongs to any tool group after this operation, it is removed (see the description of the class method Remove).

Return Value

None.

Extended Status Codes

MDLAPI_E_TOOLGROUP_NOT_EXIST
: Tool Group does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_Transaction

Description

Transactions in the HPOM for Windows model.

```
class OV_Transaction
{
        Properties:
        string Id;
        datetime TimeStarted;
        uint32 Timeout;
        boolean WmiAccessLock;

        Class Methods:

        string Start(
                [in, optional] uint32 Timeout
                );

        void Commit(
                [in] string TransId
                );

        void Rollback(
                [in] string TransId
                );

        boolean IsExisting(
                [in] string TransId
                );

        void StillAlive(
                [in] string TransId
                );
};
```

OV_Transaction-Properties

Id

Signature
    string Id

Description
    ID of the transaction.

TimeStarted

Signature
    datetime TimeStarted

Description
    Time/date of the transaction.

Timeout

Signature
    uint32 Timeout

Description
    Timeout, in seconds, after which all activities are ended. After the timeout, the WMI write lock is released automatically.

WmiAccessLock

Signature
    boolean WmiAccessLock

Description
    True if the transaction has an WMI write lock. If this property is False, use the StillAlive method to re-acquire the WMI lock.

OV_Transaction::Commit()

OV_Transaction::Commit_Trans()

```
void Commit(
        [in] string TransId)
```

Parameters

TransId
: Id of the transaction started with the Start method.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Commits the transaction.

Return Value

None.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
: Transaction with the specified ID does not exist.

OV_Transaction::IsExisting()
OV_Transaction::IsExisting_Trans()

```
boolean IsExisting(
        [in] string TransId)
```

Parameters

TransId
    Id of the transaction searched.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns true if the transaction exists.

Return Value

True if the transaction exists.

Extended Status Codes

None.

OV_Transaction::Rollback()
OV_Transaction::Rollback_Trans()

```
void Rollback(
        [in] string TransId)
```

Parameters

TransId
Id of the transaction started with Start method.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Rollback transaction.

This method should not be called for HPOM for Windows 7.50 because transactions are not fully implemented. It always returns an error (not implemented).

Return Value

None.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST
Transaction with the specified ID does not exist.

OV_Transaction::Start()

OV_Transaction::Start_Trans()

```
string Start(
        [in, optional] uint32 Timeout)
```

Parameters

Timeout
: Timeout, in seconds, after which the write lock is released automatically (unless the timeout, set as a result of the StillAlive method, is still in progress). Optional parameter. Default value is 10 minutes.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Starts the transaction.

If the WMI write lock cannot be acquired from the AccessManager, the method waits until the WMI write lock is released.

The WMI write lock is released automatically as a result of the timeout when both timeouts (set with the Timeout parameter, as a result of the StillAlive method) end.

Theoretically, it is possible for the WMI write lock to be released even though the method that uses the transaction still has some work to do on WMI. For this reason, it is very important to use the StillAlive method.

Return Value

Id of the transaction started.

Extended Status Codes

None.

OV_Transaction::StillAlive()
OV_Transaction::StillAlive_Trans()

```
void StillAlive(
        [in] string TransId)
```

Parameters

TransId
    Id of the transaction started with Start method.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Indicate that some activity is still in progress for the transaction.

This method starts the timeout for "Still Active" again.

If the WMI write lock was released as a result of a timeout, it is re-acquired, and the timeout set with the Start method is started again.

If the WMI write lock cannot be acquired from the AccessManager, the method waits until the WMI write lock is released.

The WMI write lock is released automatically as a result of a timeout when both timeouts (set with the Timeout parameter, as a result of the StillAlive method) end.

Theoretically, it is possible for the WMI write lock to be released even though the method that uses the transaction still has some work to do on WMI. For this reason, it is very important to use the StillAlive method.

Return Value

None.

Extended Status Codes

MDLAPI_E_TRANSACTION_NOT_EXIST
    Transaction with the specified ID does not exist.

MDLAPI_E_TRANSACTION_NOT_EXIST

Transaction with the specified ID does not exist.

OV_TrustedCertificates

Description

The purpose of a certificate is to provide secure communication.

```
class OV_TrustedCertificates
{
        Properties:
        string Type;
        string Subject_CN;
        string Subject_DN_L;
        string Issuer_DN_L;
        string Valid_From;
        string Valid_To;
        sint32 IsValid;

        Class Methods:

        sint32 getCertificate(
                [in] string Subject_CN,
                [out] string Certificate
                );

        sint32 installCertificate(
                [in] string Certificate
                );
};
```

OV_TrustedCertificates-Properties

Type

Signature
: string Type

Description
: Certificate type (for example, X509 certificate).

Subject_CN

Signature
: string Subject_CN

Description
: Public key of the certificate.

Subject_DN_L

Signature
: string Subject_DN_L

Description
: Node described in the certificate.

Issuer_DN_L

Signature
: string Issuer_DN_L

Description
: Issuer of the certificate.

Valid_From

Signature
: string Valid_From

Description
: Start time of the certificate's validity.

Valid_To

Signature

string Valid_To

Description

End time of the certificate's validity.

IsValid

Signature

sint32 IsValid

Description

0 = Certificate is valid.

-1 = Certificate is not valid.

OV_TrustedCertificates::getCertificate()
OV_TrustedCertificates::getCertificate_Trans()

```
sint32 getCertificate(
        [in] string Subject_CN,
        [out] string Certificate)
```

Parameters

Subject_CN
: Public key of the certificate.

Certificate
: Requested certificate PEM (Privacy Enhanced Mail).

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Returns the certificate specified by the public key. If everything goes OK, it returns 0. Otherwise, it returns -1.

Return Value

None.

Extended Status Codes

None.

OV_TrustedCertificates::installCertificate()
OV_TrustedCertificates::installCertificate_Trans()

```
sint32 installCertificate(
        [in] string Certificate)
```

Parameters

Certificate
    Certificate PEM (Privacy Enhanced Mail) to be installed.

Calling Convention

These methods can be called from a WMI class or instance object.

Description

Installs the given certificate. If everything goes OK, it returns 0. Otherwise, it returns -1.

Return Value

None.

Extended Status Codes

None.

COM Interface APIs

This appendix describes two types of COM interface APIs:

- Agent Remote Administration
- Prerequisite Check

NOTE:
The HP Operations agent API includes support for C/C++ and Java, as well as for every language that supports DCOM automation (for example, VB, VBScript, JScript, and so on). However, the agent message stream interface supports C APIs only. All of the APIs are built using Microsoft Visual Studio 2005.

Agent Command-Line API

The HPOM Automation Wrapper adds interfaces to some agent command line API tools to make these tools accessible through the Windows Scripting Host. The functionality of these tools is available through a scripting language such as VBScript.

The following command line API tools are available:

- opcmsg
- opcmon
- opcmack

In the HPOM Automation wrapper these tools are available as the following objects:

- OVOAutomation.opcmsg
- OVOAutomation.opcmon
- OVOAutomation.opcmack

Related Topics:

- Automation Wrapper: opcmsg
- Automation Wrapper: opcmon
- Automation Wrapper: ocpmack

Automation Wrapper: opcmsg

The automation interfaces for the HPOM command line tools have the same functionality as the tools themselves. The automation interface is basically a wrapper around the HPOM Interface API, which is part of the HPOM Application Integration Guide (AIG).

For example, you can submit a message to HP Operations Manager for Windows using the automation wrapper for opcmsg, as follows.

The object `OVOAutomation.opcmsg` generates a message for HP Operations Manager. Before the message is submitted, it is interpreted by the HP Operations Message Interceptor on the local managed node where the command is executed. Depending on how you configured the message interceptor, the message can be:

- Discarded
- Locally logged
- Forwarded to the management server
- Forwarded to the management server , with local logging.

The message interceptor must be configured and running on the managed node or the send() function of the object OVOAutomation.opcmsg will fail.

The following parameters and functions are available for the object:

Parameter	Access Type	Description
Application	In	Name of application or script/program which is affected by or detected the event/problem.
Object	In	Object which is affected by or has detected the event/problem.
MessageText	In	Descriptive text explaining the event/problem in more detail.
Severity	In	Specifies the severity of the message. The following severities are supported: normal, warning, minor, major, and critical. By default, severity normal is applied.

MessageGroup	In	Default message group to which the message belongs. By default, no message group is assigned.
Nodename	In	System on which the event/problem is detected. By default, the name of the current system is applied.
ServiceId	In	
OptVar	In	Optional variable to be used in the message interceptor.
MessageId	Out	Unique ID of the message generated by HP Operations Manager. Parameter cannot be retrieved before calling functions

Functions	Description
Send()	Sends the message to the Message Action Server. All mandatory parameters have to be set before calling this function.

Example

```
Dim msgObj
Set msgObj = CreateObject("OVOAutomation.Opcmsg")

msgObj.MessageText = "My automated message"
msgObj.Application = "My automated application"

msgObj.MessageGroup = "My Message Group"
msgObj.Object = "My Object"
msgObj.Nodename = "HOSTNAME"
msgObj.Severity = "warning"
msgObj.ServiceName = "My Service"

msgObj.Send()
```

Restrictions

This function can be run by any user. The message group (msg_grp), the object, and the application parameter must not be longer than 32 bytes; this is the maximum size HP Operations can handle with these parameters.

Examples

The examples are available on the management server in the following directory:

```
%OvInstallDir%\examples\OvOW\DevelopmentKit\Agent\VBScript
```

Related Topics

- opcagt
- Quick start: how to create a policy

Automation Wrapper: opcmon

The automation interfaces for the HPOM command line tools have the same functionality as the tools themselves. The automation interface is basically a wrapper around the HPOM Interface API, which is part of the HPOM Application Integration Guide (AIG).

The object OVOAutomations.Opcmon offers the following parameters and functions:

Parameter	Access Type
Object name*	In
ObjValue*	In
MsgObject*	In
OptVar	In

Functions	Description
Send()	Sends the message to the Message Action Server. All mandatory parameters have to be set before calling this function.

Example

The following script sends a monitor value to the monitor agent.

```
Dim msgObj
Set msgObj = CreateObject("OVOAutomation.Opcmon")

msgObj.Object = "MyPolicyObject"
msgObj.ObjValue = 7
msgObj.MsgObject = "My Message object"
msgObj.Send()
```

Restrictions

This function can be run by any user. The message group (msg_grp), the object, and the application parameter must not be longer than 32 bytes; this is the maximum size HP Operations can handle with these parameters.

Examples

The examples are available on the management server in the following directory:

```
%OvInstallDir%\examples\OvOW\DevelopmentKit\Agent\VBScript
```

Automation Wrapper: opcmack

The automation interfaces for the HPOM command line tools have the same functionality as the tools themselves. The automation interface is basically a wrapper around the HPOM Interface API, which is part of the HPOM Application Integration Guide (AIG).

The object OVOAutomations.Opcmack offers the following parameters and functions:

Parameter	Access Type
MessageId*	In

Functions	Description
Acknowledge	Acknowledges the specified message. All mandatory parameters have to be set before calling this function.

Example

```
Dim ackObj
Set ackObj = CreateObject("OVOAutomation.Opcmack")
ackObj.MessageId = "MY_MESSAGE_ID"
ackObj.Acknowledge()
```

Restrictions

This function can be run by any user. The message group (msg_grp), the object, and the application parameter must not be longer than 32 bytes; this is the maximum size HP Operations can handle with these parameters.

Examples

The examples are available on the management server in the following directory:

`%OvInstallDir%\examples\OvOW\DevelopmentKit\Agent\VBScript`

Agent Remote Administration

Interfaces

Agent Remote Administration can be used through the following interfaces:

- IOvRemoteAgentEvents
- IOvRemoteAgent
- IOvStdio

Script Support

All interfaces for Remote Administration are fully script compatible.

In scripting methods, the return value of parameters are defined as [out, retval]. The HRESULT return value is returned in the Err object.

For example, for VBS Scripts, see the following files:

opcragt.vbs
: Uses asynchronous functions of IOvRemoteAgent, and implements methods of _IOvRemoteAgent.

OvOWRAgtS.vbs
: Uses synchronous functions of IOvRemoteAgent.

IOvRemoteAgent

Remote administration of HP Operations agents.

The interface IOvRemoteAgent provides methods for remotely administering HP Operations agents.

There are two kinds of methods:

- Synchronous
- Asynchronous

Synchronous methods return after remote administration finishes on all HP Operations agents. They return a string with the status, as well as a Boolean value (cast to Variant) if any errors occur.

Asynchronous methods return immediately after the supplied parameters are validated. Clients that use the interface IOvRemoteAgent are notified after remote administration finishes on each HPOM for Windows managed node. In case of errors, clients are notified through event interface _IOvRemoteAgentEvents. (For a description of notification events, see the description of the _IOvRemoteAgentEvents interface.)

If you want to perform remote administration on a large set of managed nodes, it is better to use asynchronous methods. These methods provide you with continuous information about the status on each managed node. Also, you can cancel these methods.

Synchronous methods, on the other hand, return the status after remote administration finishes on all specified managed nodes.

Properties

None.

Methods

Cancel
    Interrupts the asynchronous operation.

GetManagementServerPrimaryNodeName

Get the Primary Node Name of the management server.

BecomePrimMgr
: Inform the agents on the specified systems to change the HPOM primary manager to the calling HPOM for Windows management server.

BecomePrimMgrAsync
: Inform the agents on the specified systems to change the HPOM primary manager to the calling HPOM for Windows management server.

GetStatus
: Get the current status of the agents on the specified systems, grouped by subagent ID.

GetStatusAsync
: Get the current status of the agents on the specified systems, grouped by subagent ID.

StartAgent
: Start or restart the configured agents on the specified systems.

StartAgentAsync
: Start or restart the configured agents on the specified systems.

StopAgent
: Shut down (stop) all configured agents except the HP Operations Control Agent on the specified systems.

StopAgentAsync
: Shut down (stop) all configured agents except the HP Operations Control Agent on the specified systems.

GetAgentVersion
: Return the version number of the HP Operations agent software that is currently installed on the specified systems.

GetAgentVersionAsync
: Return the version number of the HP Operations agent software that is currently installed on the specified systems.

GetConfigVar
: Return the setting of the configuration variables on the specified systems.

GetConfigVarAsync
: Return the setting of the configuration variables on the specified systems.

SetConfigVar
: Set the configuration variables to the specified value on the specified systems.

SetConfigVarAsync
: Set the configuration variables to the specified value on the specified systems.

Program ID

OvOWRmtAgt.OvRemoteAgent

IOvRemoteAgent::Cancel

Interrupts the asynchronous operation.

```
HRESULT Cancel([out, retval] VARIANT_BOOL* pbCanceling);
```

Parameters

None.

Returning Parameter

pbCanceling
: If Canceling was initiated, VARIANT_TRUE is returned on this parameter. Otherwise, VARIANT_FALSE is returned.

HRESULT Return Values

S_OK
: Canceling was initiated.

S_FALSE
: Canceling was not initiated.

(FAILED)
: An error occurred. IErrorInfo was created. A detailed error description is traced.

Description

The Cancel method is used to interrupt asynchronous operation. If there is no asynchronous operation in progress, it returns immediately. First, the OnCancel event is executed (see the description of the _IOvRemoteAgentEvents interface). Users are asked if they really want to cancel the asynchronous operation (double check). If the user decides not to cancel the operation, the method returns immediately with VARIANT_FALSE, and nothing happens. If the user decides to cancel, canceling is initiated, and the method returns with VARIANT_TRUE.

The Cancel method does not immediately interrupt the asynchronous operation. Instead, it initiates the canceling operation. After the asynchronous operation finishes with the current node, it is interrupted, and does not continue with the remaining nodes.

IOvRemoteAgent::GetManagementServerPrimaryNodeName

Get the Primary Node Name of the management server.

```
HRESULT GetManagementServerPrimaryNodeName([out, retval] BSTR* pbstrName);
```

Parameters

None.

Returning Parameter

pbstrName
: Primary Node Name of the management server.

HRESULT Return Values

S_OK
: Synchronous administration started successfully.

(FAILED)
: An error occurred. The synchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The GetManagementServerPrimaryNodeName method returns the Primary Node Name of the management server.

IOvRemoteAgent::BecomePrimMgr

Inform the agents on the specified systems to change the HPOM primary manager to the calling HPOM for Windows management server.

```
HRESULT BecomePrimMgr(
        [out] VARIANT* pvarErrors,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [in, optional, defaultvalue(0)] long nSubAgentId,
        [out, retval] BSTR* pbstrStatus
        );
```

Parameters

pvarErrors

When synchronous administration completes successfully on all nodes, VARIANT_TRUE is returned in this parameter. Otherwise, VARIANT_FALSE is returned.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds

String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups

Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

nSubAgentId

Optional parameter. ID of the subagent on which to perform the remote administration. Default is 0 (all subagents).

Returning Parameter

pbstrStatus

When synchronous administration started successfully, and is completed on all nodes, the status is returned.

HRESULT Return Values

S_OK

Synchronous administration started successfully.

(FAILED)

An error occurred. The synchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

With the BecomePrimMgr method, the calling HPOM for Windows management server becomes the HPOM primary manager on specified systems. The HPOM primary manager can be set separately for each subagent by the specified subagent ID with the parameter nSubAgentId. The default is 0 (all subagents).

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is a synchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::BecomePrimMgrAsync

Inform the agents on the specified systems to change the HPOM primary manager to the calling HPOM for Windows management server.

```
HRESULT BecomePrimMgrAsync(
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [in, optional, defaultvalue(0)] long nSubAgentId
        );
```

Parameters

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds
: String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups
: Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

nSubAgentId
: Optional parameter. ID of the subagent on which to perform the remote administration. Default is 0 (all subagents).

Returning Parameter

None.

HRESULT Return Values

S_OK
: Asynchronous administration started successfully.

(FAILED)
: An error occurred. The asynchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

With the BecomePrimMgrAsync method, the calling HPOM for Windows management server becomes the HPOM primary manager on specified systems. The HPOM primary manager can be set for each subagent separately by a specified subagent ID with the parameter nSubAgentId. The default is 0 (all subagents).

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds

- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify the string with a single backslash character ("\") as the parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is an asynchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::GetStatus

Get the current status of the agents on the specified systems, grouped by subagent ID.

```
HRESULT GetStatus(
        [out] VARIANT* pvarErrors,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [in, optional, defaultvalue(0)] long nSubAgentId,
        [out, retval] BSTR* pbstrStatus
        );
```

Parameters

pvarErrors

When synchronous administration completes successfully on all nodes, VARIANT_TRUE is returned in this parameter. Otherwise, VARIANT_FALSE is returned.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds

String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups

Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

nSubAgentId

Optional parameter. ID of the subagent on which to perform the remote administration. Default is 0 (all subagents).

Returning Parameter

pbstrStatus

When synchronous administration started successfully, and is completed on all nodes, the status is returned.

HRESULT Return Values

S_OK

Synchronous administration started successfully.

(FAILED)

An error occurred. The synchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The GetStatus method returns the current status of the agents on the specified systems, grouped by subagent ID.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is a synchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::GetStatusAsync

Get the current status of the agents on the specified systems, grouped by subagent ID.

```
HRESULT GetStatusAsync(
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [in, optional, defaultvalue(0)] long nSubAgentId
        );
```

Parameters

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds
: String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups
: Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

nSubAgentId
: Optional parameter. ID of the subagent on which to perform the remote administration. Default is 0 (all subagents).

Returning Parameter

None.

HRESULT Return Values

S_OK
: Asynchronous administration started successfully.

(FAILED)
: An error occurred. The asynchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The GetStatusAsync method returns the current status of the agents on the specified systems, grouped by subagent ID.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is an asynchronous method. For a comparison of synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::StartAgent

Start or restart the configured agents on the specified systems.

```
HRESULT StartAgent(
        [out] VARIANT* pvarErrors,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [in, optional, defaultvalue(0)] long nSubAgentId,
        [out, retval] BSTR* pbstrStatus
        );
```

Parameters

pvarErrors

When synchronous administration completes successfully on all nodes, VARIANT_TRUE is returned in this parameter. Otherwise, VARIANT_FALSE is returned.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds

String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups

Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

nSubAgentId

Optional parameter. ID of the subagent on which to perform the remote administration. Default is 0 (all subagents).

Returning Parameter

pbstrStatus

When synchronous administration starts successfully, and is completed on all nodes, the status is returned.

HRESULT Return Values

S_OK

Synchronous administration started successfully.

(FAILED)

An error occurred. The synchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The StartAgent method starts or restarts the configured agents on the specified systems.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is a synchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::StartAgentAsync

Start or restart the configured agents on the specified systems.

```
HRESULT StartAgentAsync(
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [in, optional, defaultvalue(0)] long nSubAgentId
        );
```

Parameters

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds

String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups

Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

nSubAgentId

Optional parameter. ID of the subagent on which to perform the remote administration. Default is 0 (all subagents).

Returning Parameter

None.

HRESULT Return Values

S_OK

Asynchronous administration started successfully.

(FAILED)

An error occurred. The asynchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The StartAgentAsync method starts or restarts configured agents on the specified systems.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is an asynchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::StopAgent

Shut down (stop) all configured agents except the HP Operations Control Agent on the specified systems.

```
HRESULT StopAgent(
        [out] VARIANT* pvarErrors,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [in, optional, defaultvalue(0)] long nSubAgentId,
        [out, retval] BSTR* pbstrStatus
        );
```

Parameters

pvarErrors

When synchronous administration completes successfully on all nodes, VARIANT_TRUE is returned in this parameter. Otherwise, VARIANT_FALSE is returned.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds

String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups

Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

nSubAgentId

Optional parameter. ID of the subagent on which to perform the remote administration. Default is 0 (all subagents).

Returning Parameter

pbstrStatus

When synchronous administration starts successfully, and is completed on all nodes, the status is returned.

HRESULT Return Values

S_OK

Synchronous administration started successfully.

(FAILED)

An error occurred. The synchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The StopAgent method shuts down (stops) all configured agents except the HP Operations Control Agent on the specified systems.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is a synchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::StopAgentAsync

Shut down (stop) all configured agents except the HP Operations Control Agent on the specified systems.

```
HRESULT StopAgentAsync(
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [in, optional, defaultvalue(0)] long nSubAgentId
        );
```

Parameters

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds
: String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups
: Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

nSubAgentId
: Optional parameter. ID of the subagent on which to perform the remote administration. Default is 0 (all subagents).

Returning Parameter

None.

HRESULT Return Values

S_OK
: Asynchronous administration started successfully.

(FAILED)
: An error occurred. The asynchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The StopAgentAsync method shuts down (stops) all configured agents except the HP Operations Control Agent on the specified systems.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is an asynchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::GetAgentVersion

Return the version number of the HP Operations agent software that is currently installed on the specified systems.

```
HRESULT GetAgentVersion(
        [out] VARIANT* pvarErrors,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [out, retval] BSTR* pbstrStatus
        );
```

Parameters

pvarErrors

When synchronous administration completes successfully on all nodes, VARIANT_TRUE is returned in this parameter. Otherwise, VARIANT_FALSE is returned.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds

String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups

Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

Returning Parameter

pbstrStatus

When synchronous administration starts successfully, and is completed on all nodes, the status is returned.

HRESULT Return Values

S_OK

Synchronous administration started successfully.

(FAILED)

An error occurred. The synchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The GetAgentVersion method returns the version number of the HP Operations agent software that is currently installed on the specified systems.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is a synchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::GetAgentVersionAsync

Return the version number of the HP Operations agent software that is currently installed on the specified systems.

```
HRESULT GetAgentVersionAsync(
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups
        );
```

Parameters

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds
String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups
Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

Returning Parameter

None.

HRESULT Return Values

S_OK
Asynchronous administration started successfully.

(FAILED)
An error occurred. The asynchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The GetAgentVersionAsync method returns the version number of the HP Operations agent software that is currently installed on the specified systems.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is an asynchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::GetConfigVar

Return the setting of the configuration variables on the specified systems.

```
HRESULT GetConfigVar(
        [out] VARIANT* pvarErrors,
        [in] VARIANT varVariables,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [out, retval] BSTR* pbstrStatus
        );
```

Parameters

pvarErrors

When synchronous administration completes successfully on all nodes, VARIANT_TRUE is returned in this parameter. Otherwise, VARIANT_FALSE is returned.

varVariables

Configuration variables. String or array of strings with the following form:

<config_var>[(<process_name>)]

If the <process_name> is specified, only the value for that process is returned. If a variable has different values set for different processes, all values are reported. The values are queried from the nodeinfo and opcinfo files on the managed node. If the values in these files differ, only the values from the opcinfo file are reported.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds

String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups

Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

Returning Parameter

pbstrStatus

When synchronous administration could be started and is completed on all nodes, the status is returned.

HRESULT Return Values

S_OK

Synchronous administration started successfully.

(FAILED)

An error occurred. The synchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The GetConfigVar method returns the setting of the configuration variables on the specified systems.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is a synchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::GetConfigVarAsync

Return the setting of the configuration variables on the specified systems.

```
HRESULT GetConfigVarAsync(
        [in] VARIANT varVariables,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups
        );
```

Parameters

varVariables

Configuration variables. String or array of strings with the following form:

<config_var>[(<process_name>)]

If the <process_name> is specified, only the value for that process is returned. If a variable has different values set for different processes, all values are reported. The values are queried from the nodeinfo and opcinfo files on the managed node. If the values in these files differ, only the values from the opcinfo file are reported.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds

String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths

String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds

String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups

Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

Returning Parameter

None.

HRESULT Return Values

S_OK

Asynchronous administration started successfully.

(FAILED)

An error occurred. The asynchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The GetConfigVarAsync method returns the setting of the configuration variables on the specified systems.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is an asynchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::SetConfigVar

Set the configuration variables to the specified value on the specified systems.

```
HRESULT SetConfigVar(
        [out] VARIANT* pvarErrors,
        [in] VARIANT varVariables,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups,
        [out, retval] BSTR* pbstrStatus
        );
```

Parameters

pvarErrors

When synchronous administration completes successfully on all nodes, VARIANT_TRUE is returned in this parameter. Otherwise, VARIANT_FALSE is returned.

varVariables

Configuration variables. String or array of strings with the following form:

<config_var>[(<process_name>)]=[<value>]

When <process_name> is specified, a value is set only for the process specified in (<process_name>). Process-specific settings have higher priority than general settings. To reset a value to its default setting, do not specify <value>. Resetting the general settings does not affect the specic settings, and vice versa.

Note that the values are set in the nodeinfo file on the managed node. Different settings in the opcinfo file will overwrite any settings produced with this method.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds
: String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths
: String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds
: String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups
: Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

Returning Parameter

pbstrStatus
: When synchronous administration could be started and is completed on all nodes, the status is returned.

HRESULT Return Values

S_OK
: Synchronous administration started successfully.

(FAILED)
: An error occurred. The synchronous operation could not be started. IErrorInfo was created. A detailed

error description is traced.

Description

The SetConfigVar method sets the configuration variable to the specified value on the specified systems.

After having set a configuration variable with this method, the agents must be restarted for the new values to take affect

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is a synchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

IOvRemoteAgent::SetConfigVarAsync

Set the configuration variables to the specified value on the specified systems.

```
HRESULT SetConfigVarAsync(
        [in] VARIANT varVariables,
        [in] VARIANT varNodePrimNames,
        [in] VARIANT varNodeIds,
        [in] VARIANT varNodeGroupPaths,
        [in] VARIANT varNodeGroupIds,
        [in, optional, defaultvalue(0)] VARIANT_BOOL bInclSubGroups
        );
```

Parameters

varVariables

Configuration variables. String or array of strings with the following form:

<config_var>[(<process_name>)]=[<value>]

When <process_name> is specified, a value is set only for the process specified in (<process_name>). Process-specific settings have higher priority than general settings. To reset a value to its default setting, do not specify <value>. Resetting the general settings does not affect the specic settings, and vice versa.

Note that the values are set in the nodeinfo file on the managed node. Different settings in the opcinfo file will overwrite any settings produced with this method.

varNodePrimNames

String or array of strings containing primary node names. To find data on specified nodes, members of varNodePrimNames are matched with whatever was entered as the PrimaryNodeName property of the OV_ManagedNode WMI instance.

If an exact match cannot be found, the behavior is as follows: If a node is specified without a domain, it is compared with a short name (until the first '.') in all OV_ManagedNode WMI instances that have entered FQDN as a Primary Node Name. Otherwise, if a node is specified with FQDN, a short name (until the first '.') is compared with an exact match (also a short name) in OV_ManagedNode WMI instances.

If more than one matching OV_ManagedNode WMI instance is found, an error is reported.

varNodeIds
: String or array of strings containing node IDs (names). To find data on specified nodes, members of varNodeIds are matched with whatever was entered as the Name property of the OV_ManagedNode WMI instance.

varNodeGroupPaths
: String or array of strings containing a full node group path, as shown in the console tree, starting under the nodes. The path must begin with a backslash ("\"). All node groups within the path must begin with a backslash ("\").

  Node groups in the path are identified with the node group caption (that is, the Caption property of the OV_NodeGroup WMI instance).

varNodeGroupIds
: String or array of strings containing the node group IDs (names). Node groups are identified with whatever was entered in the Name property of the OV_NodeGroup WMI instance.

bInclSubGroups
: Optional parameter. VARIANT_TRUE includes the nodes of all subgroups of the specified node groups (parameters varNodeGroupPaths and varNodeGroupIds). VARIANT_FALSE (default) does not enumerate subgroups.

Returning Parameter

None.

HRESULT Return Values

S_OK
: Asynchronous administration started successfully.

(FAILED)
: An error occurred. The asynchronous operation could not be started. IErrorInfo was created. A detailed error description is traced.

Description

The SetConfigVarAsync method sets the configuration variable to the specified value on the specified systems.

After having set a configuration variable with this method, the agents must be restarted for the new values to take affect.

Nodes are specified with some or all of the following parameters:

- varNodePrimNames
- varNodeIds
- varNodeGroupPaths
- varNodeGroupIds

If any parameter is not used, it should be passed to the method as an empty Variant.

Even when the node is included in more than one parameter, remote administration is performed only once.

To also include all subgroups of the specified node group, call the method with the parameter bInclSubGroups set to VARIANT_TRUE. It defaults to VARIANT_FALSE (subgroups will not be enumerated).

To specify all managed nodes, specify a string with a single backslash character ("\") as a parameter varNodeGroupPaths, leave the parameters varNodePrimNames, varNodeIds, and varNodeGroupIds empty, and set the parameter bInclSubGroups to VARIANT_TRUE.

This is an asynchronous method. For a comparison of the synchronous and asynchronous methods, see the description of the IOvRemoteAgent interface.

_IOvRemoteAgentEvents

Events for asynchronous remote administration of HP Operations agents

The _IOvRemoteAgentEvents interface provides events for asynchronous methods of the interface IOvRemoteAgent.

Properties

None.

Events

OnStatusChanged
    Event: Asynchronous administration finishes on the node.

OnCompleted
    Event: Asynchronous administration finishes on all nodes.

OnMessage
    Event: A message should be printed.

OnOutOfMemory
    Event: Out of memory.

OnCancel
    Event: Acknowledge cancellation of asynchronous administration.

Program ID

None.

_IOvRemoteAgentEvents::OnStatusChanged

Event: Asynchronous administration finishes on the node.

```
void OnStatusChanged(
        [in] BSTR bstrNodeName,
        [in] BSTR bstrStatus,
        [in] long nEventId
        );
```

Parameters

bstrNodeName
: Name (caption in console) of the node where the asynchronous administration finishes.

bstrStatus
: Status of the asynchronous administration on the node bstrNodeName.

nEventId
: "Event" for status on the node:

  0 - Remote administration started on the node.

  1 - Remote administration ended on the node (successfully or unsuccessfully).

Returning Parameter

None.

HRESULT Return Values

None.

Description

The OnStatusChanged event is executed each time an asynchronous administration finishes on a node (successfully or unsuccessfully). Clients that receive this event should print out only bstrStatus. bstrNodeName is already contained in bstrStatus.

_IOvRemoteAgentEvents::OnMessage

Event: A message should be printed.

```
void OnMessage(
        [in] BSTR bstrMessage,
        [in] long nSeverity
        );
```

Parameters

bstrMessage
: Message that should be printed.

nSeverity
: "Severity" of bstrMessage:

  0 - Normal message

  1 - Information

  2 - Warning

  3 - Error

Returning Parameter

None.

HRESULT Return Values

None.

Description

The OnMessage event is executed when some message (parameter bstrMessage) should be printed. The parameter nSeverity contains the "severity" of bstrMessage.

_IOvRemoteAgentEvents::OnOutOfMemory

Event: Out of memory.

```
void OnOutOfMemory();
```

Parameters

None.

Returning Parameter

None.

HRESULT Return Values

None.

Description

The OnOutOfMemory event is executed when memory allocation fails. Clients should print the error message, then exit immediately. This is a catastrophic error from which clients cannot recover.

NOTE:
The OnCompleted event was not executed when the event OnOutOfMemory occurred.

_IOvRemoteAgentEvents::OnCancel

Event: Acknowledge cancellation of asynchronous administration.

```
void OnCancel(
        [in] BSTR bstrAcknowledgeMsg,
        [out] VARIANT* pvarCancel
        );
```

Parameters

bstrAcknowledgeMsg
: Acknowledge message for canceling.

pvarCancel
: When the client sets this parameter to VARIANT_FALSE in the event handler implementation, the asynchronous method will not be canceled.

Returning Parameter

None.

HRESULT Return Values

None.

Description

The OnOutOfMemory event is executed when the Cancel method of the interface IOvRemoteAgent is called. Clients should print the message contained in the parameter bstrAcknowledgeMsg. The message asks users whether they are absolutely sure they want to cancel (that is, set the parameter pvarCancel to VARIANT_TRUE to initiate canceling). If the user does not want to cancel, set the parameter pvarCancel to VARIANT_FALSE. See also the description of the Cancel method of the IOvRemoteAgent interface.

IOvStdio

Standard input operations.

The IOvStdio interface provides methods for accessing standard input (keyboard) from scripting clients.

Properties

CancelKeys
: Get/set keys for the methods IsCancelKeyPressed and WaitOnCancelKey.

Methods

IsCancelKeyPressed
: Check if one of keys specified with the property CancelKeys is pressed.

IsKeyPressed
: Check if the key is pressed.

IsAnyKeyPressed
: Check if any key is pressed.

WaitOnCancelKey
: Wait until one of the keys specified with the property CancelKeys is pressed.

WaitOnKey
: Wait until the key is pressed.

GetKey
: Wait until any key is pressed, and then return its code.

Program ID

OvOWRmtAgt.OvStdio

IOvStdio::CancelKeys

Set keys for the methods IsCancelKeyPressed and WaitOnCancelKey:

HRESULT CancelKeys([out, retval] VARIANT *pVal);

Get keys for the methods IsCancelKeyPressed and WaitOnCancelKey:

HRESULT CancelKeys([in] VARIANT newVal);

HRESULT Return Values

S_OK
Get/set executed successfully.

(FAILED)
An error occurred. Get/set could not execute. IErrorInfo was created. A detailed error description is traced.

Description

The CancelKeys property is used in conjunction with the following methods:

- IsCancelKeyPressed
- WaitOnCancelKey

It sets/returns the code of keys.

When it is set, the parameter newVal can contain:

- 8-, 16-, or 32-bit (unsigned) integer
- String that can be converted to an unsigned long integer
- Array of 8-, 16- or 32-bit (unsigned) integers or strings that can be converted to an unsigned long integer

- Array of Variants containing 8-, 16-, or 32-bit (unsigned) integers or strings that can be converted to an unsigned long integer

When it is returned, the parameter pVal contains an array of Variants containing an unsigned long integer. If the property CancelKeys is not yet set, the parameter pVal is an empty Variant.

IOvStdio::IsCancelKeyPressed

Check if one of keys specified with the property CancelKeys is pressed.

```
HRESULT IsCancelKeyPressed([out, retval] VARIANT_BOOL* pbPressed);
```

Parameters

None.

Returning Parameter

pbPressed
: If the Cancel key was pressed, VARIANT_TRUE is returned on this parameter. Otherwise, VARIANT_FALSE is returned.

HRESULT Return Values

S_OK
: Cancel key was pressed.

S_FALSE
: Cancel key was not pressed.

(FAILED)
: An error occurred. IErrorInfo was created. A detailed error description is traced.

Description

The IsCancelKeyPressed method returns VARIANT_TRUE when any key specified with the property CancelKeys is pressed.

IOvStdio::IsKeyPressed

Check if the key is pressed.

```
HRESULT IsKeyPressed(
        [in] VARIANT varKeyCodes,
        [out, retval] VARIANT_BOOL* pbPressed
        );
```

Parameters

varKeyCodes

Key codes that are scanned. For possible values, see the description of the property CancelKeys .

Returning Parameter

pbPressed

If one of the keys in varKeyCodes was pressed, VARIANT_TRUE is returned on this parameter. Otherwise, VARIANT_FALSE is returned.

HRESULT Return Values

S_OK

One of the keys in varKeyCodes was pressed.

S_FALSE

None of the keys in varKeyCodes key was pressed.

(FAILED)

An error occurred. IErrorInfo was created. A detailed error description is traced.

Description

The IsKeyPressed method returns VARIANT_TRUE when any key specified with the parameter varKeyCodes is pressed. See the description of the property CancelKeys for possible values of the parameter `varKeyCodes`.

IOvStdio::IsAnyKeyPressed

Check if any key is pressed.

HRESULT IsAnyKeyPressed([out, retval] VARIANT_BOOL* pbPressed);

Parameters

None.

Returning Parameter

pbPressed
: If any key was pressed, VARIANT_TRUE is returned on this parameter. Otherwise, VARIANT_FALSE is returned.

HRESULT Return Values

S_OK
: Any key was pressed.

S_FALSE
: No key was pressed.

(FAILED)
: An error occurred. IErrorInfo was created. A detailed error description is traced.

Description

The IsAnyKeyPressed method returns VARIANT_TRUE when any key is pressed.

IOvStdio::WaitOnCancelKey

Wait until one of the keys specified with the property CancelKeys is pressed.

```
HRESULT WaitOnCancelKey();
```

Parameters

None.

Returning Parameter

None.

HRESULT Return Values

S_OK
    Method executed successfully.

(FAILED)
    An error occurred. IErrorInfo was created. A detailed error description is traced

Description

The WaitOnCancelKey method waits until any key specified with the property CancelKeys is pressed.

IOvStdio::WaitOnKey

Wait until the key is pressed.

```
HRESULT WaitOnKey([in] VARIANT varKeyCodes);
```

Parameters

varKeyCodes
Key codes that are scanned. For possible values, see the description of the property CancelKeys .

Returning Parameter

None.

HRESULT Return Values

S_OK
Method executed successfully.

(FAILED)
An error occurred. IErrorInfo was created. A detailed error description is traced

Description

The WaitOnKey method waits until any key specified by the parameter varKeyCodes is pressed. See the description of the property CancelKeys for possible values of the parameter varKeyCodes.

IOvStdio::GetKey

Wait until any key is pressed, and then return its code.

```
HRESULT GetKey([out, retval] long* pnKeyCode);
```

Parameters

None.

Returning Parameter

pnKeyCode
: Code of the key that was pressed.

HRESULT Return Values

S_OK
: Method executed successfully.

(FAILED)
: An error occurred. IErrorInfo was created. A detailed error description is traced

Description

The GetKey method waits until any key is pressed, and then return its code.

Prerequisite Check

Interfaces

Agent Remote Administration can be used through the following interfaces:

- IOvReqCheckSrv (since 7.50)
- IOvReqCheckSrv2 (since 8.00)
- IOvReqCheck (since 7.50)
- IOvReqCheck2 (since 8.00)

NOTE:
Clients in HPOM for Windows 8.00 should use interfaces `IOvReqCheckSrv2` and `IOvReqCheck2` , since they provide methods that are better suited for usage with new platform model in HPOM for Windows 8.00.

Script Support

All interfaces for Prerequisite Check are fully script compatible.

In scripting methods, the return value of parameters are defined as [out, retval]. The HRESULT return value is returned in the Err object.

For example, for VBS Scripts, see the following files:

ovowreqcheck.vbs
Uses the methods of IOvReqCheck.

IOvReqCheckSrv

Checking node prerequisites.

Since:
7.50

Detailed Description

The COM automation interface IOvReqCheckSrv provides the methods for checking the node prerequisites either from the HPOM for Windows server or from the HPOM for Windows console (indirectly through the OvOWReqCheckCon.exe component).

NOTE:
Clients in HPOM for Windows 8.00 should use derived interface `IOvReqCheckSrv2` , since it provides methods that are better suited for usage with new platform model in HPOM for Windows 8.00.

Properties

CheckingEnabled
Verify whether prerequisite checking is enabled

Methods

CheckNode
Check the prerequisites for the node specified as the primary node name or IP address.

CheckNodeOv
Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

CheckNodeOv2
Check the prerequisites for the node specified as the ID of the managed node.

Values of ReportFormat Enumeration

- RP_TXT (1)

- RP_XML (2)

Values of ReportType Enumeration

- RP_REQ_FAILED (1)
- RP_REQ_REC_FAILED (2)
- RP_REQ_REC_INF (3)
- RP_REQ_REC (4)
- RP_REQ (5)

Numeric values of HRESULT Return Values

- ovrcNoError (0)
- ovrcErrCannotOpenFile (0x80044000)
- ovrcErrCannotReadFile (0x80044001)
- ovrcErrCannotCloseFile (0x80044002)
- ovrcErrCannotGetConfigInfo (0x80044003)
- ovrcErrMgmtServerNotSet (0x80044100)
- ovrcErrImpersonationFailed (0x80044101)
- ovrcErrNodeIdNotFound (0x80044102)
- ovrcErrConfigFileSyntax (0x80044103)
- ovrcErrPlatformNotDefined (0x80044104)
- ovrcErrSystemNotSupported (0x80044105)

Program ID

OvOWReqCheckSrv.OvReqCheckSrv

IOvReqCheckSrv::CheckingEnabled

Verify whether prerequisite checking is enabled:

HRESULT CheckingEnabled([out, retval] VARIANT_BOOL *pVal);

Enable/disable prerequisite checking:

HRESULT CheckingEnabled([in] VARIANT_BOOL newVal);

HRESULT Return Values

S_OK
: Get/set performed successfully.

(FAILED)
: An error occurred. Get/set could not be performed. IErrorInfo was created. A detailed error description is traced.

Description

With the CheckingEnabled property, you can verify whether prerequisite checking is enabled (for GUI and PMAD operations). Prerequisite checking can be enabled or disabled by setting it to VARIANT_TRUE or VARIANT_FALSE.

IOvReqCheckSrv::CheckNode

Check the prerequisites for the node specified as the primary node name or IP address.

```
HRESULT CheckNode(
        [in] BSTR Node,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in] BSTR Platform,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node

Primary node name or IP address.

Report

String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription

Textual description of the retval (Result) parameter.

Platform

Platform of the checked node. Valid values are "UNIX" and "WINDOWS" (without quotes).

rpFormat

Specifies how detailed report should be. Valid values are:

RP_REQ_REC_INF

All prerequisites (requirements and recommendations) together with information data are returned. Each prerequisite has an associated state (PASS, FAIL, or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ_REC

All prerequisites (requirements and recommendations) are returned. Each prerequisite has an associated state (PASS, FAIL, or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ_REC_FAILED

Only prerequisites (requirements and recommendations) that have failed are returned. Each prerequisite has an associated state (FAIL or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ

All requirements are returned. Each requirement has an associated state (PASS, FAIL, or UNCHECK). Each failed requirement has an additional error description.

RP_REQ_FAILED

Only requirements that have failed are returned. Each requirement has an associated state (FAIL or UNCHECK). Each failed requirement has an additional error description.

rpType

Specifies format of report. Valid values are:

RP_XML

Report is generated in XML format (appropriate for further processing). This is not yet implemented.

RP_TXT

Report is generated in text format (appropriate for directly displaying to the user).

User

Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckSrv.exe impersonates the user account of the calling client application.

NOTE:

For UNIX nodes, you must always specify the User.

Password

Password for the User.

Returning Parameter

Result

Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:

The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

PREREQUISITES WERE CHECKED:

ovrcResAllPrereqOK

(0) - All checked prerequisites are OK.

ovrcResReqOKRecFailed

(1) - All requirements are OK. At least one recommendation has failed.

ovrcResReqOKRecNotCheked

(2) - All requirements are OK. At least one recommendation could not be checked.

ovrcResReqFailed

(3) - At least one requirement has failed.

ovrcResNoPrerequisitesSpecified

(55) - No prerequisites were specified for the <system> system.

PREREQUISTES WERE NOT CHECKED:

ovrcResNoAdminRight

(101) - Client does not have administrative privileges on the node. Prerequisites can be checked only with administrative rights.

ovrcResOneReqNotChecked

(102) - At least one requirement could not be checked (it is unknown whether the requirement is OK). All other successfully checked requirements are OK.

ovrcResNodeNotAvailable

(103) - Checked node is not available. Either there is no network connection or the firewall ports are not opened.

ovrcResNodeNotExists
(104) - Node (name) cannot be resolved.

ovrcResPlatformNotSupported
(105) - Platform on the node is not yet supported.

ovrcResPlatformNotDefined
(106) - The platform properties of the node (System Type, Operating System, and Version) are not set.

ovrcResPlatformNotDiscovered
(107) - Cannot discover the platform on the specified node.

ovrcResRexecUnavailable
(108) - Rexec communication to the node cannot be established. Either there is no rexec daemon on the node, or it is monitoring a custom defined port (other than 512).

ovrcResNoRemoteRegistry
(109) - Remote Registry Service does not run on the node. Prerequisites can be checked only when the Remote Registry Service is running.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
Method executed successfully.

ovrcErrCannotOpenFile
Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
Prerequisite check configuration file cannot be closed.

ovrcErrImpersonationFailed
OvOWReqCheckSrv.exe component cannot impersonate the specified user account.

ovrcErrConfigFileSyntax
    Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
    Another type of error occurred.

Description

The CheckNode method checks prerequisites for the node specified with the Node parameter (the primary node name or IP address). Other properties of the node are discovered, as specified in the prerequisite configuration file. A prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. If the User and Password are not provided, the OvOWReqCheckSrv.exe component "impersonates" the client.

NOTE (since 8.00):
Variables AGENT_COMM_TYPE and AGENT_BINARY_FORMAT are defined as empty strings. If tags [AGT_BIN_FRMT] and/or [AGT_COMM_TYPE] are used in the configuration file, a platform won't be recognized.

IOvReqCheckSrv::CheckNodeOv

Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

```
HRESULT CheckNodeOv(
        [in] BSTR Node,
        [in] long SystemType,
        [in] long OsType,
        [in] BSTR OSVersion,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node

Primary node name or IP address.

SystemType

HPOM for Windows node property SystemTypeId.

OsType

HPOM for Windows node property OsTypeId.

OSVersion

Caption property of OV_OsVersion instance on which points HPOM for Windows node property OsVersionId.

Report

String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription

Textual description of the retval (Result) parameter.

rpFormat
: Specifies how detailed report should be. For valid values, see CheckNode .

rpType
: Specifies the format of the report. For valid values, see CheckNode .

User
: Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckSrv.exe impersonates the user account of the calling client application.

NOTE:
For UNIX nodes, you must always specify the User.

Password
: Password for the User.

Returning Parameter

Result
: Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

For possible returned values, see CheckNode .

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile	Prerequisite check configuration file cannot be accessed.
ovrcErrCannotReadFile	Prerequisite check configuration file cannot be read.
ovrcErrCannotCloseFile	Prerequisite check configuration file cannot be closed.
ovrcErrImpersonationFailed	OvOWReqCheckSrv.exe component cannot impersonate the specified user account.
ovrcErrConfigFileSyntax	Syntax of the prerequisite check configuration file is not valid.
(other FAILED)	Another type of error occurred.

Description

The CheckNodeOv method checks the prerequisites for the node specified with the Node parameter (primary node name or IP address). The node System Type, OS Type, and OS version are specified with the parameters SystemType, OsType, and OSVersion, and are matched with the SYSTEM sections in the prerequisite configuration file to find the correct prerequisites for the specified OS. The prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. If the User and Password are not provided, the OvOWReqCheckSrv.exe component "impersonates" the client.

NOTE (since 8.00):
Variables AGENT_COMM_TYPE and AGENT_BINARY_FORMAT are defined as empty strings. If tags [AGT_BIN_FRMT] and/or [AGT_COMM_TYPE] are used in the configuration file, a platform won't be recognized.

IOvReqCheckSrv::CheckNodeOv2

Check the prerequisites for the node specified as the ID of the managed node.

```
HRESULT CheckNodeOv2(
        [in] BSTR NodeID,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

NodeID
    ID of the managed node (key property of the instance of the WMI class OV_ManagedNode).

Report
    String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription
    Textual description of the retval (Result) parameter.

rpFormat
    Specifies how detailed report should be. For valid values, see CheckNode .

rpType
    Specifies the format of the report. For valid values, see CheckNode .

User
    Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckSrv.exe impersonates the user account of the calling client application.

NOTE:
For UNIX nodes, you must always specify the User.

Password
Password for the User.

Returning Parameter

Result
Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

For possible returned values, see CheckNode .

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
Method executed successfully.

ovrcErrCannotOpenFile
Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
Prerequisite check configuration file cannot be closed.

ovrcErrImpersonationFailed
OvOWReqCheckSrv.exe component cannot impersonate the specified user account.

ovrcErrNodeIdNotFound
: Managed node with the ID (instance of the WMI class OV_ManagedNode with the key property). The NodeID does not exist.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The CheckNodeOv2 method checks the prerequisites for the node specified with the NodeId parameter (the key property of the instance of the WMI class OV_ManagedNode). The node System Type, OS Type, and OS version are read from WMI, and matched with the SYSTEM sections in the prerequisite configuration file to find the prerequisites for the specified OS. The prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined with the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. If the User and Password are not provided, the OvOWReqCheckSrv.exe component "impersonates" the client.

Since 8.00:
This method will also define variables AGENT_COMM_TYPE and AGENT_BINARY_FORMAT. In variable AGENT_COMM_TYPE value of WMI property OV_AgentCommType::Caption is copied. Instance of WMI class OV_AgentCommType is specified with property AgentCommTypeId of WMI instance OV_ManagedNode identified with parameter NodeID. In variable AGENT_BINARY_FORMAT value of WMI property OV_AgentBinaryFormat::Id is copied. Instance of WMI class OV_AgentBinaryFormat is specified with property AgentBinaryFormatId of WMI instance OV_ManagedNode identified with parameter NodeID.

IOvReqCheckSrv2

Checking node prerequisites. You can specify the agent binary format and communication type.

Since:
	8.0

Detailed Description

The COM automation interface IOvReqCheckSrv2 inherits from interface IOvReqCheckSrv. It offers all methods of IOvReqCheckSrv plus additional methods where you can specify the agent binary format and communication type.

See also description of interface IOvReqCheckSrv .

Properties (inherited from IOvReqCheckSrv)

CheckingEnabled
	Verify whether prerequisite checking is enabled

Methods

CheckNode2
	Check the prerequisites for the node specified as the primary node name or IP address.

CheckNodeOv3
	Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

CheckNodeOv4
	Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

Methods (inherited from IOvReqCheckSrv)

CheckNode
	Check the prerequisites for the node specified as the primary node name or IP address.

CheckNodeOv

Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

CheckNodeOv2
: Check the prerequisites for the node specified as the ID of the managed node.

Values of ReportFormat Enumeration

- RP_TXT (1)
- RP_XML (2)

Values of ReportType Enumeration

- RP_REQ_FAILED (1)
- RP_REQ_REC_FAILED (2)
- RP_REQ_REC_INF (3)
- RP_REQ_REC (4)
- RP_REQ (5)

Numeric values of HRESULT Return Values

- ovrcNoError (0)
- ovrcErrCannotOpenFile (0x80044000)
- ovrcErrCannotReadFile (0x80044001)
- ovrcErrCannotCloseFile (0x80044002)
- ovrcErrCannotGetConfigInfo (0x80044003)
- ovrcErrMgmtServerNotSet (0x80044100)
- ovrcErrImpersonationFailed (0x80044101)
- ovrcErrNodeIdNotFound (0x80044102)
- ovrcErrConfigFileSyntax (0x80044103)
- ovrcErrPlatformNotDefined (0x80044104)
- ovrcErrSystemNotSupported (0x80044105)

Program ID

```
OvOWReqCheckSrv.OvReqCheckSrv
```

IOvReqCheckSrv2::CheckNode2

Check the prerequisites for the node specified as the primary node name or IP address.

```
HRESULT CheckNode2(
        [in] BSTR Node,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in] BSTR Platform,
        [in, optional] VARIANT AgentCommType,
        [in, optional] VARIANT AgentBinaryFormat,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node
: Primary node name or IP address.

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription
: Textual description of the retval (Result) parameter.

Platform
: Platform of the checked node. Valid values are "UNIX" and "WINDOWS" (without quotes).

AgentCommType
: Value of this parameter (VARIANT of type string) is copied to variable AGENT_COMM_TYPE. If parameter is not specified (VARIANT of type Empty), variable is defined as empty string. Possible values are defined with tag [AGT_COMM_TYPE]. If tag [AGT_COMM_TYPE] is used and this parameter is not specified, a platform won't be recognized.

AgentBinaryFormat

Value of this parameter (VARIANT of type Int16) is copied to variable AGENT_BINARY_FORMAT. If parameter is not specified (VARIANT of type Empty), variable is defined as empty string. Possible values are defined with tag [AGT_BIN_FRMT]. If tag [AGT_BIN_FRMT] is used and this parameter is not specified, a platform won't be recognized.

rpFormat

Specifies how detailed report should be. Valid values are:

RP_REQ_REC_INF

All prerequisites (requirements and recommendations) together with information data are returned. Each prerequisite has an associated state (PASS, FAIL, or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ_REC

All prerequisites (requirements and recommendations) are returned. Each prerequisite has an associated state (PASS, FAIL, or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ_REC_FAILED

Only prerequisites (requirements and recommendations) that have failed are returned. Each prerequisite has an associated state (FAIL or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ

All requirements are returned. Each requirement has an associated state (PASS, FAIL, or UNCHECK). Each failed requirement has an additional error description.

RP_REQ_FAILED

Only requirements that have failed are returned. Each requirement has an associated state (FAIL or UNCHECK). Each failed requirement has an additional error description.

rpType

Specifies format of report. Valid values are:

RP_XML

Report is generated in XML format (appropriate for further processing). This is not yet implemented.

RP_TXT

Report is generated in text format (appropriate for directly displaying to the user).

User

Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckSrv.exe impersonates the user account of the calling client application.

NOTE:
For UNIX nodes, you must always specify the User.

Password

Password for the User.

Returning Parameter

Result

Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

PREREQUISITES WERE CHECKED:

ovrcResAllPrereqOK
(0) - All checked prerequisites are OK.

ovrcResReqOKRecFailed
(1) - All requirements are OK. At least one recommendation has failed.

ovrcResReqOKRecNotCheked
(2) - All requirements are OK. At least one recommendation could not be checked.

ovrcResReqFailed
(3) - At least one requirement has failed.

ovrcResNoPrerequisitesSpecified
(55) - No prerequisites were specified for the <system> system.

PREREQUISTES WERE NOT CHECKED:

ovrcResNoAdminRight
(101) - Client does not have administrative privileges on the node. Prerequisites can be checked only with administrative rights.

ovrcResOneReqNotChecked
(102) - At least one requirement could not be checked (it is unknown whether the requirement is OK). All other successfully checked requirements are OK.

ovrcResNodeNotAvailable
(103) - Checked node is not available. Either there is no network connection or the firewall ports are not opened.

ovrcResNodeNotExists
(104) - Node (name) cannot be resolved.

ovrcResPlatformNotSupported
(105) - Platform on the node is not yet supported.

ovrcResPlatformNotDefined
(106) - The platform properties of the node (System Type, Operating System, and Version) are not set.

ovrcResPlatformNotDiscovered
(107) - Cannot discover the platform on the specified node.

ovrcResRexecUnavailable
(108) - Rexec communication to the node cannot be established. Either there is no rexec daemon on the node, or it is monitoring a custom defined port (other than 512).

ovrcResNoRemoteRegistry
(109) - Remote Registry Service does not run on the node. Prerequisites can be checked only when the Remote Registry Service is running.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError

Method executed successfully.

ovrcErrCannotOpenFile
Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
Prerequisite check configuration file cannot be closed.

ovrcErrImpersonationFailed
OvOWReqCheckSrv.exe component cannot impersonate the specified user account.

ovrcErrConfigFileSyntax
Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
Another type of error occurred.

Description

The CheckNode2 is similar as method IOvReqCheckSrv::CheckNode . You can check the agent communication type and agent binary format.

The CheckNode2 method checks prerequisites for the node specified with the Node parameter (the primary node name or IP address). Other properties of the node are discovered, as specified in the prerequisite configuration file. A prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. If the User and Password are not provided, the OvOWReqCheckSrv.exe component "impersonates" the client.

NOTE:
Variable AGENT_COMM_TYPE / AGENT_BINARY_FORMAT is defined as empty string if parameter AgentCommType / AgentBinaryFormat is not specified.

IOvReqCheckSrv2::CheckNodeOv3

Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

```
HRESULT CheckNodeOv3(
        [in] BSTR Node,
        [in] long SystemTypeId,
        [in] long OsTypeId,
        [in] BSTR OSVersion,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in, optional] VARIANT AgentCommType,
        [in, optional] VARIANT AgentBinaryFormat,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node
: Primary node name or IP address.

SystemTypeId
: HPOM for Windows node property SystemTypeId.

OsTypeId
: HPOM for Windows node property OsTypeId.

OSVersion
: Caption property of OV_OsVersion instance on which points HPOM for Windows node property OsVersionId.

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription
: Textual description of the retval (Result) parameter.

AgentCommType
: Value of this parameter (VARIANT of type string) is copied to variable AGENT_COMM_TYPE. If parameter is not specified (VARIANT of type Empty), variable is defined as empty string. Possible values are defined with tag [AGT_COMM_TYPE]. If tag [AGT_COMM_TYPE] is used and this parameter is not specified, a platform won't be recognized.

AgentBinaryFormat
: Value of this parameter (VARIANT of type Int16) is copied to variable AGENT_BINARY_FORMAT. If parameter is not specified (VARIANT of type Empty), variable is defined as empty string. Possible values are defined with tag [AGT_BIN_FRMT]. If tag [AGT_BIN_FRMT] is used and this parameter is not specified, a platform won't be recognized.

rpFormat
: Specifies how detailed report should be. For valid values, see CheckNode .

rpType
: Specifies the format of the report. For valid values, see CheckNode .

User
: Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckSrv.exe impersonates the user account of the calling client application.

NOTE:
For UNIX nodes, you must always specify the User.

Password
: Password for the User.

Returning Parameter

Result
: Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be

checked.

For possible returned values, see CheckNode .

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile
: Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
: Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
: Prerequisite check configuration file cannot be closed.

ovrcErrImpersonationFailed
: OvOWReqCheckSrv.exe component cannot impersonate the specified user account.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The CheckNodeOv3 is similar as method IOvReqCheckSrv::CheckNodeOv . You can specify the agent communication type and agent binary format.

The CheckNodeOv3 method checks the prerequisites for the node specified with the Node parameter (primary node name or IP address). The node System Type, OS Type, and OS version are specified with the parameters SystemTypeId, OsTypeId, and OSVersion, and are matched with the SYSTEM sections in the prerequisite configuration file to find the correct prerequisites for the specified OS. The prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. If the User and Password are not provided, the OvOWReqCheckSrv.exe component "impersonates" the client.

NOTE:
Variable AGENT_COMM_TYPE / AGENT_BINARY_FORMAT is defined as empty string if parameter AgentCommType / AgentBinaryFormat is not specified.

IOvReqCheckSrv2::CheckNodeOv4

Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

```
HRESULT CheckNodeOv4(
        [in] BSTR Node,
        [in] long SystemTypeId,
        [in] long OsTypeId,
        [in] BSTR OSVersion,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in, defaultvalue(0)] long AgentCommTypeId,
        [in, defaultvalue(0)] long AgentBinaryFormatId,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node
: Primary node name or IP address.

SystemTypeId
: HPOM for Windows node property SystemTypeId.

OsTypeId
: HPOM for Windows node property OsTypeId.

OSVersion
: Caption property of OV_OsVersion instance on which points HPOM for Windows node property OsVersionId.

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription

Textual description of the retval (Result) parameter.

AgentCommType

HPOM for Windows node property AgentCommTypeId.
Value of this parameter identifies (property Id) instance of WMI class OV_AgentCommType, from where property Caption is copied to variable AGENT_COMM_TYPE. If parameter has value 0 (OV_AgentCommType.Caption="n/a"), variable is defined as empty string. Possible string equivalents are defined with tag [AGT_COMM_TYPE], If tag [AGT_COMM_TYPE] is used and this parameter is not specified, a platform won't be recognized.

AgentBinaryFormat

HPOM for Windows node property AgentBinaryFormatId.
Value of this parameter identifies (property Id) instance of WMI class OV_AgentBinaryFormat, from where property Caption is copied to variable AGENT_BINARY_FORMAT. If parameter has value 0 (OV_AgentBinaryFormat.Caption="n/a"), variable is defined as empty string. Possible string equivalents are defined with tag [AGT_BIN_FRMT]. If tag [AGT_BIN_FRMT] is used and this parameter is not specified, a platform won't be recognized.

rpFormat

Specifies how detailed report should be. For valid values, see CheckNode .

rpType

Specifies the format of the report. For valid values, see CheckNode .

User

Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckSrv.exe impersonates the user account of the calling client application.

NOTE:
For UNIX nodes, you must always specify the User.

Password

Password for the User.

Returning Parameter

Result

Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

For possible returned values, see CheckNode .

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError	Method executed successfully.
ovrcErrCannotOpenFile	Prerequisite check configuration file cannot be accessed.
ovrcErrCannotReadFile	Prerequisite check configuration file cannot be read.
ovrcErrCannotCloseFile	Prerequisite check configuration file cannot be closed.
ovrcErrImpersonationFailed	OvOWReqCheckSrv.exe component cannot impersonate the specified user account.
ovrcErrConfigFileSyntax	Syntax of the prerequisite check configuration file is not valid.
(other FAILED)	Another type of error occurred.

Description

The CheckNodeOv4 is similar as method IOvReqCheckSrv::CheckNodeOv . You can specify the agent communication type and agent binary format. You specify the OS Version with the numeric parameter OsVersionId instead of the string parameter OSVersion. Conversion to string is performed by matching parameter OsVersionId to property Id of instance of WMI class OV_OsVersion. From matched OV_OsVersion instance then Caption property is used for OS version string.

The CheckNodeOv4 method checks the prerequisites for the node specified with the Node parameter (primary node name or IP address). The node System Type, OS Type, and OS version are specified with the parameters SystemTypeId, OsTypeId, and OSVersion, and are matched with the SYSTEM sections in the prerequisite configuration file to find the correct prerequisites for the specified OS. The prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. If the User and Password are not provided, the OvOWReqCheckSrv.exe component "impersonates" the client.

NOTE:
Variable AGENT_COMM_TYPE / AGENT_BINARY_FORMAT is defined as empty string if parameter AgentCommType / AgentBinaryFormat is not specified or 0.

IOvReqCheck

Checking node prerequisites.

Since:
7.50

Detailed Description

The COM automation interface IOvReqCheck provides methods for checking node prerequisites from the HPOM for Windows console. Prerequisites can be checked in the context of the client or in the context of the provided username and password. If the prerequisite check finishes in the context of the provided username and password, the OvOWReqCheck.exe component acts as a proxy to the OvOWReqCheckSrv.exe component residing on the server. If the username and password are not provided when checking prerequisites, the OvOWReqCheck.exe server "impersonates" its clients, and checks prerequisites directly.

The identity must be set to the Launching user. Lunch and access permissions are granted to everyone. Access permission are checked with SecLib within the COM server. Only HPOM for Windows administrators can use this COM server.

NOTE:
Clients in HPOM for Windows 8.00 should use derived interface `IOvReqCheck2` , since it provides methods that are better suited for usage with new platform model in HPOM for Windows 8.00.

NOTE:
For performance reasons, clients using the OvOWReqCheck.exe COM server on multiple threads (for example, Node Editor) should create just one instance in the main thread, and pass a pointer to the working threads.

Properties

CheckingEnabled
Verify whether prerequisite checking is enabled

Methods

CheckNode

Check the prerequisites for the node specified as the primary node name or IP address.

CheckNodeOv
: Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

CheckNodeOv2
: Check the prerequisites for the node specified as the ID of the managed node.

UseManagementServer
: Specify which management server should be used.

GetRequirements
: Return the list of prerequisites for the specified System.

GetAllRequirements
: Return a list of prerequisites for all supported systems.

GetSupportedSystems
: Return a list of supported OS versions.

Values of ReportFormat Enumeration

- RP_TXT (1)
- RP_XML (2)

Values of ReportType Enumeration

- RP_REQ_FAILED (1)
- RP_REQ_REC_FAILED (2)
- RP_REQ_REC_INF (3)
- RP_REQ_REC (4)
- RP_REQ (5)

Numeric values of HRESULT Return Values

- ovrcNoError (0)
- ovrcErrCannotOpenFile (0x80044000)

- ovrcErrCannotReadFile (0x80044001)
- ovrcErrCannotCloseFile (0x80044002)
- ovrcErrCannotGetConfigInfo (0x80044003)
- ovrcErrMgmtServerNotSet (0x80044100)
- ovrcErrNodeIdNotFound (0x80044102)
- ovrcErrConfigFileSyntax (0x80044103)
- ovrcErrPlatformNotDefined (0x80044104)
- ovrcErrSystemNotSupported (0x80044105)

Program ID

OvOWReqCheck.OvReqCheck

IOvReqCheck::CheckingEnabled

Verify whether prerequisite checking is enabled:

HRESULT CheckingEnabled([out, retval] VARIANT_BOOL *pVal);

Enable/disable prerequisite checking:

HRESULT CheckingEnabled([in] VARIANT_BOOL newVal);

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Get/set performed successfully.

ovrcErrMgmtServerNotSet
: The UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

(other FAILED)
: Another type of error occurred.

Description

With the CheckingEnabled property, you can verify whether prerequisite checking is enabled (for GUI and PMAD operations). Prerequisite checking can be enabled or disabled by setting it to VARIANT_TRUE or VARIANT_FALSE.

IOvReqCheck::CheckNode

Check the prerequisites for the node specified as the primary node name or IP address.

```
HRESULT CheckNode(
        [in] BSTR Node,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in] BSTR Platform,
        [in, defaultvalue (RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node
: Primary node name or IP address.

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription
: Textual description of the retval (Result) parameter.

Platform
: Platform of the checked node. Valid values are "UNIX" and "WINDOWS" (without quotes).

rpFormat
: Specifies how detailed report should be. Valid values are:

    RP_REQ_REC_INF
    : All prerequisites (requirements and recommendations) together with information data are returned. Each prerequisite has an associated state (PASS, FAIL, or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ_REC

All prerequisites (requirements and recommendations) are returned. Each prerequisite has an associated state (PASS, FAIL, or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ_REC_FAILED

Only prerequisites (requirements and recommendations) that have failed are returned. Each prerequisite has an associated state (FAIL or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ

All requirements are returned. Each requirement has an associated state (PASS, FAIL, or UNCHECK). Each failed requirement has an additional error description.

RP_REQ_FAILED

Only requirements that have failed are returned. Each requirement has an associated state (FAIL or UNCHECK). Each failed requirement has an additional error description.

rpType

Specifies format of report. Valid values are:

RP_XML

Report is generated in XML format (appropriate for further processing). This is not yet implemented.

RP_TXT

Report is generated in text format (appropriate for directly displaying to the user).

User

Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckCon.exe impersonates the user account of the calling client application. Otherwise, the call is routed to the OvOWReqChecSrv component.

NOTE:

For UNIX nodes, you must always specify the User.

Password

Password for the User.

Returning Parameter

Result

Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

PREREQUISITES WERE CHECKED:

ovrcResAllPrereqOK
(0) - All checked prerequisites are OK.

ovrcResReqOKRecFailed
(1) - All requirements are OK. At least one recommendation has failed.

ovrcResReqOKRecNotCheked
(2) - All requirements are OK. At least one recommendation could not be checked.

ovrcResReqFailed
(3) - At least one requirement has failed.

ovrcResNoPrerequisitesSpecified
(55) - No prerequisites were specified for the <system> system.

PREREQUISTES WERE NOT CHECKED:

ovrcResNoAdminRight
(101) - Client does not have administrative privileges on the node. Prerequisites can be checked only with administrative rights.

ovrcResOneReqNotChecked
(102) - At least one requirement could not be checked (it is unknown whether the requirement is OK). All other successfully checked requirements are OK.

ovrcResNodeNotAvailable
(103) - Checked node is not available. Either there is no network connection or the firewall ports are not opened.

ovrcResNodeNotExists
(104) - Node (name) cannot be resolved.

ovrcResPlatformNotSupported
(105) - Platform on the node is not yet supported.

ovrcResPlatformNotDefined
(106) - The platform properties of the node (System Type, Operating System, and Version) are not set.

ovrcResPlatformNotDiscovered
(107) - Cannot discover the platform on the specified node.

ovrcResRexecUnavailable
(108) - Rexec communication to the node cannot be established. Either there is no rexec daemon on the node, or it is monitoring a custom defined port (other than 512).

ovrcResNoRemoteRegistry
(109) - Remote Registry Service does not run on the node. Prerequisites can be checked only when the Remote Registry Service is running.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
Method executed successfully.

ovrcErrCannotOpenFile
Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet
The UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check

configuration should be used.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The CheckNode method checks prerequisites for the node specified with the Node parameter (the primary node name or IP address). Other properties of the node are discovered, as specified in the prerequisite configuration file. A prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. The call is routed to the OvOWReqCheckSrv.exe component on the server that actually checks requirements. If the User and Password are not provided, the OvOWReqCheck.exe component "impersonates" the client, and contacts node directly.

NOTE (since 8.00):
The variables `AGENT_COMM_TYPE` and `AGENT_BINARY_FORMAT` are defined as empty strings. If the node to be checked is not set up as managed node on the HPOM management server, OVReqCheck cannot check the node. In this case you must use IOvReqCheck2::CheckNode2 to get results.

IOvReqCheck::CheckNodeOv

Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

```
HRESULT CheckNodeOv(
        [in] BSTR Node,
        [in] long SystemType,
        [in] long OsType,
        [in] BSTR OSVersion,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node
: Primary node name or IP address.

SystemType
: HPOM for Windows node property SystemTypeId.

OsType
: HPOM for Windows node property OsTypeId.

OSVersion
: Caption property of OV_OsVersion instance on which points HPOM for Windows node property OsVersionId.

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription

Textual description of the retval (Result) parameter.

rpFormat
: Specifies how detailed report should be. For valid values, see CheckNode .

rpType
: Specifies the format of the report. For valid values, see CheckNode .

User
: Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckCon.exe impersonates the user account of the calling client application. Otherwise, the call is routed to the OvOWReqChecSrv component.

  NOTE:
  For UNIX nodes, you must always specify the User.

Password
: Password for the User.

Returning Parameter

Result
: Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

  NOTE:
  The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

  For possible returned values, see CheckNode .

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile
: Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
: Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet
: UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The CheckNodeOv method checks the prerequisites for the node specified with the Node parameter (primary node name or IP address). The node System Type, OS Type, and OS version are specified with the parameters SystemType, OsType, and OSVersion, and are matched with the SYSTEM sections in the prerequisite configuration file to find the correct prerequisites for the specified OS. The prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. The call is routed to the OvOWReqCheckSrv.exe component on the server, which actually checks requirements. If the User and Password are not provided, OvOWReqCheck.exe component "impersonates" the client, and contacts the node directly.

NOTE (since 8.00):
The variables `AGENT_COMM_TYPE` and `AGENT_BINARY_FORMAT` are defined as empty strings. If the node to be checked is not set up as managed node on the HPOM management server, CheckNodeOv cannot check the node. In this case you must use IOvReqCheck2::CheckNodeOv3 to get results.

IOvReqCheck::CheckNodeOv2

Check the prerequisites for the node specified as the ID of the managed node.

```
HRESULT CheckNodeOv2(
        [in] BSTR NodeID,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

NodeID
: ID of the managed node (key property of the instance of the WMI class OV_ManagedNode).

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription
: Textual description of the retval (Result) parameter.

rpFormat
: Specifies how detailed report should be. For valid values, see CheckNode .

rpType
: Specifies the format of the report. For valid values, see CheckNode .

User
: Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckCon.exe impersonates the user account of the calling client application. Otherwise, the call is routed to the OvOWReqChecSrv component.

NOTE:
For UNIX nodes, always specify the user.

Password
Password for the User.

Returning Parameter

Result
Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

For possible returned values, see CheckNode .

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
Method executed successfully.

ovrcErrCannotOpenFile
Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet
UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from where the prerequisite check configuration should be used.

ovrcErrNodeIdNotFound
: Managed node with the ID (instance of the WMI class OV_ManagedNode with the key property). The NodeID does not exist.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The CheckNodeOv2 method checks the prerequisites for the node specified with the NodeId parameter (the key property of the instance of the WMI class OV_ManagedNode). The node System Type, OS Type, and OS version are read from WMI, and matched with the SYSTEM sections in the prerequisite configuration file to find the prerequisites for the specified OS. The prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined with the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. The call is routed to the OvOWReqCheckSrv.exe component on the server, which actually checks requirements. If the User and Password are not provided, OvOWReqCheck.exe component "impersonates" the client, and contacts the node directly.

NOTE (since 8.00):
This method also defines the variables `AGENT_COMM_TYPE` and `AGENT_BINARY_FORMAT` :

- `AGENT_COMM_TYPE`

  The value of the WMI property `OV_AgentCommType::Caption` is copied into the variable `AGENT_COMM_TYPE` . An instance of the WMI class `OV_AgentCommType` is specified with the property `AgentCommTypeId` of WMI instance `OV_ManagedNode` , which is identified with the parameter `NodeID` .

- `AGENT_BINARY_FORMAT`

  The value of the WMI property `OV_AgentBinaryFormat::Id` is copied into the variable `AGENT_BINARY_FORMAT` . An instance of the WMI class `OV_AgentBinaryFormat` is specified with the property `AgentBinaryFormatId` of WMI instance `OV_ManagedNode` , which is identified with the parameter `NodeID` .

IOvReqCheck::UseManagementServer

Specify which management server should be used.

```
HRESULT UseManagementServer([in] BSTR Server);
```

Parameters

Server
: Name (NetBIOS) of the management server to use.

Returning Parameter

None.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile
: Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
: Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
: Prerequisite check configuration file cannot be closed.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)

Another type of error occurred.

Description

The UseManagementServer method must be called before any other method provided by IOvReqCheck interface. This method sets the management server that should be used.

NOTE:
The console can connect to a different management server. As a result, it is necessary to specify which management server should be used.

IOvReqCheck::GetRequirements

Return the list of prerequisites for the specified System.

```
HRESULT GetRequirements(
        [in] BSTR System,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [out, retval] BSTR* Requirements
        );
```

Parameters

System
: Identifies the System for which to return prerequisites. It directly relates to the [VER] tag in the prerequisite check configuration file.

rpFormat
: Specifies the format of the returned list of prerequisites. For valid values, see CheckNode .

Returning Parameter

Requirements
: String containing the list of prerequisites for the specified System.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile
: Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile

Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
: Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet
: UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The GetRequirements method returns the list of prerequisites for the specified System as a string formatted (XML/TXT) according to the rpFormat parameter.

NOTE (since 8.00):
Variables AGENT_COMM_TYPE and AGENT_BINARY_FORMAT are defined as empty strings.

IOvReqCheck::GetAllRequirements

Return a list of prerequisites for all supported systems.

```
HRESULT GetAllRequirements(
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [out, retval] BSTR* Requirements
        );
```

Parameters

rpFormat
: Specifies the format of the returned list of prerequisites For valid values, see CheckNode .

Returning Parameter

Requirements
: String containing a list of the prerequisites for all supported systems.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile
: Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
: Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
: Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet
: UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The GetAllRequirements method returns the list of prerequisites for all of the supported systems. The format (XML/TXT) of the report is defined with the rpFormat parameter.

NOTE (since 8.00):
Variables AGENT_COMM_TYPE and AGENT_BINARY_FORMAT are defined as empty strings.

IOvReqCheck::GetSupportedSystems

Return a list of supported OS versions.

```
HRESULT GetSupportedSystems([out, retval] VARIANT* SupportedSystems);
```

Parameters

None.

Returning Parameter

SupportedSystems
: Array of strings with the supported OS versions.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile
: Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
: Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
: Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet
: UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The GetSupportedSystems method returns the list of supported OS versions in the array of strings. The list of supported OS versions is retrieved from the prerequisite check configuration file ([VER] tag).

IOvReqCheck2

Checking node prerequisites. You can specify the agent binary format and communication type.

Since:
    8.0

Detailed Description

The COM automation interface IOvReqCheckSrv2 inherits from interface IOvReqCheckSrv. It offers all methods of IOvReqCheckSrv plus additional methods where you can specify the agent binary format and communication type.

See also description of interface IOvReqCheck .

Properties (inherited from IOvReqCheck)

CheckingEnabled
    Verify whether prerequisite checking is enabled

Methods

CheckNode2
    Check the prerequisites for the node specified as the primary node name or IP address.

CheckNodeOv3
    Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

CheckNodeOv4
    Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

GetRequirements2
    Return the list of prerequisites for the specified System.

GetAllRequirements2
    Return a list of prerequisites for all supported systems.

Methods (inherited from IOvReqCheck)

CheckNode
: Check the prerequisites for the node specified as the primary node name or IP address.

CheckNodeOv
: Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

CheckNodeOv2
: Check the prerequisites for the node specified as the ID of the managed node.

UseManagementServer
: Specify which management server should be used.

GetRequirements
: Return the list of prerequisites for the specified System.

GetAllRequirements
: Return a list of prerequisites for all supported systems.

GetSupportedSystems
: Return a list of supported OS versions.

Values of ReportFormat Enumeration

- RP_TXT (1)
- RP_XML (2)

Values of ReportType Enumeration

- RP_REQ_FAILED (1)
- RP_REQ_REC_FAILED (2)
- RP_REQ_REC_INF (3)
- RP_REQ_REC (4)
- RP_REQ (5)

Numeric values of HRESULT Return Values

- ovrcNoError (0)
- ovrcErrCannotOpenFile (0x80044000)
- ovrcErrCannotReadFile (0x80044001)
- ovrcErrCannotCloseFile (0x80044002)
- ovrcErrCannotGetConfigInfo (0x80044003)
- ovrcErrMgmtServerNotSet (0x80044100)
- ovrcErrNodeIdNotFound (0x80044102)
- ovrcErrConfigFileSyntax (0x80044103)
- ovrcErrPlatformNotDefined (0x80044104)
- ovrcErrSystemNotSupported (0x80044105)

Program ID

OvOWReqCheck.OvReqCheck

IOvReqCheck2::CheckNode2

Check the prerequisites for the node specified as the primary node name or IP address.

```
HRESULT CheckNode2(
        [in] BSTR Node,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in] BSTR Platform,
        [in, optional] VARIANT AgentCommType,
        [in, optional] VARIANT AgentBinaryFormat,
        [in, defaultvalue (RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node
: Primary node name or IP address.

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription
: Textual description of the retval (Result) parameter.

Platform
: Platform of the checked node. Valid values are `UNIX` and `WINDOWS` .

AgentCommType
: Optional parameter (VARIANT of type string) that defines the agent communication type of the node. This parameter is required if the node to be checked is not set up as managed node on the HPOM management server. Valid values are `DCE` and `HTTPS` .

AgentBinaryFormat

Optional parameter (VARIANT of type Int16) that defines the agent binary format of the node. This parameter is required if the node to be checked is not set up as managed node on the HPOM management server.

rpFormat

Specifies how detailed report should be. Valid values are:

RP_REQ_REC_INF

All prerequisites (requirements and recommendations) together with information data are returned. Each prerequisite has an associated state (PASS, FAIL, or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ_REC

All prerequisites (requirements and recommendations) are returned. Each prerequisite has an associated state (PASS, FAIL, or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ_REC_FAILED

Only prerequisites (requirements and recommendations) that have failed are returned. Each prerequisite has an associated state (FAIL or UNCHECK). Each failed prerequisite has an additional error description.

RP_REQ

All requirements are returned. Each requirement has an associated state (PASS, FAIL, or UNCHECK). Each failed requirement has an additional error description.

RP_REQ_FAILED

Only requirements that have failed are returned. Each requirement has an associated state (FAIL or UNCHECK). Each failed requirement has an additional error description.

rpType

Specifies format of report. Valid values are:

RP_XML

Report is generated in XML format (appropriate for further processing). This is not yet implemented.

RP_TXT

Report is generated in text format (appropriate for directly displaying to the user).

User

Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckCon.exe impersonates the user account of the calling client application. Otherwise, the call is routed to the OvOWReqChecSrv component.

NOTE:
For UNIX nodes, you must always specify the User.

Password

Password for the User.

Returning Parameter

Result

Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

PREREQUISITES WERE CHECKED:

ovrcResAllPrereqOK

(0) - All checked prerequisites are OK.

ovrcResReqOKRecFailed

(1) - All requirements are OK. At least one recommendation has failed.

ovrcResReqOKRecNotCheked

(2) - All requirements are OK. At least one recommendation could not be checked.

ovrcResReqFailed

(3) - At least one requirement has failed.

ovrcResNoPrerequisitesSpecified

(55) - No prerequisites were specified for the <system> system.

PREREQUISTES WERE NOT CHECKED:

ovrcResNoAdminRight
: (101) - Client does not have administrative privileges on the node. Prerequisites can be checked only with administrative rights.

ovrcResOneReqNotChecked
: (102) - At least one requirement could not be checked (it is unknown whether the requirement is OK). All other successfully checked requirements are OK.

ovrcResNodeNotAvailable
: (103) - Checked node is not available. Either there is no network connection or the firewall ports are not opened.

ovrcResNodeNotExists
: (104) - Node (name) cannot be resolved.

ovrcResPlatformNotSupported
: (105) - Platform on the node is not yet supported.

ovrcResPlatformNotDefined
: (106) - The platform properties of the node (System Type, Operating System, and Version) are not set.

ovrcResPlatformNotDiscovered
: (107) - Cannot discover the platform on the specified node.

ovrcResRexecUnavailable
: (108) - Rexec communication to the node cannot be established. Either there is no rexec daemon on the node, or it is monitoring a custom defined port (other than 512).

ovrcResNoRemoteRegistry
: (109) - Remote Registry Service does not run on the node. Prerequisites can be checked only when the Remote Registry Service is running.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile

Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile

Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile

Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet

The UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

ovrcErrConfigFileSyntax

Syntax of the prerequisite check configuration file is not valid.

(other FAILED)

Another type of error occurred.

Description

The CheckNode2 is similar as method IOvReqCheck::CheckNode . You can also specify the agent communication type and agent binary format.

The CheckNode2 method checks prerequisites for the node specified with the Node parameter (the primary node name or IP address). Other properties of the node are discovered, as specified in the prerequisite configuration file. A prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of the report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. The call is routed to the OvOWReqCheckSrv.exe component on the server that actually checks requirements. If the User and Password are not provided, the OvOWReqCheck.exe component "impersonates" the client, and contacts node directly.

IOvReqCheck2::CheckNodeOv3

Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

```
HRESULT CheckNodeOv3(
        [in] BSTR Node,
        [in] long SystemTypeId,
        [in] long OsTypeId,
        [in] BSTR OSVersion,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in, optional] VARIANT AgentCommType,
        [in, optional] VARIANT AgentBinaryFormat,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node
: Primary node name or IP address.

SystemTypeId
: HPOM for Windows node property SystemTypeId.

OsTypeId
: HPOM for Windows node property OsTypeId.

OSVersion
: Caption property of OV_OsVersion instance on which points HPOM for Windows node property OsVersionId.

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription
: Textual description of the retval (Result) parameter.

AgentCommType
: Optional parameter (VARIANT of type string) that defines the agent communication type of the node. This parameter is required if the node to be checked is not set up as managed node on the HPOM management server. Valid values are `DCE` and `HTTPS` .

AgentBinaryFormat
: Optional parameter (VARIANT of type Int16) that defines the agent binary format of the node. This parameter is required if the node to be checked is not set up as managed node on the HPOM management server then.

rpFormat
: Specifies how detailed report should be. For valid values, see CheckNode .

rpType
: Specifies the format of the report. For valid values, see CheckNode .

User
: Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckCon.exe impersonates the user account of the calling client application. Otherwise, the call is routed to the OvOWReqChecSrv component.

NOTE:
For UNIX nodes, you must always specify the User.

Password
: Password for the User.

Returning Parameter

Result
: Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

For possible returned values, see CheckNode .

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError

Method executed successfully.

ovrcErrCannotOpenFile

Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile

Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile

Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet

UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

ovrcErrConfigFileSyntax

Syntax of the prerequisite check configuration file is not valid.

(other FAILED)

Another type of error occurred.

Description

The CheckNodeOv3 is similar as method IOvReqCheck::CheckNodeOv . You can specify the agent communication type and agent binary format.

The CheckNodeOv3 method checks the prerequisites for the node specified with the Node parameter (primary node name or IP address). The node System Type, OS Type, and OS version are specified with the parameters SystemTypeId, OsTypeId, and OSVersion, and are matched with the SYSTEM sections in the prerequisite configuration file to find the correct prerequisites for the specified OS. The prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. The call is routed to the OvOWReqCheckSrv.exe component on the server, which actually checks requirements. If the User and Password are not provided, OvOWReqCheck.exe component "impersonates" the client, and contacts the node directly.

IOvReqCheck2::CheckNodeOv4

Check the prerequisites for the node specified with the properties of the OV_ManagedNode WMI class.

```
HRESULT CheckNodeOv4(
        [in] BSTR Node,
        [in] long SystemTypeId,
        [in] long OsTypeId,
        [in] BSTR OSVersion,
        [out] VARIANT* Report,
        [out] VARIANT* ResultDescription,
        [in, defaultvalue(0)] long AgentCommTypeId,
        [in, defaultvalue(0)] long AgentBinaryFormatId,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [in, defaultvalue(RP_REQ_FAILED)] ReportType rpType,
        [in, defaultvalue("")] BSTR User,
        [in, defaultvalue("")] BSTR Password,
        [out, retval] long* Result
        );
```

Parameters

Node
: Primary node name or IP address.

SystemTypeId
: HPOM for Windows node property SystemTypeId.

OsTypeId
: HPOM for Windows node property OsTypeId.

OSVersion
: Caption property of OV_OsVersion instance on which points HPOM for Windows node property OsVersionId.

Report
: String containing prerequisite check report (list of requirements and recommendations with the status).

ResultDescription
: Textual description of the retval (Result) parameter.

AgentCommType
: HPOM for Windows node property AgentCommTypeId.
Value of this parameter identifies (property Id) instance of WMI class OV_AgentCommType, from where property Caption is copied to variable AGENT_COMM_TYPE. If parameter has value 0 (OV_AgentCommType.Caption="n/a"), variable is defined as empty string. This parameter is required if the node to be checked is not set up as managed node on the HPOM management server then.

AgentBinaryFormat
: HPOM for Windows node property AgentBinaryFormatId.
Value of this parameter identifies (property Id) instance of WMI class OV_AgentBinaryFormat, from where property Caption is copied to variable AGENT_BINARY_FORMAT. If parameter has value 0 (OV_AgentBinaryFormat.Caption="n/a"), variable is defined as empty string. This parameter is required if the node to be checked is not set up as managed node on the HPOM management server.

rpFormat
: Specifies how detailed report should be. For valid values, see CheckNode .

rpType
: Specifies the format of the report. For valid values, see CheckNode .

User
: Optional user account (with domain) that has administrative privileges on the node where prerequisites are checked. If not specified, OvOWReqCheckCon.exe impersonates the user account of the calling client application. Otherwise, the call is routed to the OvOWReqChecSrv component.

NOTE:
For UNIX nodes, you must always specify the User.

Password
: Password for the User.

Returning Parameter

Result
: Status of the prerequisite checking. A textual description is returned in the ResultDescription parameter.

NOTE:
The returned status is 2 if at least one requirement fails, and if any other requirement could not be checked.

For possible returned values, see CheckNode .

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError	Method executed successfully.
ovrcErrCannotOpenFile	Prerequisite check configuration file cannot be accessed.
ovrcErrCannotReadFile	Prerequisite check configuration file cannot be read.
ovrcErrCannotCloseFile	Prerequisite check configuration file cannot be closed.
ovrcErrMgmtServerNotSet	UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.
ovrcErrConfigFileSyntax	Syntax of the prerequisite check configuration file is not valid.
(other FAILED)	Another type of error occurred.

Description

The CheckNodeOv4 is similar as method IOvReqCheck::CheckNodeOv . You can specify the agent communication type and agent binary format. You specify the OS Version with the numeric parameter OsVersionId instead of the string parameter OSVersion. Conversion to string is performed by matching parameter OsVersionId to property Id of instance of WMI class OV_OsVersion. From matched OV_OsVersion instance then Caption property is used for OS version string.

The CheckNodeOv4 method checks the prerequisites for the node specified with the Node parameter (primary node name or IP address). The node System Type, OS Type, and OS version are specified with the parameters SystemTypeId, OsTypeId, and OSVersion, and are matched with the SYSTEM sections in the prerequisite configuration file to find the correct prerequisites for the specified OS. The prerequisite check report (list of requirements and recommendations with the status) is returned as a string in the Report parameter. The content and format of report are determined by the rpFormat and rpType parameters. If the User and Password parameters are provided, the prerequisite check is performed in the context of the provided user. The call is routed to the OvOWReqCheckSrv.exe component on the server, which actually checks requirements. If the User and Password are not provided, OvOWReqCheck.exe component "impersonates" the client, and contacts the node directly.

IOvReqCheck2::GetRequirements2

Return the list of prerequisites for the specified System.

```
HRESULT GetRequirements2(
        [in] BSTR System,
        [in, optional] VARIANT AgentCommType,
        [in, optional] VARIANT AgentBinaryFormat,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [out, retval] BSTR* Requirements
        );
```

Parameters

System

Identifies the System for which to return prerequisites. It directly relates to the [VER] tag in the prerequisite check configuration file.

AgentCommType

Value of this parameter (VARIANT of type string) is copied to variable AGENT_COMM_TYPE. If parameter is not specified (VARIANT of type Empty), variable is defined as empty string. Possible values:

- "DCE"
- "HTTPS"

AgentBinaryFormat

Value of this parameter (VARIANT of type Int16) is copied to variable AGENT_BINARY_FORMAT. If parameter is not specified (VARIANT of type Empty), variable is defined as empty string. Possible values are defined with tag [AGT_BIN_FRMT]. If tag is not specified, validation of this parameter is not performed.

rpFormat

Specifies the format of the returned list of prerequisites. For valid values, see CheckNode .

Returning Parameter

Requirements

String containing the list of prerequisites for the specified System.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile
: Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
: Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
: Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet
: UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The GetRequirements2 is similar as method IOvReqCheck::GetRequirements . You can specify the agent communication type and agent binary format.

The GetRequirements2 method returns the list of prerequisites for the specified System as a string formatted (XML/TXT) according to the rpFormat parameter.

NOTE:
Variable AGENT_COMM_TYPE / AGENT_BINARY_FORMAT is defined as empty string if parameter

AgentCommType / AgentBinaryFormat is not specified.

IOvReqCheck2::GetAllRequirements2

Return a list of prerequisites for all supported systems.

```
HRESULT GetAllRequirements2(
        [in, optional] VARIANT AgentCommType,
        [in, optional] VARIANT AgentBinaryFormat,
        [in, defaultvalue(RP_TXT)] ReportFormat rpFormat,
        [out, retval] BSTR* Requirements
        );
```

Parameters

AgentCommType

Value of this parameter (VARIANT of type string) is copied to variable AGENT_COMM_TYPE. If parameter is not specified (VARIANT of type Empty), variable is defined as empty string. Possible values:

- "DCE"
- "HTTPS"

AgentBinaryFormat

Value of this parameter (VARIANT of type Int16) is copied to variable AGENT_BINARY_FORMAT. If parameter is not specified (VARIANT of type Empty), variable is defined as empty string. As opposite to method GetRequirements2, this method doesn't validate parameter AgentBinaryFormat – tag [AGT_BIN_FRMT] is ignored.

rpFormat

Specifies the format of the returned list of prerequisites For valid values, see CheckNode .

Returning Parameter

Requirements

String containing a list of the prerequisites for all supported systems.

HRESULT Return Values

If not ovrcNoError, IErrorInfo is created, and a detailed error description is traced.

ovrcNoError
: Method executed successfully.

ovrcErrCannotOpenFile
: Prerequisite check configuration file cannot be accessed.

ovrcErrCannotReadFile
Prerequisite check configuration file cannot be read.

ovrcErrCannotCloseFile
: Prerequisite check configuration file cannot be closed.

ovrcErrMgmtServerNotSet
: UseManagementServer method was not called. Each client should first call the UseManagementServer method to set the management server from which the prerequisite check configuration should be used.

ovrcErrConfigFileSyntax
: Syntax of the prerequisite check configuration file is not valid.

(other FAILED)
: Another type of error occurred.

Description

The GetAllRequirements2 is similar as method IOvReqCheck::GetAllRequirements . You can specify the agent communication type and agent binary format.

The GetAllRequirements2 method returns the list of prerequisites for all of the supported systems. The format (XML/TXT) of the report is defined with the rpFormat parameter.

NOTE:
Variable AGENT_COMM_TYPE / AGENT_BINARY_FORMAT is defined as empty string if parameter AgentCommType / AgentBinaryFormat is not specified.

- Main Page
- Modules
- Classes
- Files
- Related Pages

Policy Management and Deployment (PMAD) API Documentation

HP Operations Manager for Windows 8.00

This documentation provides a detailed description of the COM Interfaces of the Policy Management and Deployment component.

Some Sample VB scripts document how to use the PMAD API.

Introduction to the HPOM Web Services

HP Operations Manager (HPOM) Web Services enable you to develop remote clients that access HPOM management servers using industry-standard terminology and technical standards, instead of product-specific interfaces.

HPOM provides the following web services:

- Incident Web Service

  Enables clients to access HPOM messages.

- Tool Web Service

  Enables clients to execute tools from an HPOM management server.

HPOM Incident Web Service

In HPOM, a message is a structured, readable notification that HPOM generates after detecting, filtering, and correlating one or more events. These events relate to changes in application, system, network, or service status. HPOM messages are conceptually similar to incidents, as defined by the IT Infrastructure Library (ITIL) version 3 framework for service management. An ITIL-compliant incident is an event that could interrupt a service or degrade the quality of a service.

The HPOM Incident Web Service exposes HPOM messages as ITIL-compliant incidents. The service enables remote clients that you develop to get incidents from HPOM, and update and create incidents in HPOM. Clients can also trigger incident state changes in HPOM, for example, to close an incident.

HPOM Tool Web Service

HPOM provides an action system that enables users to start applications, scripts, and commands on managed nodes. The HPOM Tool Web Service enables remote clients that you develop to launch tools on managed nodes using the HPOM action system. To launch a tool, the client creates a new tool execution using the service. After a tool is executed on a managed node, the agent sends back a result code and any output. Clients can subscribe to receive details of updated tool executions.

Features and Benefits

The web services provide the following features and benefits:

- Consistent interfaces on both HP Operations Manager for UNIX and HP Operations Manager for Windows management servers. Clients can integrate seamlessly with HPOM management servers on both platforms.

- Compliance with the Distributed Management Task Force (DMTF) Web Services for Management (WS-Management) standard. This compliance enables you to develop clients for the web service using tools that support WS-Management, for example Wiseman.
- Support for client development using other web service development tools (for example Apache Axis and Windows Communication Framework (WCF)).
- Standard operations that WS-Management specifies (for example Get, Create), and in addition custom operations for incidents (for example Close, Reopen).

Prerequisite Knowledge

Developing a client to access the HPOM Web Services requires knowledge of the standards and tools that you will use. Discussion of these standards and tools is outside the scope of this help .

You may want to consult the following external resources:

- Web services definition language: www.w3.org
- ITIL: www.itil-officialsite.com
- WS-Management: www.dmtf.org (document number DSP0226)
- Wiseman: wiseman.dev.java.net
- Apache Axis: ws.apache.org
- WCF: msdn.microsoft.com

Incident to Message Mappings

The HPOM Incident Web Service exposes HPOM messages as ITIL-compliant incidents. HPOM messages are conceptually similar to incidents, but do not have identical attributes. To enable operations on incidents for HPOM, the service maps HPOM message attributes into incident subelements, and incident subelements into HPOM messages attributes.

For some standard incident subelements, there are conceptually equivalent HPOM message attributes, and so the service maps them directly. The service also includes an extension to the standard incident definition, which makes it possible to map additional HPOM message attributes to incidents elements. Although an incident can contain other extensions, the service ignores them.

For other incident subelements, there are no conceptually equivalent HPOM message attributes. In some cases, the service maps these incident attributes to a custom message attribute (CMA). Each custom message attribute is a name-value pair. In HPOM, each message can have any number of custom message attributes attached to it.

Mapping of Incident Subelements to HPOM Message Attributes shows the mapping of Incident subelements to HPOM message attributes.

Mapping of Incident Subelements to HPOM Message Attributes

Incident Subelement	HPOM Message Attribute	Description
IncidentID	ID	Unique identifier for this incident. HPOM generates this ID.
Description	CMA with the name “Description”	Detailed description of the incident.
Title	Text	Brief description of the event that this incident relates to.
LifeCycleState	Not mapped to an HPOM message attribute. The service uses the following rules to set the LifeCycleState in incidents that the service returns: ▪ If the message is acknowledged, LifeCycleState contains “closed”. ▪ If the message is owned, LifeCycleState contains “work in progress”. ▪ If the message is not acknowledged or owned, LifeCycleState contains “open”. If the message’s State property is “Unknown”, LifeCycleState contains “Undefined”.	

Incident Subelement	HPOM Message Attribute	Description
Severity	Severity	Severity of the event that the incident relates to.
Solution	CMA with the name “Solution”	Description of steps taken in response to the incident.
Category	Message Group	String used for organizing incidents. Incidents that have some logical connection have the same category.
SubCategory	CMA with the name “Subcategory”	String used for more detailed organization of incidents that have the same category.
ProductType	CMA with the name “Product Type”	String that may be used for integration with a service management product. The service management product defines the string’s value and purpose.
ProblemType	CMA with the name “Problem Type”	String that may be used for integration with a service management product. The service management product defines the string’s value and purpose.
CollaborationMode	CMA with the name “Collaboration Mode”	String that may be used for integration with a service management product. The service management product defines the string’s value and purpose.
EmittingCI.ID	Service Id	ID of the service that the incident relates to. The severity of an incident can affect the status of a service that it relates to.
EmittingNode.DnsName	Primary Node Name	Name of the node generating the incident.
AssignedOperator.Name	User of Last State Change	Name of the HPOM user that is currently responsible for the incident. If the incident has an owner, HPOM prevents other users from starting some tasks on that incident.
AffectedCI	*Not mapped to HPOM messages*	
RequesterReference	*Not mapped to HPOM messages*	
Type	Message Type	String used for organizing incidents, for example, to group different types of incident within a category.

Mapping of OperationsExtension Subelements to HPOM Message Attributes shows the mapping of OperationsExtension subelements.

Mapping of OperationsExtension Subelements to HPOM Message Attributes

OperationsExtension Subelements	HPOM Message Attribute	Description
Application	Application	Name of the application to which the incident relates.
Object	Object	Name of the object to which the incident relates.
StateChangeTime	Time of Last State Change	Time at which the State message attribute last changed.
CreationTime	Time First Created on Node	Time at which the agent created the incident.
ReceivedTime	Time First Received on Server	Time at which the management server received the incident.
NumberOfDuplicates	Number of Duplicates	Number of duplicates that the management server has detected for the incident. If duplicate detection is disabled, the value of this is 0.
CorrelationKey	Message Key	String that enables other processes to identify incidents that relate to each other. Related incidents have similar message keys. Message keys are not unique.
ConditionMatched	Unmatched	Indicates whether the incident was sent to the server because of a matched condition in a policy or template.
AutomaticActionStatus	Automatic Command	If the message has an automatic command, the service sets the contents of AutomaticActionStatus to available . Otherwise, the service sets the contents of AutomaticActionStatus to notAvailable .
OperatorActionStatus	Operator Command	If the message has an operator-initiated command, the service sets the contents of OperatorActionStatus to available . Otherwise, the service sets the contents of OperatorActionStatus to notAvailable .
EscalationStatus	Not available	Escalation status is not available on HPOM for Windows.
OriginalEvent	Original Event	Details of the event that is the cause of this incident.
CustomAttributes	Custom Message Attributes	Each incident can have any number of custom message attributes. Each custom message attribute is a name-value pair. The service excludes the following custom message

OperationsExtension Subelements	HPOM Message Attribute	Description
		attributes from the CustomAttributes subelement: ■ Description ■ Solution ■ Subcategory ■ Product Type ■ Problem Type ■ Collaboration Mode The service maps these custom message attributes to Incident subelements instead (see Mapping of Incident Subelements to HPOM Message Attributes).
NumberOfAnnotations	Number of annotations	Number of annotations that have been added to the incident. An annotation is a short note about the incident. For example, a user can add an annotation to summarize actions taken in response to the message.
Source	Policy	Contains the name and version of the policy that created the message.

For an example incident in XML, see Pull Response SOAP Envelope Example .

XML Namespaces

Prefixes and XML Namespaces Used in this Document lists the prefixes used in this document to show the namespaces of element types.

Prefixes and XML Namespaces Used in this Document

Prefix	XML Namespace
inc	http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
incExt	http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident
incFil	http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/IncidentFilter
ismCi	http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/ConfigurationItem
ismNode	http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Node
ismWorkItem	http://schemas.hp.com/ism/ServiceOperation/Common/1/WorkItem
s	http://www.w3.org/2003/05/soap-envelope
tool	http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution
toolFil	http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecutionFilter
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing
wse	http://schemas.xmlsoap.org/ws/2004/08/eventing
wsen	http://schemas.xmlsoap.org/ws/2004/09/enumeration
wsman	http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd
xs	http://www.w3.org/2001/XMLSchema

Endpoint References

The WS-Management specification uses WS-Addressing endpoint references (EPRs) as the addressing model for individual resource instances. XML Namespaces lists the prefixes used in this section to show the namespace of each element type.

An EPR is an XML element of the type *wsa:EndpointReferenceType* . An EPR element contains subelements that, in combination, provide the full reference to a resource instance. In terms of the HPOM Incident Web Service, this means that an EPR provides the full reference to an individual incident on a particular management server. In terms of the HPOM Tool Web Service, an EPR provides the full reference one execution of a tool.

The following XML element shows an example of an EPR for the HPOM Incident Web Service:

```
<wsa:EndpointReference>

  <wsa:Address>
    https://manager1.example.com:443/opr-webservice/Incident.svc/
  </wsa:Address>
  <wsa:ReferenceParameters>
    <wsman:ResourceURI>
      http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
    </wsman:ResourceURI>
    <wsman:SelectorSet>
      <wsman:Selector Name="IncidentID">
          2b9cdf48-d6ca-71db-0d4b-1039228b0000
      </wsman:Selector>
    </wsman:SelectorSet>
  </wsa:ReferenceParameters>
</wsa:EndpointReference>
```

The above EPR contains of the following conceptual elements:

- Address at which the service is available on a management server. This is in the format:

  https://< *server_name* >:< *port* >/opr-webservice/Incident.svc/

- Unique identifier for the type of resource. This is the following URI:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Selector set that identifies an incident. This contains one selector, which contains an incident ID.

When you create a new incident using the service, the service returns an EPR for the new incident. You can store this EPR, and use it to uniquely identify the incident if you need to update it later. (See Create .)

WS-Addressing specifies that, to use an EPR in a request to a service, clients should add the contents of the EPR's *wsa:Address* subelement as the contents of the *wsa:To* subelement in the SOAP header. Where possible, clients should not rely on the *wsa:ReferenceParameters* subelement containing any particular subelements. Clients should instead treat the *wsa:ReferenceParameters* subelement as opaque, by unwrapping its contents and adding them all to the SOAP header of the request.

The following XML fragment shows an example of a SOAP header that contains the EPR of an incident:

```
<s:Header>
...
  <wsa:To>
    https://manager1.example.com:443/opr-webservice/Incident.svc/
  </wsa:To>
  <wsman:ResourceURI>
    http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  </wsman:ResourceURI>
  <wsman:SelectorSet>
    <wsman:Selector Name="IncidentID">
        2b9cdf48-d6ca-71db-0d4b-1039228b0000
    </wsman:Selector>
  </wsman:SelectorSet>
...
</s:Header>
```

Configuring the HPOM Web Services

HPOM for Windows 8.10 and later installs the HPOM Web Services by default with the management server.

The service provides several configuration parameters that enable you to configure how the service responds to client requests. You can also configure the port that the service uses to listen for connections.

The following topics provide more details:

- Service Configuration
- Port Configuration
- IIS Configuration

Service Configuration

The HPOM Web Services provide several configuration parameters that enable you to configure how each service responds to client requests. You can change the values of these parameters to suit your environment.

You can configure the parameters in the Server Configuration dialog box. The same set of parameters are available in the Incident Web Service and Tool Web Service namespaces.

General Service Configuration Parameters lists the parameters that are common to both web services.

General Service Configuration Parameters

Name	Default Value	Description
Default expiration time for enumerations	10 (minutes)	Default duration that enumeration contexts are valid for.
Maximum expiration time for enumerations	60 (minutes)	Maximum duration that enumeration contexts are valid for. If a client specifies a longer duration, the service overrides the client and uses the value of this parameter instead.
Default expiration time for subscriptions	60 (minutes)	Default duration that subscriptions are valid for.
Maximum expiration time for subscriptions	1440 (minutes)	Maximum duration that subscriptions are valid for. If a client specifies a longer duration, the service overrides the client and uses the value of this parameter instead.
Default maximum number of items returned by service	100	Maximum number of items that the service returns for a PullOp operation if the client does not specify a value.
Maximum allowed maximum number of items returned by service	500	Maximum number of items that service returns for a PullOp operation. If a client specifies a larger number, the service uses this value instead.
Event queue size	1000	Maximum number of events that an event queue can store. If the event queue contains the maximum number of events, and a new event occurs, the service discards the oldest event from the queue before it adds the new event to the end of the queue.

Tool Service Configuration Parameters lists the additional parameters for the HPOM Tool Web Service.

Tool Service Configuration Parameters

Name	Default value	Description
Expiration time for finished tool executions	300 (seconds)	Duration that the Tool Web Service stores details of finished tool executions. Clients must read the result of a tool execution within the defined number of seconds after the tool execution finishes.
Expiration time for running tool executions	43200 (seconds)	Duration that the Tool Web Service stores details of running tool executions. If an agent does not send a tool execution result within the defined number of seconds after the tool execution starts, the service deletes the tool execution.

Port Configuration

The port that the HPOM Incident Web Service is available on depends on the configuration of Internet Information Services on HPOM for Windows management servers .

The service listens for HTTPS connections on port 443 by default.

You can configure the service to listen for HTTPS connections on a different port in Microsoft Internet Information Services Manager. Open the properties for the default web site, and then change the SSL port.

IIS Configuration

By default in IIS, the default application pool recycles worker processes every 1740 minutes (29 hours). This means that on HPOM for Windows management servers, IIS restarts the HPOM Web Services every 29 hours. This restart may cause problems with any operations that are in progress. In particular, restarting the service releases any enumeration contexts. After it releases an enumeration context for a subscription, the service does not maintain an event queue for incidents or tool executions until the client subscribes again. (For the Incident Web Service, see SubscribeOp . For the Tool Web Service, see SubscribeOp .)

You can avoid this problem by disabling worker process recycling. In addition, you should develop clients that can handle a restart of the service.

To disable worker process recycling:

1. In Internet Information Services (IIS) Manager, right-click **DefaultAppPool** , and then click **Properties** . The DefaultAppPool Properties dialog box opens.
2. Clear the **Recycle worker processes** check box.
3. Click **OK** .

Incident Operations Reference

The HPOM Incident Web Service provides a Web Services Definition Language (WSDL) document, which describes the service. After you install the service, the WSDL is available from the following location on the HPOM management server:

`https://<`*`server_name`*`>:<`*`port`*`>/opr-webservice/Incident.svc?wsdl`

This WSDL document refers to the other WSDL documents and associated XML Schema Documents (XSDs), and gives their location on the management server. These documents provide complete information about the operations that the service supports. The details of how your client uses these operations depends on the web service client development toolkit that you choose.

HP provides unsupported examples for several client development toolkits. The examples are also available on the HPOM management server after you install the service.

The following read-me files provide more information on how to develop clients with each toolkit:

- Apache Axis2

  <install_dir `>/contrib/oprweb/clients/axis/readme.txt` .

- Windows Communication Foundation

  `<install_dir>/contrib/oprweb/clients/wcf/readme.txt` .

- Wiseman

  `<install_dir>/contrib/oprweb/clients/wiseman/readme.txt` .

Alternatively, you can develop a client using any other suitable toolkit or programming language.

This chapter provides a generic reference to the operations that the service provides. For specific examples of SOAP envelopes that the service receives and sends, see SOAP Envelope Examples

XML Namespaces lists the prefixes used in this section to show the namespace of each element type.

The WSDL document contains the following operations that the service does not actually support:

- Delete
- SubscriptionEndOp

If your client attempts to use these operations, the service returns an ActionNotSupported fault.

Get

This operation returns one incident, which is identified by the incident ID.

Input

SOAP Body

Empty.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance to get.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *inc:Incident* of type *inc:Incident* .

Fault

If it cannot return the incident, the service returns a fault according to the WS-Management standard.

Put

This operation updates an existing incident, which is identified by the incident ID.

Input

SOAP Body

Element *inc:Incident* of type *inc:Incident* . *inc:Incident* can contain any of the following the subelements to update on the server:

- Element *inc:Title* of type *xs:string* .
- Element *inc:Severity* of type *xs:string* . This can be one of the following strings:
  - `Normal`
  - `Warning`
  - `Minor`
  - `Major`
  - `Critical`

inc:Incident can also contain other valid subelements, although this operation ignores them and returns an incident with the contents of those subelements unchanged. In other words, if Incident contains immutable subelements, the operation does not return a fault.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.xmlsoap.org/ws/2004/09/transfer/Put`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance to get.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *inc:Incident* of type *inc:Incident* , which contains the updated incident.

Fault

If it cannot update the incident, the service returns a fault according to the WS-Management standard.

Create

This operation stores a new incident on the management server.

Input

SOAP Body

Element *inc:Incident* of type *inc:Incident* . *inc:Incident* must contain the following subelements:

- *inc:Title*
- *inc:EmittingNode/ismNode:DnsName*
- *Incident* can contain any of the following subelements:
- *inc:Description*
- *inc:Severity* (default: “Unknown”)
- *inc:Solution*
- *inc:Category*
- *inc:SubCategory*
- *inc:ProductType*
- *inc:ProblemType*
- *inc:CollaborationMode* (default: “FYI”)
- *inc:EmittingCI/ismCi:ID*
- *inc:Type*
- *inc:Extensions/incExt:OperationsExtension/incExt:Application*
- *inc:Extensions/incExt:OperationsExtension/incExt:Object*
- *inc:Extensions/incExt:OperationsExtension/incExt:CreationTime* (default: the current time)
- *inc:Extensions/incExt:OperationsExtension/incExt:CorrelationKey* (default: null)
- *inc:Extensions/incExt:OperationsExtension/incExt:CustomAttributes* (default: an empty list)

inc:Incident can also contain other valid subelements, although this operation ignores them.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.xmlsoap.org/ws/2004/09/transfer/Create`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

For an example, see Create Request SOAP Envelope Example .

Output

The SOAP body contains an element *wsa:EndpointReference* of type *wsa:EndpointReferenceType* . This element contains the subelements *wsman:ResourceURI* and *wsman:SelectorSet* , which you can use to uniquely identify the incident in subsequent operations.

For an example, see Create Response SOAP Envelope Example .

Fault

If the service cannot store the incident, it returns a fault according to the WS-Management standard.

Close

This operation sets the lifecycle state of an existing incident to closed. The incident is identified by the incident ID. Alternatively, you can use the CloseMany operation to close multiple incidents in one operation. (See CloseMany .)

Input

SOAP Body

Empty.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/Common/1/Close`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance to close.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

For an example, see Close Request SOAP Envelope Example .

Output

The SOAP body is empty. For an example, see Close Response SOAP Envelope Example .

Fault

If it cannot close the incident, the service returns a fault according to the WS-Management standard.

Reopen

This operation sets the lifecycle state of a closed incident to open. The incident is identified by the incident ID.

Alternatively, you can use the ReopenMany operation to open multiple incidents in one operation. (See ReopenMany .)

Input

SOAP Body

Empty.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/Common/1/Reopen`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance to reopen.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot reopen the incident, the service returns a fault according to the WS-Management standard. The service returns a fault if the incident is already open.

EnumerateOp

This operation returns an enumeration context, which you can then use with the PullOp operation to get batches of incidents from the service. You can specify a filter, so that the enumeration context contains only specific incidents.

Input

SOAP Body

Element *Enumerate* of type *wsen:Enumerate* . This element can contain the following subelements:

- Optional element *wsen:Expires* of type *wsen:ExpirationType* .

  Contains a duration, for which the client requires the enumeration context. If you omit this subelement, the service uses the value of the Default expiration time for enumerations parameter on the management server. If you specify a duration that exceeds the value of the Maximum expiration time for enumerations parameter, the service uses the value of EnumerationExpirationMaximum Maximum expiration time for enumerations instead. (See Service Configuration .)

- Optional element *wsen:Filter* of type *wsen:FilterType* or *wsman:Filter* of type *wsman:dialectableMixedDataType* . For compatibility with different toolkits, the service supports a filter of either type, but you must specify only one of them.

  Filter attribute Dialect must have the following value:

  `http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/IncidentFilter`

  Filter must contain the subelement *incFil:IncidentEnumerationFilter* of type *incFil:IncidentEnumerationFilter* . *IncidentEnumerationFilter* can contain any of the following subelements:

  - Optional element *incFil:Severity* of type *inc:Severity_OpenType*
  - Optional element *incFil:EmittingNode* of type *incFil:EmittingNode*
  - Optional element *incFil:Category* of type *xs:string*
  - Optional element *incFil:Application* of type *xs:string*
  - Optional element *incFil:Object* of type *xs:strin* g
  - Optional element *incFil:EmittingCI* of type *incFil:EmittingCI*
  - Optional element *incFil:CorrelationKey* of type *xs:string*

- Optional element *incFil:EscalationStatus* of type *xs:string*
- Optional element *incFil:ConditionMatched* of type *xs:boolean*
- Optional element *incFil:ReceivedTime* of type *incFil:TimeFilter*
- Optional element *incFil:Title* of type *incFil:KeywordFilter*
- Optional element *incFil:CustomAttributes* of type *incFil:CustomAttributes*

The service enumerates incidents that match the contents of the *incFil:IncidentEnumerationFilter* subelements that you specify. For more details on incident attributes, see Incident to Message Mappings .

If you omit *Filter* , the service enumerates all incidents that have the status open or work in progress.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

For an example, see Enumerate Request SOAP Envelope Example .

Output

The SOAP body contains an element *wsen:EnumerateResponse* (of anonymous type), which contains the following subelements:

- Element *wsen:Expires* of type *wsen:ExpirationType* .

  Contains a duration, for which the enumeration context is valid. You can use the ReleaseOp operation to cancel the enumeration context early. (See ReleaseOp .)

- Element *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

Contains a string that identifies the enumeration context. Use *wsen:EnumerationContext* with the PullOp operation to get batches of incidents from the service. (See PullOp .)

For an example, see Enumerate Response SOAP Envelope Example .

Fault

If it cannot return the enumeration context, the service returns a fault according to the WS-Management standard.

PullOp

This operation returns a batch of incidents from an enumeration context.

Input

SOAP Body

Element *wsen:Pull* (of anonymous type), which contains the following subelements:

- Element *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

  Contains a string that identifies the enumeration context. Use a *wsen:EnumerationContext* that one of the following operations returns:

  - EnumerateOp (see EnumerateOp)
  - SubscribeOp (see SubscribeOp)
  - PullOp (see below)

- Optional element *wsen:MaxElements* of type *xs:positiveInteger* .

  Contains an integer that indicates the maximum number of incidents to return in this batch. If you omit this subelement, the service uses the value of the Default maximum number of items returned by service parameter on the management server. (See Service Configuration .)

The HPOM Incident Web Service does not support the subelements *wsen:MaxTime* or *wsen:MaxCharacters* . You should omit these subelements.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  For enumeration contexts from the SubscribeOp operation, if the event queue remains empty for the specified OperationTimeout, PullOp returns a TimedOut fault.

For an example, see Pull Request SOAP Envelope Example .

Output

The SOAP body contains an element *wsen:PullResponse* (of anonymous type), which contains the following subelements:

- Element *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

  Contains a string that identifies the enumeration context. This is the same as the input *wsen:EnumerationContext.*

- Element *wsen:Items* of the type *wsen:ItemListType* .

  Contains multiple *inc:Incident* elements of type *inc:Incident* .

- Optional element *wsen:EndOfSequence* of type *wsen:attributableEmpty* .

  This element is empty. If *wsen:EndOfSequence* is present, there are no remaining items to pull for this enumeration context and the enumeration context is no longer valid.

For an example, see Pull Response SOAP Envelope Example .

Fault

If it cannot return the batch of incidents, the service returns a fault according to the WS-Management standard.

ReleaseOp

This operation cancels an existing enumeration context early (that is, before the client has pulled all the incidents, and before the enumeration context has expired).

You can use this operation to cancel an enumeration context from an enumerate operation. To cancel an enumeration context from a subscription operation, unsubscribe instead. (See UnsubscribeOp .)

Input

SOAP Body

Element *wsen:Release* (of anonymous type), which contains the subelement *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

wsen:EnumerationContext contains a string that identifies the enumeration context. Use a *wsen:EnumerationContext* that one of the following operations returns:

- EnumerateOp (see EnumerateOp)
- PullOp (see PullOp)

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/09/enumeration/Release
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot release the enumeration context, the service returns a fault according to the WS-Management standard.

SubscribeOp

This operation returns an enumeration context, which you can then use with the PullOp operation to get batches of new or updated incidents from the service. The service maintains an event queue of new or updated incidents. If an existing incident changes several times, the service adds the incident to the event queue several times (the service does not consolidate the events).

The value of the Event queue size parameter on the management server constrains the size of the event queue. (See Service Configuration .) To prevent the service from deleting events from the queue, you must pull batches of updated incidents at appropriate intervals. Consider using the following procedure to manage a subscription:

1. Use the SubscribeOp operation to get an enumeration context.
2. Immediately after SubscribeOp returns the enumeration context, start a PullOp operation. As soon as the event queue contains events, PullOp returns a batch of incidents.

   If the event queue remains empty for the specified OperationTimeout, PullOp returns a TimedOut fault.
3. Immediately after the PullOp returns incidents or a fault, start another PullOp operation.

   This step ensures that you always have a PullOp running, which should prevent the event queue from becoming too large.

CAUTION:
If you restart the HPOM Incident Web Service, the service releases any enumeration contexts. The service does not maintain an event queue of new or updated incidents until the client subscribes again. This situation can also arise if the management server is part of a cluster and a failover occurs.

Input

SOAP Body

Element *wse:Subscribe* of type *wse:SubscribeType* , which contains the following subelements:

- Element *wse:Delivery* of type *wse:DeliveryType* . *wse:Delivery* attribute Mode must have the following value:

  `http://schemas.dmtf.org/wbem/wsman/1/wsman/Pull`

- Optional element *wse:Expires* of type *wse:ExpirationType* .

  Contains either a duration that specifies how long the client requires the subscription, or a date and time at which the subscription should expire. If you omit this subelement, the service uses the value of the Default expiration time for subscriptions parameter on the management server. If you specify a duration that exceeds the value of the Maximum expiration time for subscriptions parameter, the service uses the value of Maximum expiration time for subscriptions instead. (See Service Configuration .)

- Optional element *wse:Filter* of type *wse:FilterType* or *wsman:Filter* of type *wsman:dialectableMixedDataType* . For compatibility with different toolkits, the service supports a filter of either type, but you must specify only one of them.

  Filter attribute Dialect must have the following value:

  `http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/IncidentFilter`

  Filter must contain the subelement *incFil:IncidentEventingFilter* of type *incFil:IncidentEventingFilter* . *IncidentEventingFilter* can contain any of the following subelements:

  - Optional element *incFil:Severity* of type *inc:Severity_OpenType*
  - Optional element *incFil:EmittingNode* of type *incFil:EmittingNode*
  - Optional element *incFil:Category* of type *xs:string*
  - Optional element *incFil:Application* of type *xs:string*
  - Optional element *incFil:Object* of type *xs:strin* g
  - Optional element *incFil:EmittingCI* of type *incFil:EmittingCI*
  - Optional element *incFil:CorrelationKey* of type *xs:string*
  - Optional element *incFil:Type* of type *xs:string*
  - Optional element *incFil:EscalationStatus* of type *xs:string*
  - Optional element *incFil:ConditionMatched* of type *xs:boolean*
  - Optional element *incFil:ForwardToTroubleTicket* of type *xs:boolean*

    incFil:ForwardToTroubleTicket enables you to filter incidents, depending on whether the corresponding HPOM message is flagged for forwarding to an external trouble ticket system.

  - Optional element *incFil:ForwardToNotification* of type *xs:boolean*

    incFil:ForwardToNotification enables you to filter incidents depending on whether the corresponding HPOM message is flagged for forwarding to an external notification system.

  - Optional element *incFil:Title* of type *incFil:KeywordFilter*

- Optional elements *incFil:ChangeType* of type *xs:string*

  incFil:ChangeType enables you to filter incidents depending on how they have changed (whether they are new, updated, closed, or reopened). *incFil:ChangeType* must contain one of the following strings:

  - new
  - modified
  - closed
  - reopened

  For example, if you are interested in new incidents only, you specify a filter with a *incFil:ChangeType* subelement that contains new . To subscribe to a combination of change types, specify several *incFil:ChangeType* subelements.

  To subscribe to all change types, you can omit the *incFil:ChangeType* subelements.

- Optional element *incFil:CustomAttributes* of type *incFil:CustomAttributes*

The service enumerates all incidents that match the contents of the *incFil:IncidentEventingFilter* subelements that you specify. For more details on incident attributes, see Incident to Message Mappings .

If you omit *Filter* , the service enumerates all incidents that have the status open or work in progress.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

For an example, see SOAP Envelope Examples .

Output

The SOAP body contains an element *wse:SubscribeResponse* (of anonymous type), which contains the following subelements:

- Element *wse:SubscriptionManager* of type *wsa:EndpointReferenceType* .

  Contains the subelement *wsa:ReferenceParameters/wse:Identifier* of type *wse:Identifier* , which you use to identify the subscription if you unsubscribe or renew it. (See UnsubscribeOp and RenewOp .)

- Element *wse:Expires* of type *wse:ExpirationType* .

  Contains a duration, for which the service maintains the subscription. You can use the RenewOp operation to renew the subscription before it expires. (See RenewOp .)

- Element *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

  Contains a string that identifies the enumeration context. Use *EnumerationContext* with the PullOp operation to get batches of updated incidents from the service. (See PullOp .)

Fault

If it cannot return the enumeration context, the service returns a fault according to the WS-Management standard.

UnsubscribeOp

This operation cancels a subscription before the subscription expires.

Input

SOAP Body

Element *wse:Unsubscribe* of type *wse:UnsubscribeType.* This element is required, but should be empty.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.xmlsoap.org/ws/2004/08/eventing/Unsubscribe`

- Element *wse:Identifier* of type *xs:anyURI* .

  Contains the identifier of an active subscription. You can use the element *wse:SubscriptionManager/wsa:ReferenceParameters/wse:Identifier* from the output of a SubscribeOp operation. (See SubscribeOp .)

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot cancel the subscription, the service returns a fault according to the WS-Management standard.

RenewOp

This operation renews a subscription before it expires.

Input

SOAP Body

Element *wse:Renew* (of anonymous type), which contains the optional subelement *wse:Expires* of type *wse:ExpirationType* .

Contains either a duration that specifies how long the client requires the subscription, or a date and time at which the subscription should expire. If you omit this subelement, the service uses the value of the SubscriptionExpiration parameter on the management server. If you specify a duration that exceeds the value of the SubscriptionExpirationMaximum parameter, the service uses the value of SubscriptionExpirationMaximum instead. (See Service Configuration .)

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/08/eventing/Renew
  ```

- Element *wse:Identifier* of type *xs:anyURI* .

  Contains the identifier of an active subscription. You can use the element *wse:SubscriptionManager/wsa:ReferenceParameters/wse:Identifier* from the output of a SubscribeOp operation. (See SubscribeOp .)

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *wse:RenewResponse* (of anonymous type), which contains the subelement *wse:Expires* of type *wse:ExpirationType* .

wse:Expires contains the new date and time at which the subscription expires.

Fault

If it cannot renew the subscription, the service returns a fault according to the WS-Management standard.

OwnMany

This operation sets the value of the *ismWorkItem:AssignedOperator/ismWorkItem:PersonAttributes/Name* element for one or more incidents. The incidents are identified by incident IDs. The operation sets value of the *Name* element to the user name of the currently authenticated HPOM user. (User Authentication .)

Input

SOAP Body

Element *incExt:IncidentIDs* of type *incExt:IncidentIDs* , which contains any number of *incExt:id* elements of type *xs:string* .

Each *incExt:id* contains the ID of an incident to own.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/OwnMany
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot own the incidents, the service returns a fault according to the WS-Management standard.

DisownMany

This operation clears the value of the *ismWorkItem:AssignedOperator/ismWorkItem:PersonAttributes/Name* element for one or more incidents. The incidents are identified by incident IDs.

Input

SOAP Body

Element *incExt:IncidentIDs* of type *incExt:IncidentIDs* , which contains any number of *incExt:id* elements of type *xs:string* .

Each *incExt:id* contains the ID of an incident to disown.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/DisownMany
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Optional element *OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot disown the incidents, the service returns a fault according to the WS-Management standard.

GetAnnotations

This operation returns the annotations for one incident, which is identified by the incident ID.

Input

SOAP Body

Empty.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/GetAnnotations
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance to get annotations for.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *incExt:Annotations* of type *incExt:Annotations* , which contains any number of *incExt:Annotation* elements of type *incExt:Annotation* .

Each *incExt:Annotation* contains the following subelements:

- Optional element *incExt:Author* of type *xs:string* .
- Element *incExt:Text* of type *xs:string* .
- Optional element *incExt:Date* of type *xs:dateTime* .
- Optional element *incExt:ID* of type *xs:string* .

Fault

If it cannot return the annotations, the service returns a fault according to the WS-Management standard.

AddAnnotation

This operation stores a new annotation to an existing incident. The incident is identified by incident ID.

Input

SOAP Body

Element *incExt:AnnotationText* of type *xs:string* , which contains the text of the annotation to add. (HPOM sets the annotation's *incExt:ID* , *incExt:Date* , and *incExt:Author* automatically.)

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/AddAnnotation
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance to add the annotation to.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains element *incExt:AnnotationId* of type *xs:string* . This is the ID that HPOM generates for the annotation.

Fault

If it cannot add the annotation, the service returns a fault according to the WS-Management standard.

UpdateAnnotation

This operation updates an annotation to an existing incident. The annotation to update is identified by annotation ID. The existing incident is identified by incident ID.

Input

SOAP Body

Element *incExt:UpdateAnnotation* of type *incExt:UpdateAnnotation* , which contains the following subelements:

- Element *incExt:AnnotationId* of type *xs:string* .

  Contains the ID of the annotation to update.

- Element *incExt:AnnotationText* of type *xs:string* .

  Contains the updated text for the annotation.

(HPOM updates the annotation's *incExt:Date* and *incExt:Author* automatically.)

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/UpdateAnnotation`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

    - *wsman:Selector* attribute Name must have the value IncidentID .
    - *wsman:Selector* must contain the ID of the incident instance that contains the annotation.
- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot update the annotation, the service returns a fault according to the WS-Management standard.

DeleteAnnotation

This operation deletes an annotation from an existing incident. The annotation to delete is identified by annotation ID. The existing incident is identified by incident ID.

Input

SOAP Body

Element *incExt:AnnotationId* of type *xs:string* , which contains the ID of the annotation to delete.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/DeleteAnnotation`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance that contains the annotation to delete.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot delete the annotation, the service returns a fault according to the WS-Management standard.

SetCustomAttribute

This operation creates or updates a custom attribute for an existing incident. The existing incident is identified by incident ID.

Input

SOAP Body

Element *incExt:CustomAttribute* of type *incExt:CustomAttribute* , which contains the following subelements:

- Element *incExt:Key* of type *xs:string* .

  Contains the key for the custom attribute. If a custom attribute with this key exists, the operation updates the existing a custom attribute. If a custom attribute with the specified key does not exist, the operation stores an new custom attribute.

- Element *incExt:Text* of type *xs:string* .

  Contains the value of the custom attribute.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/SetCustomAttribut
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance that contains the custom attribute.
- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot create or update the custom attribute, the service returns a fault according to the WS-Management standard.

DeleteCustomAttribute

This operation deletes a custom attribute from an existing incident. The existing incident is identified by incident ID.

Input

SOAP Body

Element *incExt:CustomAttributeKey* of type *xs:string* , which contains the key of the custom attribute to delete.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/DeleteCustomAttr:`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:Selector* :

  - *wsman:Selector* attribute Name must have the value IncidentID .
  - *wsman:Selector* must contain the ID of the incident instance that contains the custom attribute to delete.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot delete the custom attribute, the service returns a fault according to the WS-Management standard.

CloseMany

This operation sets the lifecycle state of multiple existing incidents to closed. The incidents are identified by incident IDs.

Input

SOAP Body

Element *incExt:IncidentIDs* of type *incExt:IncidentIDs* , which contains any number of *incExt:id* elements of type *xs:string* .

Each *incExt:id* contains the ID of an incident to close.

SOAP Header

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot close the incidents, the service returns a fault according to the WS-Management standard.

ReopenMany

This operation sets the lifecycle state of multiple closed incidents to open. The incidents are identified by incident IDs.

Input

SOAP Body

Element *incExt:IncidentIDs* of type *incExt:IncidentIDs* , which contains any number of *incExt:id* elements of type *xs:string* .

Each *incExt:id* contains the ID of an incident to close.

SOAP Header

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot reopen the incidents, the service returns a fault according to the WS-Management standard. If any incident is already open, the service returns a fault and stops processing the list of incidents. The service does not roll back changes to incidents that it completed successfully before the fault occurred.

StartAction

This operation starts the automatic or operator-initiated action of an existing incident. The incident is identified by the incident ID.

Input

SOAP Body

Element *incExt:ActionType* of type *xs:string* . This can be one of the following strings:

- AutomaticAction
- OperatorAction

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/StartAction`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident`

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  wsman:Selector attribute Name must have the value IncidentID .

  wsman:Selector must contain the ID of the incident instance to get.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot start the action, the service returns a fault according to the WS-Management standard.

StopAction

This operation stops the automatic or operator-initiated action of an existing incident. The incident is identified by the incident ID.

Input

SOAP Body

Element *incExt:ActionType* of type *xs:string* . This can be one of the following strings:

- AutomaticAction
- OperatorAction

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incident/StopAction
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
  ```

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the incident instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  wsman:Selector attribute Name must have the value IncidentID .

  wsman:Selector must contain the ID of the incident instance to get.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot stop the action, the service returns a fault according to the WS-Management standard.

Tool Operations Reference

The HPOM Tool Web Service provides a Web Services Definition Language (WSDL) document, which describes the service. After you install the service, the WSDL is available from the following location on the HPOM management server:

https://< *server_name* >:< *port* >/opr-toolwebservice/ToolWebService.svc?wsdl

This WSDL document refers to the other WSDL documents and associated XML Schema Documents (XSDs), and gives their location on the management server. These documents provide complete information about the operations that the service supports. The details of how your client uses these operations depend on the web service client development toolkit that you choose.

You can develop a client using any suitable toolkit or programming language. For example, you can develop your client with one of the following web service client development toolkits:

- Apache Axis2
- Windows Communication Foundation
- Wiseman

This chapter provides a generic reference to the operations that the service provides. Prefixes and XML Namespaces Used in this Document lists the prefixes used in this section to show the namespace of each element type.

Create

This operation requests the management server to execute a specified command on a specified node. Consider using the following procedure to manage tool executions:

1. Use the SubscribeOp operation to get an enumeration context.
2. Create a tool execution.
3. Start a PullOp operation. As soon as the event queue contains events, PullOp returns a batch of updated tool executions. The updated tool executions contain the current *tool:LifecycleState* . When *tool:LifecycleState* is `finished` , the service inserts the results into *tool:Output* , *tool:ResultCode* , and *tool:FinishTime* .

   If the event queue remains empty for the specified OperationTimeout, PullOp returns a TimedOut fault.
4. Immediately after the PullOp returns updated tool executions or a fault, start another PullOp operation.

Input

SOAP Body

Element *tool:ToolExecution* of type *tool:ToolExecution* . *tool:ToolExecution* must contain the following subelements:

- *tool:Command* of type *xs:string* .

  Contains a string that specifies the tool to execute. The command can be either of the following:

  - the unique ID of an existing tool on the management server
  - an executable with all parameters
  - a script

  The command must be in the format that the agent expects. The service does no parameter replacement or other modification on this string.
- *tool:CommandType* of type *tool:CommandType_OpenType*

  Contains a string that specifies the type of command. This string can be any value, including the following strings:

  - `server-defined`

    To use this command type, *tool:Command* must contain the unique ID of a tool.
  - `executable`

- `vbscript`
- `jscript`
- `perl`
- `wshost`

- *tool:Node* /*ismNode:DnsName*

  Contains a string that specifies the fully qualified domain name of the node on which to execute the command. The node must exist on the HPOM management server.

tool:ToolExecution can contain any of the following subelements:

- *tool:AdditionalParameters*

  Contains a string that specifies parameters for an existing tool on the management server. The *tool:CommandType* must be `server-defined` , and the existing tool must allow operators to change parameters.

- *tool:ReplacementVariables* of type tool:ReplacementVariables

  Contains any number of subelements of type *tool:ReplacementVariable* . Each *tool:ReplacementVariable* contains a name-value pair, which the service uses to replace an environment variable in the command of an existing tool on the management server. The *tool:CommandType* must be `server-defined` . When you launch a tool from the HPOM user interface, the management server automatically replaces the environment variables with values. When you execute a tool using the tool with the tool web service, you have to specify the values.

- *tool:User* of type *xs:string*

  Contains a string that specifies the name of the user to execute the command on the node. *tool:User* can contain the name of a real user or the string $AGENT_USER . The behavior is the same as when you execute a tool from within the HPOM user interface.

- *tool:Password* of type *xs:string*

  Contains the password of the specified *tool:User* . If you specify *tool:User* without *tool:Password* , the agent attempts to switch to the specified user without the password. After the service starts the tool execution, the service replaces the password with asterisks (*).

- *tool:Context* of type *xs:string*

  Contains any string that you want to store with the tool execution. The tool web service does not use this string.

- *tool:SessionId* of type *xs:string*

  Contains any string that you want to store with the tool execution. The tool web service does not use this string.

- *tool:Display* of type *xs:string*

  Contains a display environment variable, which the agent sets before it starts the tool.You can use this to redirect the display for X programs on nodes with a UNIX or Linux operating system.

tool:ToolExecution can also contain other valid subelements, although this operation ignores them.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *wsa:EndpointReference* of type *wsa:EndpointReferenceType* . This element contains the subelements *wsman:ResourceURI* and *wsman:SelectorSet* , which you can use to uniquely identify the tool execution in subsequent operations.

Fault

If the service cannot execute the tool, it returns a fault according to the WS-Management standard.

Delete

This operation deletes an existing tool execution, which is identified by the tool execution ID. The management server sends a request to the agent to stop the tool.

Input

SOAP Body

Empty.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution`

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the tool execution instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute **`Name`** must have the value **`ID`** .
  - *wsman:Selector* must contain the ID of the tool execution to delete.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If the service cannot stop the tool execution, it returns a fault according to the WS-Management standard.

EnumerateOp

This operation returns an enumeration context, which you can then use with the PullOp operation to get batches of tool executions from the service. You can specify a filter, so that the enumeration context contains only specific tool executions.

Input

SOAP Body

Element *Enumerate* of type *wsen:Enumerate* . This element can contain the following subelements:

- Optional element *wsen:Expires* of type *wsen:ExpirationType* .

  Contains a duration, for which the client requires the enumeration context. If you omit this subelement, the service uses the value of the Default expiration time for enumerations parameter on the management server. If you specify a duration that exceeds the value of the Maximum expiration time for enumerations parameter, the service uses the value of Maximum expiration time for enumerations instead. (See Service Configuration .)

- Optional element *wsen:Filter* of type *wsen:FilterType* or *wsman:Filter* of type *wsman:dialectableMixedDataType* . For compatibility with different toolkits, the service supports a filter of either type, but you must specify only one of them.

  Filter attribute **`Dialect`** must have the following value:
  `http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecutionFilter`

  Filter must contain the subelement *toolFil:ToolExecutionEnumerationFilter* of type *toolFil:ToolExecutionEnumerationFilter* . *ToolExecutionEnumerationFilter* can contain any of the following subelements:

  - Optional element *tool:ID* of type *xs:string*
  - Optional element *tool:SessionId* of type *xs:string*

  The service enumerates tool executions that match the contents of the *toolFil:ToolExecutionEnumerationFilter* subelements that you specify.

  If you omit *Filter* , the service enumerates all tool executions.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

`http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution`

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *wsen:EnumerateResponse* (of anonymous type), which contains the following subelements:

- Element *wsen:Expires* of type *wsen:ExpirationType* .

  Contains a duration, for which the enumeration context is valid. You can use the ReleaseOp operation to cancel the enumeration context early. (See ReleaseOp .)

- Element *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

  Contains a string that identifies the enumeration context. Use *wsen:EnumerationContext* with the PullOp operation to get batches of tool executions from the service. (See PullOp .)

Fault

If it cannot return the enumeration context, the service returns a fault according to the WS-Management standard.

Get

This operation returns one tool execution, which is identified by the tool execution ID.

Input

SOAP Body

Empty.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
  ```

- Element *wsman:SelectorSet* of type *wsman:SelectorSetType*

  Contains one element of type *wsman:SelectorType* , which identifies the tool execution instance. You can use the element *wsa:ReferenceParameters/wsman:SelectorSet* from an element of type *wsa:EndpointReferenceType* . (See Endpoint References .)

  Alternatively, create a *wsman:SelectorSet* that contains one *wsman:Selector* element of type *wsman:SelectorType* :

  - *wsman:Selector* attribute **`Name`** must have the value **`ID`** .
  - *wsman:Selector* must contain the ID of the tool execution to get.

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *tool:ToolExecution* of type *tool:ToolExecution* .

Fault

If it cannot return the tool execution, the service returns a fault according to the WS-Management standard.

SubscribeOp

This operation returns an enumeration context, which you can then use with the PullOp operation to get batches of updated tool executions from the service.

Input

SOAP Body

Element *wse:Subscribe* of type *wse:SubscribeType* , which contains the following subelements:

- Element *wse:Delivery* of type *wse:DeliveryType* . *wse:Delivery* attribute **`Mode`** must have the following value:

  `http://schemas.dmtf.org/wbem/wsman/1/wsman/Pull`

- Optional element *wse:Expires* of type *wse:ExpirationType* .

  Contains either a duration that specifies how long the client requires the subscription, or a date and time at which the subscription should expire. If you omit this subelement, the service uses the value of the Default expiration time for subscriptions parameter on the management server. If you specify a duration that exceeds the value of the Maximum expiration time for subscriptions parameter, the service uses the value of Maximum expiration time for subscriptions instead. (See Service Configuration .)

- Optional element *wsen:Filter* of type *wsen:FilterType* or *wsman:Filter* of type *wsman:dialectableMixedDataType* . For compatibility with different toolkits, the service supports a filter of either type, but you must specify only one of them.

  Filter attribute **`Dialect`** must have the following value:
  `http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecutionFilter`

  Filter must contain the subelement *toolFil:ToolExecutionEventingFilter* of type *toolFil:ToolExecutionEventingFilter* . *ToolExecutionEventingFilter* can contain any of the following subelements:

  - Optional element *tool:ID* of type *xs:string*
  - Optional element *tool:SessionId* of type *xs:string*

  The service enumerates tool executions that match the contents of the *toolFil:ToolExecutionEventingFilter* subelements that you specify.

  If you omit *Filter* , the service enumerates all tool executions.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution`

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *wse:SubscribeResponse* (of anonymous type), which contains the following subelements:

- Element *wse:SubscriptionManager* of type *wsa:EndpointReferenceType* .

  Contains the subelement *wsa:ReferenceParameters/wse:Identifier* of type *wse:Identifier* , which you use to identify the subscription if you unsubscribe or renew it. (See UnsubscribeOp and RenewOp .)

- Element *wse:Expires* of type *wse:ExpirationType* .

  Contains a duration, for which the service maintains the subscription. You can use the RenewOp operation to renew the subscription before it expires. (See RenewOp .)

- Element *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

  Contains a string that identifies the enumeration context. Use *EnumerationContext* with the PullOp operation to get batches of updated tool executions from the service. (See PullOp .)

Fault

If it cannot return the enumeration context, the service returns a fault according to the WS-Management standard.

RenewOp

This operation renews a subscription before it expires.

Input

SOAP Body

Element *wse:Renew* (of anonymous type), which contains the optional subelement *wse:Expires* of type *wse:ExpirationType* .

Contains either a duration that specifies how long the client requires the subscription, or a date and time at which the subscription should expire. If you omit this subelement, the service uses the value of the SubscriptionExpiration parameter on the management server. If you specify a duration that exceeds the value of the SubscriptionExpirationMaximum parameter, the service uses the value of SubscriptionExpirationMaximum instead.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/08/eventing/Renew
  ```

- Element *wse:Identifier* of type *xs:anyURI* .

  Contains the identifier of an active subscription. You can use the element *wse:SubscriptionManager/wsa:ReferenceParameters/wse:Identifier* from the output of a SubscribeOp operation. (See SubscribeOp .)

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body contains an element *wse:RenewResponse* (of anonymous type), which contains the subelement *wse:Expires* of type *wse:ExpirationType* .

wse:Expires contains the new date and time at which the subscription expires.

Fault

If the service cannot renew the subscription, the service returns a fault according to the WS-Management standard.

UnsubscribeOp

This operation cancels a subscription before the subscription expires.

Input

SOAP Body

Element *wse:Unsubscribe* of type *wse:UnsubscribeType.* This element is required, but should be empty.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.xmlsoap.org/ws/2004/08/eventing/Unsubscribe`

- Element *wse:Identifier* of type *xs:anyURI* .

  Contains the identifier of an active subscription. You can use the element *wse:SubscriptionManager/wsa:ReferenceParameters/wse:Identifier* from the output of a SubscribeOp operation. (See SubscribeOp .)

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution`

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot cancel the subscription, the service returns a fault according to the WS-Management standard.

PullOp

This operation returns a batch of tool executions from an enumeration context.

Input

SOAP Body

Element *wsen:Pull* (of anonymous type), which contains the following subelements:

- Element *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

  Contains a string that identifies the enumeration context. Use a *wsen:EnumerationContext* that one of the following operations returns:

  - EnumerateOp (see EnumerateOp)
  - SubscribeOp (see SubscribeOp)
  - PullOp (see below)

- Optional element *wsen:MaxElements* of type *xs:positiveInteger* .

  Contains an integer that indicates the maximum number of tool executions to return in this batch. If you omit this subelement, the service uses the value of the Default maximum number of items returned by service parameter on the management server. (See Service Configuration .)

The HPOM Tool Web Service does not support the subelements *wsen:MaxTime* or *wsen:MaxCharacters* . You should omit these subelements.

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  `http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull`

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  `http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution`

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

For enumeration contexts from the SubscribeOp operation, if the event queue remains empty for the specified OperationTimeout, PullOp returns a TimedOut fault.

Output

The SOAP body contains an element *wsen:PullResponse* (of anonymous type), which contains the following subelements:

- Element *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

  Contains a string that identifies the enumeration context. This is the same as the input *wsen:EnumerationContext.*
- Element *wsen:Items* of the type *wsen:ItemListType* .

  Contains one or more *tool:ToolExecution* elements of type *tool:ToolExecution* .
- Optional element *wsen:EndOfSequence* of type *wsen:attributableEmpty* .

  This element is empty. If *wsen:EndOfSequence* is present, there are no remaining items to pull for this enumeration context and the enumeration context is no longer valid.

Fault

If the service cannot return the batch of tool executions, the service returns a fault according to the WS-Management standard.

ReleaseOp

This operation cancels an existing enumeration context early (that is, before the client has pulled all the tool executions, and before the enumeration context has expired).

You can use this operation to cancel an enumeration context from an enumerate operation. To cancel an enumeration context from a subscription operation, unsubscribe instead. (See UnsubscribeOp .)

Input

SOAP Body

Element *wsen:Release* (of anonymous type), which contains the subelement *wsen:EnumerationContext* of type *wsen:EnumerationContextType* .

wsen:EnumerationContext contains a string that identifies the enumeration context. Use a *wsen:EnumerationContext* that one of the following operations returns:

- EnumerateOp (see EnumerateOp)
- PullOp (see PullOp)

SOAP Header

- Element *wsa:Action* of type *wsa:ActionType* that contains the following string:

  ```
  http://schemas.xmlsoap.org/ws/2004/09/enumeration/Release
  ```

- Element *wsman:ResourceURI* of type *wsman:AttributableURI*

  Contains a string that identifies the resource type. You can use the element *wsa:ReferenceParameters/wsman:ResourceURI* from an element of type *wsa:EndPointReferenceType* .

  Alternatively, create a *wsman:ResourceURI* element that contains the following string:

  ```
  http://schemas.hp.com/opr/ws/ServiceOperation/ToolManagement/1/ToolExecution
  ```

- Optional element *wsman:OperationTimeout* of type *wsman:AttributableDuration*

  On an HPOM for Windows management server, the service ignores this element.

Output

The SOAP body is empty.

Fault

If it cannot release the enumeration context, the service returns a fault according to the WS-Management standard.

Security for Web Service Clients

When the client that you develop connects to an HPOM Web Service, it must use HTTP basic authentication to specify the username and password of a valid HPOM user. To ensure that malicious users cannot intercept these credentials and other communications, HP recommends that you use HTTPS connections.

The following topics provide more details:

- User Authentication
- Secure HTTP Connections

User Authentication

A client must connect to an HPOM Web Service using HTTP basic authentication. Use your normal client development toolkit to specify a username and password in the client's HTTP requests.

The username and password that you specify must be those of a valid HPOM user. The user must have appropriate rights to perform the operations that the client attempts.

On HPOM for Windows, if the user that you specify is an operator, the client may have more permissions than the operator would have in the HPOM console. In particular, any restrictions from user roles do not apply. For example, the Incident Web Service can enumerate all messages, and change messages that are not already owned by another user.

In contrast, on HPOM for UNIX, if the user that you specify is an operator, the client has the same permissions as the operator would have in the HPOM console. For example, the Incident Web Service EnumerateOp returns only incidents that the operator would see as messages in the console.

Secure HTTP Connections

HP recommends that you connect to the HPOM Web Services using HTTPS connections, which require a suitable certificate on the server. The management server installation creates a self-signed certificate for HTTPS communication, but you can replace this with a different certificate if necessary. The port that the service uses for HTTPS communication depends on the configuration of the HPOM management server.

The default HTTPS port number on HPOM for Windows is 443 .

For further security, HP recommends that you verify the hostname and certificate for each HTTPS connection. To verify the certificate for an HTTPS connection, the client system must trust the server's certificate. You may need to export the server's certificate and import it to the client system.

Export the certificate from the security settings in Internet Information Services Manager using the following options:

- Do not export the private key.
- Select the export format DER encoded binary X.509 (.CER).

After you export the server's certificate from the management server, you must import it using the appropriate tools for your client environment. You can then program your client to verify HTTPS connections using the methods that your client development toolkit provides.

Basic Troubleshooting

If you experience problems with the HPOM Web Services, check that your client can connect to the service over the network, and that the service is available.

The following topics provide more details:

- Troubleshoot Connectivity
- Troubleshoot on HPOM for Windows

Troubleshoot Connectivity

- Check the network connectivity from your client system to the HPOM management server. For example:
  - Use nslookup to check that the client system can resolve the management server’s hostname.
  - Use ping to check that the client system can reach the management server.
- Check the connectivity from your client system to the HPOM Web Service.

  Open a web browser and navigate to the following locations on the HPOM management server:

  - `https://<server_name>:<port>/opr-webservice/Incident.svc?wsdl`
  - `https://<server_name>:<port>/opr-toolwebservice/ToolWebService.svc?wsdl`

  The port that the service is available on depends on the configuration of the HPOM management server. The service listens for HTTPS connections on port 443 by default.

  You may need to accept a certificate and provide credentials for the HPOM management server. If your browser cannot open the WSDL document, check that the service is installed and running. (See Troubleshoot on HPOM for Windows .)

Troubleshoot on HPOM for Windows

- The HPOM Web Services service run within Microsoft Internet Information Services (IIS). Check the status of the World Wide Web Publishing Service to ensure that it is started. The minimum you must do to restart the service is restart the default web site using Microsoft Internet Information Services Manager. To ensure a clean startup, you can restart the World Wide Web Publishing Service.
- Check the installation of the HPOM Incident Web Service as follows:
  - Open Windows Explorer, and then navigate to the following folder:

    `%OvInstallDir%www\webapps\omws-incident`

    The folder should contain the files `Incident.svc` , `PrecompiledApp.config` , `Web.config` , and the subfolders bin and `Metadata` .

    The bin subfolder should contain nine DLL files.

    If any of the files or folders are missing, reinstall the service.
  - Verify the configuration of Microsoft Internet Information Server (IIS):

    In the Internet Information Services Manager console tree, click Application Pools → DefaultAppPool . Verify that the application HP Operations Manager Incident Web Service exists.

    In the console tree, click Web Sites →Default Web Site . Check that the web site omws exists.

    If either the application or web site is missing, reinstall the service.

  To reinstall the service:

  a. Copy the installation package (`HPOprWsInc-<`*`version`*` >-WinNT4.0-release.msi`) to a temporary folder on the HPOM for Windows management server.
  b. Open a command prompt, and navigate to the temporary folder that contains the installation package. Type the following command:

     `msiexec /i HPOprWsInc-<` *`version`* **`>-WinNT4.0-release.msi /qn`**
- Check the following log file for errors and warnings:

  `%OvShareDir%server\log\om\incident-ws.trace.txt`

  This file exists only after the service receives the first request from a client.

SOAP Envelope Examples

The following examples show SOAP envelopes that the HPOM Incident Web Service receives and sends for various operations.

Create Request SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/09/transfer/Create</a:Act
    <h:OperationTimeout xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
      PT10M0.00S
    </h:OperationTimeout>
    <h:ResourceURI xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
      http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
    </h:ResourceURI>
    <a:MessageID>urn:uuid:2be9d179-a8be-476d-8558-576e5d2283b8</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://manager1.example.com/opr-webservice/Incident.svc</a:To>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
    <Incident xmlns="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident">
      <Title>My custom app has serious problems</Title>
      <Severity>Major</Severity>
      <Category>CustomApplications</Category>
      <CollaborationMode>fyi</CollaborationMode>
      <EmittingCI>
        <ConfigurationItemProperties
          xmlns="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Configur
          <ID>92384-1232-234-2134</ID>
        </ConfigurationItemProperties>
      </EmittingCI>
      <EmittingNode>
        <NodeProperties xmlns="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagem
          <DnsName>host.example.com</DnsName>
        </NodeProperties>
      </EmittingNode>
      <Extensions>
        <OperationsExtension xmlns="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManag
```

```
          <Application>My custom app</Application>
          <Object>Custom Apps</Object>
          <NumberOfDuplicates>0</NumberOfDuplicates>
          <ConditionMatched>false</ConditionMatched>
          <AutomaticActionStatus>notAvailable</AutomaticActionStatus>
          <OperatorActionStatus>notAvailable</OperatorActionStatus>
          <EscalationStatus>notEscalated</EscalationStatus>
          <CustomAttributes>
            <CustomAttribute>
              <Key>Customer</Key>
              <Text>VIP</Text>
            </CustomAttribute>
          </CustomAttributes>
          <NumberOfAnnotations>0</NumberOfAnnotations>
        </OperationsExtension>
      </Extensions>
    </Incident>
  </s:Body>
</s:Envelope>
```

Create Response SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateRespons
    <a:RelatesTo>urn:uuid:2be9d179-a8be-476d-8558-576e5d2283b8</a:RelatesTo>
    <a:To s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous<
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
    <ResourceCreated xmlns="http://schemas.xmlsoap.org/ws/2004/09/transfer">
      <a:Address>http://manager1.example.com/opr-webservice/Incident.svc</a:Address>
      <a:ReferenceParameters>
        <SelectorSet xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
          <Selector Name="IncidentID">a1a822f0-d485-40d5-96bb-2e42b6dd961a</Selector>
        </SelectorSet>
        <ResourceURI xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
          http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
        </ResourceURI>
      </a:ReferenceParameters>
    </ResourceCreated>
  </s:Body>
</s:Envelope>
```

Close Request SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.hp.com/ism/ServiceOperation/Common/1/Close</a
    <h:ResourceURI xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
      http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
    </h:ResourceURI>
    <h:SelectorSet xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd" xmlns:xsi="http://www.w3.org/2001/
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <Selector Name="IncidentID">527c6290-c5b8-71dc-1691-103941590000</Selector>
    </h:SelectorSet>
    <h:OperationTimeout xmlns:h="http://schemas.xmlsoap.org/ws/2005/06/management"
      xmlns="http://schemas.xmlsoap.org/ws/2005/06/management">
      PT10M0.00S
    </h:OperationTimeout>
    <a:MessageID>urn:uuid:73f6adda-7b30-4d4d-b1ae-d0c2848527b9</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://manager1.example.com/opr-webservice/Incident.svc</a:To>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
</s:Envelope>
```

Close Response SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.hp.com/ism/ServiceOperation/Common/1/CloseRes
    <a:RelatesTo>urn:uuid:73f6adda-7b30-4d4d-b1ae-d0c2848527b9</a:RelatesTo>
    <a:To s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous<
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
</s:Envelope>
```

Enumerate Request SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate<
    <h:OperationTimeout xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
      PT10M0.00S
    </h:OperationTimeout>
    <h:ResourceURI xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
      http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
    </h:ResourceURI>
    <a:MessageID>urn:uuid:9e8b81d7-a725-4693-9af3-f025045750c6</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://manager1.example.com:8081/opr-webservice/Incident.svc</a:
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
    <Enumerate xmlns="http://schemas.xmlsoap.org/ws/2004/09/enumeration">
      <Filter Dialect="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incid
        <IncidentEnumerationFilter
          xmlns="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/IncidentFil
          <Severity>Critical</Severity>
          <Severity>Major</Severity>
          <Category>Database</Category>
          <CustomAttributes>
            <CustomAttribute>
              <Key>Customer</Key>
              <Text>VIP</Text>
            </CustomAttribute>
          </CustomAttributes>
        </IncidentEnumerationFilter>
      </Filter>
    </Enumerate>
  </s:Body>
</s:Envelope>
```

Enumerate Response SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerateR
    <a:RelatesTo>urn:uuid:f449fd93-be02-43cf-a8f3-2b32b3355ce8</a:RelatesTo>
    <a:To s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous<
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
    <EnumerateResponse xmlns="http://schemas.xmlsoap.org/ws/2004/09/enumeration">
      <Expires>2008-01-18T12:41:20.968+01:00</Expires>
      <EnumerationContext>60d863fd-f058-41b1-8195-341652bd458a</EnumerationContext>
    </EnumerateResponse>
  </s:Body>
</s:Envelope>
```

Pull Request SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull</a:Ac
    <h:OperationTimeout xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
      PT1000M0.00S
    </h:OperationTimeout>
    <h:ResourceURI xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
      http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
    </h:ResourceURI>
    <a:MessageID>urn:uuid:42f16c88-6adc-49f0-8881-558fdecfecfe</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://manager1.example.com/opr-webservice/Incident.svc</a:To>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
    <Pull xmlns="http://schemas.xmlsoap.org/ws/2004/09/enumeration">
      <EnumerationContext>60d863fd-f058-41b1-8195-341652bd458a</EnumerationContext>
    </Pull>
  </s:Body>
</s:Envelope>
```

Pull Response SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/09/enumeration/PullRespon
    <a:RelatesTo>urn:uuid:42f16c88-6adc-49f0-8881-558fdecfecfe</a:RelatesTo>
    <a:To s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous<
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
    <PullResponse xmlns="http://schemas.xmlsoap.org/ws/2004/09/enumeration">
      <EnumerationContext>60d863fd-f058-41b1-8195-341652bd458a</EnumerationContext>
      <Items>
        <Incident xmlns="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incide
          <AssignedOperator xmlns="http://schemas.hp.com/ism/ServiceOperation/Common/1/WorkItem
            <PersonAttributes>
              <Name xmlns="" />
            </PersonAttributes>
          </AssignedOperator>
          <IncidentID>527c6290-c5b8-71dc-1691-103941590000</IncidentID>
          <Title>
            MS IIS Server (WWW): (1013) A process serving application pool 'DefaultAppPool' exc
            shut down. The process id was '5736'.
          </Title>
          <Type />
          <LifecycleState>open</LifecycleState>
          <Severity>Warning</Severity>
          <Category>WINOSSPI-INTERNET_SERVICE</Category>
          <CollaborationMode>fyi</CollaborationMode>
          <EmittingCI>
            <ConfigurationItemProperties
              xmlns="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Conf
              <ID>WINOSSPI:IIS60@@{9768285B-3AA5-4F49-B6DE-CB654F5AD67B}</ID>
            </ConfigurationItemProperties>
          </EmittingCI>
          <EmittingNode>
            <NodeProperties xmlns="http://schemas.hp.com/ism/ServiceTransition/ConfigurationMan
              <DnsName>host.example.com</DnsName>
            </NodeProperties>
          </EmittingNode>
          <Extensions>
            <OperationsExtension xmlns="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentM
              <Application>IIS 6.0</Application>
              <Object>W3SVC</Object>
              <StateChangeTime>2008-01-18T12:27:16</StateChangeTime>
              <CreationTime>2008-01-18T12:25:47</CreationTime>
              <ReceivedTime>2008-01-18T12:27:16</ReceivedTime>
              <NumberOfDuplicates>0</NumberOfDuplicates>
              <CorrelationKey>host.example.com:W3SVC:0x800003F5</CorrelationKey>
              <ConditionMatched>true</ConditionMatched>
              <AutomaticActionStatus>notAvailable</AutomaticActionStatus>
              <OperatorActionStatus>notAvailable</OperatorActionStatus>
              <EscalationStatus>notEscalated</EscalationStatus>
              <Source>WINOSSPI-IIS60_FwdAllSystemWarnError(10.0)</Source>
              <NumberOfAnnotations>0</NumberOfAnnotations>
              <OriginalEvent>
                Computer: host Source: W3SVC Category: None Type: Warning Event ID: 1013 Descri
                application pool 'DefaultAppPool' exceeded time limits during shut down. The pr
              </OriginalEvent>
            </OperationsExtension>
         </Extensions>
        </Incident>
```

```
      <Incident xmlns="http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incide
        <AssignedOperator xmlns="http://schemas.hp.com/ism/ServiceOperation/Common/1/WorkItem
          <PersonAttributes>
            <Name xmlns="" />
          </PersonAttributes>
        </AssignedOperator>
        <IncidentID>a4669be0-c5b6-71dc-1209-103941590000</IncidentID>
        <Title>(ctrl-45) Component 'opcacta' with pid 4704 exited. Restarting component.</Tit
        <Type />
        <LifecycleState>open</LifecycleState>
        <Severity>Major</Severity>
        <Category>OpenView</Category>
        <CollaborationMode>fyi</CollaborationMode>
        <EmittingCI>
          <ConfigurationItemProperties
            xmlns="http://schemas.hp.com/ism/ServiceTransition/ConfigurationManagement/1/Conf
            <ID />
          </ConfigurationItemProperties>
        </EmittingCI>
        <EmittingNode>
          <NodeProperties xmlns="http://schemas.hp.com/ism/ServiceTransition/ConfigurationMan
            <DnsName>host.example.com</DnsName>
          </NodeProperties>
        </EmittingNode>
        <Extensions>
          <OperationsExtension xmlns="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentM
            <Application>OpenView</Application>
            <Object>ovcd</Object>
            <StateChangeTime>2008-01-18T12:15:14</StateChangeTime>
            <CreationTime>2008-01-18T12:15:14</CreationTime>
            <ReceivedTime>2008-01-18T12:15:14</ReceivedTime>
            <NumberOfDuplicates>0</NumberOfDuplicates>
            <CorrelationKey />
            <ConditionMatched>true</ConditionMatched>
            <AutomaticActionStatus>notAvailable</AutomaticActionStatus>
            <OperatorActionStatus>notAvailable</OperatorActionStatus>
            <EscalationStatus>notEscalated</EscalationStatus>
            <Source></Source>
            <NumberOfAnnotations>0</NumberOfAnnotations>
            <OriginalEvent />
          </OperationsExtension>
        </Extensions>
      </Incident>
    </Items>
    <EndOfSequence />
  </PullResponse>
  </s:Body>
</s:Envelope>
```

Subscribe Request SOAP Envelope Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe</a:
    <h:ResourceURI xmlns:h="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
      xmlns="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
      http://schemas.hp.com/ism/ServiceOperation/IncidentManagement/1/Incident
    </h:ResourceURI>
    <h:OperationTimeout xmlns:h="http://schemas.xmlsoap.org/ws/2005/06/management"
      xmlns="http://schemas.xmlsoap.org/ws/2005/06/management">
      PT1000M0.00S
    </h:OperationTimeout>
    <a:MessageID>urn:uuid:2822dea3-af62-4e26-a1de-31d738f2bc59</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://manager1.example.com:8081/opr-webservice/Incident.svc</a:
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/20
    <Subscribe xmlns="http://schemas.xmlsoap.org/ws/2004/08/eventing">
      <Delivery Mode="http://schemas.dmtf.org/wbem/wsman/1/wsman/Pull"/>
      <Filter Dialect="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/Incid
        <IncidentEventingFilter
          xmlns="http://schemas.hp.com/opr/ws/ServiceOperation/IncidentManagement/1/IncidentFil
          <Severity>Critical</Severity>
          <Severity>Major</Severity>
          <EmittingNode>
            <NodeProperties>
              <DnsName>host.example.com</DnsName>
            </NodeProperties>
          </EmittingNode>
          <CustomAttributes>
            <CustomAttribute>
              <Key>Customer</Key>
              <Text>VIP</Text>
            </CustomAttribute>
          </CustomAttributes>
          <ChangeType>new</ChangeType>
          <ChangeType>closed</ChangeType>
        </IncidentEventingFilter>
      </Filter>
    </Subscribe>
  </s:Body>
</s:Envelope>
```

Understanding the operator's role

The view you see in your console has been preconfigured by your administrator to display information received from managed nodes and services about conditions in your environment.

Your primary daily tasks might require you to:

- Monitor the status of a service or node using predefined views.
- Troubleshoot a problem or isolate it using the appropriate combination of filters to research the problem or the appropriate view to drill down to it.
- Resolve the problem by using the appropriate tool or action. A tool is an application or program that is not associated with the event.

Your administrator-configured view allows you to perform many tasks from a single location:

- View messages to monitor the health of your managed nodes.
- Detect problems early and take corrective actions before end users are impacted.
- Select from a variety of tools, actions, information sources, and problem-solving techniques to resolve problems.
- Maintain a history for all problems you resolve by documenting the steps you took to resolve them.

Related Topics:

- Browsing messages
- Diagnosing Problems with the Map View
- Applying tools to managed nodes and services

Steps to problem resolution

Problem resolution is a multi-step process that begins when you receive an important message.

- Detect: A message indicating a problem appears in your active messages browser. From within the browser, you can see what has happened, the severity level of the message, and whether there are preconfigured actions you can take to resolve the problem.
- Evaluate: Examine the problem to determine what the problem is and where it occurred. Your administrator may have predefined instructions associated with the message to help you diagnose the problem. Several options are available to you. Use the map view to see all managed devices for a node, find the root cause of the problem, or see a view of services impacted by the problem.
- Resolve: Correct the problem. Your administrator may have preconfigured automatic actions to correct the problem. If so, the action will automatically occur. Other kinds of actions must be initiated by you. You can choose when to start these operator-initiated actions, depending on what your administrator has configured for you. When the action is completed, review to see whether the actions you started corrected the problem. For example, the initial message about the problem will no longer appear in the message browser and a new message describing the new status will appear. You can also check to see whether a tool application succeeded by looking at the Tool Status dialog box.
- Document: When the problem is resolved, make note of your results using annotations to the message and acknowledge the message. Acknowledging moves the message to the history database.

Managing your workspace

Your administrator configures the view of nodes, tools, services, and messages that you see in your console and may place certain limits on the activities you can perform. Within these administrator-defined constraints however, you are free to customize your view and save the changed configuration to use again. You might change your view in these ways:

- Define filters to display only the messages you specify.
- Define your screen layout in terms of:
  - number of windows
  - window arrangement
  - message browser sorting
  - message browser column placement
- Configure the console default view for nodes, node groups, maps, and services
- Configure map background color
- Turn error message display on or off
- Set the number of levels a map can expand
- Set the maximum number of messages that are cached

You can save your customized view for later use. Name your view with an `.msc` extension.

Related Topics:

- Save your workspace
- Configure the console
- Filter browser messages

Browse messages with the web console

The HP Operations Manager web console provides a quick and convenient way to view and respond to messages that result from events that occur on your managed nodes. From any location, use your Internet Explorer or Netscape browser to instantly see the severity of a message and act to correct the problem that caused it.

To access the complete HP Operations Manager for Windows functionality, use the native console. For quick access to the most commonly used functions from any browser, open the web console as described below.

To log in to the web console

1. From the management server, click the Start button, then select Programs → HP → HP BTO Software → Web Console to open a browser window and display the message browser.
2. From a remote console, in your web browser address or location field, enter a URL in the following format:

   `http://hostname/OVOWEB/`

   Where `hostname` is the name of the management server. So for example, if your server were named `myserver` , then your URL would be:

   `http://myserver/OVOWEB/`
3. After you enter the URL, the Enter Network Password dialog opens. Enter the Windows domain name and user name in the following form.

   `domainname\myusername`

   If your domain name is bora bora and your user name is traveler, your login would look like this:

   bora-bora\traveler

   You must be a member one of the following HP user groups:

   HP-OVE-OPERATORS

   HP-OVE-ADMINS
4. Alternatively, you can log in as a local user with your user name. You must be in one of the groups listed above.
5. When your user name and password are accepted, the browser opens and displays the web console.

In addition to the native console and the web console, you can access the console functionality using Microsoft's terminal services.

To start the web console using HTTPS

By default, the web console uses HTTP to communicate with the management server. You can, however, configure the web console and the management server to communicate using HTTPS. See the *HP Operations Manager for Windows Firewall Configuration* white paper for more information.

Configure the console

You can configure the appearance and behavior of your console by setting preferences in the Configure Console dialog box for the following areas:

- Number of levels of expansion for maps
- Map background color
- Download of node and service information at startup
- Number of WMI instances to be retrieved
- Location of results views
- Data presentation properties
- Message browser properties
- Notification properties

To configure console properties

1. Open the Configure Console dialog box if it is not already open.
2. Specify your preferences in the tabs.
3. Click Apply to apply your settings without closing the dialog box.
4. Click OK to confirm your choices and close the dialog box.

Changes take effect immediately and apply to all the consoles available to the user making the changes.

Related Topics:

- Specify console General properties
- Specify console Data Presentation properties
- Specify console Message Browser properties
- Specify console Notification properties

Specify console General properties

You can specify the appearance of maps in HPOM using the General tab of the Console Properties dialog box to set preferences for how many levels the map view will expand, map background color, and model pre-caching. In addition, you can specify how many WMI instances can be retrieved at any one time, how results views are opened, and whether the Agent Installation dialog box opens when you deploy a policy or package to a node that does not yet have an agent installed.

To specify console general properties

1. Open the Configure Console dialog box if it is not already open.
2. In the General tab, specify how the map view expands using the Number of Levels to Automatically Expand box, enter a number. You can specify from one to five levels of expansion.

   In map views where large numbers of items are displayed in the lower levels, configuring the map to expand one level at a time can increase performance. The new settings will not take effect until the map is redisplayed. Change the view to the message browser, for example, then return to the map view to see the change.
3. To change the light gray default background color of your maps, click Change Color to open the Microsoft Windows Color dialog box. Select a basic color or define a custom color, then click OK to close the Color dialog box. In the Console Properties General tab, click Apply to see the new background color.
4. Select Model Pre-caching to upload all node and service information to the console at startup. Clear this check box if you do not want this information uploaded at startup.
5. The number in the WMI instance retrieval count box determines the maximum size of the data packets that are exchanged between the management server and the console (more specifically, between the server and the node, tool, and service configuration editors). The default value of 100 means that a maximum of 100 objects (messages, nodes, or services, for example) are transmitted in each packet. This default value is valid for most HPOM environments, with the following exceptions:
   - Fast network connections: If a fast network connection is available, you can decrease the default value to a minimum of ten. Low values mean that fewer objects are transmitted in each packet so that the first packet arrives more quickly. However, because more packets are transmitted, the overall transportation overhead is higher.
   - Slow network connections: If the network connection is slow, increase the value to a maximum of 500. Higher values mean that more objects are transmitted in each packet, which slows down the arrival of the first packet, but overall, fewer packets are transmitted.
6. Clear Show result views in new window to display result views in the details pane rather than in a new window. Result views are, for example, map views, reports, and graphs.

7. The Enable agent installation dialog option determines if HPOM checks prior to each deployment whether an agent is already installed on the target nodes. If no agent is installed, the Agent Installation dialog box opens and asks you for the credentials of the node. Depending on the number of nodes, the agent installation check may take a long time to complete, during which the console is inaccessible.

   When you disable the Agent Installation dialog box, all deployment jobs start immediately. However, policy deployment jobs may fail if no agent is installed. To correct the problem, restart the failed deployment job by right-clicking the job and selecting All Tasks → Restart job with advanced options .

Related Topics:

- Configure the console
- Specify console Data Presentation properties
- Specify console Message Browser properties
- Specify console Notifications properties
- Restart job with new options
- Disable automatic agent installation

Specify console Data Presentation properties

Set preferences for data presentation in your console using the Data Presentation tab of the Configure Console dialog box. You can specify default views for nodes, node groups, and services with and without subservices.

To set default view preferences

1. Open the Configure Console dialog box if it is not already open. ▼
2. Select the Data Presentation tab.
3. *Optional.* Select the Position duplicate messages as new message in browser check box. If you select this check box, when a duplicate message arrives, the duplicate count of the original message increases, and the message moves to the bottom of the message list.
4. Select the default view for Nodes from the list. Choose Maps, Message Browser, List, Impacted, or Root Cause.
5. Choose a filter for the selected Node view from the list. Options depend on the selection you made in step 4.

   If you select Maps, you can choose from Contained By or Hosting Services Map as your default view. See Understanding Uses and Contains Relationships for more information about available map types. If you select Message Browser, you can display either the active or acknowledged messages browser in your default view.

   If you select List, Impacted, or Root Cause, no filtered view is available.
6. Follow the same procedure for Node Groups .

   If you select Maps, you can select your default view from Contains or Contained By map views.

   If you select Message Browser, you can display either the active or acknowledged messages browser in your default view.

   If you select List, Impacted, or Root Cause, no filtered view is available.
7. Select the default view for Service without subservice. Choose Maps, Message Browser, or List.
8. Choose a filter for the selected Service with subservice view. Options depend on the selection you made in step 7.

   If you select Map you can select your default view from Contains/Uses, Contained By/Used By, Uses, Used By, Contains, ContainedBy and Service Hosted On map views.

   If you select Message Browser , you can display either the active or acknowledged messages browser in your default view.

If you select List, no filtered view is available.

9. Select the default view for Service without subservice . Choices are the same as those in step 8.
10. Choose a filter for the selected Service without subservice view. Choices are the same as in step 8.
11. Select Position duplicate messages as new messages in local browser to treat duplicate messages the same way you treat new messages.
12. Click Apply to see your changes take effect. Click OK to confirm your choices and close this dialog box.

Related Topics:

- Configure the console
- Specify console General properties
- Specify console Message Browser properties
- Specify console Notifications properties
- Configure duplicate message suppression

Specify console message browser properties

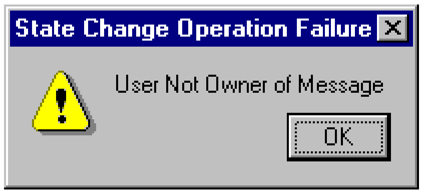
You can specify message browser appearance using the Message Browser tab of the Console Properties dialog box to set preferences for whether error messages display, the color of messages, and the maximum number of displayed messages.

To specify message browser properties

1. Open the Configure Console dialog box if it is not already open.
2. In the Message Browser tab, check the Display Error Dialogs from Message Browser check box to display error messages. Clear the check box if you do not want to view these error messages.

   Turn the display of messages on or off in your active messages browser for messages on which you perform mass operations. This can save you time so that you do not repeat the closure of the same error message box for a block of messages.

   For example, assume your active messages browser has a block of 1,000 messages that are owned by another operator. You decide to select all 1,000 messages to perform a mass operation. Before the operation can complete, the following error message displays:

   
   State Change Operation Failure

   User Not Owner of Message

   To continue working, you would have to select OK and close each of these error messages. To avoid having to click OK for each error message, you can turn off the display of all the error messages by clearing the Display Error dialogs checkbox. This turns off the errors associated with all messages that are owned, disowned, acknowledged, and unacknowledged. It can also be used on messages where the severity has been changed.
3. To specify the coloring of messages in the message browser, click one of the following options:
   - Do not color messages
     Messages have no background color.
   - Color messages, highlight owned messages
     Messages that you own have a background color according to their status. Messages that other users own have a light gray background.
   - Color messages, highlight unowned messages
     Unowned messages have a background color according to their status. Messages that other users own have a light gray background.

4. To set the number of messages that you want to display, enter the number in the Maximum Number of Displayed Messages text box and click OK . There is no limit to the number of messages you can display.

   To verify your changes, display the active messages browser. The display limit appears in the status bar in the lower right corner of your message browser window.

   When the default message limit is exceeded, the oldest messages are removed from view at the rate of 10% at a time. The browser removes all messages in those views not being looked at (inactive views) and also removes the view. For example, if you have an active view and three inactive views in the background, to remain within the message cache setting, the browser would first remove the inactive views, then remove 10% of the messages in the active view.

Related Topics:

- Configure the console
- Specify console General properties
- Specify console Data Presentation properties
- Specify console Notifications properties

Specify console Notifications properties

You can specify how to be notified in the console when new messages arrive using the Notifications tab of the Console Properties dialog box, using one or all of the following options:

- System tray popup message
- System tray icon display
- Sound notification

To specify notifications properties

1. Open the Configure Console dialog box if it is not already open.
2. In the Notifications tab, select the Enable system tray popup to display a notice that a new message has arrived in the message browser. The default is to show the message for all message severities. Use the list box to restrict the message display to the severity level you prefer.
3. Select Enable system tray icon to receive notification in the Windows system tray. The default is to show the icon for all message severities. The list box contains the same severity levels as the Enable system tray popup .
4. Select Enable sounds to receive an audible notification when a new message arrives. Use the Browse button to select the sound you prefer. The default is to play the sound for all message severities. The list box contains the same severity levels as the Enable system tray popup .

Related Topics:
- Configure the console
- Specify console General properties
- Specify console Data Presentation properties
- Specify console Message Browser properties

Console menu

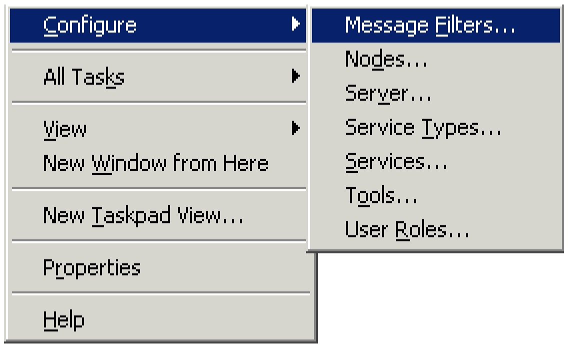
The console menu that appears when you right-click a selected item in the console tree displays various options, depending on your original selection. In other words, its options change according to the context in which you open it. In addition, the administrator's view provides options for configuring services, nodes, tools, service types, and user roles.

The console menu can also be opened by selecting the Action menu in the Microsoft Management Console (MMC) toolbar.

The MMC and HP Operations Manager for Windows make extensive use of context, or shortcut menus. These menus may contain commands provided by:

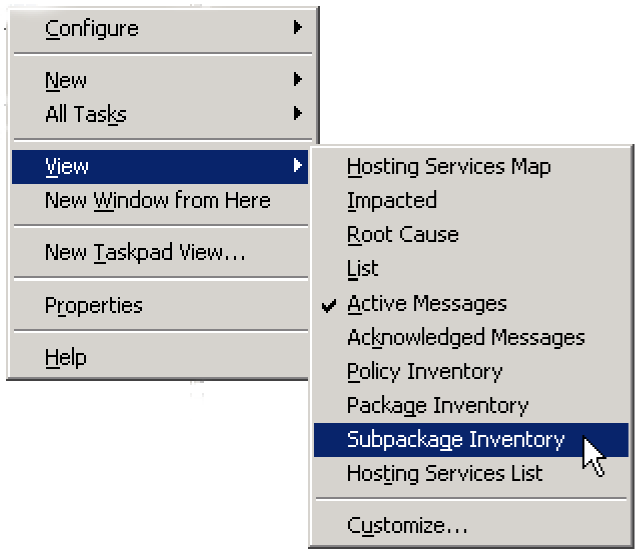
- The MMC console
- HP Operations Manager for Windows

Menus may display different commands depending on the context in which they are opened and the way they are selected. A typical console menu is shown below.



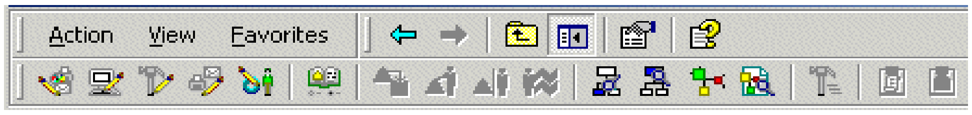
Command	Description
Configure:	Open configuration editors to configure message filters, nodes, the management server, service types, services, tools, and user roles.
All Tasks:	Provides a collection of frequently used tasks, such as launching tools, filtering messages, displaying graphs and reports, deploying instrumentation, and updating and restarting jobs and policies.
View:	Provides several options for changing the appearance of your view, including things like showing and hiding columns, icon size, type of map display, and active or acknowledge message browser display.
New Window from Here:	Opens a new windows in the context of an existing one.
New Taskpad View:	Opens the Taskpad Wizard, in which you can create a taskpad view of a console tree item.
Properties:	Opens the Properties dialog box for the selected item in the console tree.
Help:	Opens the HP Operations Manager for Windows help system, with help for the appropriate item shown as selected in the console tree.

If you right-click an item in the console tree without first selecting it with a left-click, a slightly different menu displays. This is standard behavior within the MMC. This example shows the console menu with the View submenu displayed.



Using the HP Operations Manager toolbar

The HP Operations Manager toolbar provides convenient access to many frequently performed tasks that are also available from the shortcut menu in the console tree. Click an icon in the interface for a tooltip explanation of its meaning.



Using the toolbar, you can:

- Open an editor to configure services, nodes, tools, service types, or user roles.
- Display a map view to perform root cause or impact analysis and to launch tools and display graphs, reports and messages from the map.
- Open an active or acknowledged messages browser.
- Apply tools to selected services or nodes.
- Apply filters.
- Display an inventory of deployed policies and packages.

Other toolbars allow you to customize your message browser and map views:

- Message Browser Toolbar
- Map View Toolbar

Save your workspace

The view of nodes, services, tools, and messages that you see in your console has been preconfigured by your administrator, but you are free to make certain changes to customize your display. You can save the changed configuration to use again.

You might want to save your workspace if you have:

- Filtered the display of messages in the browser to show only messages of a certain severity, ownership, or certain period of time.
- Modified the sort order of the received time and date column. The sort order of the other columns is not saved.
- Added, removed, or rearranged console views and want to preserve the views to use the next time your workspace file (`.msc`) is loaded.

To save changes to your workspace

1. From the console menu, select Save as to open the Save as ...dialog box.
2. Type a name for your customized view with the file extension `.msc` .
3. Specify the directory where you want to keep the file.
4. Click OK to confirm your selections and close the dialog box.

You can return to any customized view you create by opening its `.msc` file. Select File → Open from the MMC console menu to open a saved view.

Related Topics:

- Managing your workspace
- Modify message attributes
- Filter browser messages

Sort columns in the details pane

You can sort the information displayed in the columns in the details pane so that data appears in either ascending or descending order, indicated by either an up or down arrow at the top of the column.

The contents of the details pane will vary, depending on what is selected in the console tree. Column headings in the details pane also change according to the console tree selection. The Microsoft Management Console may show items from the console tree in the details pane, but these are not sorted. Only items that are programmatically inserted in the details pane are actually sorted.

To sort columns in the details pane

1. Click in the column heading of the column you want to sort.
2. Use the up arrow to sort in ascending order.
3. Use the down arrow to sort in descending order.

Find nodes and services in the console tree

You can search the console tree for a specific node or service. This helps you to locate a node or service more quickly, for example if you want to apply a tool.

To find nodes in the console tree

1. Right-click the Nodes folder in the console tree.
2. Select All Tasks → Find Node... . The Find Node dialog box opens.
3. Type the search string in the Search for box.
4. Select Display Name if you want to search all display names.
5. Select Primary Node Name if you want to search all primary node names.
6. *Optional.* Select the Case sensitive check box if you want your search results to match exactly the characters you typed into the Search for box.
7. *Optional* . Select the Only nodes in outage check box if you want your search results to match only nodes that are currently in outage.
8. Click OK . The Results dialog box opens.
9. Select a node and click Select . The node is selected in the console tree.

To find services in the console tree

1. Right-click the Services folder in the console tree.
2. Select All Tasks → Find Service... . The Find Service dialog box opens.
3. Type the search string in the Search for box.
4. Select Display Name if you want to search all display names.
5. Select Service Type if you want to search all services with a particular service type.
6. *Optional.* Select Case sensitive if you want your search results to match exactly the characters you typed into the Search for box.
7. *Optional* . Select the Only services in outage check box if you want your search results to match only services that are currently in outage.
8. Click OK . The Results dialog box opens.
9. Select a service and click Select . The service is selected in the console tree.

Related Topics:

- Viewing properties
- Applying tools to nodes, services, and messages
- Configuring outage information

Viewing service, node, and tool properties

The services, nodes, and tools your administrator configures to display in the console tree have properties that you can view when you want more information about a selected item.

Properties are set when the administrator configures the item.

To view properties

1. Select a service, node, tool, or group in the console tree.
2. Right-click to open the shortcut menu.
3. Select Properties to open a read-only dialog box that displays properties for the selected item.

Related Topics:

- Services properties
- Node group properties
- Node General properties
- External node General properties
- Tool group properties
- Tool General properties

Services properties

Properties for items in the Services area of the console tree are administrator-configured and can be viewed in a read-only dialog. The Services folder is located in the console tree directly beneath the HP Operations Manager entry.

NOTE:
Your administrator may rename the Services folder to something that is more meaningful in your specific environment. However, the folder location remains the same.

To view Services properties

1. Right-click Services in the console tree to open the shortcut menu.
2. Select Properties to open the Services Properties dialog box, which displays the following information:
   - General
   - Reports and Graphs
   - Tools
   - Outage

View services General properties

Properties for Services items in the console tree are administrator-configured and can be viewed in a read-only dialog box. The Services folder is located in the console tree directly beneath the HP Operations Manager entry.

NOTE:
Your administrator may rename the Services folder to something that is more meaningful in your environment. However, the folder location remains the same.

To view general information for Services properties

1. Right-click the Services folder in the console tree to open the shortcut menu.
2. Select Properties to open the Services Properties dialog box, which displays the General tab by default.
   - Icon: Displays the name of the icon that represents the service.
   - Display Name: Displays the name of the selected service.
   - Description: Displays a description of the selected service if your administrator has provided one.
   - Service ID: Displays the unique ID for the service.
   - Service Type: Displays the name of the service type for the selected service.
   - Service Hosting: Specifies a virtual or hosted on service.Displays the name of the node that hosts the service if Hosted On is specified.

Related Topics:

- View services Reports and Graph properties
- View services Tool properties
- View services Outage properties

View services Reports and Graphs properties

Properties for Services items in the console tree are administrator-configured and can be viewed in a read-only dialog box. The Services folder is located in the console tree directly beneath the HP Operations Manager entry.

NOTE:
Your administrator may rename the Services folder to something that is more meaningful in your environment. However, the folder location remains the same.

To view report and graph information for Services properties

1. Right-click Services in the console tree to open the shortcut menu.
2. Select Properties to open the Services Properties dialog box, which displays the General tab by default.
3. Select the Reports and Graphs tab, which displays the family and category of any reports and graphs configured by your administrator to be launched for the service.

Related Topics:
- View services General properties
- View services Tool properties
- View services Outage properties

View services Tools properties

Properties for Services items in the console tree are administrator-configured and can be viewed in a read-only dialog box. The Services folder is located in the console tree directly beneath the HP Operations Manager entry.

NOTE:
Your administrator may rename the Services folder to something that is more meaningful in your environment. However, the folder location remains the same.

To view tools information for Services properties

1. Right-click Services in the console tree to open the shortcut menu.
2. Select Properties to open the Services Properties dialog box, which displays the General tab by default.
3. Select the Tools tab, which displays the name and description of any tools configured by your administrator to run on the node the service is hosted on.

Related Topics:

- View services General properties
- View services Reports and Graph properties
- View services Outage properties

View service Outage properties

Properties for services in the console tree Services folder are configured by your administrator and can be viewed in a read-only dialog box. The Outage tab displays the settings established for both unplanned and scheduled outages.

A scheduled outage meets these criteria:

- Planned to happen
- Recurs at regular intervals for maintenance
- Configured by a policy

An unplanned outage is unexpected and can occur randomly.

NOTE:
Your administrator may rename the Services folder to something that is more meaningful in your environment. However, the folder location remains the same.

To view Outage properties for a selected service

1. Right-click Services in the console tree to open the shortcut menu.
2. Select Properties to open the Services Properties dialog box, which displays the General tab by default.
3. Select the Outage tab to view the following information:
    - Current Outage State:

      Displays the outage status of the selected node. This status will be either "ON " or "OFF".
    - Unplanned Outage Configuration:

      Displays the action that should be taken for Incoming Messages During Scheduled Outage . Messages can be either deleted or acknowledged.
    - Scheduled Outage Configuration:

      Displays the action that should be taken for Incoming Messages During Outage . Messages can be either deleted or acknowledged.

Related Topics:

- View services General properties
- View services Tools properties
- View services Reports and Graph properties

Nodes group properties

Properties for node groups in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box. Properties are set when the administrator configures the node group. If you are an administrator, you can make configuration changes for certain commonly used tasks by opening the Properties dialog box for a selected node group from the scope pane.

To view node group properties

1. Right-click the node group in the console tree to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected node group, which displays the following information:
   - General
   - Tools
   - Reports and Graphs
   - Deployment

To view properties for an individual node, right-click the node name in the console tree, then select Properties from the shortcut menu.

Related Topics:

- View General information for node group properties
- View Tools information for node group properties
- View Reports and Graphs information for node group properties
- View Deployment information for node group properties

Node group general information properties

General properties for node groups in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box.

Properties are set when the administrator configures the item. If you are an administrator, you can make configuration changes for certain commonly used tasks by opening the Properties dialog box for a selected node group from the scope pane. You can configure all items if you open this dialog box from the Node Configuration editor.

To view General information for node group properties

1. Right-click a node group in the Nodes folder in the console tree to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected node group, which displays the General tab by default. If you are an administrator, the fields of the dialog box will be available for editing.
   - Unique ID: Displays the unique ID for the selected node group.
   - Display Name: Displays the name of the selected node group.
   - Description: Displays a description of the selected node group if your administrator has provided one.
3. Select the Tools tab for more information about assigned tools.

Related Topics:

- Node group properties
- View Tools information for node group properties
- View Reports and Graphs information for node group properties
- View Deployment information for node group properties

Node group Tools properties

Tools properties for node groups in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box opened from the scope pane.

Properties are set when the administrator configures the item. If you are an administrator, you can make configuration changes for certain commonly used tasks by opening the Properties dialog box for a selected node group from the scope pane. As administrator, you can configure all items if you open this dialog box from the Node Configuration editor.

To view Tools information for node group properties

1. Right-click a node group in the Nodes folder in the console tree to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected node group, which displays the General tab by default.
3. Select the Tools tab to display the name and description of any tools configured by your administrator for this node group.

Related Topics:

- Node group properties
- View General information for node group properties
- View Reports and Graphs information for node group properties
- View Deployment information for node group properties

Node group Reports and Graphs information properties

Reports and graph properties for node groups in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box opened from the scope pane.

Properties are set when the administrator configures the item. If you are an administrator, you can make configuration changes for certain commonly used tasks by opening the Properties dialog box for a selected node folder from the scope pane. As administrator, you can configure all items if you open this dialog box from the Node Configuration editor.

To view reports and graphs information for node group properties

1. Right-click a node group in the Nodes folder in the console tree to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected node group.
3. Select the Reports and Graphs tab. If you are an administrator, the Graphs section of the dialog box will be available for editing.
4. In the Graphs section, click Select to open the Graph Selector dialog box. Use it to choose a family or category of graphs that can be generated for every node in your selected node group. The top level entries are families; the subordinate level entries are categories.
5. To remove a graph family or category, click Remove .
6. If you opened this dialog box from the scope pane, the Reports section will be read-only. Open this dialog box from within the Node Configuration Editor to select reports.
7. To add a report family or category, click Add to open the Report Selector dialog box. Use it to choose a family or category of reports that can be generated for every node in your selected node group.
8. To remove a report family or category, click Remove .
9. Click OK to confirm your choices and close this dialog box.

Related Topics:

- Node group properties
- View General information for node group properties
- View Tools information for node group properties
- View Deployment information for node group properties

Node group Deployment properties

Deployment properties for node groups in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box opened from the scope pane.

Properties are set when the administrator configures the item. If you are an administrator, you can make configuration changes for certain commonly used tasks by opening the Properties dialog box for a selected node group from the scope pane. As administrator, you can configure all items if you open this dialog box from the Node Configuration editor.

To view Deployment information for node group properties

1. Right-click a node group in the Nodes folder in the console tree to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected node group, which displays the General tab by default.
3. Select the Deployment tab to display the names and paths of policy groups your administrator configured for deployment to each node in the selected node group. If you opened this dialog box from the scope pane, the Deployment fields will be read-only. Open this dialog box from within the Node Configuration Editor to add or remove policy groups.
4. To add policy groups to the list of those you want to deploy, click Add to open the Policy Selector dialog box. Select one or more policies and click OK . The policy groups you selected appear in the Policy Groups box in the Deployment tab.
5. To remove a listed policy, select the policy group name and click Remove .
6. Click OK to confirm your choices and close this dialog box.

Related Topics:

- Node group properties
- View General information for node group properties
- View Tools information for node group properties
- View Reports and Graphs information for node group properties
- Disable policy autodeployment

View node General properties

Properties for nodes in the console tree Nodes folder are configured by your administrator. HPOM administrator rights are required to modify the general properties of nodes.

To view General properties for a selected node

1. Right-click the name of the node in the console tree to open the shortcut menu.
2. Select Properties to open the properties dialog box for the selected node, which displays the General tab by default.
   - Unique ID: Displays the automatically generated unique ID for the selected node.
   - Display Name: Displays the name of the selected node.
   - Description: Displays a description of the selected node if your administrator has provided one.
   - Owner Name: Displays the name of the critical contact, usually the node owner or administrator.
   - Contact Details: Phone number, pager number, or email address for the person entered in the Owner Name box.
   - Manufacturer: Displays the hardware maker's name, as configured by your administrator.
   - Model: Displays the model name or number of the selected node (for example, Kayak XA for an HP system.
   - Advanced Configuration: Opens the Advanced Configuration dialog box, which displays the following information:
     - Modify Agent ID: The GUID of the agent that resides on this node.
     - Modify Certificate State: Shows whether the HTTPS node has the certificates it requires to communicate securely with the management server.

       View certificate states

Related Topics:

- View Network information for Node properties
- View System information for Node properties
- View Tools information for Node properties
- View Outage information for Node properties

View node Network properties

Properties for nodes in the console tree Nodes folder are configured by your administrator and can be viewed in the Properties dialog box. Some of the information on the tabs for this dialog box is configurable and some is read-only.

To view and configure Network properties for a selected node

1. Right-click the name of the node in the console tree to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected node, which displays the General tab by default.
3. Select the Network tab.
4. In the Primary Node Name box, enter a unique node name. This information is required and is automatically entered if you selected the node to be managed by dragging and dropping the node name from the Discovered Nodes list to the Nodes list in the Configure Managed Nodes dialog box.

   When this information is entered or changed, the System Type , OS Type , and OS Version are automatically modified in the System tab.
5. Specify your preferences in the Server To Node Communications box. Check Node IP address obtained automatically (DHCP) if you do not want to use a static IP address.

   The server will not cache an IP address it once received for a system, but will do a name resolution every time a system is contacted. This means that if you specify the name of a system as the communication path value, and you have checked this box, the given name is always resolved (using an external name resolution service like DNS) if the node is contacted.

   NOTE:
   If you do not check this box, the server caches the first IP address it received for a node from a name resolution service. On subsequent contacts, the server uses the cached IP address instead of doing another external name resolution. This increases performance for nodes using a static IP address.

   Check Notify management server if node communication address changes if you want the server to be aware of IP address changes.

   The agent checks each time you start the system to see if the IP address of the system you are running on has changed. If the answer is yes, this information is sent to the management server, where the communication path of the corresponding node is updated with the new IP address.

NOTE:

If the address space of the managed node is different from the address space of the management server (because they are located in different subnets), it is possible that the management server cannot use the new IP address it received from an agent. In this case, to prevent errors, do not check the Notify management server if node communication address changes check box.

If you check this box, selections below it in the dialog box appear dimmed and are unavailable.

6. If you did not make a selection in the Server to Node Communications box, click either the IP Address or Domain Name FQDN button to specify a communications path. If you select IP Address , you must enter the IP address manually.

   By default, the Domain Name (FQDN) box displays the primary node name. If you change the primary node name, the change is reflected in the Domain Name (FQDN) box. You can also enter the domain name for the node you are configuring in the space provided. This information is optional.

   If you have multiple IP addresses for a particular node and the communications path would differ from the Primary Node Name, specify the particular IP address or DNS name that you want.

7. Heartbeat Polling sends a signal to the managed node or contacts the agent on the node to check whether the node is offline.

   System Default means that the management server uses the heartbeat polling setting from the Server Configuration dialog box. This setting is valid for all managed nodes and is by default "ICMP & Agent". To override the system default for a managed node, set Polling to Custom , then specify the Ping Protocol you want to use:

   - ICMP & Agent

     With this option, the server first attempts to contact the node using ICMP packages to find out if the node is reachable. If this succeeds, it will contact the agent on the node to find out if the agent processes are running. When this fails, it will use ICMP packages again to find out if, at least, the system is alive. As soon as this succeeds, the agent is contacted again. This option is not recommended for nodes outside of a firewall because ICMP calls are usually blocked by firewalls.

   - Agent Only

     The management server does not actively contact the node with ICMP pings, but still contacts the agent on the node. This is the recommended setting for nodes outside of a firewall. The disadvantage is that in the event of a system outage, the network load is higher than with normal heartbeat monitoring because the agent connection is still being tried.

   - ICMP Only

     The management server sends ping packages (using ICMP) to verify the availability of the agent. This option is not recommended for nodes outside a firewall because ICMP calls are usually blocked by firewalls.

8. Set the polling Interval in seconds. This is the interval at which the management server checks whether the managed node is offline. The minimum is polling interval is 60 seconds, and the maximum is 28800 (8 hours).

9. Clear Enable Auto Deployment if you do not want HPOM to automatically deploy policies to the node. HPOM automatically deploys policies to a node when the agent is correctly installed. For HTTPS agents, a correctly installed certificate is also required; otherwise the deployment job fails.

   By default, HPOM automatically deploys certain core policies to nodes. The core policies include autodiscovery policies that gather service information on nodes. This information is sent back to the management server to generate a service tree. (You can also automatically deploy additional groups of policies by associating policy groups with node groups and service types.)

10. Click Apply to confirm your changes.

11. Click OK to apply your changes and close this dialog box.

12. Click Cancel to close this dialog box without saving your changes.

Related Topics:

- View General information for Node properties
- View Network information for Node properties
- View System information for Node properties
- View Tools information for Node properties
- Agent health checks
- Disable policy autodeployment

View node Outage properties

Properties for nodes in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box. The Outage tab displays the settings established for both unplanned and scheduled outages.

A scheduled outage meets these criteria:

- Planned to happen
- Recurs at regular intervals for maintenance
- Configured by a policy

An unplanned outage is unexpected and can occur randomly.

To view Outage properties for a selected node

1. Right-click the name of the node in the console tree to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected node, which displays the General tab by default.
3. Select the Outage tab to view the following information:
   - Current Outage State:

     Displays the outage status of the selected node. This status will be either "ON " or "OFF".
   - Unplanned Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Scheduled Outage and Heartbeat Polling During Outage . Messages can be deleted or acknowledged. Heartbeat polling will be set either to "ON" or "OFF".
   - Scheduled Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage and Heartbeat Polling During Outage . Messages can be deleted or acknowledged. Heartbeat polling will be set either to "ON" or "OFF".

Related Topics:

- View Node General Properties
- View Network information for Node properties
- View Node System information for Node properties
- View Node Tools information for Node properties

View node System properties

Properties for nodes in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box.

To view System properties for a selected node

1. Right-click the name of the node in the console tree to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected node, which displays the General tab by default.
3. Select the System tab, which displays the following information:
    - System Type: Displays the system type for the selected node. If the node is of a type that is not supported by HP Operations Manager, this box will display a system type of Other.
    - Agent Comm Type: Displays the agent comm type configured for this system. Possibilities include DCE and HTTPS.
    - Operating System: Displays the operating system type. If the value cannot be determined, this box displays an operating system of Unknown.
    - Bit Length: Displays the configured bit length for this system. Possibilities include 32 and 64 bit.
    - Agent Binary Format: Displays the agent binary format configured for this system.
    - Version: The information has been automatically entered. If the value cannot be determined or is unsupported, this box remains blank. The list box shows the available versions for the selected operating system. You can select from the list or type in a version not available from the list (for example, a newly released version).
    - Automatically grant certificate: The HTTPS agent requires certificates, which enable it to communicate securely with the management server. The node can request these certificates from the management server. Check this box if you want the management server to automatically grant certificate requests for this node.

      For this property to take effect, you must also configure the management server to grant certificate requests automatically.
    - Certificate State: Shows whether the HTTPS node has the certificates it requires to communicate securely with the management server.
      View certificate states
    - Prerequisite check: Shows whether the node has passed the prerequisite check. If Passed is not selected, the prerequisite check has not yet run or has failed.

Related Topics:

- View General information for Node properties
- View Network information for Node properties
- View Tools information for Node properties

View Node Tools properties

Node properties for items in the console tree Tools folder are configured by your administrator and can be viewed in a read-only dialog box.

To view Tools information for a selected node

1. Right-click the name of the node in the console tree to open the shortcut menu.
2. Select Properties to open the properties dialog box for the selected node, which displays the General tab by default.
3. Select the Tools tab, which displays the name and description of any tools configured by your administrator for this node.

Related Topics:

- View General information for Node properties
- View Network information for Node properties
- View System information for Node properties

View external node General properties

Properties for external nodes in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box.

To view General properties for a selected external node

1. Right-click the name of the external node in the console tree to open the shortcut menu.
2. Select Properties to open the External Nodes Properties dialog box, which displays the General tab by default.
3. The automatically generated unique ID (GUID) for the selected node appears at the top of the dialog box.
4. The Display Name box displays the label (display name) for the external node.
5. The Description box displays any comments or additional information that your administrator has entered.

Related Topics:

- View external node Details properties
- View external node Order properties
- View external node Outage properties

View external node Details properties

Properties for external nodes in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box.

To view Details properties for a selected external node

1. Right-click the name of the external node in the console tree to open the shortcut menu.
2. Select Properties to open the External Nodes Properties dialog box, which displays the General tab by default.
3. Select the Details tab.
4. The Owner Name box displays the name of the critical contact, usually the node owner or administrator.
5. The Contact Details box displays information how to contact the owner (the phone number, for example).
6. The Pattern box displays the pattern that a message must match so that it is associated with this external node.

   The pattern acts as a filter that defines the external node. For example, if the pattern is ROS* then all messages with node IDs that match this pattern will be associated with the external node and will only appear in the browser for that node. The first match found determines to which node the message belongs. When an incoming message matches a node pattern, the evaluation stops and the message appears in the browser for that node.
7. The Check pattern against group box defines what the pattern should match. This selection clarifies any possible ambiguity in the pattern. For example, a pattern like *15* could refer to an IP address like 15.1.2.2 , or to a node name like ROS15test . By selecting IP Address or Node Name , any confusion about the meaning of the pattern is avoided.
   - Fully Qualified Domain Name

     If Fully Qualified Domain Name is the pattern choice, the filter will ignore the IP address for this pattern.
   - IP Address

     If IP Address is the pattern choice, only valid IP strings are evaluated.
   - Node Name

     If Node Name is selected, the filter matches the node name.

Related Topics:
- View external node Order properties

- View external node Outage properties
- View external node General properties

View external node Order properties

Properties for external nodes in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box.

To view order properties for a selected external node

1. Right-click the name of the external node in the console tree to open the shortcut menu.
2. Select Properties to open the External Nodes Properties dialog box, which displays the General tab by default.
3. Select the Order tab.
4. If the Check Before Managed Nodes check box is selected, external nodes are evaluated before managed nodes. This can be faster, because it avoids some of the checking that occurs with configured node names. When this box is checked, messages with node IDs that match the pattern specified in the Details tab are associated with the external node. If this box is not checked, the managed node takes precedence.
5. The two lists display all external nodes that have been created. The order of the two list shows the order of evaluation set for incoming messages. Nodes are evaluated in descending order; the node at the top of the list has the highest priority and will be evaluated first. After the first node has matched, the message is assigned to that node and no further evaluation takes place.

Related Topics:

- View external node Outage properties
- View external node General properties
- View external node Details properties

View external node Outage properties

Properties for external nodes in the console tree Nodes folder are configured by your administrator and can be viewed in a read-only dialog box.

Use the Outage tab of the External Node Properties dialog box to view outage information for an external node. This read-only tab displays the settings established for both unplanned and scheduled outages.

A scheduled outage meets these criteria:

- Planned to happen
- Recurs at regular intervals for maintenance
- Configured by a policy

An unplanned outage is unexpected and can occur randomly.

Only administrators can put an external node into maintenance mode.

To view Outage properties for a selected external node

1. Expand the Nodes folder in the console tree, then select HP Defined Groups → External to display a list of any external nodes that have been configured for your environment.
2. Right-click the name of the external node for which you want to display properties; this opens the context menu.
3. Click Properties to open the Properties dialog box for the selected external node, which displays the General tab by default.
4. Select the Outage tab to view the following information:
   - Current Outage State:

     Displays the outage status of the selected external node. This status will be either "ON " or "OFF".
   - Unplanned Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage . Messages can be either deleted or acknowledged.
   - Scheduled Outage Configuration:

     Displays the action that should be taken for Incoming Messages During Outage . Messages can be either deleted or acknowledged.

Related Topics:

- View external node General properties
- View external node Details properties

- View external node Order properties

Tool and tool group properties

General properties for tools and tool groups in the console tree Tools folder are configured by your administrator and can be viewed in a read-only dialog box. Properties are set when the administrator configures the item. If you are an administrator, you can make configuration changes for certain commonly used tasks by opening the Properties dialog box for a selected tool or tool group from the console tree. You can view properties for tool groups and for individual tools.

To view properties for an individual tool, right-click the tool name in the details pane, then select Properties from the shortcut menu.

To view general information for tool group properties

1. Right-click the tool group in the Tools folder in the console tree to open the shortcut menu.
2. Select Properties to open the Tool Group Properties dialog box, which displays the General tab.
    - Unique ID: Displays the unique ID for the selected tool group.
    - Display Name: Displays the name of the selected tool group.
    - Description: Displays a description of the selected tool group if your administrator has provided one.

Related Topics:

- View General information for tool properties
- View Details information for tool properties
- View Target information for tool properties
- View Nodes information for tool properties

View Tool General properties

General properties for tools in the details pane are administrator-configured and can be viewed and modified, depending on the user roles and permissions assigned to you by your administrator.

To view General information for a selected tool

1. Right-click the name of the tool in the details pane to open the shortcut menu.
2. Select Properties to open the Properties dialog box for the selected tool. Properties include:
   - General information about the selected tool, such as the display name of the tool. The General tab displays by default.
   - Details about the specific tool you selected, such as the type of tool, the executable name, and script details, if any.
   - Target nodes on which the tool will run.
   - Nodes and node folders associated with this tool.
3. The Display Name box shows the name for the tool as it appears in the Tools list.
4. The Description box shows any comments or additional information the administrator wants you to have available.
5. If Show tool in message context is selected, the tool can be launched in the context of all messages.

Related Topics:

- Configure Tools

View Tool Details properties

Details properties for tools in the details pane are administrator-configured and can be viewed in a read-only dialog box.

To view properties for an individual tool, right-click the tool name in the details pane, then select Properties from the shortcut menu.

To view Details information for a selected tool

1. Right-click the name of the tool in the console tree to open the shortcut menu.
2. Select Properties to open the Tools Properties dialog box, which displays the General tab by default.
3. Select the Details tab to view the following information:
   - Command Type: Displays the command type configured by your administrator. Command types include executable, URL, VBScript, JScript, Perl, and Windows Scripting Host (WSH).
   - Command Generates Output: If this box is checked, you can see the results of applying your tool or command in the Tool Status dialog box.
   - Allow Operator to Change Parameters: If this box is checked, you have permission to change tool parameters.
   - Allow Operator to Change the Login: If this box is checked, you have permission to change the user for this launch of this tool.
   - Password Required: If this box is checked, you must enter a password for this launch of this tool.
   - Command: Displays the name of the executable (for example, notepad.exe).
   - Parameters: Displays any parameters for the tool as configured by your administrator.

   Related Topics:

   - View General information for Tool properties
   - View Target information for Tool properties
   - View Nodes information for Tool properties

View tool Target properties

Target properties for tools in the details pane are configured by your administrator and can be viewed in a read-only dialog box.

To view Target information for a selected tool

1. Right-click the name of the tool in the details pane to open the shortcut menu.
2. Select Properties to open the properties dialog box for the selected tool.
3. Select the Target tab to view the following information:
   - Execute On: Displays the administrator-configured target location where the tool will run. Locations include console, management server, node list, and selected node.
   - User Name: Indicates whether a user name is required to execute this tool.
   - Run As Agent User indicates that the user name is set to $AGENT_USER. This parameter is later replaced with the name of the agent user account specified on the server. The account on the management server is initially set to Local System.
   - Password: Indicates whether a password is required to execute this tool.
4. Predefined Node List: Displays a list of predefined nodes or node groups. The contents of this list will vary, depending on the target location specified in the Execute On: box.

Related Topics:

- View General information for Tool properties
- View Details information for Tool properties
- View Nodes information for Tool properties

View Tool Nodes properties

Nodes properties for tools in the details pane Tools are administrator-configured and can be viewed in a read-only dialog box.

To view Nodes information for a selected tool

1. Right-click the name of the tool in the details pane to open the shortcut menu.
2. Select Properties to open the properties dialog box for the selected tool.
3. Select the Nodes tab, which displays the name and description of any nodes or node groups associated with this tool.

Related Topics:

- View General information for tool properties
- View Details information for tool properties
- View Target information for tool properties

Applying tools to managed nodes, services, and messages

Tools are applications, scripts, and commands that help you perform necessary tasks in your environment, such as launching corrective actions or diagnostic tools. HPOM provides many such out-of-the-box tools to make managing your environment easier. The tools available to you are determined by your administrator and are contained in tool groups in the Tools folder in the HP Operations Manager console tree. To view tools, select a tools folder to display the tools it contains in the details pane.

NOTE:
Predefined tools in HPOM have no versioning. If you modify one of the predefined tools, and later perform an upgrade to your current product version, this upgrade will overwrite any modifications you made to the tool. Only tools that you created will be saved and will reappear after the upgrade.

You can apply tools to selected managed services, nodes, or messages. In some cases, you can view the results of the tool application to see if it failed or succeeded using the Tool Status dialog box. If the tool was configured to run on your console, status is not displayed. For tools running in locations other than the console, you can see the output of a tool application if Command Generates Output was specified when your administrator configured the tool.

For example, you might specify that a report be generated for a selected service, then launched to run on the node where the service is running. You would then view the results of the launch and take further action, such as stopping or rerunning the tool, if necessary.

You can apply tools using either of these methods:

- Select a node, service, or message, then apply a tool
- Select a tool, then select a node or service

NOTE:
To use HP Operations Manager tools for UNIX nodes, you must first deploy instrumentation to those nodes.

Related Topics:

- View tool application status

Select nodes, services, or messages to apply tools

Tools are applications, scripts, or commands that you can run on selected nodes, services, and messages. A tool can be an application such as Notepad, a script that performs some automatic action, or a command, such as *ping* .

The tools available to you are configured by your administrator and are contained in tool groups in the Tools folder in the HP Operations Manager console tree. To view tools, select a tools folder to display the tools it contains in the details pane. You can apply one or more tools to one or more selected nodes, services, or messages.

To apply tools by selecting nodes, services, or messages

1. Select a node, service, or message:
   - From the console tree Services or Nodes folder, select one or more nodes or services to which you want to apply tools.
   - In the message browser, select one or more messages to which you want to apply tools.
2. Right-click to open the shortcut menu.
3. Select All Tasks →Launch Tool to open the Select the Tool to Execute dialog box, which displays a list of available tools. The list shows the tools as configured by your administrator for the selected node, service, or message.

   Details about each tool display when the tool is selected. These details are part of the configuration the administrator specifies when creating the tool, including a description of the tool and its function.

   If you are unable to see the tool you want, you can search for it. Right-click to open the shortcut menu and select Find to open the Find Item dialog box. Enter the tool name in the Search Text box and click Search . The tool name will be selected in the list of available tools.

   You can also search within the console tree by typing in the name of the node you want to find. As you type, the system locates the node for you.
4. Select the tool or group of tools that you want to apply to the selected node, service, or message.
5. Click Launch to apply the tool to your selected nodes, services, or messages. The Select the Tool to Execute dialog box closes.

   If the administrator has specified that operators can change parameters or login information, the Parameters or the Login dialog box opens. Make any necessary changes here and click Launch . If you need to configure both, parameters and login information, click Next in the Parameters page to go to the Login page. Then click Launch .
6. If the tool was applied to a remote node, the Tool Status dialog box opens to display the results of

the launch operation.

The Tool Status dialog box does not display if the tool has been configured to run on your console.

7. Click Cancel to close the Select Tool to Execute dialog box without launching a tool.

Server or cluster failover behavior

HPOM for Windows monitors the availability of server components such as the Windows Management Interface (WMI), the server itself, and the cluster resource group (if your environment contains clusters.)

In the event of a failure of one or more of these components, some activities may be temporarily unavailable, until WMI or the server is restored. See Related Topics: for details.

Related Topics:

- Edit parameters
- Edit login
- View tool application status
- Apply tools by selecting tools
- Server or cluster failover behavior

Select tools to apply

Tools are applications, scripts, or commands that you can run on selected managed nodes or services. A tool can be an application, a script that performs some action, or a command, such as *ping* . The tools available for a managed node or service are configured by your administrator and are contained in tool groups in the Tools folder in the HP Operations Manager console tree.

To select tools to apply

1. From the console tree Tools folder, select the folder that contains the tool you want to apply. Available tools are listed in the details pane.

   TIP:
   To search the console tree for tools, right-click Tools in the console tree, and then click All Tasks →Find Tool... . The Find Tool dialog box appears, which enables you to search for tools according to the display name, description, or command.

2. You can launch a tool in the following ways:

   - Drag the tool and drop it onto a node, node group, or service. If the administrator allows you to select target nodes for this tool, the tool launches on these nodes. If you drop a tool onto a service, the tool launches on the node that hosts the service. (You cannot launch a tool on a virtual service.)
   - Right-click the tool that you want to apply, and then click All Tasks → Launch Tool... . If the administrator allows you to select target nodes for this tool, the Select Where to Launch This Tool dialog box opens, which contains a tree of nodes and services.

     Select the check box for each target node, node group, and service. If you select a service, the tool launches on the node that hosts the service. (You cannot launch a tool on a virtual service.)

     If you are unable to see the managed node or service that you want, you can search for it. Right-click the tree, and then click Find... . The Find dialog box opens. Enter the node or service name you want to locate in Search for , and then click Find Next . Click OK to close the Find dialog box. The node or service is selected.

     Click Launch... to apply the tool to your selected nodes or services. The Select Where to Launch This Tool dialog box closes.

3. If the administrator allows you to specify or change parameters or login information, the Parameters or the Login dialog box opens. Make any necessary changes in these dialog boxes and click Launch... to launch the tool. If you need to configure both, parameters and login information, click Next in the Parameters page to go to the Login page. Then click Launch... .

4. If the tool was applied to a node, the Tool Status dialog box opens to display the status of the tool launch and any output. This dialog box also enables you to restart tools, by selecting one or more tools in the list, and then clicking Rerun . You can also save the results in a text file, by select one or more tools in the list, and then clicking Save .

   The Tool Status dialog box does not appear if the tool runs on your console.

Server or cluster failover behavior

HPOM for Windows monitors the availability of server components such as the Windows Management Interface (WMI), the server itself, and the cluster resource group (if your environment contains clusters.)

In the event of a failure of one or more of these components, some activities may be temporarily unavailable, until WMI or the server is restored. See Related Topics: for details.

Related Topics:

- Edit login/parameters
- View tool application results
- Apply tools by selecting nodes or services
- Server or cluster failover behavior

Edit parameters

When configuring a tool, your administrator can configure the tool so that users can pass parameters to the tool. When you launch the tool, the Parameters dialog box opens for you to make your changes. The buttons in the dialog box vary depending on the options checked. The dialog box may show Back and Next buttons if both parameters and login information must be supplied. If you do not need to enter login information, the dialog box shows the Launch button. The dialog box displays the name of the tool as it appears in the console tree.

If you select a node or service in the scope pane, or a message in a message browser, and right-click to Launch Tool , the Select the Tool to Execute dialog box opens so that you can select the tool you want to launch. Select the tool and click Launch . The Parameters dialog box opens, where you can enter parameters.

To edit parameters

NOTE:
This example explains how to edit parameters. The availability of this option depends on how the tool was configured by the administrator, how the tool was selected, and which options were checked on the Tool Properties Details tab.

1. The Command box displays the name of the command, script, or application to be executed.
2. In the Parameters: box, type in the parameters you want to associate with this tool. Double angle brackets (<< >>) enclose variables that will be replaced by your selection in the Select the nodes or services to replace box.
3. In the Select the nodes/services to replace box, select the check box beside the node or service name that you want to have replace the parameter you specified. For example, the parameter $OPC_NODEID will be replaced with the selected node or the node the selected service is hosted on.
4. If you also need to enter login information, the Parameters dialog box displays a Next button, which you need to click to go to the Login page.
5. Click Launch to apply the tool. The Tool Status dialog box appears when Parameters closes and displays the status of the tool application.

Possible scenarios

Behavior and available options of the Parameters dialog box vary depending on selections you make in the Details tab of the Tool Properties dialog box. Some examples follow.

Check Password Required:

When you launch the tool, the Login dialog box opens. The User Name field is read-only, but a password is required.

Check Allow Operator to Change Parameters and Allow Operator to Change Login:

In this configuration, when you launch the tool, the Parameters page opens first. Change the parameters and click Next to go to the Login page, where you can change the user name.

Check all checkboxes:

When all check boxes are checked, the Parameters page opens first. Change the parameters and click Next to go to the Login page, where you can change the user name and enter the password.

Related Topics:

- Edit login
- View tool application status
- Select nodes, services, or messages to apply tools
- Apply tools by selecting tools

Edit login

When configuring a tool, your administrator can configure the tool so that users can:

- Change the login.
- Enter a password.

With one or both of these options configured, when you launch the tool, the Login dialog box opens for you to make your changes. The buttons in the dialog box vary depending on the options checked. The dialog box may show a Back button if both parameters and login information must be supplied. If you do not need to enter parameters, the dialog box shows only the Launch button. The dialog box displays the name of the tool as it appears in the console tree.

If you select a tool folder in the scope pane and a tool from that folder in the details pane, when you right-click to launch this tool, the Select where to launch this tool dialog box opens if the administrator has configured the tool to execute on Selected Node. Use the dialog box to select the node or nodes on which to run the tool. After you select the node or nodes, the Login dialog box opens, where you can specify login information.

To specify login information

If the administrator has specified a user name, password, or both for this tool, enter them in the Login dialog box. The fields are dimmed if this information is not required. If you need to change the password for this launch of this tool, you can change it in the Login dialog box.

1. The Execute On: box displays the location where the tool will run. Depending on the target location, a correct user name, password, or both as specified by the administrator may be required. Security information and behavior are the same for the management server and for the managed node.
2. In the User Name: box, enter the correct user ID if required. If you are logged in as a local user, you may need to preface your user name with your domain name to run the tool. You can use either of these formats:

   domain\user

   user@domain

   If you change the user name, you must enter a password before the tool can be launched.
3. In the Password: box, enter the correct password if required.

   NOTE:
   On target nodes with a UNIX operating system, the agent checks only the first eight characters of the password before it runs the tool. If you specify a password that is longer than eight characters, the agent ignores the extra characters.

4. If you also need to enter parameters, click Back to go to the Parameters page, where you can change the parameters before launching the tool.
5. Click Launch to launch this tool. The Tool Status dialog box appears when the Login dialog box closes and displays the status of the tool application.

For information about passwords, logins, and their behavior with the Local System account, see the topic, Agent Users and Actions .

Possible scenarios

Behavior and available options of the Login dialog box vary depending on selections you make in the Details tab of the Tool Properties dialog box. Some examples follow.

Check Allow Operator to Change the Login:

If you change the user name from the one that was configured for the tool, you will see the message "Since User Name has changed, Password is required."

Check Allow Operator to Change the Login and Password Required:

If both check boxes are checked, enter the new user name in the Target tab. Return to the Details tab, apply your changes, and close the Tool Properties dialog box. When you launch the tool, the Login dialog box opens; the new user name is in the User Name box. If you launch the tool without entering a password, you will receive an error message stating that a password is required. You must enter a valid password to continue.

Check Password Required:

When you launch the tool, the Login dialog box opens. The User Name field is read-only, but a password is required.

Check Allow Operator to Change Parameters and Allow Operator to Change Login:

In this configuration, when you launch the tool, the Parameters page opens first. Change the parameters and click Next to go to the Login page, where you can change the user name.

Check all checkboxes:

When all check boxes are checked, the Parameters page opens first. Change the parameters and click Next to go to the Login page, where you can change the user name and enter the password.

Related Topics:

- Edit parameters
- View tool application status
- Select nodes, services, or messages to apply tools
- Apply tools by selecting tools

Agent users and actions

Tools can either run as the agent user ($AGENT_USER), or an alternative user. In either case, the user must have permission to launch the tool.

Operators can interact with tools in the following ways:

- If an operator does not modify the user of a tool, the tool runs with the user configured by the administrator.
- An operator cannot modify the user of a tool without specifying a password.
- An operator can only run a tool with a changed login if he knows the user name and password of a user who is allowed to run the tool on the managed node.
- An operator cannot change the user to $AGENT_USER.
- If the administrator leaves both the user name and password blank, and the operator does not specify a user:
  - On Windows nodes, the tool runs with the operator's account. The operator must be a domain user. (In addition, the security authentication module must be available on the domain controller.)
  - On UNIX nodes, the tool runs under the user account that the agent itself is currently running under.

Related topics:

- Security authentication module

View tool application status

When you launch a tool, HP Operations Manager automatically displays the outcome in the Tool Status dialog box if the tool was configured to run on a node other than the console.

The dialog box gives information about the apply tool operation:

- Status: Success or failure of the apply tool operation.
- Action: Tool name.
- Start/Finish Time: Initially the time when the tool was started. Changes to finish time when the tool has completed.
- Node: Name of the node or service to which the tool was applied.
- Command: The executed command.

Messages about the apply operation appear in the Tool Output box.

To stop or rerun a tool

You can cancel the application of a tool by clicking Stop . If the application fails to complete, the Stop button changes to the Rerun button, which you can click to reapply the tool.

NOTE:
When stopping a tool launch on a node that is not responding, you may see a "Server Busy" message and be unable to select another node in the list. This condition clears in a few minutes.

To save tool output

You can also save the results of the apply tool operations. Select one or more lines in the Launched Tools box and click Save . The output is saved in text format.

Filter browser messages

The messages that appear in your active messages browser are received from the nodes managed by the management server, as configured by your administrator. However, by setting filters you can further customize this display to show only those messages that match the criteria you set.

Messages can be filtered by one or more of these criteria:

- Message text: Filter messages containing specific text.
- Severity: Filter messages of the selected severity.
- Ownership: Filter messages of various ownership.
- Unmatched: Filter messages that either do or do not match any of the message conditions or suppressed conditions defined in the policies deployed on the managed nodes.
- Time: Filter messages created at particular dates and times.
- Message attributes: Filter messages on the application, object, or message group message attribute.
- Message source: Filter messages of the selected service, node, or node group.
- Custom message attribute (CMA): Filter messages on custom message attributes.

Because you can select multiple properties for a filter, you can create a filter to display just those messages that matter most to you. For example, you could create a filter that would display only messages from a particular service on a particular node, created between specified dates and times.

There are two types of filters:

- Public filters: Public filters are available to all users, but only HPOM administrators can create, modify, save, and delete them.
- Private filters: Any user can create, modify, save, and delete private filters for their own, personal use, if they have been granted the right to do so.

Related Topics:

- Apply filters
- Create filters
- Set General filter properties
- Set Time filter properties
- Set Message Source filter properties
- Set Message Properties filter properties
- Set Message CMA Properties filter properties

Apply a filter

When you apply a message filter, the set of messages in the active or acknowledged messages browsers changes to display only those messages that match the criteria you set. In addition, the message browser status line shows the name of the filter that is currently applied.

One filter can be applied to each selected node or service available in the console tree.

To apply a filter

You can apply a filter in one of two ways:

- Toolbar menu
  a. Select a node, service, or message in the console.
  b. Click on the console toolbar.
     If one or more message filters are already defined, a menu opens. If no filters are defined yet, create a filter first.
  c. In the toolbar menu, select Activate Message Filter and the name of the filter that you want to apply.
- Filter Messages dialog box
  a. Select a node, service, or message in the console.
  b. Click on the console toolbar.
     If one or more message filters are already defined, a menu opens. If no filters are defined yet, create a filter first.
  c. In the toolbar menu, select Configure Message Filter to open the Filter Messages dialog box.
  d. In the Filter Messages dialog box, select the filter that you want to apply and click Activate .
     The filter icon changes in the Message Filter dialog box to indicate that the filter is currently applied.

To remove a filter

You can remove a filter in one of two ways:

- Toolbar menu
  a. Select a node, service, or message in the console.
  b. Click on the console toolbar.
  c. Select Activate Message Filter and deselect the name of the filter that you remove.

- Filter Messages dialog box
  a. Select a node, service, or message in the console.
  b. Click on the console toolbar.
  c. Select Configure Message Filter to open the Filter Messages dialog box.
  d. In the Filter Messages dialog box, select the filter that you want to remove and click Deactivate .

     The filter icon changes in the Message Filter dialog box to indicate that the filter is turned off. The filter criteria are no longer applied to incoming messages.

Related Topics:

- Create filters
- Edit filters

Create filters

Filters control the display of messages within your console by matching incoming messages to the criteria you set. One filter can be applied to each selected node or service available in the console tree. You can filter messages by one or more of these properties:

- Severity
- Ownership
- Matched conditions
- Date and time
- Message source (service, node, or node group)
- Message properties (application, object, or message text)
- Custom message attributes

To create a new filter

1. Open the Filter Messages dialog box if it is not already open.
2. Click New to open the Filter Properties dialog box and specify filter properties in these areas:
    - General
    - Time
    - Message Source
    - Message Properties
    - Message CMA Properties
3. In the Filter Properties dialog box, click Apply to apply your changes.
4. Click OK to apply your changes and close the Filter Properties dialog box.

    The filter name appears in the list of available filters in the Message Filter dialog box. If a filter is being used, its icon changes in the Message Filters list and its name is selected.
5. If you want to apply the filter to the active messages browser, click Activate .
6. In the Message Filter dialog box, click Close to return to your console view.

    If you already have a message browser open, you should see only those messages that match the applied filter. The message browser status line shows which filter, if any, is applied.

Related Topics:

- Apply filters
- Edit filters

Set General filter properties

Use the General tab of the Message Filter dialog box to specify general information about a filter. All information is optional.

You can combine several filtering criteria in one filter. For example, you could select both critical severity and an unmatched criteria.

To specify General information for a filter

1. Open the Filter Messages dialog box if it is not already open.
2. Click New to open the Filter Properties dialog box. The General tab displays by default.
3. Enter a name for the filter in the Name box.
4. Select Public filter if you are an HPOM administrator and want the filter to be available to all users.
5. If you want to filter by message text, enter the text in the Message Text Includes: box. Only messages that contain this specified text will display in the browser. For example, if you enter "error" in the Message Text Includes box, messages will be filtered on message text that contains the word "error," regardless of case.
6. To filter on severity criteria, check one or more check boxes in the Severity group.
7. To filter on ownership criteria, check one or more check boxes in the Ownership group.
8. In the Condition criteria, check one or both the Unmatched and Matched check boxes to filter for messages that either match or do not match any of the message conditions or suppressed conditions defined in the policies deployed on the managed nodes. Inform your administrator of any unmatched messages so that the corresponding message can be improved or corrective action provided.
9. Click Apply to apply your changes.
10. Click OK to apply your changes and close this dialog box.

Related Topics:

- Set Time filter properties
- Set Message Source filter properties
- Set Message Properties filter properties
- Set Message CMA Properties filter properties
- Create filters
- Apply filters

Set Time filter properties

Use the Time tab of the Message Filter dialog box to filter messages created on a particular date, at a particular time, or during a specified time period.

To filter by time, date, or time period

1. Open the Filter Messages dialog box if it is not already open.
2. Select the Time tab.
3. Select All Messages to display messages regardless of the time they were created.
4. If you prefer to filter messages created at a specific time, on a specific date, or during a specific interval of time, select Message Received .
   - Select specific start and end dates by clicking the down arrows in the To and From date boxes to open the calendar.

     In the calendar, click the start date you want. The calendar closes and the date you selected appears in the To field. Repeat using the From date box to open the calendar again. Select an end date.
   - Select specific start and end times by clicking the down arrows in the To and From time boxes.
   - Specify start and end times using the up and down arrows in the time fields. Place your cursor in area of the time that you want to change (hours, minutes, or seconds) and use the arrows to increase or decrease the time and change from am or pm.
   - To filter by a time period such as months or days, rather than specific dates, click Within: and enter the number of months or days you prefer.
5. Click Apply to apply your changes.
6. Click OK to apply your changes and close this dialog box.

Related Topics:
- Set Message Source filter properties
- Set Message Properties filter properties
- Set Message CMA Properties filter properties
- Set General filter properties
- Create filters
- Apply filters

Set Message Source filter properties

Use the Message Source tab of the Message Filter dialog box to filter for messages received from specified message sources such as services and nodes. Only messages from the services and nodes you select will display in the message browser.

To set message source filter properties

1. Open the Filter Messages dialog box if it is not already open.
2. Select the Message Source tab.
3. Select one or more services or nodes from the Select Services, Nodes, or Node Groups tree to display only messages from those nodes or services in the message browser. Services can be selected independently of their position in the tree's hierarchy.

   If you select a node group, all nodes included in the group are also selected. To deselect an item, click the check box.
4. Click Apply to apply your changes without closing this dialog box.
5. Click OK to apply your changes and close this dialog box.

Related Topics:

- Set Message Properties filter properties
- Set Message CMA Properties filter properties
- Set General filter properties
- Set Time filter properties
- Create filters
- Apply filters

Set Message Properties filter properties

Use the Message Properties tab of the Filter Properties dialog box to filter on application, object, and message group properties. Only messages with those properties will display in the message browser.

An application is defined as the name of the application which was affected by or detected the problem. An object is defined as a specific object which was affected by, detected, or caused the message. A message group is defined as a group of messages that belong to the same task or have some logical connection (for example, messages from backup tasks or messages having a common policy).

Attributes you can filter on include:

- Applications, such as XP, Oracle, Exchange, or OpC.
- Objects, such as a print server or device manager, a graph template name, or a metric such as cpu_utilization.
- Message groups, such as OpenView, OpC, or VP_SM.

To set Message Properties filter properties

1. Open the Filter Messages dialog box if it is not already open.
2. Select the Message Properties tab.
3. From the Browser Property dropdown list, select the attribute for which you want to filter (application, object, or message group).
4. In the Value box, type the application, object, or message group name, or select it from the dropdown list. Message filters in the console limit the message group length to 32 characters (the same length limit of the Field box in messages).
5. Click Add to add the property to the Property list.
6. If you selected a message in step 1, Get Values is available. Click Get Values to add attributes to the Property list. If multiple messages were selected, the Value column shows a unique value for each attribute. For example, a SAP application might show values for:

   SAP DB server#1
   SAP APP server#2
7. To add more properties to the Property list, repeat the previous steps.
8. To remove a property from the list being filtered for, select the property and click Delete .
9. Click Apply to apply your changes without closing this dialog box.

10. Click OK to apply your changes and close this dialog box.

Related Topics:

- Set Message CMA Properties filter properties
- Set General properties
- Set Time properties
- Set Message Source properties
- Create filters
- Apply filters

Set Message CMA Properties filter properties

Use the Message CMA Properties tab of the Filter Properties dialog box to filter for messages that match specified custom message attributes (CMA) properties. Only messages that match the names and values that you select will display in the message browser.

To set message CMA filter properties

1. Open the Filter Messages dialog box if it is not already open.
2. Select the Message CMA Properties tab.
3. Select the CMA Name and Value pair for which you want to filter.

   The list contains all the known CMA names. CMAs are available only if they have been received on the server and are already attached to messages in the database.
4. Click Add to add your selections to the Property list.
5. To add more properties to the Property list, repeat the previous steps.
6. To remove a property from the list, select the property and click Delete .
7. Click Apply to apply your changes without closing this dialog box.
8. Click OK to apply your changes and close this dialog box.

Related Topics:
- Set General filter properties
- Set Time filter properties
- Set Message Source filter properties
- Set Message Properties filter properties
- Create filters
- Apply filters

Edit filters

You can easily edit any filter you have created.

To edit a filter

1. Open the Filter Messages dialog box if it is not already open.
2. From the Filter Name list, select the name of the filter that you want to edit.
3. Click Edit to open the Filter Properties dialog box.
4. Edit the existing information in the following tabs:
   - General
   - Time
   - Message Source
   - Message Properties
   - Message CMA Properties
5. When you finish editing the filter, click Apply to save your changes.
6. Click OK to save your edited filter and close the Filter Messages dialog box.
7. In the Filter Messages dialog box, click Close .

Related Topics:

- Apply filters
- Create filters

Browsing messages

HP Operations Manager for Windows provides two message browsers in which you can view messages that result from events occurring on managed nodes.

Active Messages: The active messages browser displays active (unacknowledged) messages that can be acted on by operators to resolve problems in the managed environment. Operators can review information about a selected message, display maps that show the root cause of a problem or services impacted by the problem, and take action to resolve problems and improve performance.

Acknowledged Messages: The acknowledged messages browser displays messages that have been acknowledged. Typically, a message is acknowledged when the problem that caused that message to appear has been resolved. As in the active messages browser, you can display maps that show the root cause of a problem or services impacted by the problem. You cannot launch any actions to correct the problem from this browser. If you need to perform tasks that are unavailable in the acknowledged messages browser, you can unacknowledge the message to move it back to the active messages browser. More options are available to you for problem resolution in the active messages browser.

Using the message browser interface

The message browser interface displays currently active messages or acknowledged messages, depending on the browser you have selected. Within the selected browser, the headline displays a number of message details in one view. You can also view message attributes, severity, and status levels at a glance. The following topics describe the browser interface in detail:

- Browsing active messages
- Browsing acknowledged messages
- Read the browser headline
- Message browser toolbar
- Message attributes key
- Message severity and status

Browsing active messages

When first launched, the active message_browser displays all active (unacknowledged) messages for an operator's managed environment. At a glance, you can see status and severity for managed nodes and services and take action to resolve problems that impact critical services.

The active message browser is your administrator-configured view of all the active messages received at your console. HP Operations Manager for Windows receives events from managed nodes and services and displays them as messages in your view. To see acknowledged messages (messages that have been resolved), switch to the acknowledged message browser.

Using the active message browser, you can:

- Evaluate all current messages according to color-coded status.
- Read message text.
- Review instructions.
- View a map in context of a selected message. Show uses/contains relationships, root cause, or impacted services or nodes, depending on your selections.
- Add, delete, and modify annotations for selected messages.
- Determine impacted services for a selected message.
- Find the root cause of a message's severity.
- Initiate corrective actions.
- Review status of operator-initiated actions.
- Display a graph or report in the context of a selected message.

You can simplify the message display by showing only messages from nodes or services of special interest to you. The browser headline displays the message attributes in a condensed format that you can interpret quickly.

You can customize the appearance of your active messages browser by arranging columns in the order you prefer and specifying the sort order of information in the columns. These settings are maintained from session to session.

By default, unowned messages have a background color according to their status. Messages that other users own have a light gray background. You can customize message coloring by changing the message browser properties (in the Console Properties dialog box).

Related Topics:

- Browsing acknowledged messages overview

- Interpret the browser headline
- View and edit message properties
- View messages for a selected system or service
- Specify console message browser properties

Browsing acknowledged messages

The acknowledged messages browser displays all messages that have been acknowledged in the active messages browser. Acknowledged messages are moved from the active messages browser to the acknowledged messages browser.

Using the acknowledge messages browser, you can:

- Unacknowledge a message and return it to the active messages browser.
- Read message text.
- Review instructions.
- Add, delete, and modify annotations for selected messages.
- View message properties.

If you need to perform tasks that are unavailable in the acknowledged messages browser, you can unacknowledge the message to move it back to the active messages browser where more options are available to you.

- Browser overview
- Browsing messages

Read the Browser Headline

The browser headline is a banner across the top of the message browser that labels the columns of information the browser displays. Use the headline to quickly identify these message details:

Severity	Color-coded icons give at-a-glance message status. The console displays six levels of message severity.
Duplicates	Message counters are displayed in the Duplicates column by default. You can configure their display using the Options dialog box which you can open by clicking in the browser headline.
SUIAON	The status column displays attributes for message states. The flags in the columns indicate state and the availability of instructions, annotations, and actions for each message.
Received	Specifies the date and time the message was received on the management server.
Created	Specifies the date and time the message was created.
Service	Specifies the service that issued the message. For example, a server or node name.
Node	Specifies the name of the node that issued the message.
Application	Specifies the application that detected the message or was affected by it.
Object	Specifies the object that caused the message or was affected by it.
Group	Specifies the message group to which the message belongs.

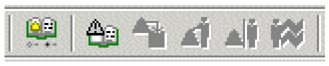
You can get more detailed information about a message and change some message attributes using the Message Properties dialog box.

Related Topics:

- View and edit message properties

Using the message browser toolbar

The message browser toolbar allows you to customize the appearance of your active and acknowledged messages browsers and perform tasks associated with selected messages. Click an icon in the interface for a tooltip explanation of its meaning.



Using the toolbar, you can:

- Filter messages that appear in the acknowledged and unacknowledged message browsers.
- Acknowledge and unacknowledge a message.
- Own and disown messages.
- Run an operator-initiated command.

Other toolbars allow you to customize your HP Operations Manager console and map views:

- HP Operations Manager for Windows console toolbar
- Configuration toolbar
- Map view toolbar

Message attributes key

Message attributes are described in the Message Properties dialog box and displayed graphically in the message browser headline. The SUIAON column in the message browser shows which attributes are available for a selected message. Flags in the columns provide further information.

If an attribute is not available for a message, the column for that attribute displays a hyphen (-).

Value	Flag
S	Owned Message State . A flag in this column indicates that a user has taken note (Marked) or ownership (Owned) of a message. The flags you might see in this column indicate that a message is: ■ O: owned by operator ■ X: owned by others ■ -: unowned ■ A: acknowledged
U	Unmatched Message . A message that does not match either a message condition or a suppress condition. An X in this column indicates an unmatched message.
I	Help Instructions . The administrator provides instructions for messages to help with problem resolution. If available, you can view these instructions in the Instructions tab of the Message Properties dialog box.
A	Automatic Command . Indicates that an automatic command has been configured for the message and gives the status of the command. The attribute value shows whether a command: ■ X: Available ■ S: Successful ■ F: Failed ■ R: Running ■ N: Not started ■ D: Discarded ■ C: Console command
O	Operator-Initiated Command . Indicates that an operator-initiated command has been configured for the message and gives the status of the command. The attribute value shows whether a command is: ■ X: Available ■ S: Successful ■ F: Failed ■ R: Running ■ N: Not started ■ D: Discarded ■ C: Console command

N Annotations: Indicates if annotations exist for this message. You review the annotations to find procedures that resolved similar problems in the past.

Message severity and status levels

The console displays six levels of message severity, color-coded so you can assess their importance at a glance. Severity levels are assigned to messages by your administrator, based on their importance in your environment. The numbers beside the severity level show the corresponding severity level as it appears in WMI messages.

Severity Level	Icon	Meaning
Critical (32)	❌	A condition affecting a service or node has occurred and immediate corrective action is needed.
Major (16)	⚠	Problem severity is high and normal use of the affected object is likely to be impeded.
Minor (8)	⚠	Problem severity is relatively low and should not impede normal use of the object.
Warning (4)	▲	A potential or impending service-affecting fault has occurred. Take action to diagnose and correct the problem.
Normal (2)	ℹ	Message output is as expected. A process might be starting or completing or status information displayed.
Unknown (0)	↔	A severity level was not defined in the policy running on the managed node for the event that generated the message.

The following table shows the possible states of the message and the values as they map to WMI message values.

State	Numeric Value
Undefined	1
Unowned	2
Owned	3
Acknowledged	4
Node Deleted	5
Deleted	6

Acting on messages

From within the active messages browser, you can perform a number of operations on existing messages. For example, you can acknowledge, annotate, edit, own, and disown messages, edit message text attributes, and launch commands and policies, as well as perform other functions. See "Acting on Messages" in the Table of Contents for the complete list.

More limited options are available to you from within the acknowledged messages browser. In this browser, you can unacknowledge the message, view message properties, text, and instructions, and act on annotations for selected messages.

Related Topics:

- Acknowledge a message
- Active messages browser menu
- Assign messages
- Annotate a message
- Edit annotation text
- Disown a message
- Edit message text
- Launch commands
- Launch policies from the message browser
- Mass operations on messages
- Modify message attributes
- Own a message
- Policy editing options table
- Sort message information
- Specify duplicate message criteria
- Unacknowledge a message
- View actions
- View messages for selected nodes or services

Acknowledge a message

Why acknowledge a message?

To acknowledge a message

1. In the details pane, select one or more messages that you want to acknowledge.
2. Right-click to open the shortcut menu.
3. Select Acknowledge . The message you acknowledged no longer appears in the active messages browser view.

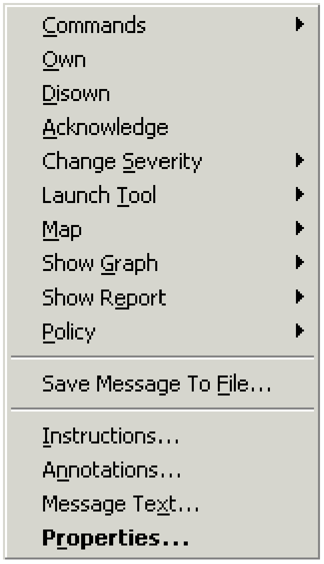
You can move acknowledged messages back into the active messages browser by unacknowledging them in the acknowledged messages browser.

Related Topics:

- Unacknowledge a message

Active messages browser menu

You can perform a number of actions quickly from the active messages browser using the shortcut menu shown below. More details are available about a selected message from the Message Properties dialog box.



Command	Description
Commands:	Start (execute) and stop automatic and operator-initiated commands.
Own:	Take ownership of a message to resolve a problem.
Disown:	Give up ownership of a message.
Acknowledge:	Send a message for a resolved problem to the acknowledged message browser and remove it from the active message browser view.
Change Severity:	Change the severity of a selected message.
Launch Tool:	Apply a tool to a selected message, node or service. You can select multiple messages, nodes, or services, and apply the tool to all of them. If the selected node is an external node, the Launch Tool option is not available for services and nodes. Tools cannot be launched on external nodes. You can launch tools on messages if the tool is a console tool or it is started on the management server.
Map:	Display a map of a selected service or node. Show the root cause of a problem, services or nodes impacted by the problem, or the relationship of the selected service or node to other services or nodes.
Show Graph:	Display a graph associated with the service, service type, or node group of a selected message.

Show Report:	Display a report associated with the service, service type, or node group of a selected message.
Policy:	Edit a policy and create suppress or match conditions. Policy deployment is not supported on external nodes.
Save Message To File...	Save messages to a .cvs or .txt file. You can save just the current selected message or all messages in the current view. The file includes all message properties and is stored on the console.
Instructions:	View instructions prepared by your administrator to help you solve the problem associated with the selected message.
Annotations:	View and edit annotations associated with the selected message.
Message Text:	View and edit text associated with the selected message.
Properties:	Open the Message Properties dialog box to view and edit message properties.

Assign messages

Administrators can assign ownership of messages to other users. Administrators can also give operators the right to assign ownership of messages for particular message groups.

The owner of a message is responsibile for performing the actions associated with that message. This ability to assign ownership of messages enables managers, for example, to delegate work to other users.

To assign messages

1. In the message browser , select one or more messages .
2. Right-click the messages and then click Assign... . The Select User dialog box opens.
3. Click the user to whom you want to assign the messages, and then click OK .

Related Topics:

- Own a message
- Disown a message
- Set permitted operations for message groups

Annotate a message

Why annotate a message?

To annotate a message

1. In the details pane, select the message you want to annotate.
2. Right-click to open the shortcut menu.
3. Select Annotations to open the Annotations tab of the Message Properties dialog box. Existing annotations for the selected message are listed here, along with the originator of the annotation and the time it was created.
4. To create an annotation, click New to open the New Annotation dialog box.
5. Enter your comments and click OK . The annotation appears in the Text box in the Annotations tab of the Message Properties dialog box.
6. Click OK to save the annotation. The annotation count is incremented in the message browser.
7. Click Cancel to close the dialog box without saving your annotation.
8. To delete an annotation, click Delete . The annotation is removed from the message in the active messages browser.

   NOTE:
   Annotations added by the system as a result of an automatic action specified by a policy cannot be deleted. If you try to delete a SYSTEM annotation, a message displays telling you that the delete operation failed.

Edit annotation text

You can edit annotation text and resize the text window where the text appears.

To edit annotation text

1. In the details pane, right-click a message in the message browser to open the shortcut menu.
2. Select Properties to open the Message Properties dialog box. The Text tab displays by default.
3. Select the Annotations tab.
4. Select a message and click Edit to open the Edit Annotations dialog box.
5. Make any edits to the text and drag the window to resize it, if desired. You can also double-click a message name to open the Edit Annotations dialog box.
6. Click OK to save your changes.
7. To view your changes, select the annotation. The annotation you edited appears in the text box.

Related Topics:
- View and edit message properties
- View message annotations

Disown a message

Only the owner of a message or the administrator can disown a message.

To disown a message

1. In the active messages browser, select the message or messages you want to disown.
2. Right-click the message to open the shortcut menu and select Disown .

   The browser headline displays a hyphen in the S column to indicate that the message is disowned.

Related Topics:

- Own a message

Edit message text

You can change message text as long as no one else owns the message. If you change the message text of an unowned message, you become the owner of the message.

You might modify a message because:

- You want to clarify the message.
- You want to add additional details.

To edit message text

1. In the details pane of the message browser, select the message you want to edit.
2. Right-click to open the shortcut menu.
3. Select Message Text to open the Text tab of the Message Properties dialog box.
4. In the Message box, enter the new message information.
5. Click OK . The dialog box closes and your changes appear in the message browser for others to read.
6. Click Cancel to close this dialog box without saving your changes.

NOTE:
You can add hyperlinks to URLs in your message or instruction text. Insert your cursor in the text of an active message on the Text tab and type the required URL. Apply and save your changes. This feature is useful for linking to external sites of all types, such as support sites, documentation repositories, troubleshooting information, and similar sites.

Related Topics:

- Annotate a message
- View instructions
- View state
- Launch actions
- Change message severity

Launch commands

Why launch a command?

To launch a command

1. In the details pane, select the message for which you want to run a command.
2. Right-click to open the shortcut menu.
3. To rerun an automatic command, select Actions → Start → Automatic You can verify that the command was successful by looking in the message browser status column. The A column displays an S if the command was successful.
4. To run an operator-initiated command, select Actions → Start → Operator Initiated . You can verify that the command was successful by looking in the message browser status column. The O column displays an S if the command was successful.

Related Topics:

- Stop commands

Stop a command

You can easily stop any command you have launched.

To stop a command

1. From the details pane, select the message for which you want to stop a command.
2. Right-click to open the shortcut menu.
3. To stop an automatic command, select Commands → Stop → Automatic . You can verify that the command was successful by looking in the message browser status column. The A column displays an S if the command was successful, an F if the command failed.
4. To stop an operator-initiated command, select Commands → Stop → Operator Initiated . You can verify that the command was successful by looking in the message browser status column. The O column displays an S if the command was successful, an F if the command failed.

Launch policies from message browser

You can edit the policy that generates a specific message directly from the message browser, saving steps and time.

To launch the policy editor directly from the browser

1. In the message browser, select the message for which you want to edit the policy.
2. Right-click to open the context menu.
3. Select Edit Policy to open the policy editor.
4. Edit the policy as necessary.

Related Topics:
- Configure sources
- Configure rules
- Configure options

Mass operations on messages

Some operations can be performed on many messages at once, saving you time and effort when performing repetitive actions. Mass operations you can perform on messages include:

- Own
- Disown
- Acknowledge
- Unacknowledge
- Change severity
- Use the operator-initiated command

NOTE:
To stop an operator-initiated command, you must stop it one message at a time.

To perform a mass operation

1. Select a block of messages or the desired set of messages in the message browser.
2. Right-click to open the shortcut menu.
3. Select the command for the operation you want to perform.

Related topics

- Turn off browser error dialog boxes for messages
- Turn on browser error dialog boxes for messages

Modify message attributes

You can modify the text of the message that appears in your browser. Some attributes are modified from the Properties dialog box and some only from the shortcut menu. After you modify an unowned message, you become the owner of the message. Attributes you can modify include:

- Annotation
- Message text
- Severity

To modify message attributes

1. In the message browser, select the message you want to modify.
2. Right-click the message to open the shortcut menu.

   To modify severity, select Change Severity to open a shortcut menu. Select Critical, Major, Minor, Warning, or Normal to change the severity of the selected message.

   To modify annotation or message text for a selected message, select Properties from the shortcut menu to open the Message Properties dialog box. If you prefer, you can select the attribute you want to change from the shortcut menu and go directly to that tab in the Properties dialog box.
3. Modify one or more attributes using the tabs in the dialog box.
4. Click OK to close the dialog box. Your changes are immediately applied to the selected message and are visible to other users.
5. Click Cancel to close the dialog box without saving your changes.

Related Topics:
- View message properties

Own a message

Why own a message?

To own a message

1. In the message browser, select the message you want to own.
2. Right-click the message to open the shortcut menu and select **Own**.
3. An owned message displays as orange in the message browser and the browser headline displays a letter O in the S column to indicate that the message is owned. Other operators will see an X in that column.

Related Topics:

- Disown a message
- Annotate a message
- Modify message text
- Acknowledge a message
- Rerun automatic commands
- Perform operator-initiated commands

Policy editing options table

The editing options available for various policies are shown in the following table. Users working in the message browser can modify the policy that generated a selected message by choosing one of the following menu options. Menu options vary depending on the message selected.

- Edit Policy (if no conditions were set in the policy, only this menu option is available)
- Create Suppress Condition
- Create Match Condition

The ConfigFile, Service Auto-Discovery, Flexible Management, and Node Info policy types do not generate messages, so are not included in this table.

	Edit policy (Opens Policy Editor to policy only)	Edit policy condition (Opens Policy Editor to policy and condition)	Create match condition	Create suppress condition
Open Message Interface	no	yes	yes	yes
Log File Entry	no	yes	yes	yes
Windows Event Log	no	yes	yes	yes
Measurement Threshold	no	yes	no	no
SNMP Interceptor	no	yes	no	no
Scheduled Task	yes	no	no	no
Windows Management Interface	no	yes	no	no
Service/Process Monitoring	yes	no	no	no

Sort message information

You can sort the information that appears in the columns in the active and acknowledged messages browsers so that data appears in either ascending or descending order, indicated by either an up or down arrow at the top of the column. Except for the Received column, all new information appears at the bottom of the message browser.

Received: This column sorts date and time. Select ascending order to see the oldest messages at the top of the list and new messages at the bottom. Select descending order to see the newest messages at the top of the list.

Severity: In the Severity column, ascending order places the least severe messages at the top of the list. Descending order places the most severe messages at the top.

All other columns are sorted alphabetically in either ascending or descending order.

To sort message information

1. In either the active or acknowledged messages browser, click in the column heading of the column you want to sort.
2. Use the up arrow to sort in ascending order.
3. Use the down arrow to sort in descending order.

Specify duplicate message criteria

For a message to be considered a duplicate of an existing message, selected criteria must match those same criteria in the original message. By specifying characteristics such as node name, you can specify which criteria must be matched. By doing so, you can control the number of duplicate messages received in the message browser.

Messages with identical message keys are always considered duplicates. For messages without message keys, you can specify one or all of the following criteria that a message must meet to be considered a duplicate. By default, all options are selected.

- Severity
- Service ID
- Message Text
- Condition
- Message Group
- Node Name
- Application
- Object

To select the criteria a duplicate message must possess:

1. Check the check box for each criteria that must match the original message. Click Select All to check all boxes. All checked criteria or the Message Key (if available) will be used for duplicate message suppression. Click Clear All if the Message Key should be the only criteria for duplicate message suppression.
2. Click OK to close the dialog box and confirm your changes. When you return to the message browser, the options you checked will take effect.

Unacknowledge a message

Unacknowledge messages when you want to return them to the active messages browser for further investigation. Unacknowledging a message removes it from the acknowledged messages browser and returns it to the active messages browser.

You might revisit acknowledged messages to:

- Read message annotations.
- Use the acknowledged messages browser as a resource for solving problems.

To unacknowledge a message

1. In the console tree, select the node or service you are interested in to display only those messages in the details pane.
2. From the console tree, right-click to open the shortcut menu.
3. Select Acknowledged to open the acknowledged messages browser in the details pane.
4. In the details pane, select the message you want to unacknowledge. Right-click to display the shortcut menu.
5. Select Unacknowledge . The message you unacknowledged is removed from the history database and returned to the active messages browser.

Related Topics:

- Acknowledge a message

View automatic and operator-initiated commands

Commands to resolve problems are associated with messages by your administrator. There are two types of commands; each message can be associated with one command of each type:

- Automatic commands: run automatically when certain conditions are met (as soon as an event is detected).
- Operator-initiated commands: launched by an operator and may require some user input.

The Commands tab of the Message Properties dialog box displays any automatic and operator-initiated commands configured by the administrator for the selected message and also shows the node on which the command runs and the status of the command. You can start or stop the configured automatic or operator-initiated command using the Start and Stop buttons. Commands are launched from the active messages browser.

To view automatic and operator-initiated commands

1. In the details pane of the active messages browser, select the message for which you want to view commands.
2. Right-click to open the shortcut menu.
3. Select Properties to open the Message Properties dialog box.
4. Select the Commands tab to view the commands configured for the selected message, the location where the command runs, and the status of the command.
5. To start or stop a command, use the Start and Stop buttons for automatic and operator-initiated commands.

   Click Refresh to see the status of the command you launched. If the command runs on the console, you will not see a status change; status displays as Available. Commands running on the console are not launched from the management server, which reports the status information.
6. Click OK to close the dialog box.

You can also check the success or failure of a command by looking at the message attributes in the Status column of the message browser headline.

Related Topics:

- Launch commands

View active messages for selected nodes or services

When first opened, the active message browser displays all active messages. To focus more precisely on areas of interest to you, you can specify those nodes or services for which you want to view messages.

To view messages from a selected node, node group, or service

1. From the console tree, select the node, node group, or service for which you want to view messages.
2. Right-click to open the shortcut menu.
3. Select View → Active Messages to open the browser in the details pane. The browser displays active messages from the selected node or service and its subcomponent nodes or services.

To view acknowledged messages, see the acknowledged messages browser.

Related Topics:

- Browsing active messages overview
- Browsing acknowledged messages

Message storm detection and suppression

A message storm is the phenomenon that occurs when an unusually high number of new messages arrive on the management server within a short time interval and flood the active message browser. During a message storm, the disc space consumption for the database increases significantly.

Frequently, message storms can lead to management server outages. It could take a significant amount of time to reset the management server to a consistent state.

Message storm root causes

Message storms can occur for the following reasons:

- Wrongly designed policies that generate a high number of messages. In many cases the messages describe the same event.
- Due to some network problems or long-lasting maintenance tasks, agents were disconnected from the management server. During this time, the agents detected multiple problems, generated a high number of messages, and buffered them locally. When the communication to the management server is re-established, the agents send the buffered messages within a short time interval.
- If network devices are included in the managed environment, these devices might generate considerably more SNMP traps than usual in case of serious network failures. If the monitoring policies do not consider this by applying suppression rules, then the system forwards SNMP events endlessly. This can lead to message storms.

Related Topics:

- Detect and suppress message storms
- Configure the message storm detection mechanism

Detect and suppress message storms

As a system administrator, you can take the necessary precautions to detect and, optionally, suppress message storms by configuring the message storm detection mechanism and defining appropriate automatic and operator initiated actions. The message storm detection mechanism is based on the following two message properties:

- Detection based on TimeCreated

  Agents create a high number of messages within a short time interval. The message storm detection is based on the TimeCreated property of the message, which is set on the managed node when the agent creates the message.

  If there are many messages from a certain node where the TimeCreated values are close by, this indicates a message storm. This is the classical case for a message storm. In most cases, the root cause might be wrongly defined policies.

- Detection based on TimeReceived

  Agents send a high number of messages within a short time due to a large backlog of buffered messages on managed nodes. The detection is based on the TimeReceived property of the message, which is set on the management server when the message arrives.

  If there are many messages from a certain node where the TimeReceived values are close by, but the TimeCreated values show normal deltas, then this indicates a message storm. However, this is a less common cause for a message storm.

As administrator, you can configure what to interpret as a message storm for both message properties by setting configuration values in the Server Configuration dialog box in the following namepaces:

- Message Storm Detection Based on Time Created
- Message Storm Detection Based on Time Received

You can enable the detection for both properties in parallel or for only one. As soon as a message storm is detected, a high priority notification message is sent to the console and the automatic action assigned to the message is launched.

NOTE:
By default, the message storm detection mechanism is disabled. Message storm detection works on a per node basis.

Related Topics:

- Message storm detection and suppression
- Configure the message storm detection mechanism

Configure the message storm detection mechanism

Message storm detection is disabled by default. By setting appropriate values in the Server Configuration dialog box, you can configure the message storm detection mechanism according to the individual needs of your managed environment. You can configure what to interpret as a message storm by setting configuration values in the following namepaces:

- Message Storm Detection Based on Time Created
- Message Storm Detection Based on Time Received

The following table shows the values, which are the same in both namespaces used by the message storm detection mechanism. Note that these values have to be defined separately for each of the two available mechanisms. The assigned configuration values however can differ between two mechanisms.

Value Name	Possible Values	Description
Enable message storm detection based on time created Enable message storm detection based on time received	True, False	Used to enable or disable message storm detection. Set it to true to enable, set to false to disable.
Time interval to analyze	Any value from 1 to 604800	Time interval in seconds over which the message flow is analyzed.
Number of messages for beginning of message storm	Any value from 2 to 1000000	Used in combination with the time interval to analyze. If based on the message property TimeCreated, then it specifies the maximum number of messages which can be created on a managed node within the defined time interval. If based on the property TimeReceived, then it specifies the maximum number of messages which can be received on the management server from a managed node within the defined time interval. If this value is exceeded, then this indicates a message storm. In this case the mgmt server automatically creates a high priority message and sends it immediately to the console. This message contains information about the affected node, message count, and time interval. The severity of this message can be configured with the value "Severity of the message storm begin message". Optionally, both an automatic and operator- initiated

		action can be associated with this message by using the values "Automatic action of the message storm begin message" and "Operator-initiated action of the message storm begin message". The actions could contain the command to start the "opcragt" tool to stop the agent on the affected node on demand.
Suppress messages during message storm	True, False	Defines whether to suppress messages for a managed node for which a message storm has been detected. If the option suppression is selected, then all messages for the affected managed node will be suppressed until the message storm is over. The end of a message storm is detected automatically. Use the value false if you do not want messages to be suppressed; use the value true if you want messages to be suppressed. Suppressed messages are not stored in the database and are lost.
Number of messages for end of message storm	Any value from 1 to 1000000	Defines the number of messages to fall below until a message storm is indicated as over. If the message storm is over, then the message suppression is stopped automatically for the affected node. Then the management server creates a high priority message and sends it immediately to the console to inform operators about the end of the message storm. This message contains information about the affected node, message count, and time interval. The severity of this message can be configured with the registry value "Severity of the message storm end message". Optionally, both an automatic and operator- initiated action can be associated with this message by using the registry values "Automatic action of the message storm end message" and "Operator-initiated action of the message storm end message".

Severity of the message storm begin message	2,4, 8, 16, 32	Severity of the high priority message sent to the console to indicate the start of a message storm. Use the following numeric values:
		
		`2  normal`
		`4  warning`
		`8  minor`
		`16 major`
		`32 critical`

Severity of the message storm end message	2, 4, 8, 16, 32	Severity of the high priority message sent to the console to indicate that a message storm is over. Use the following numeric values: `2  normal`

		`4 warning`
		`8 minor`
		`16 major`
		`32 critical`

Automatic action of the message storm begin message	String	Name of a command file (.cmd or .bat) or name of a script file (.vbs) to be associated as an automatic action to the high priority message that is sent when a message storm is detected. Run time options and variables can be optionally added following the command/script file name. The automatic action is executed on the management server immediately as a high priority action. The action execution does not involve the agent. It is also executed if the agent is stopped or is not installed on the management server. Possible tasks for the automatic action could be: ■ Stop the agent on the affected managed node by using the "opcragt" tool. ■ Send an email to the operators. ■ Page the operators. Assign an empty string if no automatic action should be assigned to the message. Use the file %OvInstallDir\bin\OvMsgStormStartAutoTmpl.cmd as the template for a command file that could be assigned as an automatic action to a high priority message.
Automatic action of the message storm end message	String	Name of a command file (.cmd or .bat) or name of a script file (.vbs) to be associated as an automatic action to the high priority message that is sent when a message storm is over. Run time options and variables can be optionally added following the command/script file name. The automatic action is executed on the management server immediately as a high priority action. The action execution does not involve the agent. It is also executed if the agent is stopped or is not installed on the management server. A possible task for the automatic action could be to send an email to the operators to indicate the end of a message storm. Assign an empty string if no automatic action should be assigned to the message.
Operator-initiated action of the message storm begin message	String	Name of a command file (.cmd or .bat) or name of a script file (.vbs) to be associated as an operator-initiated action for the high priority message that is sent when a message storm is detected. Run time options and variables can be optionally

		added following the command/script file name. A possible task in the operator-initiated action could be to restart the agent on the affected managed node by using the "opcragt" tool. Assign an empty string if no operator-initiated action should be assigned to the message. Use the file %OvInstallDir\bin\OvMsgStormStartOperatorTmpl.cmd as a template for a command file that could be assigned as an operator-initiated action for a high priority message.
Operator-initiated action of the message storm end message	String	Name of a command file (.cmd or .bat) or name of a script file (.vbs) to be associated as an operator-initiated action for the high priority message that is sent when a message storm is over. Run time options and variables can be optionally added following the command/script file name. Possible tasks depend on the managed environment and user requirements. Assign an empty string if no operator initiated action should be assigned to the message.

Message key prefix for message storm messages	Any string	User-defined prefix of the message key to add to the high priority messages generated when a message storm is detected and when a message storm is over. One of the following keys is constructed when a message storm is detected:
		
		`<MsgKeyPrefix>:TimeCreated:start:<NodeID>`
		
		`<MsgKeyPrefix>:TimeReceived:start:<NodeID>`
		
		One of the following keys is constructed when a message storm is over:
		
		`<MsgKeyPrefix>:TimeCreated:end:<NodeID>`
		
		`<MsgKeyPrefix>:TimeReceived:end:<NodeID>`
		
		NOTE:
		Defining this value ensures that a message created at the end of a message storm automatically acknowledges the message created when the message storm was detected.
		
		Assign an empty string if no message key should be created and to suppress automatic acknowledgement.

Service name for message storm messages	Any string	Service name that should be assigned to the high priority messages. The following format is used: `<ServiceName>@@<NodeID>`

NOTE:
If message storm detection is enabled, then it is necessary to insert all values correctly. Missing incorrect values will disable message storm detection.

Change the configuration dynamically

After the initial configuration of the message storm detection mechanism, the OvEpMessageActionServer service must be restarted to read the configuration values.

Subsequent changes to the configuration values, however, do not require a restart of the OvEpMessageActionServer service. Values can be changed at any time during production and the new values take effect immediately, so it is possible to dynamically enable and disable the message storm detection mechanism.

NOTE:
Any dynamic change to the configuration values while the message storm detection mechanism is enabled will reset the message flow analysis by starting a new measuring time interval. Metrics collected up to this point are ignored.

Related Topics:

- Message storm detection and suppression
- Variables used in message storm detection

Variables used in message storm detection

Variables can be added as run time options when starting the command files or scripts assigned to high priority messages. See the values:

- Automatic action of the message storm begin message
- Automatic action of the message storm end message
- Operator-initiated action of the message storm begin message
- Operator-initiated action of the message storm end message

The following table lists the variables that you can use.

Variable	Description
<$MSGSTORM_MSG_ID>	Returns the unique identity number of the high priority notification message, as generated by the message storm detection mechanism.
<$MSGSTORM_NODE>	Returns the IP address of the managed node that caused the message storm.
<$MSGSTORM_NODE_NAME>	Returns the node name of the managed node that caused the message storm.
<$MSGSTORM_NODE_ID>	Returns the GUID of the managed node that caused the message storm.
<$MSGSTORM_TYPE>	Returns the type of the message storm. Value 1 is returned for message storms based on the message property "TimeCreated." Value 2 is returned for message storms based on the message property "TimeReceived"
<$MSGSTORM_NUM_MESSAGES>	If a message was generated to indicate that a message storm has begun, then this variable returns the number of messages received within the time interval for which the message flow has been analyzed. If a message was generated to indicate that a message storm is over, then this variable returns the number of messages received within the time interval for which the message flow has been analyzed. The returned number must be below the number specified with the registry value "RecoverCount."
<$MSGSTORM_TIME_INTERVAL>	Returns the time interval in seconds during which the message flow was analyzed
<$MSGSTORM_EVENT>	Returns the value 1 if message has been created to indicate a message storm; returns the value 2 if message indicates that the message storm is over.

<$OPC_MGMTSV>	Returns the node name of the management server where the message storm has been detected.

Related Topics:

- Detect and suppress message storms
- Configure the message storm detection mechanism

High priority messages and actions

Message storms are exceptional situations which can lead to disruption in your environment. It is important that the notification message generated by the message storm detection mechanism be sent immediately to the management consoles and that the automatic action assigned to it be executed without any delay. The management server achieves this by using high priority messages, which have the following characteristics:

- They are generated and sent directly by the management server, without involving the agent.
- They are not placed into the message queue but are sent immediately.
- Automatic actions assigned to high priority messages are treated as emergency actions and are executed immediately without involving the agent. Consequently, the agent on the management server does not have to run to handle high priority messages and actions.
- High priority messages cannot be created by policies.

NOTE:
High priority messages do not automatically have Critical severity. Their severity can be defined by using the values "Severity of the message storm begin message" and "Severity of the message storm end message".

Related Topics:

- Message storm detection and suppression
- Variables used in message storm detection
- Messages generated by message storm detection

Messages generated by message storm detection

If a message storm is detected, then the message storm detection mechanism creates a high priority message with the following properties.

Property	Value
Severity	Set to the severity as defined with the value "Severity of the message storm begin message"
Group	OVO
Application	OVO
Object	OVOMsgStorm
Message key	<MsgKeyPrefix>:TimeCreated:start:<NodeID> Or <MsgKeyPrefix>:TimeReceived:start:<NodeID>
Service name	<ServiceName>@@<NodeID>
Text	Message storm detected on <NodeName>. Detection was based on the message property <TimeCreated | Time Received>. Node Id: <NodeID> Number of received messages: <Nbr of messages received from affected node> Time interval: <Nbr of seconds>

NOTE:
The output of the automatic and operator-initiated actions is automatically added as annotations to the message and the message reflects the execution states of these actions.

If a message storm is over, then a high priority message with the following properties is created by the message storm detection mechanism.

Property	Value
Severity	Set to the severity as defined with the value "Severity of the message storm end message"
Group	OVO
Application	OVO
Object	OVOMsgStorm
Message key	<MsgKeyPrefix>:TimeCreated:end:<NodeID> Or <MsgKeyPrefix>:TimeReceived:end:<NodeID>
Service name	<ServiceName>@@<NodeID>
Text	Message storm on node <NodeName> is over, because the number of received messages is below the configured maximum number. Detection was based on the message property <TimeCreated | Time Received>. Node Id: <NodeID> Configured maximum number of messages: <RecoverCount> Number of received messages: <Nbr of messages received from affected node> Time interval: <Nbr of seconds>

NOTE:
The output of the automatic and operator-initiated actions is automatically added as annotations to the message and the message reflects the execution states of these actions.

Special considerations for detecting the end of message storms

The detection of both the start and the end of message storms is based on analyzing the message flow. However, if your administrator configured automatic actions to stop the agents on the affected nodes, then agents will not send messages until they are restarted, so message flow is stopped.

In order to detect the end of message storms when the message flow is stopped, the message storm detection mechanism maintains a list of managed nodes for which a message storm has been detected. This list is checked every five minutes to see whether new messages have arrived from the nodes. If no messages have arrived in the last five minutes and the configured time interval for message storm recovery has passed, then a high priority message is generated to indicate that the message storm is over.

Related Topics:

- Detect and suppress message storms
- Configure the message storm detection mechanism
- Variables used in message storm detection
- High priority messages and actions

Managing message delays

A message delay is the phenomenon that occurs when new, incoming messages that arrive on the management server are propagated with a significant time delay to different internal consumers (for example, WMI, console, status engine).

The biggest contributor to message delays is the OvEpMessageFilter component of the OvEpMessageActionServer service. The OvEpMessageFilter component validates whether messages originate from authorized managed nodes. All messages must pass this validation before they can be routed to consumers. OvEpMessageFilter uses the OvEpNodeCache component which is responsible for determining valid managed nodes.

Message validation is fast for all messages that include an AgentId and where the sending node is set up as an HPOM managed node. In all other cases the validation must pass a multiple level name resolution process which can cause significant delays.

Possible root causes for message delays are:

- Messages from deleted nodes

  Messages are sent from nodes that have been deleted on the management server but the agents have not been uninstalled and are still sending messages.

- Messages from agentless nodes

  A proxy node gathers information (for example, SNMP traps) from agentless nodes and forwards this information as HPOM messages to the management server. The validation of messages from proxy nodes may also take considerable time as these messages do not contain an AgentId.

In the worst case, delivery of new messages stops completely and the message pipeline hangs.

Related Topics:

- Detect nodes that cause a message delay
- Configure message delay detection

Detect nodes that cause a message delay

To detect nodes that contribute to message delays, you can measure the time it takes to resolve node names and the time it takes to route a message from the message queue to all consumers.

- Measure the time for node name resolution (OvNameResMsgFilter)

  To measure the node name resolution, you define a threshold value (in milliseconds) that the node name resolution is allowed to take. If the measured value exceeds the threshold value, the logging mechanism stores the resolution information (node name, IP address, time to resolve, and so on) in a user-configurable log file. Recommended threshold values range from 50 to 200 milliseconds.

- Measure the time for message routing (OvMessageFlow)

  To measure the time for message routing, you define a threshold value (in milliseconds) it can take to route a message to all consumers. (This includes the time used for the node name resolution.) If the measured value exceeds the threshold value, the logging mechanism stores the routing information (node name, IP address, proxy information, message ID, first 40 characters of the message, discard information, time to validate the sending node, and so on) in a user-configurable log file. Recommended threshold values range from 200 to 1000 milliseconds.

NOTE:
Enabling message delay detection does not affect the behavior of node resolution or message routing. It only generates log information when thresholds are exceeded.

Log information for node name resolution

The following lines show an extract of a log file for measuring node name resolution (OvNameResMsgFilter):

```
Measure: OvNameResMsgFilter
Component: OvEpMessageFilter
Time Started:  2007-05-19 14:20:18.413
Node Name: node_x
Resolved: no
Time Finished: 2007-05-19 14:20:25.319
Elapsed time: 6906
State: Finished with threshold exceeded
*-*-*-*-*-*-*-*-*-*
Measure: OvNameResMsgFilter
Component: OvEpMessageFilter
Time Started:  2007-05-19 17:33:18.406
Agent Id: 25da9f90-fe5c-71d7-0422-0f887c4c0000
```

```
Node Name: node_y
Ip Addr: 15.136.124.76
Resolved: yes
Time Finished: 2007-05-19 17:33:18.609
Elapsed time: 203
State: Finished with threshold exceeded
*-*-*-*-*-*-*-*-*-*
```

The first log entry shows that node "node_x" could not be resolved. The elapsed time for this measure interval was 6906 milliseconds. The message did not contain an AgentId or IP address, but a node name only and the node does not exist as managed node.

The second log entry shows that node "node_y" with ip 15.136.124.76 could be resolved as a managed node but the resolution time exceeded the configured threshold value.

Log information for message routing

The following lines show an extract of a log file for measuring message routing (OvMessageFlow):

```
Measure: OvMessageFlow
Component: OvEpMessageFilter
Time Started:  2007-05-19 11:07:25.155
Node Name: node_y
Ip Addr: 15.136.124.76
Time Created: May 19 11:07:25
Time Received: May 19 11:07:25
OvEpMessage: Id="f1da7a30-a973-71d8-1afb-0f887c4c0000", Text="This is a good message..."
Proxied: no
NodeResTime: 0
Discarded: no
Time Finished: 2007-05-19 11:07:25.217
Elapsed time: 63
State: Finished with threshold exceeded
*-*-*-*-*-*-*-*-*-*
Measure: OvMessageFlow
Component: OvEpMessageFilter
Time Started:  2007-05-19 11:16:32.337
Node Name: node_x
Time Created: May 19 11:16:25
Time Received: May 19 11:16:32
OvEpMessage: Id="37ffd310-a975-71d8-1afb-0f887c4c0000", Text="Wrong node ..."
Proxied: yes
NodeResTime: 6938
Discarded: yes
Time Finished: 2007-05-19 11:16:39.274
```

```
Elapsed time: 6938
State: Finished with threshold exceeded
*-*-*-*-*-*-*-*-*-*
```

The first log entry shows that a message has been routed to all consumers in 63 milliseconds, but it exceeded the configured threshold. The time for the node resolution (field NodeResTime) was below 1 millisecond. (The name resolution worked well.) The message came from a managed node and was not discarded.

The second log entry shows that a message was discarded because it was proxied and the node name of the originating node was not known on the management server. The resolution time for the node was 6938 milliseconds and this exceeded the threshold.

NOTE:
The logging mechanism always logs discarded messages, whether they exceed the threshold or not.

Related Topics:

- Configure message delay detection

Configure message delay detection

Message delay detection is disabled by default. By setting appropriate values in the Server Configuration dialog box, you can configure the message delay detection mechanism according to the individual needs of your managed environment. You can configure how to detect message delays by setting configuration values in the following namepaces:

- Message Delay Detection Measuring Node Name Resolution
- Message Delay Detection Measuring Complete Message Routing

The following table shows the values, which are the same in both namespaces used by the message delay detection mechanism. These values have to be defined separately for each of the two available mechanisms. The assigned configuration values however can differ between two mechanisms.

Value Name	Possible Values	Default Value	Description
Enable message delay detection measuring node name resolution Enable message delay detection measuring the complete message routing	True, False	False	This value enables or disables message delay detection based on the time it takes to resolve a node name or the time it takes to route a message through the HPOM message flow. Set it to True to enable, set to False to disable.
Log file name	String	C:\Program Files\HP\HP BTO Software\ Support\ MeasureLog.txt	The fully qualified name of the log file. If the file does not exist, it will be created automatically. The directory where the file will be stored must already exist. All new entries are appended at the end of the file. (Make sure sufficient disk space is available.) You can use the same log file for measuring both node name resolution and message routing.
Measuring type	NORMAL, CRITICAL	NORMAL	A NORMAL measure waits until the measure interval has finished, then compares the elapsed time with the threshold and generates log information if the threshold has been exceeded. Only one single log entry is created per measure interval. This is the recommended measure type.

			A CRITICAL measure immediately generates log information as soon as the threshold value has been exceeded and does not wait until the end of the measure interval. It generates two log entries: the first one when the threshold has been exceeded and a second one when the measure interval has finished. If the second log entry is missing, then this indicates a hang during the measure interval. However the first entry contains sufficient information for finding out which message caused the hang. CRITICAL measures should only be used for tracking down hangs while routing messages and the threshold value should be set to a high value (more than 60000 milliseconds).
Threshold value (in milliseconds)	Integer	Node name resolution: 100 Message routing: 500	The threshold value in milliseconds. If the threshold is exceeded then appropriate log records will be written to the log output file. Recommended values: Node name resolution: 50 to 200 milliseconds Message routing: 200 to 1000 milliseconds

Related Topics:

- Detect nodes that cause a message delay
- Change server configuration values

Configuring the browser display

You can customize the appearance and content of your message browser in several ways:

- Change the message browser column display options
- Change the default message browser limit
- Configure policies from the message browser
- Stop a command
- Turn on browser error dialog boxes
- Turn off browser error dialog boxes

For information about applying filters to limit the display of messages, see the help topic Filter browser messages .

Change message browser column display options

You can choose to show any combination of the following message attributes in the columns of your message browsers:

- Severity
- Duplicates
- State
- Unmatched
- Instructions Available
- Automatic Command
- Operator-initiated Command
- Annotations Available
- Received
- Created
- Service
- Node
- Application
- Object
- Text
- Message Group
- Policy
- Policy Type
- Origin
- Sender
- User of last state change
- Message key

You can also show up to ten columns of custom message attributes (CMAs).

To change the column display

1. Right-click the column headings in the message browser, and then click **Options...** . The Message Header Options dialog box opens.

2. To toggle display of a column, select or clear the check box beside the column you want to show or hide.

   TIP:
   You can also hide an individual column by right-clicking the column name in the message browser, and then clicking **Hide Column** .

3. To create a custom message attribute column, type the title of the column that you want to add to the

browser, and then click Add . The column name appears in the Custom Message Attributes group box. Use the Remove and Remove All buttons to delete selected column names you have created.

4. Click OK to close the dialog box and confirm your changes. When you return to the message browser, the options you checked will take effect.

Related Topics:

- Browsing messages
- View and edit custom message attributes
- Specify console message browser properties

Configure policies from message browser

HPOM consolidates a number of tasks related to the policies that generate messages to one convenient interface accessible from the message browser itself. By launching the policy editor directly from the message generated by the policy itself, you can fine tune a new policy to monitor a new application, isolate and modify a policy that is generating messages during message storms, and reconfigure a policy to recognize an unknown message. Specifically, from the message browser, you can:

- Quickly suppress message generation
- Edit the policy condition that generated a message with a mouse click
- Quickly match unknown messages
- Deploy policies directly from the policy editor
- Deploy a policy based on version already deployed.

The Policy Type Table shows which features are available for specific policy types. See the related topics for details on performing specific policy-related tasks.

Related Topics:

- Launch policies from message browser
- Create a suppress condition from within the message browser
- Create a match condition from within the message browser

Change default message browser limit

The active and acknowledged messages browsers have a default limit of 50,000 messages that can be viewed at one time. Each open message browser, map view, or list view constitutes an MMC view that appears in a separate window within the MMC. The limit applies to the combined open views, so that if you have two message browser views open (or a message browser and a map view), the combined number of messages displayed in both views cannot exceed 50,000 messages.

A view can be active or inactive. An active view appears on top of other windows in the MMC and its title bar is colored blue. Any inactive views are located behind the active view, and their title bars are colored gray. To change an inactive view to the active view, left-click once in the inactive view. It will become active and appear on top of any other views.

When the default message limit is exceeded, the oldest messages are deleted, at the rate of 10% at a time. The browser removes all messages in those views not being looked at (inactive views) and also removes the view. For example, if you have an active view and three inactive views in the background, to remain within the 50,000 message limit, the browser would first remove the inactive views, then remove 10% of the messages in the active view.

To configure the browser cache limit:

1. From the console tree, select Operations Manager.
2. Right-click to open the shortcut menu.
3. Select Properties to open the Properties dialog box which you can use to configure console settings.
4. In the General tab, specify the maximum level of cached messages.
5. Click OK to save your changes and close the dialog box.

Create a suppress condition from within the message browser

From within the message browser, you can create a new condition for a message that suppresses future generation of the message. When you select a message to create a suppress condition, as described below, the system creates a policy condition that instructs the agent to suppress generation of messages that match the selected message's original event text.

In this way, you can control the type and number of messages that appear in the message browser and can respond quickly to message storms by suppressing messages generated from a particular policy.

To create a suppress condition

1. From within the message browser, select the message for which you want to create the suppress condition and right-click to open the context menu.
2. Select Policy → Create Suppress Condition from the menu to open the appropriate policy editor, which displays the New Rule dialog with the Condition tab active and the generated default displayed. Accept the defaults.
3. Select the Actions tab. The default setting for here for suppress conditions is Do nothing: stop evaluation.
4. If you prefer, you can change the default to suppress the message based on some criteria other than the message text. Provide a description of the rule and specify its conditions.
5. Click OK . Your new rule will appear on the Rules tab of the policy editor dialog box. Incoming messages matching the criteria in this rule will be suppressed and will not appear in the message browser.

NOTE:
Not all policy types offer the Create message suppression menu option.because they do not have condition IDs. Scheduled policy type is an example. For these policy types, the Policy → Edit Policy menu option opens the appropriate policy editor to the specific policy that generated the message.

Details on setting conditions for several specific policy types can be found in help in this path:

Administering your Environment → Event Policy Editors → Configure Rules → Configure Conditions .

Create a match condition from within the message browser

From within the message browser, you can create a new condition for a message or modify the condition that matches a generated message. When you select a message to create a match condition, as described below, the system creates a new condition that matches on the selected message's original event text. After you save and deploy the policy containing the new condition, the agent on the managed node generates a message when an event it receives matches the new condition.

The message generated by the event can be preconfigured to have specific properties, such as severity, instruction text, and the like. The message can also be configured to launch automatic and operator-initiated commands.

To create a match condition

1. From within the message browser, select the message for which you want to create the match condition.
2. Right-click to open the context menu.
3. Select Policy → Create Match Condition from the menu to open the appropriate policy editor, which displays the New Rule dialog with the Condition tab active and the generated default displayed. Accept the defaults.
4. To modify the default, select the Actions tab and provide a description of the rule and specify its conditions.
5. Click OK . Your new rule will appear on the Rules tab of the policy editor dialog box. Incoming messages matching the criteria in this rule will appear in the message browser. The Actions tab default setting for match conditions is Send Message... .
6. To modify the actions associated with this message, double-click the rules description on the Rules tab to open the Rule dialog.

   Select the Actions tab, then select the location where the message should be sent and indicate any associated automatic or operator-initiated actions.
7. Click OK to confirm your changes and close the dialog box.

NOTE:
Not all policy types offer the Create Match Condition menu option because they do not have condition IDs. Scheduled policy type is an example. For these policy types, the Policy → Edit Policy menu option opens the appropriate policy editor to the specific policy that generated the message.

Details on configuring messages and adding automatic and operator-initiated commands to messages can be found in help in this path:

Administering your Environment → Event Policy Editors → Configure Rules → Configure Actions .

Turn on browser error dialog boxes

Use this tool to turn on the display of error messages for messages that you perform mass operations on in your active messages browser. If your browser has been set to hide the error message display, you can turn the display back on using this tool.

To turn on browser error dialog boxes

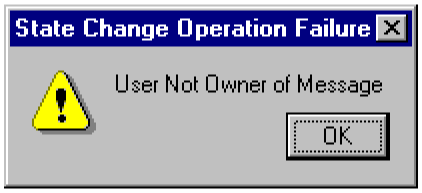
1. In the console tree, select Operations Manager .
2. Right-click to open the shortcut menu.
3. Select Properties to open the Properties dialog box.
4. In the General tab, check the Display error dialogs from message browser check box to enable the display of error messages.
5. If you prefer not to display these error messages, be sure this check box is cleared.
6. Click OK to save your changes and close this dialog box.

Related Topics:
- Mass operations on messages
- Turn off browser error dialog boxes

Turn off browser error dialog boxes

Use this tool to turn off the display of error messages for messages that you perform mass operations on in your active messages browser. This tool helps you save time and skip repeating the closure of the same error message box for a block of messages.

For example, assume your active messages browser has a block of 1,000 messages that are owned by another operator. You decide to select all 1,000 messages to perform a mass operation.. Before the operation can complete, the following error message displays:



State Change Operation Failure

User Not Owner of Message

To continue working, you would have to select OK and close each of these error messages. To avoid having to OK each error message, you can turn off all the error messages by using the Browser Error Dialogs Off tool. This tool turn off the errors associated with all messages that are owned, disowned, acknowledged, and unacknowledged. It can also be used on messages where the severity has been changed.

To turn off browser error dialog boxes

1. In the console tree, select Operations Manager .
2. Right-click to open the shortcut menu.
3. Select Properties to open the Properties dialog box.
4. In the General tab, check the Display error dialogs from message browser check box to enable the display of error messages.
5. If you prefer not to display these error messages, be sure this check box is cleared.
6. Click OK to save your changes and close this dialog box.

Related Topics:
- Mass operations on messages
- Turn on browser error dialog boxes

View and edit message properties

You can view message properties and edit certain message attributes using the Message Properties dialog box, which displays information about a message in greater detail than is shown in the message browser headline. Only one Message Properties dialog box displays at a time per each message browser view.

To view message properties

Double-click the message in the messages browser or follow the steps below.

1. In the details pane, right-click a message in the message browser to open the shortcut menu.
2. Select Properties to open the Message Properties dialog box. The Text tab displays by default.
3. Review general information, annotations, instructions, message text, state, custom message attributes, and actions for the selected message.

To edit the annotations and text message properties, select the appropriate tab in the Message Properties dialog box and enter your changes.

Use the Previous (Up arrow), Next (Down arrow), and Acknowledge buttons to move the focus back and forth in the acknowledged or unacknowledged messages browsers to make and apply changes. The icon on the Acknowledge button changes, depending on which message browser you are working from.

If you leave the message browser up, you can work from within the Message Properties dialog box to modify the instructions, state, text, commands, annotations, duplicates, and custom message attributes for a selected message, acknowledge and unacknowledge a selected message, and see your changes immediately take effect in the browser window.

Related Topics:

- View general message properties
- View message annotations
- Edit annotation text
- View duplicates
- View message instructions
- View and edit message text
- View message state
- View and launch actions
- View and edit custom message attributes (CMA)

View general message properties

Use the General tab in the Message Properties dialog box to view details about a selected message. Properties are read-only, with the exception of status. You can change severity from this tab.

- ID: The unique identifier for this message.
- Dropdown list and status icon: The list allows you to change the severity of the selected message. The icon indicates the severity of the selected message.
- Time First Created on Node: The date and time the message was generated on the agent system. This time always displays using the time zone of the agent at creation time (for example, 11:30 (CET/winter). This means that this time always displays in this fixed time zone.
- Time First Received on Server: The date and time the message was received on the server.
- Number of Duplicates: The number of duplicates for the selected message.
- Time Last Received on Server: The date and time the message was received on the server. This time always displays using the current time zone of the HPOM for Windows server (for example, 02:30 (PST/winter). This means that the time zone in which this time displays may change if the HPOM server time zone changes (for daylight savings time, for example). During daylight savings time, the Time from the example would be recalculated to 03:30 (PST/summer).
- Sender: The name of the management server that forwarded the message to this management server.
- Origin: The name of the management server that originally received this message, and then forwarded it to one or more other management servers.
- Primary Node Name: The name of the node generating the message, if any.
- Service Id: Uniquely identifies a service and maps messages to that service. All messages contributing to the severity calculation of this service contain this service's Service ID as a message attribute.
- Message Group: The name of the group this message was assigned to by the administrator.
- Message Type: Shows the subgroup a message has been assigned to (for example, to show the occurrence of a specific problem).
- Message Key: The identifier assigned to all messages produced by a particular policy rule. The same message key might be assigned to messages produced by different rules in different policies.
- Acknowledge Message with Message Key: A message key that this message searches for. This message has acknowledged any messages with this message key that were in the active messages browser when this message was received.
- Policy: The name of the policy originating the message.
- Policy Type: The policy type the originating policy belongs to.

- Application: The name of the application generating the message, if any.
- Object: The name of the object generating the message (for example, CPU).
- Unmatched: If checked, this check box indicates that the message does not match any known conditions and does not belong to any policy type.

To view general message properties

1. From the details pane, select a message and right-click to open the shortcut menu.
2. Select Properties to open the Message Browser dialog box. The General tab displays by default.
3. If desired, change the severity of the selected message.
4. When you finish looking at this tab, you can select another tab to view more details about the message you selected.
5. Click OK to close this dialog box.
6. Click Cancel to close this dialog box without saving your changes.

View and edit custom message attributes (CMA)

Use the Custom Message Attributes tab in the Message Browser Properties dialog box to create, edit, and remove additional attributes for messages that you have selected. For example, you might add a company name, contact information, or a city location to a message; a CMA can be any information that is meaningful to you and you can have more than one CMA attached to a single message.

This attribute information displays in the message browser in a column you have previously created to contain it. See the help topic Change message browser column display options for details on creating additional columns in the browser.

You can choose from the list of static attributes and currently available CMAs to display in the message browser. CMAs will be available only if they have been received on the server and are already attached to messages in the database.

To create a custom message attribute:

1. In the details pane, select the message for which you want to create a custom message attribute.
2. Right-click the message to open the Message Properties dialog box.
3. Select the Custom Message Attributes tab.
4. Click New to open the New CMA dialog box.
5. Type a name for the new CMA in the Name box.
6. Type the CMA attribute to associate with this name in the Value box.
7. Click OK . The new CMA attribute displays in the browser in the CMA column you have previously created using the Options dialog box.

To edit a custom message attribute

1. In the details pane, select the message for which you want to edit the custom message attributes.
2. Right-click the message to open the Message Properties dialog box.
3. Select the Custom Message Attributes tab.
4. Select the CMA you want to edit and click Edit to open the Edit CMA dialog box.
5. Edit the attribute in the Value box. The new value appears on the Custom Message Attributes tab.
6. Click Apply , then OK to confirm your choice and close this dialog box. The edited attribute appears in

the message browser column you previously created.

To delete a custom message attribute

1. In the details pane, select the message for which you want to edit the custom message attributes.
2. Right-click the message to open the Message Properties dialog box.
3. Select the Custom Message Attributes tab.
4. Select the name of the attribute you want to delete.
5. Click Delete to remove the attribute from the Custom Message Attributes tab.
6. Click Apply , then OK to remove the attribute from the message browser column.

View and edit message annotations

Annotations are short notes that summarize the actions taken to resolve problems and can be used by others to resolve similar situations. You can add annotations to a message at any time and can review existing annotations using the Annotations tab in the Message Properties dialog box.

Messages that have annotations associated with them display an X in the N column of the message browser.

To create, edit, and delete annotations

1. In the active messages browser, select a message.
2. Right-click to open the shortcut menu.
3. Select Properties to open the Message Properties dialog box and click the Annotations tab. You can also open the Annotations tab directly from the shortcut menu.
4. Annotations associated with the selected message are listed in order of their creation. The text of the annotation appears in the Text box and can be edited. Select an annotation from the list to see its complete text in the Text box.
5. To create a new annotation, click New... to open the New Annotation dialog box.
6. Enter your comments and click OK . The annotation appears in the Text box in the Annotations tab of the Message Properties dialog box. You can edit your annotation now or at a later date by selecting the message on the Annotations tab and clicking Edit . This displays a text box containing your message, which can be edited.
7. Click OK to save the annotation and close this dialog box. The annotation count is incremented in the message browser.
8. To delete an annotation, select it and click Delete . The annotation is removed from the message in the active messages browser.

   NOTE:
   Annotations added by the system as a result of an automatic action specified by a policy cannot be deleted. If you try to delete a SYSTEM annotation, a message displays telling you that the delete operation failed.

9. Click Cancel to close this dialog box without saving your annotation.

View duplicates

The number of duplicate messages appears in the Duplicates column of the MMC and Web console browsers. Stored duplicates (duplicate messages) are listed in the Duplicates tab of the Message Properties dialog box. The list is read-only; you cannot add, delete, or edit the list.

The list displays the message ID and time the duplicate was created. You can view the text, severity, object, application, message group, and node for the message by clicking on the items in the list. The information displays in the Text area at the bottom of the list.

The first item in the list contains the information for the original message. Subsequent listings contain the information for duplicate messages. Use the Up and Down arrows to scroll through the messages listed in the browser.

You can acknowledge a selected message by clicking the Acknowledge button located below the Down arrow.

View message instructions

Message instructions are configured by your administrator to help you solve common problems. They might include details describing:

- Automatic actions.
- Operator-initiated actions.
- Manual steps to follow for problem resolution.

Instructions are read-only and cannot be edited.

To view message instructions

1. In the details pane, select the message for which you want to view instructions.
2. Right-click the message to open the shortcut menu.
3. Select Properties to open the Message Browser dialog box.
4. Select the Instructions tab to view instructions configured by your administrator for this message.
5. When you finish viewing the instructions, click OK to close this dialog box.

View or change message state

Use the State tab in the Message Properties dialog box to view or change the state of a selected message. Details include:

- Owned: If checked, indicates the message is owned by the person shown in User of Last State Change: box. If you are the owner of the message, and you want to disown it, clear the Owned: check box. Results of the change appear in the browser.
- Acknowledged: If checked, indicates that the message was acknowledged by the person shown in the User of Last State Change: If you are the owner of this message and you want to unacknowledge it, clear the Acknowledged: check box. Results of the change appear in the browser.
- User of Last State Change: box. Displays the node name and user ID for the last user to own or acknowledge the message.
- Time of Last State Change: Displays the time the state last changed. This time is always displayed using the current time zone of the HPOM for Windows server and is similar to the handling of the Received box in the General tab.
- Click OK to confirm your changes and close this dialog box.

Diagnosing problems using the map view

Problems you are alerted to in the message browser can often best be solved using the perspective provided by the map view of your environment. The map view presents a graphical view of your entire service or node hierarchy, including any subsystems or subservices. Nodes, services, and their components are represented as icons, color-coded to indicate the current state of the node or service.

When changes occur in the environment, HP Operations Manager automatically updates the map view to present the latest configuration and status. Opening the map view from the message browser displays a map in a new window. You can tile the windows horizontally to view both the message browser and the map view at the same time.

For a closer look at a specific area of concern, you can select a node or service on the map and left-click to redraw the map with the selected item at the center of the map. All subcomponent details such as subservices or managed devices are displayed below your selection.

Selecting any item in the map view displays messages from that item and any subcomponents below it in the hierarchy. Left-click an object in the map view to place that object in the center of the map. If you click in a blank area of the map, the point where you clicked becomes the center of the map and objects are repositioned accordingly. If your administrator has provided descriptions of the items in the map, they display as ToolTips. Captions can contain up to 500 characters.

The map view offers specialized views that help you diagnose problems:

- Root cause analysis performs a top-down investigation of the hierarchy of your selected service or node and stops at the hierarchical level of the service or node that caused the status to change. There may be more levels below that, which are not displayed because the services or nodes on those levels did not contribute to the problem.
- Impacted analysis works in the other direction, from the bottom up, by searching through the service hierarchy to display all other services or nodes that are impacted by the change in status. The nature of the impact is determined by the status propagation and calculation rules configured by your administrator. The impact path of a service or node also displays in the console tree.

Because the impact analysis only considers negative impacts (status changes from good to bad) the impact graph does not necessarily display the top-level service or node, but stops where impact occurred.

Related Topics:

- Show root cause
- Show impacted services or nodes
- Open a map view of nodes or services

Open a map view of nodes or services

The map view presents a graphical view of your entire service or node hierarchy, including any subsystems or subservices. Use the map view to drill down to the level in your node or service hierarchy where a problem is occurring.

You can open the map view from the console tree, active and acknowledged message browsers, or from another map view. Depending on your original selection from the console tree and the type of map you select to view, your map view will show different results:

- If you select a node group from the Nodes folder in the console tree, your map view show a node hierarchy starting with the selected node at the top of the view and show nodes only.
- If you select a specific node, you can view a hosting services map that shows both nodes and services, or a root cause or impacted services map, which show only nodes.
- If you select a service from the Services folder in the console tree, your map view displays a service hierarchy, beginning with the service you selected.

You can select a message in either the active or acknowledged message browsers and right-click to choose a root cause, impacted, or uses/contains map for the selected message. From the resulting map view, you can right-click again to make further selections of contained By or Hosting Services map views.

The icons and lines in your map are color-coded to indicate the severity levels of items in the map and to show status propagation. Black lines in the map indicate that status is not propagated up from a subservice or node. Solid lines show containment relationships . Dotted lines show dependency relationships and are also used to indicate a service hosted on a node.

To open a map view

From the console tree (View alternate method)

1. From the console tree, right-click an item in the Nodes or Services folder to open the shortcut menu.
2. Select Map to open a new window that displays a map of your selected service or node and any related subcomponents or subservices.
3. To further investigate the problem, right-click the icon for a node or service to open the shortcut menu.
4. Select Root Cause or Impacted to view another map that pinpoints the location where status changed or other services are impacted.
5. To locate a specific node or service in a crowded map view, use the Find option.
    - Within the map, right-click to open the shortcut menu. Select Find to open the Search dialog box.

- Type in the name of the node or service you want to locate in the Search for Caption box.
- Click OK . A red arrow appears beside the caption in the map and the map is redrawn to center the node or service you searched for.

Related Topics:

- Diagnosing Problems Using the Map View
- Show root cause
- Show impacted services

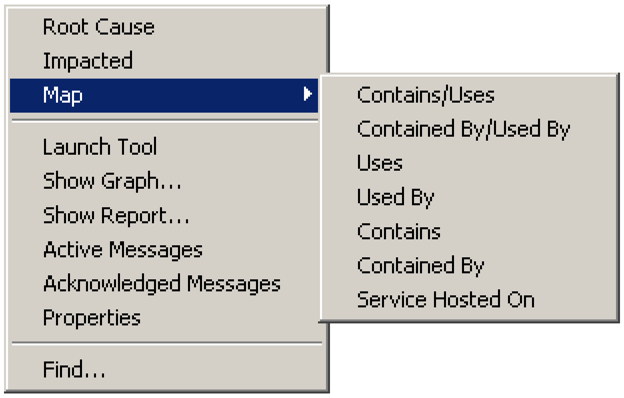
Keyboard commands in map view

Custom key commands provide the following functionality within a map view.

Key/Key Sequence	Function
Home	Selects and centers the root node of the map.
Left Arrow	Selects and centers the sibling map node to the left of the currently selected node.
Right Arrow	Selects and centers the sibling map node to the right of the currently selected node.
Down Arrow	Selects and centers the first (leftmost)child map node, relative to the currently selected node.
Up Arrow	Selects and centers the parent map node, relative to the currently selected node.
Shift+Up Arrow	Collapses the children of the currently selected node, if expanded.
Shift+Down Arrow	Expands the children of the currently selected node, if collapsed.
Ctrl+Up Arrow	Moves the currently selected node toward the top of the window.
Ctrl+Down Arrow	Moves the currently selected node toward the bottom of the window.
Ctrl+Left Arrow	Moves the currently selected node to the left.
Ctrl+Right Arrow	Moves the currently selected node to the right.

Map view menu

Use the map view menu commands to customize the map display to focus only on the items of special interest to you. You can view all the relationships for a selected item or can choose to show only those components that are contained by another service or node or are used by another service or node.

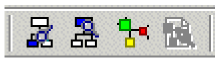


To display a map, start from within an open map. Right-click to open the shortcut menu and select the relationship you want to display from the menu.

Command	Description
Root Cause:	Displays a map showing the root cause of a problem with a service or node.
Impacted:	Display a map showing all services or nodes impacted by a problem with a selected node or service.
Map:	Opens a context menu from which you select the type of relationships you want to display in the map. You can view only services or nodes that are contained by other services or nodes or you can display only services or nodes that are used by other components.
Launch Tool:	Launch a tool on a selected service or node.
Show Graph:	Display a graph associated with the selected item in the map view.
Show Report:	Display a report associated with the selected item in the map view.
Active Messages:	Open an active messages browser. Only active messages for the selected service or node are displayed.
Acknowledged Messages:	Open the acknowledged messages browser. Only acknowledged messages for the selected service or node are displayed.
Properties:	View properties for a service or node selected in the map.
Find:	Open the Search dialog box.

Using the Map View toolbar

From within a map view, the map view toolbar is available to provide convenient access to frequently performed tasks that are also available from the shortcut menu. Click an icon in the interface for a tooltip explanation of its meaning.



Other toolbars allow you to customize your HP Operations Manager console and map views:

- HP Operations Manager for Windows Console Toolbar
- Configuration Toolbar
- Message Browser Toolbar

Show root cause

Many factors contribute to the severity status of a node or service. In a complex environment with many hierarchical levels, it can be difficult to determine whether the service or node itself or one or more subservices or nodes has caused a severity change.

To help you determine the source of a problem, HP Operations Manager provides root cause analysis to take you quickly to the service or node that is not performing. Root cause analysis starts at the level of your selected node or service, stops at the level where the cause of the problem lies, and draws a map that shows the source of the problem and the nodes or services affected.

You can view a root cause map from within the console tree, a map view, or the active and acknowledged messages browsers.

To show the root cause of a problem

From the console tree (View alternate method)

1. From the console tree, click the service, node, or node group that you want to investigate.
2. Right-click the service, node, or node group to open the shortcut menu.
3. Click View → Root Cause to display a top-down map view of the selected node or service in the current details pane. Click the + sign to expand the map to show its subservices or subsystems. The last item in the hierarchy represents the root cause of the problem.

Related Topics:

- Show impacted services
- Diagnosing Problems Using the Map View

Show impacted services or nodes

In a complex environment with many hierarchical levels, it can be difficult to determine how a severity change on one node or service may impact other nodes or services or their subcomponents. Because a subservice or subsystem can be contained in a node or service and also be used by another service or node, a severity change affects:

- The immediate parent service or node
- All other services that use that subservice

To help you resolve problems more efficiently, HP Operations Manager for Windows provides a map view of services or nodes that are impacted by a change in severity. You can see at a glance the services or nodes that are affected. The Show Impacted Services analysis tool starts at the level of the selected service or node and searches upward to display all other services or nodes affected by the status change.

You can view an impacted services map from within the console tree, a map view, or the active and acknowledged messages browsers. When opened from the console tree, the map displays in the details pane. When opened from the message browser or map view, the map displays in a new window.

To show the services impacted by a problem

1. From the console tree, map view, or active and acknowledged messages browsers, select the service or node you want to investigate.
2. Right-click the service or node to open the shortcut menu.
3. Select View → Impacted to show a map view of the selected service or node.
4. The map displays the service or node you selected and all services or nodes affected by the problem.

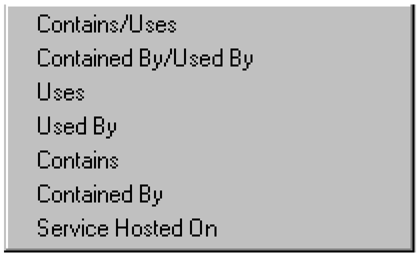
Related Topics:
- Show root cause
- Diagnosing Problems Using the Map View

Understanding uses and contains relationships

You can change your map view to show only certain relationships among the services and nodes in your environment, which can help you to resolve problems. Views fall into two main categories; each category offers further customization of your map view:

- Uses: A service that is used by another service is indicated on the map by a dotted line relationship. A service can be used by several other services, but is contained by only one service.
- Contains: A service or node that is contained by another service or node group is indicated on the map by a solid line relationship. A service can only be contained by one service; a node can be contained by more than one node group.

To display a submap showing a selected relationship, you must start from within an open map. Right-click to open the shortcut menu and select the relationship you want to display from the menu. Click any menu option for a description of the command. The submap appears in a new window.



Monitoring HP Operations Manager for Windows

The vpstat tool is a command line utility consisting of several tools for checking system status, file version, critical services, registry entries, DCOM servers, RPC servers, minimal system requirements and critical processes for HP Operations Manager for Windows.

Each tool within vpstat is launched with its own command. Help is available with the -? option. For example, to view help for the -3 command (List Services), type vpstat -3 -? at the Windows command prompt.

Items that are verified depend on the contents of the configuration file. All commands except the -1 command use this file. The requirements are read from it and compared with the actual state of the HPOM for Windows program. In case of a mismatch, noncompliant items are displayed to the screen. The default configuration file, `vpstat.conf` , is provided with the tool. The file `vpstat.conf` should only be modified by support personnel.

The file `vpstat.conf` is version dependent. The [CHKVER] section of the configuration file lists all the HPOM for Windows files that are part of the product. If the product is updated, then some of these files may also be updated and show new versions or different time stamps. In short, for every new version and/or patch and/or build, the [CHKVER] section of the configuration file `vpstat.conf` should be updated. If with the new version some new services and/or processes are introduced, other sections should also be updated.

Related Topics:

- vpstat command syntax

vpstat command syntax

To issue a vpstat tool command, type the number for the command preceded by the minus sign. You can optionally use a parameter list, which is specific for a command. Output from the vpstat tool can be redirected through a standard output stream redirection using the '> filename' option at the command line.

For example, to get System, Memory, Disk, Network, and Security Info for the local node written to the file MyNode.txt, type:

```
vpstat -1 >MyNode.txt
```

Common command syntax

You can pass more than one command with appropriate parameters (if any) to the vpstat tool. The common vpstat command line syntax is:

```
vpstat.exe [-f CfgFile] '-command' [-?|]
```

- -f NewCfg.conf : parameter specifies a configuration file other than the default. The letter 'f', preceded by a minus sign, followed by a space, followed by the file name (path), instructs vpstat to load this file as a new configuration file. Note: When used, specify this parameter as a first parameter at the command line. This parameter is optional.
- -command : command number preceded by the minus sign (number from 1 to 9). If no command is specified, all possible commands are displayed to the screen.
- -? : The question mark, preceded by the minus sign in combination with the command number (example: `vpstat -3 -?`), prints out a command line help for the command. If used without the command number, all possible commands of the tool are listed. This parameter is not valid for the '-1' command. (No help is available for this command.)
- <parameter_list> : parameter list is specific for each command. All commands, with the exception of the '-1' command, have the ability to take additional parameters. To list possible parameters for individual commands, use:

  ```
  vpstat -command -?
  ```

  Detailed descriptions of the <parameter_list> for each of the commands are shown in the examples.

Related Topics:
- Launch vpstat with no commands and no parameters
- Launch vpstat with the -3 and -? options
- Get System, Memory, Disk, Network, and Security Information
- Check File Version
- List Services

- List Registry Entries
- List DCOM servers
- List RPC servers
- Managed Node Requirements Check
- List Processes
- Get Management Server/Console Version
- List Patch History
- Run All Options
- vpstat Return Codes

vpstat return codes

All vpstat commands that verify a condition (like services running) return codes based on results of verification. Normally, vpstat return codes indicate whether checked items (defined with the vpstat command) have passed verification.

HPOM includes the following vpstat return codes:

- 0: there was no error and checking was successful.
- 1: there was no error and checking was not successful.
- negative value: if there was an error - execution of the tool failed.

Examples

- Checking single process if running:

  ```
  vpstat -8 opcctla.exe
  ```

- Returns 0 if opcctla.exe process is running; otherwise it returns 1.
- Checking all processes (configured in vpstat configuration file):

  ```
  vpstat -8
  ```

- Returns 0 if all process listed in [PROCESSLIST] section of the configuration file are running; otherwise it returns 1.

If some internal error occurs during vpstat tool execution, a negative return code is returned. The following table includes all possible values.

Error Code	Description
-100	Retrieving of system platform type failed.
-101	Retrieving of system memory status failed.
-102	Retrieving of system disk space status failed.
-103	Retrieving of system network settings failed.
-104	Retrieving of system security settings failed.
-110	Retrieving of information from configuration file (vpstat.conf) failed.
-111	System runs out of physical memory when creating report.
-112	Com internal error (cannot initialize COM library).
-113	Wrong parameter is specified.
-114	Specified folder was not found.
-115	Error in configuration file (vpstat.conf).
-116	Internal error occurs.

-117	Caught internal exception.
-120	Opening of service control manager database failed.
-121	Enumerating services under specific service control manager failed.
-122	Opening of specific service failed.
-123	Retrieving configuration parameters of specified service failed.
-124	Retrieving the name and status of each service that depends on the specified service failed.
-130	Establishing connection to predefined registry key failed.
-131	Opening of specified registry key failed.
-132	Retrieving of data for specific registry key failed.
-133	Converting specified string into original class identifier failed.
-140	Creating of an enquiry context for viewing the elements in an endpoint map failed.
-141	Server is not listening for remote procedure calls.
-150	Cannot connect to specified node.
-151	Operating system is not Windows.
-152	Whenever establishing connection to predefined registry key failed or property under specified registry key does not exist.
-160	Try to kill process failed, because process name or PID cannot be found.
-170	Retrieving of handle to current process failed.
-171	Enumerating of process modules failed.
-172	Retrieving "exe" module failed (vpstat.exe).
-180	Net-bios computer name cannot be retrieved.
-190	Specified file cannot be found on specified path.
-191	Patch configuration file is not specified.
-192	Inserting of lines fails while patching configuration file.
-193	Patching of REGLIST section in vpstat.com failed.
-194	Patching of RPCLIST section in vpstat.com failed.
-195	Patching of COMLIST section in vpstat.com failed.
-196	Specified file cannot be found on specified path.
-197	Patching of REQLIST section in vpstat.com failed.
-198	Patching of PROCLIST section in vpstat.com failed.
-199	Patching of CHKVER section in vpstat.com failed.
-200	Patching of DEPLIST section in vpstat.com failed.
-210	Saving changes to configuration file (vpstat.conf) failed.

List patch history

This option displays a list of the patches that have been installed and deinstalled.

Example:

```
command prompt : vpstat -p

result:

----------- PATCH DETAILS SECTION -----------

Patch Name  : OVOW_00228
Superseded by : OVOW_00244, 9-17-2007, 14:37:15

Patch Name  : OVOW_00229
Description : Mgmt Server/Console
Installed   : 6-19-2007, 10:00:16

Patch Name  : OVOW_00230
Superseded by : OVOW_00254, 9-17-2007, 15:01:04

Patch Name  : OVOW_00244
Description : Mgmt Server/Console
Installed   : 9-17-2007, 14:37:15

Patch Name  : OVOW_00246
Superseded by : OVOW_00254, 9-17-2007, 15:01:04

Patch Name  : OVOW_00247
Superseded by : OVOW_00254, 9-17-2007, 15:01:04

-------- END OF PATCH DETAILS SECTION --------
```

Run all options

This option runs all the commands and prints the output to a file called `'vpstatout.txt'` . First, it looks for an environment variable named `'TEMP'` . If the variable is found, it stores the file in that directory (usually `C:\TEMP`). Otherwise, it stores the file in the current working directory. Here is a list of the parameters used for each option.

-1, -2, -3 -d, -4 -l, -5 -l, -6 -l, -7 -d, -8 -d, -9, -p

```
Example:

    command prompt : vpstat -z

    result:

 c:\>vpstat -z
 generating file 'C:\TEMP\vpstatout.txt'...
 c:\>
```

Launch vpstat with no commands and no parameters

If you run vpstat with no commands and no parameters, vpstat runs all the commands and prints the output to the screen:

```
   command prompt : vpstat

   result:

============================================================
==================== System Information ====================
============================================================

System Summary for Computer VMBERT6
   Processor 0
    Identifier : x86 Family 6 Model 15 Stepping 8
    ~MHz       : 2327

...

============================================================
==================== Patches Information ===================
============================================================



----------- PATCH DETAILS SECTION -----------

    No Patches have been installed or deinstalled.

-------- END OF PATCH DETAILS SECTION --------
```

Launch vpstat with -3 and -? options

This example shows how to launch the vpstat tool with the -3 (List Services) command and with the -? option. The parameter list, accepted by the -3 command, is displayed to the screen.

command prompt : vpstat -3 -?

result:

```
List Services
  <param_list>:[ServiceNames][ -n NodeName][ -d][ -s State][ -l][ -a]
  or          :[ServiceNames][ -n NodeName][ -d] -r [action]
  Name1 Name2 ... : internal ServiceName of services that will be checked for
                    State. If none is specified, list of services is read from
                    [SERVICELIST] section of configuration file.
  -n NodeName|IP  : Lists/checks service on NodeName|IP, if none is specified,
                    listing is performed on the local system.
  -d              : Detailed output.
  -r [action]     : Restarts services with optional START or STOP only action.
  -s State        : Lists/checks only services with state equal to
                    State: ALL | ACTIVE | STOP. Default = ACTIVE
  -l              : List checked services.
  -a              : Lists all services. No checking is performed.
```

Get system, memory, disk, network, and security information

This command is the only command that does not use the configuration file. It does not perform any checking; it gathers several kinds of information about a local computer node, such as:

- Current computer System (CPU type, Speed)
- Current global memory status
- Current number of installed disks on the system and their space availability.
- Current network settings by capturing the output of ipconfig /all. (The utility ipconfig.exe must be available in the system path.)
- Current user security summary (User account, Group accounts, Account privileges)

Launch vpstat using the following syntax:

command prompt: vpstat -1

See the example for the results of the command.

Example

Launch vpstat to get system, memory, disk, network, and security information

Example: get system, memory, disk, network, and security information

command prompt: `vpstat -1`

result:

```
============================================================
==================== System Information ====================
============================================================

System Summary for Computer VMBERT6
  Processor 0
    Identifier : x86 Family 6 Model 15 Stepping 8
    ~MHz       : 2327
Global Memory Status
  MemoryLoad    :         77 percent.
  TotalPhys     : 1047998464 bytes
  AvailPhys     :  235884544 bytes
  TotalPageFile : 1226825728 bytes
  AvailPageFile :  620417024 bytes
  TotalVirtual  : 2147352576 bytes
  AvailVirtual  : 2126385152 bytes
Disk space summary
  Drive C:\
    Client Quota :   3806642176 bytes
    Total Free   :   3806642176 bytes
    Total Disk   :  10725732352 bytes
Network settings
  Output for "ipconfig /all"

Windows IP Configuration

   Host Name . . . . . . . . . . . . : VMBert6
   Primary Dns Suffix  . . . . . . . : deu.hp.com
   Node Type . . . . . . . . . . . . : Unknown
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : deu.hp.com
                                       hp.com

Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : VMware Accelerated AMD PCNet Adapter
   Physical Address. . . . . . . . . : 00-0C-29-2C-3F-1E
   DHCP Enabled. . . . . . . . . . . : No
   IP Address. . . . . . . . . . . . : 16.57.36.253
   Subnet Mask . . . . . . . . . . . : 255.255.240.0
   Default Gateway . . . . . . . . . : 16.57.32.1
   DNS Servers . . . . . . . . . . . : 16.14.64.51
                                       16.8.64.51

Security Summary
  User account   :
    VMBERT6\Administrator
  Group accounts :
    VMBERT6\None
    \Everyone
    VMBERT6\HP-OVE-ADMINS
    BUILTIN\Administrators
    BUILTIN\Users
    NT AUTHORITY\INTERACTIVE
    NT AUTHORITY\Authenticated Users
    NT AUTHORITY\This Organization
    NONE_MAPPED
    \LOCAL
    NT AUTHORITY\NTLM Authentication
  Account privileges
    SeChangeNotifyPrivilege
    SeSecurityPrivilege
    SeBackupPrivilege
    SeRestorePrivilege
    SeSystemtimePrivilege
    SeShutdownPrivilege
    SeRemoteShutdownPrivilege
    SeTakeOwnershipPrivilege
    SeDebugPrivilege
    SeSystemEnvironmentPrivilege
    SeSystemProfilePrivilege
    SeProfileSingleProcessPrivilege
    SeIncreaseBasePriorityPrivilege
    SeLoadDriverPrivilege
    SeCreatePagefilePrivilege
    SeIncreaseQuotaPrivilege
    SeUndockPrivilege
    SeManageVolumePrivilege
    SeImpersonatePrivilege
    SeCreateGlobalPrivilege
```

Check file version

The Check File Version command performs file version and time stamp (last write and link time) checking. It reads the necessary information from the [CHKVER] section of the configuration file and compares the listed file's version and time stamp with the actual information for the file installed on the system.

Time is expressed in Universal Coordinated Time (UTC). Use the following command:

command prompt : vpstat -2

result:

```
============================================================
==================== Check File Version ====================
============================================================

Configuration file: C:\Program Files\HP\data\conf\vpstat\vpstat.conf
  Ver: 0.8.00.201
  Date: Wed Sep  5 02:29:14 2007
  Remark: HP Operation Manager - Performance.

C:\Program Files\HP\bin\CclMsgs.dll
version             A.4.0.18.17             date    08/16/2007 10:24:17    pass

C:\Program Files\HP\bin\DnsDscr.exe
version             A.22.0.14.5             date    08/21/2007 07:41:40    pass


. . .

----------- CHECK VERSION SUMMARY SECTION -----------
None to report


----------- WARNINGS

FILE                         WARNING   ACTUAL VERSION/DATE  --  VPSTAT.CONF VER/DATE

OvOWNPReg.exe               NEWER VER  A.2.0.16.7           --  A.2.0.14.21
OvPmdPolicyEditorFrame.exe  NEWER VER  0.0.0.0              --  A.5.0.13.94
ovconfreporter.exe          NEWER VER  0.0.0.0              --  A.22.0.26.1
ovpmutil.exe                NEWER VER  0.0.0.0              --  A.22.0.26.1
```

```
PATH:
C:\Program Files\HP\bin
C:\WINDOWS\system32
C:\WINDOWS
C:\WINDOWS\System32\Wbem
c:\Program Files\Microsoft SQL Server\90\Tools\binn\
C:\Program Files\HP\lib
c:\Program Files\HP\bin\OpC




******* FILE VERSION CHECK STATUS:    WARNING


----------- END OF CHECK VERSION SUMMARY -----------
```

Example: check file version and display help

command prompt: `vpstat -2 -?`

result:

```
Check File Version:
  This option lists the status of a version and date check for the files
  listed in the [CHKVER] section of the vpstat.conf file.  Any failed files
  are listed in the details section along with the system's path.
  **There are no suboptions for this command.
```

List services

The List Services command lists the Windows services state on the target managed node and allows starting and stopping of particular services. There are several optional parameters for using this command. The complete parameter list is:

```
vpstat -3 [ServiceNames][ -n NodeName][ -d][ -s State][ -l][ -a]
```

or

```
vpstat -3 [ServiceNames][ -n NodeName][ -d] -r [action]
```

As defined above, there are two operational modes for this command. The first lists and checks for proper state The second mode uses the -r parameter to stop or start particular services.

By default (without any optional parameter), this command checks for critical HP Operations Manager for Windows services that need to be in active state on the node where the management server is installed. The services that must be up and running are configured in the `vpstat.conf` file.

With the optional ServiceName as the first parameter on the command line, you can specify the services that are checked directly. If a critical service that is not registered or not in active state is found, it is reported as an error.

Examples

- Example: list services using the -n option (check remote node)
- Example: list services using the -a option (list all services)
- Example: list services using the -r option (stop/start services)

Example: list services using the -n option (check remote node)

Use the -n option to check the management server running on a remote node. You can specify a name or node IP to identify the target node. In the following example the HP Operations Manager for Windows management server services are checked on a remote node named SVR0123. The generated output shows that no critical services were installed and thus are not running.

command prompt: `vpstat -3 -n SVR0123`

result:

```
Configuration file: C:\Program Files\HP\data\conf\vpstat\vpstat.conf
 Ver: 0.8.00.201
 Date: Wed Sep  5 02:29:14 2007
 Remark: HP Operation Manager - Performance.

 Services on     : VMBERT6

 (NT Services) :

 OK: All services : SERVICE_ACTIVE
```

Only the services with errors are displayed. You can override this with the -l option, which forces all the services that are examined to display.

The -d option displays a service's details, such as current state, logon account, startup type, dependencies, and other registration information.

Example: list services using the -a option (list all services)

Use the -a option to list all registered services, not only services which are critical to HP Operations Manager for Windows.

In conjunction with this option, the -s State option can be used to list only the services that are in a particular state. By default, those are ACTIVE services (running and start pending). Use the following command to list all active services:

command prompt: `vpstat -3 -a`

result:

```
============================================================
==================== Services Information ==================
============================================================

List All Services...
Services on     : LOCAL MACHINE

(NT Services) :
  SERVICE_RUNNING  : AeLookupSvc ( Application Experience Lookup Service )
  SERVICE_RUNNING  : Alerter ( Alerter )
  SERVICE_RUNNING  : appmgr ( Remote Server Manager )
  SERVICE_RUNNING  : AudioSrv ( Windows Audio )
  SERVICE_RUNNING  : Browser ( Computer Browser )
  SERVICE_RUNNING  : ccEvtMgr ( Symantec Event Manager )
  SERVICE_RUNNING  : ccSetMgr ( Symantec Settings Manager )
  SERVICE_RUNNING  : COMSysApp ( COM+ System Application )
  SERVICE_RUNNING  : CryptSvc ( Cryptographic Services )
  SERVICE_RUNNING  : DcomLaunch ( DCOM Server Process Launcher )
  SERVICE_RUNNING  : DefWatch ( Symantec AntiVirus Definition Watcher )
  SERVICE_RUNNING  : Dhcp ( DHCP Client )
  SERVICE_RUNNING  : dmserver ( Logical Disk Manager )
  SERVICE_RUNNING  : Dnscache ( DNS Client )
  SERVICE_RUNNING  : elementmgr ( Web Element Manager )
  SERVICE_RUNNING  : ERSvc ( Error Reporting Service )
  SERVICE_RUNNING  : Eventlog ( Event Log )
  SERVICE_RUNNING  : EventSystem ( COM+ Event System )
  SERVICE_RUNNING  : helpsvc ( Help and Support )
  SERVICE_RUNNING  : HTTPFilter ( HTTP SSL )
  SERVICE_RUNNING  : IISADMIN ( IIS Admin Service )
  SERVICE_RUNNING  : lanmanserver ( Server )
  SERVICE_RUNNING  : lanmanworkstation ( Workstation )
  SERVICE_RUNNING  : LmHosts ( TCP/IP NetBIOS Helper )
  SERVICE_RUNNING  : MSDTC ( Distributed Transaction Coordinator )
  SERVICE_RUNNING  : MSFtpsvc ( FTP Publishing Service )
  SERVICE_RUNNING  : Netman ( Network Connections )
  SERVICE_RUNNING  : Nla ( Network Location Awareness (NLA) )
  SERVICE_RUNNING  : PlugPlay ( Plug and Play )
  SERVICE_RUNNING  : PolicyAgent ( IPSEC Services )
  SERVICE_RUNNING  : ProtectedStorage ( Protected Storage )
  SERVICE_RUNNING  : RemoteRegistry ( Remote Registry )
  SERVICE_RUNNING  : RpcSs ( Remote Procedure Call (RPC) )
  SERVICE_RUNNING  : SamSs ( Security Accounts Manager )
  SERVICE_RUNNING  : SavRoam ( SavRoam )
  SERVICE_RUNNING  : Schedule ( Task Scheduler )
  SERVICE_RUNNING  : seclogon ( Secondary Logon )
  SERVICE_RUNNING  : SENS ( System Event Notification )
  SERVICE_RUNNING  : ShellHWDetection ( Shell Hardware Detection )
  SERVICE_RUNNING  : Spooler ( Print Spooler )
  SERVICE_RUNNING  : srvcsurg ( Remote Administration Service )
  SERVICE_RUNNING  : Symantec AntiVirus ( Symantec AntiVirus )
  SERVICE_RUNNING  : TermService ( Terminal Services )
  SERVICE_RUNNING  : TrkWks ( Distributed Link Tracking Client )
  SERVICE_RUNNING  : VMTools ( VMware Tools Service )
  SERVICE_RUNNING  : W32Time ( Windows Time )
  SERVICE_RUNNING  : W3SVC ( World Wide Web Publishing Service )
  SERVICE_RUNNING  : winmgmt ( Windows Management Instrumentation )
  SERVICE_RUNNING  : wuauserv ( Automatic Updates )
  SERVICE_RUNNING  : WZCSVC ( Wireless Configuration )
  SERVICE_RUNNING  : HPOvTrcSvc ( HP Software Shared Trace Service )
  SERVICE_RUNNING  : OvCtrl ( HP OpenView Ctrl Service )
  SERVICE_RUNNING  : SQLWriter ( SQL Server VSS Writer )
  SERVICE_RUNNING  : MSSQL$OVOPS ( SQL Server (OVOPS) )
  SERVICE_RUNNING  : OvowWmiPlatProv ( OvowWmiPlatProv )
  SERVICE_RUNNING  : OvPmad ( OvPmad )
  SERVICE_RUNNING  : OvSecurityServer ( OvSecurityServer )
  SERVICE_RUNNING  : OvMsmAccessManager ( OvMsmAccessManager )
  SERVICE_RUNNING  : OvAutoDiscovery Server ( OvAutoDiscovery Server )
  SERVICE_RUNNING  : OvEpMessageActionServer ( OvEpMessageActionServer )
  SERVICE_RUNNING  : OvEpStatusEngine ( OvEpStatusEngine )
  SERVICE_RUNNING  : OvOWReqCheckSrv ( OvOWReqCheckSrv )
  SERVICE_RUNNING  : OvDnsDscr ( OvDnsDscr )
  SERVICE_RUNNING  : OvServerMonitor ( OvServerMonitor )
  SERVICE_RUNNING  : OVServiceLogger ( OVServiceLogger )
```

Example: list services using the -r option (stop/start services)

Use the -r option to restart (refresh) HP Operations Manager for Windows services. By default, this command option stops and restarts the services.

To stop the services add the STOP action qualifier. To start services, use the START action qualifier. When stopping a service, the vpstat utility considers the service that is not responding to be blocked and terminates its process after 20 seconds. For example to start the OvDnsDiscovery service, use the following command:

command prompt: `vpstat -3 ovdnsdscr -r START`

result:

```
Services on     : LOCAL MACHINE
Starting services...
  ovdnsdscr
Waiting for services to start...
  ovdnsdscr
   Wait 1000 [ms] of 20000 [ms], CheckPoint: 0
```

List registry entries

Use the -4 command to list and check all registry entries that are needed by HP Operations Manager for Windows. The necessary keys are configured in the `[REGLIST]` section of the `vpstat.conf` configuration file.

By default, only those that are not registered are displayed as output. You can override this with the -l option, which forces all the keys that are examined to display. The optional -r parameter allows you to list all the subkeys and their contents for the examined key.

List DCOM servers

The command for listing DCOM servers is -5. It checks the DCOM classes by their class IDs on the node where the management server is installed. The necessary classes are taken from the `[COMLIST]` section of the configuration file.

By default, only those that are not registered are displayed as output. You can override this with the -l option, which forces all the COM servers that are examined to display. The optional -d parameter allows you to view details for each listed COM server. This includes full registry sub-tree output for CLSID, TypeLib, and AppID.

Example

List DCOM Servers using the `-1` option

Example: list DCOM servers using -l option

By default, only those registry subentries that are not registered are displayed as output. You can override this with the -l option, which forces all the COM servers that are examined to display.

command prompt: `vpstat -5 -l`

result:

```
Configuration file: C:\Program Files\HP\data\conf\vpstat\vpstat.conf
  Ver: 0.8.00.201
  Date: Wed Sep  5 02:29:14 2007
  Remark: HP Operation Manager - Performance.


OK:ALL CRITICAL COM/DCOM SERVERS ARE REGISTERED !
```

List RPC servers

This command is invoked with the -6 option. It checks for all critical HP Operations Manager for Windows RPC servers that should be registered.

By default, only those that are not registered are displayed as output. You can override this with the `-l` option, which forces all the RPC servers that are examined to display.

The -d option allows you to view more detailed RPC registration parameters, such as a server's uuid and the bindings. By default, this command checks only the RPC servers that are in the [RPCLIST] section of the configuration file.

The -a option allows you to list all the registered RPC servers found on the system.

Example

List RPC Servers using the -d option

Example: list RPC servers using the -d option

In this example, two registered RPC servers with registration details and one missing server are listed at the end of the output.

command prompt: `vpstat -6 -d`

result:

```
Configuration file: C:\Program Files\HP\data\conf\vpstat\vpstat.conf
  Ver: 0.8.00.201
  Date: Wed Sep  5 02:29:14 2007
  Remark: HP Operation Manager - Performance.


OK:ALL CRITICAL RPC SERVERS ARE REGISTERED !
```

Managed node requirements check

This command performs a requirements check on a local or remote node. For this operation, vpstat reads the `[REQLIST]` section of the configuration file.

Example

- Managed node requirements check using the -? option
- Managed node requirements check using the -d, -n options

Example: managed node requirements check using the -? option

command prompt: `vpstat -7 -?`

result:

```
Managed node requirements check using OvOWReqCheckSrv tool
  <param_list>: [-n NodeName] [-s][ -d]
  -n Name|IP  : Checks specified node.If not specified, the local node is checked.
  -s          : Displays status ( OK | NOK ) per requirements.
  -d          : Displays status ( OK | NOK ) per requirements detailed output.
```

Use the -? option of this command to print out help on the command.

Option	Description
-n Name | IP	With the -n option, followed by a space and by a computer name or an IP address, vpstat is instructed to connect to a specified machine and to perform the requirements checklist for that machine.
-s	With the -s option vpstat is instructed to display to the screen the result of the MC line requirements check in a short form (OK or NOK).
-d	With the -d option vpstat is instructed to display the result of the MC line requirements check in a detailed form. Besides the short print out (OK or NOK), the demanded and the current values of the requirement are also displayed.

Example: managed node requirements check with -d, -n options

This example shows the requirement check of the remote node sangre, through all MC lines with detailed output.

command prompt: `vpstat -7 -d -n sangre`

result:

```
Prerequisites checked for: Windows2003

Requirements:
-------------
[ PASS  ] Default system share (e.g. C$, D$) accessible
[ PASS  ] 40MB hard disk space required for installation
[ PASS  ] Default launch permissions
[ PASS  ] Event Log Service
[ PASS  ] Remote Procedure Call (RPC) Service
[ PASS  ] Plug & Play Service
[ PASS  ] Security Accounts Manager Service
[ PASS  ] Net Logon Service
[ PASS  ] Remote Registry Service
[ PASS  ] Server Service
[ PASS  ] Workstation Service

Recommendations:
----------------
[ PASS  ] Windows Management Instrumentation Service
[ PASS  ] SNMP Trap Service
[ PASS  ] SNMP Service
[ PASS  ] NT LM Security Support Provider Service



STATUS:

All checked prerequisites are OK.
```

If a certain requirement in the MC line is not specified, that requirement is not verified and has automatically passed. If all requirements defined in the MC line match the actual state, the checked node meets minimal requirements.

List processes

The processes required by HP Operations Manager for Windows to be running are checked with the -8 command. The output shows missing processes that should be running on the checked system. Also other processes can be displayed with this command. The full syntax is:

```
vpstat -8 [ProcessName...][ -d][ -l][ -m][ -a][ -k PID|Name]
```

When invoked without any optional parameter, the necessary process names are taken from the `[PROCESSLIST]` section of the configuration file.

Example

List Processes using the -l option

Example: list processes using the -l option

By default, only the missing processes are displayed. To also show the processes that are found on the system, use the `-l` switch. For example, the following command displays the HP Operations Manager for Windows processes that are running:

command prompt: `vpstat -8 -l`

result:

```
Configuration file: C:\Program Files\HP\data\conf\vpstat\vpstat.conf
  Ver: 0.8.00.201
  Date: Wed Sep  5 02:29:14 2007
  Remark: HP Operation Manager - Performance.

ID:1456  OVTRCSVC.EXE          06.00.035       C:\Program Files\HP\bin
ID:1536  SQLSERVR.EXE          2005.090.3042.0 c:\Program Files\HP\MSSQL.1\MSSQL\Binn
ID:1612  OVCD.EXE              06.00.030       C:\Program Files\HP\bin
ID:1748  DNSDSCR.EXE           A.22.0.14.5     C:\Program Files\HP\bin
ID:1800  SERVICELOGGER.EXE     2, 6, 0, 0      c:\Program Files\HP\bin
ID:2244  OVMSMACCESSMANAGER.E  A.5.0.22.5      C:\Program Files\HP\bin
ID:2452  OVOWREQCHECKSRV.EXE   A.22.0.22.6     C:\Program Files\HP\bin
ID:2532  OVOWWMIPLATPROV.EXE   A.1.1.12.5      C:\Program Files\HP\bin
ID:2676  OVPMAD.EXE            A.5.0.58.10     C:\Program Files\HP\bin
ID:2952  OVSECURITYSERVER.EXE  A.3.0.29.2      C:\Program Files\HP\bin
ID:3012  OVAUTODISCOVERYSERVE  A.4.5.10.21     C:\Program Files\HP\bin
ID:3060  OVEPMSGACTSRV.EXE     A.5.0.45.26     C:\Program Files\HP\bin
ID:3180  OVEPSTATUSENGINE.EXE  A.5.0.26.5      C:\Program Files\HP\bin

OK:ALL CRITICAL PROCESSES ARE RUNNING
```

The processes you want to check or list can be specified directly as a first ProcessName parameter on the command line or by using the -a option, which lists all the processes on the system.

You can get additional information about the process by using the optional -d and `-m` parameters. The -d option displays process details such as current and peak working set size, page faults, image size, base address, and entry point.

The `-m` option lists all modules (for example, DLLs) that are loaded into the process address space with the corresponding version and image file name.

The -k option allows you to terminate a particular process (forced kill). This must be specified either by its PID or a process name.

Get management server/console version

To get the HP Operations Manager for Windows management server and console version, use the -9 command. The command can retrieve the version information of the installed product from the local node as well as from the remote node. The full syntax is:

```
vpstat -9 [-n NodeName]
```

When invoked without any optional parameter, the version information of the installed HPOM for Windows version is retrieved from the local node.

Examples

- Example: server and console installation on local node
- Example: console only installation on a remote node

Example: get server/console version on local node

On a local machine named VMBert6, a management server and console installation was performed. Issuing the -9 command, both versions (server and console version) would be displayed.

command prompt: vpstat -9

result:

```
Connecting ...
Product version on node: \\VMBert6
  Management server  : A.08.00
  Management console : A.08.00
```

Example: get console version on remote node

On a remote node named SVR5678, a remote console installation was performed. Issuing the -9 command and specifying the node name, only the console version displays. For the management server version, the message 'Not Installed' displays.

command prompt: `vpstat -9 -n SVR5678`

result:

```
Connecting ...
Product version on node: \\SVR5678
  Management server  : Not Installed
  Management console : A.08.00
```

HPOM Self Management

HPOM self management monitors the availability and performance HPOM itself.

HPOM self management can automatically:

- discover HPOM's own agents and servers and model them in a service hierarchy .
- update the discovered service hierarchy.
- deploy policies that monitor the discovered agents and servers.

Related Topics:

- Synchronizing self management
- Self-management policies

Synchronizing self management

HPOM self management discovers management servers and agents on managed nodes and automatically deploys policies that monitor their availability and performance. Self management automatically constructs a service hierarchy that models your servers and agents. This first takes place during installation, and you can easily update the service hierarchy by synchronizing it.

During synchronization, self management discovers any new agents or servers and adds them to the service hierarchy. It automatically deploys policies to monitor the new services. If any agents or servers are no longer available, they are removed from the service hierarchy, and the policies are automatically removed.

You can:

- Schedule synchronization
- Start synchronization manually
- Disable self-management functions

Schedule synchronization

To schedule regular synchronization for HPOM self management:

1. Click the Policy management icon in the console tree.
2. Click the Agent policies grouped by type icon.
3. Click the Scheduled Task icon.
4. Right-click the VP_SM-Server_SynchAgentServices policy.
5. Select All Tasks → Edit .
6. Select the Schedule tab and specify how often you want to synchronize your service hierarchy.
7. Click the Save and Close button.
8. Deploy the new version of the policy to the management server .

NOTE:
By default, self-management synchronization is scheduled to start daily at 3:00am, and the policy is already deployed to the management server.

Related Topics:

- Scheduled task overview
- Deploy a policy or policy group

Start synchronization manually

To start synchronization for self management manually:

1. Click the Tools icon in the console tree.
2. Open the HPOM Self Management tool group.
3. Right-click the Synchronize Agent Services tool.
4. Select All Tasks → Launch tool .
5. Type in a valid User Name and Password for the management server.
6. Click the Launch button. The Tool Status window appears and shows details of the synchronization operation. This may take some time.

Related Topics:

- Synchronizing self management
- Schedule synchronization

Disable self-management functions

HPOM self management discovers management servers and agents on managed nodes and creates a service hierarchy that models them. The management server then automatically deploys policies that monitor the availability and performance of these services.

Although it is not recommended, you can disable the creation of the services, and the automatic deployment of the policies.

CAUTION:
The self-management functions minimize the administrative effort required to monitor the management server and agents deployed on both the server and managed nodes. Before disabling either, or both, of these features make sure you understand how to administer the self-management manually.

To disable self-management service creation

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Self Management . A list of values appears.
3. Set the value of Create Services to False .

To disable automatic deployment of self-management policies

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click Self Management . A list of values appears.
3. Set the value of Auto Deploy Policies to False .

NOTE:
These procedures apply only to self-management service creation and policy deployment. The procedures do not disable automatic service discovery or policy deployment supplied by Smart Plug-ins (SPIs).

Related Topics:

- Server self-management policies
- Agent self-management policies
- Disable policy autodeployment

Self-management policies

HPOM provides self-management policies that monitor the availability and performance of the management server and agents on managed nodes .

By default, these policies are automatically deployed as follows:

- Agent policies

  The policies in the group HPOM Self Management → Agent are auto-deployed to a node if the Operations Agent package is installed.

- Server policies

  The policies in the group HPOM Self Management → Server are auto-deployed are auto-deployed to HPOM management servers.

- Database server policies

  The policies in the group HPOM Self Management → Database Server are auto-deployed to the HPOM management servers if a local database is used. If a remote database is used and the database system is managed by HPOM, you must deploy these policies manually.

NOTE:
If necessary, you can disable deployment or removal of self-management policies .

Related Topics:

- Server self-management policies
- Agent self-management policies
- Database server self-management policies

Server self-management policies

HPOM includes a number of self-management policies that monitor the management server . These policies monitor processes that need to be running on the management server, and make sure you that know about any problems.

- VP_SM_OVOWServices

  The service/process monitors check approximately every five minutes whether the necessary services and process are still running on the management server. If not, a message is sent to the active message browser. The operator can restart the service using an operator-initiated command. When the service is running again, the message is acknowledged.

  The following services are monitored by the VP_SM_OVOWServices policy:

  - OvAutoDiscovery Server
  - OvDnsDscr
  - OvEpMessageActionServer
  - OvEpStatusEngine
  - OvMsmAccessManager
  - OvPmad
  - OvSecurityServer
  - OvOWReqCheckSrv
  - Windows Management Instrumentation

- OvSvcDiscServerLog

  Checks the log file of the service discovery server for errors.

- VP_SM-Server_EventLogEntries

  Checks the Application Event Log for errors.

- VP_SM-Server_SyncAgentServices

  Synchronizes the agent services according to the specified schedule.

- VP_SM-WMI-Restart

  Tries to stop and restart OVMsmAccessManager and OVEpStatusEngine if WMI terminates unexpectedly.

Agent self-management policies

HPOM includes a self-management policy that monitors the service discovery agent. The service discovery agent must be running on every node that you want to manage. This policy makes sure you know about any problems with the service discovery agent and helps you to solve them.

- OvSvcDiscErrorLog

  The OvSvcDiscErrorLog policy checks approximately every five minutes for any entries in the error log file of the service discovery agent. If entries match the search pattern, a message is sent to the active message browser.

Database server self-management policies

HPOM includes self-management policies that monitor and back up the database.

- VP_SM_CHK_OVODB

  If a local SQL Server Express database is used, this policy checks the size of the database every five minutes. The limit of the database size is 4GB.

  - If the database size is greater than or equal to 90% of the 4GB limit, the policy sends a critical message to the active message browser. If the database size subsequently falls to below 90% of this limit, the policy sends a normal message to the acknowledged message browser.
  - If the database size is greater than or equal to 80% of the 4GB limit, the policy sends a warning message to the active message browser. If the database size subsequently falls to below 80% of this limit, the policy sends a normal message to the acknowledged message browser.

- VP_SM_DB_Backup

  This policy performs a daily backup of the openview database.

Related Topics:

- Back up the HPOM database

Message queue file self-monitoring

When the HPOM management server receives messages, it stores them temporarily in a message queue file before processing them. The maximum size of a this file is 2 GB. The operating system components that the management server uses do not support larger files. If the file size grows beyond 2 GB, the file becomes corrupt. The management server loses messages from the file, and cannot process new messages.

Under normal conditions this problem does not occur. However, the following circumstances could lead the message queue file to grow uncontrollably:

- Message storms (if message storm detection is not enabled).
- Many new messages arriving, but slow processing of the message queue due to DNS name resolution problems.
- Processing of the message queue is suspended due to WMI problems. In this case new messages are still accepted.

To prevent the message queue file becoming corrupt, the management server monitors the size of this file. By default, if the file size approaches 2 GB, the management server rejects new messages until the file size returns to normal. Agents start to buffer rejected messages locally, and try to resend them later. Therefore, you do not lose the messages that the management server rejects.

The management server sends a high-priority message to connected consoles when the file approaches 2 GB, and also when the file drops back to a safe size. When it sends these high-priority messages, the management server bypasses the agent and the message queue. (You cannot create high-priority messages using policies.)

You can change this default behavior in the following ways:

- Define whether the management server continues to accept messages after the message queue file size reaches the maximum.
- Define different message queue file size thresholds.
- Configure the severity and message key of the high-priority messages.
- Configure automatic and operator-initiated commands for the high-priority messages.

Configuration values for message queue file self-monitoring

You can configure message queue file monitoring by changing values in the Message Queue File Monitoring namespace using the Server Configuration dialog box. The following table describes the values that you can change.

Value	Default	Description

Message queue monitoring mode	2	This value configures the action that the management server takes when the message queue file size reaches the configured maximum. Set one of the following values: ▪ 0 - Ignore this problem and let the queue file keep growing. If the file size grows beyond 2 GB, the file becomes corrupt. ▪ 1 - Send a high-priority message to connected consoles, but continue accepting messages. Reject messages only if the file size reaches 2 GB. ▪ 2 - Send a high-priority message to connected consoles, and reject new messages, so that agents buffer messages locally. Accept messages again only if the message queue file size drops below the configured Message queue rearm file size .
Message queue maximum file size	2147473407	This value configures the maximum file size to which the message queue file is allowed to grow before the monitor takes the action defined in Message queue monitoring mode . Set a value in bytes between 20000000 (about 19 MB) and 2147473407 (2 GB minus 10 KB). If you configure a value that is outside this range, the management server uses the default value.
Message queue rearm file size	2146435071	This value configures the file size to which the message queue file size must return after reaching the configured maximum. When the message queue file returns to a size below this threshold, the management server sends a high priority message to connected consoles. If the value Message queue monitoring mode is 2, the management server also begins accepting messages again. Set a value in bytes that is smaller than the Message queue maximum file size . The value must also be between 256000 (about 250 KB) and 2146435071 (2 GB minus 1 MB). If you configure a value that is outside this range, the management server uses a value that is half the defined message queue maximum file size.
Severity of queue warning messages	32	This value configures the severity of the high-priority message that the management server sends when the message queue file size reaches the configured maximum. Set one of the following numeric values: ▪ 2 - Normal ▪ 4 - Warning

		■ 8 - Minor ■ 16 - Major ■ 32 - Critical NOTE: The message that the management server sends when the message queue file size drops below the configured Message queue rearm file size has normal severity. You cannot configure the severity of this message.
Message key prefix for queue warning messages	Empty string (no message key defined)	This value configures a prefix for the message key of the high-priority messages about the message queue file size. The management server constructs the message key in the format: `<UserDefinedPrefix>:QueueFileSize:MsgQueue` . If you configure this value, the message that the management server sends when the message queue file size returns to below Message queue rearm file size automatically acknowledges the previous high-priority message. If you leave this value empty, the message key remains blank, which prevents automatic acknowledgement of these messages.
Automatic action for queue warning messages	Empty string (no automatic action defined)	This value specifies the name of a command file (.cmd or .bat) or script file (.vbs) to add as an automatic command to high-priority messages about the message queue file size. (This enables you, for example, to configure an automatic command that pages an operator or sends an urgent email.) The management server bypasses the agent and runs the automatic command immediately. The management server adds the results of the automatic command as an annotation to the high-priority message. When the management server runs the command file or script, it automatically adds the following parameters, which your command file or script can evaluate: <ReasonOfCall> Specifies the reason why the automatic command was called. This can be one of the following values: ■ 0 - The message queue file size is back to normal. ■ 1 - The message queue file size has reached the configured maximum. ■ 2 - The message queue file size has reached the system limit of 2 GB.

		<MessageId> ID of the high-priority message. <CurrFileSize> Current size of the message queue file in bytes (before adding the new message). <ReqFileSize> Required size of the message queue file in bytes (if the new message were added). <QueueFileName> Fully qualified name of the message queue file. You can use the following command file as a template: `%OvInstallDir%\bin\OvMsgQueueMonAutoActionTmpl.cmd` .
Operator-initiated action for queue warning messages	empty string (no operator-initiated action defined)	This value specifies the name of a command file (.cmd or .bat) or script file (.vbs) to add as an operator-initiated command to high-priority messages about the message queue file size. When the operator starts this command, the management server bypasses the agent and runs the command immediately. The management server automatically appends parameters for your command file or script to evaluate. These are the same parameters that the management server appends to automatic commands (see above). You can use the following command file as a template: `%OvInstallDir%\bin\OvMsgQueueMonOperatorActionTmpl.cmd`

Testing message queue file self-monitoring

Before you use customized message queue file self-monitoring in a production environment, test whether it works as you expect. Consider the following questions:

- Do you receive high-priority messages when the message queue file size reaches the configured maximum and then when the file returns to a size below the configured rearm file size? Does the first message have the configured priority?
- Do the high-priority messages have the configured automatic and operator-initiated commands? Do the commands run correctly and is the output correct?
- Does automatic acknowledgement work correctly?
- Does the management server reject incoming messages when the message queue file size reaches the configured maximum? Does the management server accept messages again when the file returns to a size below the configured rearm file size?

Normally, it is difficult to create tests that cause the message queue file size to reach the configured maximum. The permitted ranges for configuration values suit production environments, but for testing it is desirable to configure smaller values.

In a test environment, you can specify values that are smaller than normally permitted for Message queue maximum file size and Message queue rearm file size by defining negative values. The management server interprets these negative values as test values and automatically converts to them to positive values without checking whether the values are permitted.

CAUTION:
Perform tests in a test environment. Do not use negative values in production environments.

The following example explains a test case that sends a high-priority message if the message queue file size reaches 20000 bytes. In this example, the management server continues to accept new messages. When the message queue file returns to size below 10000 bytes, the management server sends a second high-priority message.

1. Set the following values appropriately:
   - Severity of queue warning messages
   - Message key prefix for queue warning messages
   - Automatic action for queue warning messages
   - Operator-initiated action for queue warning messages
2. Set Message queue monitoring mode to 1 .
3. Set Message queue maximum file size to -20000
4. Set Message queue rearm file size to -10000 .
5. Use `opcmsg` to create test messages and use Windows Explorer to check the size of the file `%OvShareDir%\tmp\queues\MsgQueue` . When the file size reaches 20000 bytes (which happens quickly), check that a high-priority message appears in the console for the management server node.
6. Check whether the automatic command completes successfully and that the results appear as an annotation to the high-priority message.
7. Start the operator-initiated command, and check the results.
8. Continue to send new messages until the message queue file size reaches approximately 265000 bytes. The file size automatically reduces. Check that a second high-priority message appears in the console.
9. Check whether the automatic and operator-initiated commands complete successfully for the second high-priority message.
10. Check that the second high-priority message automatically acknowledges the first high-priority message.

Related Topics:

- Message storm detection and suppression
- Optimize HPOM node name resolution
- Generic server configuration
- opcmsg

Agent health checks

NOTE:
The agent health checks described here check the health of the HPOM agent and of all its subagents.

Self management monitors the health of the agents on each managed node using the following mechanisms:

- The control agent checks the health of its subagents and reports aborting agents by sending a message to the message browser .
- After a configurable interval (300 seconds by default), the management server checks the agent health. The management server attempts to contact the agent with either an ICMP ping or a call to the control agent, or both.

The management server reports the health of the agents either to the active message browser, or to the Windows event log. An event log policy that is deployed to the management server evaluates events in the event log and forwards them to the message browser. Message correlation acknowledges *Node down* messages automatically when a *Node up* message arrives.

Some sample message generated by the server are:

Node down -messages:

- *Could not contact RPC server of agent on radish. RPC server of agent not registered. Agent is probably not running.*
- *Node rhubarb is maybe down. Even to contact it with ping-packages failed.*
- *Message agent on node radish is not running*

Node up -messages:

- *Message agent on node radish is now running*
- *Control agent on node radish is now running.*

NOTE:
The management server does not check the health of nodes that have an empty package inventory. Nodes can have an empty package inventory if, for example, you install the agent manually, or if you upload the node configuration from another management server. If you want the management server to start checking the health of these nodes, synchronize the package inventory.

To configure advanced agent health check options

1. In the console tree, right-click Operations Manager , and then click Configure → Server... . The Server Configuration dialog opens.
2. Click Namespaces , and then click Agent Health Check . A list of values appears.
3. Change any of the values in the following table:

Values	Value type	Unit	Default value	Description
Health check ping protocol	List	■ DISABLED ■ AGENTONLY ■ ICMPONLY ■ ENABLED	ENABLED	This value configures the default ping protocol. You can change the default for each node in the Node Properties dialog. ■ DISABLED means that the management server performs no agent health check at all. ■ AGENTONLY means that the server does not actively contact the node with ICMP pings, but still contacts the agent on the node. This is useful for nodes behind a firewall. ■ ICMPONLY means that the server does not contact the agent, but only uses ICMP pings. This is useful for managed nodes like SNMP devices that do not have an agent installed. ■ ENABLED means that all aspects of agent health check are used.
Enable health check	Boolean	■ True ■ False	True	Enables or disables all aspects of the health check.
Time interval to check agent health	Integer	Number of seconds	300	The default interval at which the management server checks the health of each agent. You can change the default for each node in the Node Properties dialog.
Maximum number of parallel checks	Integer	Number of threads	100	The maximum number of parallel threads that are used to do the active check (server pings the node). After you have changed this value, restart the OvEpMessageActionServer service for the change to take effect.

Health check retries	Integer	0 to 3 retries	0	This value configures the number of health check ping retries to do immediately if an agent could not be reached. The node is considered down when all retries have been unsuccessful. Increase this value if you have an unreliable network infrastructure.
Target for agent health problem messages	List	▪ SERVER ▪ EVENTLOG ▪ SERVER_EVENTLOG	SERVER	The target for messages that indicate problems with agent health checking. ▪ SERVER means that these messages are directly written to the active message browser on the management server, without passing any policy-based message filter. ▪ EVENTLOG means that these messages are written to the application event log so that they can be picked up by a Windows Event Log policy. The VP_SM-Server_EventLogEntries policy already contains two rules for these health messages named "forwards all health check...". These rules can be easily adapted or used as templates for your own health checking rules. ▪ SERVER_EVENTLOG combines SERVER and EVENTLOG.
Severity of agent health problem messages	List	▪ Normal ▪ Warning ▪ Minor ▪ Major ▪ Critical	Critical	The severity for messages that indicate problems with agent health checking. For example, "Node xxx may be down. Failed to contact it using ping." If you configure the Target for agent health problem messages to include the event log, this value sets the event types as follows: ▪ Normal results in information events. ▪ Warning, minor, and major result in warning events. ▪ Critical results in error events.
Health check report buffering	Boolean	▪ True ▪ False	True	This value configures whether to report that an agent is buffering messages.

Severity of buffering for this management server	List	■ Normal ■ Warning ■ Minor ■ Major ■ Critical	Major	This value configures the severity of messages that indicate that the agent is buffering messages for this management server.
Severity of buffering for other management servers	List	■ Normal ■ Warning ■ Minor ■ Major ■ Critical	Warning	This value configures the severity of messages that indicate that the agent is buffering messages for a management server other than this one.
Enable access denied warning for raw socket creation	Boolean	■ True ■ False	True	This value configures whether to write a warning to the system event log if the management server cannot accept alive packets from agents. (See Accepting alive packets below.)

Accepting alive packets

On nodes that have the DCE agent, the message agent sends an alive packet to the management server at a configurable interval. However, in HPOM 8.10, the management server is no longer able to receive these alive packets by default. The management server runs under the HP-OVE-User account, which no longer has administrative rights. Without administrative rights, the management server cannot open the raw socket that it needs to receive alive packets.

To continue receiving alive packets, you must add the HP-OVE-User to the local administrators group on the management server. Before you give the HP-OVE-User administrative rights, check the security requirements of your organization.

If the management server can accept alive packets, it checks whether it received a packet from a node before it contacts that node by ICMP ping or call to the control agent. If the management server has received an alive packet, it does not attempt to contact the node.

You can change the frequency with which each node sends alive packets. You do this by configuring the value for OPC_HBP_INTERVAL_ON_AGENT in a nodeinfo policy, which you deploy to the agent. The agent sends an alive packet at an interval equal to two-thirds of the configured value. On nodes that have the DCE agent, the default value of OPC_HBP_INTERVAL_ON_AGENT is 280, so the agent sends an alive packet every 120 seconds.

If the management server cannot accept alive packets, change the default value of OPC_HBP_INTERVAL_ON_AGENT to 0 on nodes with DCE agents. The agent stops sending alive packets, which prevents unnecessary network load. On nodes that have the HTTPS agent, the value is not set by default, so the HTTPS agent sends no alive packets by default.

Changing agent health check behavior

- To reduce network traffic by monitoring less frequently, increase Time interval to check agent health and OPC_HBP_INTERVAL_ON_AGENT. Time interval to check agent health should remain greater than the value for OPC_HBP_INTERVAL_ON_AGENT to ensure that the server looks for the alive packet after the node sends it.
- To increase monitoring, decrease Time interval to check agent health and OPC_HBP_INTERVAL_ON_AGENT. Time interval to check agent health should remain greater than the value for OPC_HBP_INTERVAL_ON_AGENT to ensure that the server looks for the alive packet after the node sends it.
- To monitor nodes through a firewall, or if the ICMP-port cannot be used, set Health check ping protocol to AGENTONLY to switch off the check with PING-packets. The active check with RPCs will still be done. Note that this increases the network traffic because the server will check the health of the agent with an RPC call each time the Time interval to check agent health is exceeded. (RPC calls require more bandwidth than ping.)
- To reduce CPU load and memory consumption, reduce the value of Maximum number of parallel checks . This might be necessary when monitoring large environments, or if the management server has limited resources.
- To stop the health check entirely, set Enable health check to false.

Related Topics:

- Configure network information for managed nodes
- Node Info Policy Type
- Scalable Architecture for Multiple Management Servers
- Synchronize packages

HP Performance Manager Integration

HP Performance Manager is a web-based tool that enables you to analyze system performance and resource utilization using graphs. HP Performance Manager provides preformatted graphs and also enables you to design your own graphs.

For details of the versions supported for integration with HP Operations Manager for Windows, see the support matrix at HP Software Support Online.

This version of HPOM does not include a built-in graph designer or viewer. However, the license to use HPOM includes a separate license key for HP Performance Manager. The included license key enables you to install and use HP Performance Manager on the same computer as your HPOM management server . (The license key for HP Performance Manager is different to the license key for the HPOM management server.)

If you install HP Performance Manager on the same computer as your HPOM management server, the Graphs snap-in appears in the HPOM console tree. You can access the graphs that HP Performance Manager generates, directly from the HPOM console. Also, HP Performance Manager can load the list of nodes and node groups from HPOM.

HP Operations agents gather and store performance data locally on nodes. HP Performance Manager can connect to any nodes that have an HPOM agent, and request performance data to generate graphs from that node.

You can associate graphs with node groups, services, and service types. This enables you to more easily access appropriate reports for those node groups and services. Each time you request a graph, HP Performance Manager requests the performance data from the node, and generates the graph immediately.

Related Topics:

- Configure HP Performance Manager integration
- Show graphs
- Node group reports and graphs properties
- Configure reports and graphs for services
- Configure reports and graphs for service types

Configure HP Performance Manager integration

If you install HP Performance Manager on the same computer as your HPOM management server, the installation configures the integration between HPOM and HP Performance Manager automatically. For more details, see the *HP Performance Manager Installation Guide* , which is available from the HP Performance Manager installation media or the product manuals search page at HP Software Support Online.

After you install HP Performance Manager, the Graphs snap-in appears in the HPOM console tree. You can configure whether operators can view graphs by setting the Graphs can be viewed check box in their user roles.

Reconfigure the HP Performance Manager ports in HPOM

The default ports that HPOM uses to connect to HP Performance Manager are preconfigured on the HPOM management server. If you reconfigure HP Performance Manager to use different ports, you can reconfigure HPOM to connect to a these ports.

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click HP Performance Manager Integration . A list of values appears.
3. *Optional.* Set the value of HTTP Port .
4. *Optional.* Set the value of HTTPS Port .
5. Click Apply .

Enable file downloads in Group Policy

To be able to export graphs from the HP Performance Manager Graph Design Wizard from within the HPOM console, you must first configure your web browser to allow file downloads from the Internet Zone. You can do this locally on the computer that runs the remote console, or globally for all computers in the domain.

1. Start Group Policy Object Editor.
2. Navigate to the following policy object:

   Local Computer Policy → User Configuration → Administrative Templates → Windows Components → Internet Explorer → Internet Control Panel → Security Page → Internet Zone → Allow file downloads
3. Enable the Allow file downloads policy to allow file downloads from the Internet Zone.

NOTE:
The HPOM console requests graphs directly from the HP Performance Manager server using standard

Windows web browsing components. For each computer that runs an HPOM console, the Internet Options in Windows must be correctly configured to communicate with the HP Performance Manager web server.

Related Topics:

- Configure general information for user roles

Show graphs

HPOM enables you to access the graphs that HP Performance Manager generates, directly from the HPOM console. You can do this from the following starting points:

- the Graphs snap-in
- a node group
- a service
- a message

You can configure whether reports appear in a new window or the details pane. To do this, open the Console Properties dialog and set the Show results in new window check box.

To show a graph from the Graphs snap-in

1. In the console tree, expand Graphs . The console tree shows the available graph families.
2. Expand a graph family. The console tree shows the graph categories.
3. Click a graph category. The details pane shows the available graphs.
4. Double-click a graph. The Show Graph wizard opens.
5. In the node tree, select the nodes that you want to see a graph for. To open several graphs at the same time, select node groups or multiple nodes. Click Next .
6. Select a Date Range that corresponds to the period you want to analyze.
7. Select a Granularity to specify how much detail you need.
8. *Optional.* Select the Periodically update data in graph checkbox to have your graph automatically refreshed. Clearing this check box can increase graph performance.
9. Click Show . A new window opens, which shows the graph. If you selected node groups or multiple nodes, each graph appears on a separate tab in this window.

TIP:
Expand Temporary Graphs in the console tree for a list of reports that you viewed recently. Click any graphs that you want to review.

To show a graph for a node group:

1. In the console tree, right-click a node group, and then click All Tasks → Show Graph... The Show

Graph wizard opens.

2. The Show Graph wizard shows a tree that contains all graphs that are available for this node group. Expand the graph family and graph category for the graph that you need.
3. Click a graph in the tree, and then click Next . A node tree appears.
4. In the node tree, the nodes in the node group are preselected. You can optionally change the preselection. Click Next .
5. Select a Date Range that corresponds to the period you want to analyze.
6. Select a Granularity to specify how much detail you need.
7. *Optional.* Select the Periodically update data in graph checkbox to have your graph automatically refreshed. Clearing this check box can increase graph performance.
8. Click Show . A new window opens, which shows the graph for each node that you selected. Each graph appears on a separate tab in this window.

NOTE:
You can only show a graph for a node group after the administrator associates graphs with the node group. (For more information, see Node group reports and graphs properties .)

To show a graph from a service:

1. In the console tree, right-click the service, and then click All Tasks→Show Graph... *or* in the service map, right-click the service, and then click Show Graph... . The Show Graph wizard opens.
2. The Show Graph wizard shows a tree that contains all graphs that are available for this node group. Expand the graph family and graph category for the graph that you need.
3. Click a graph in the tree, and then click Next . A node tree appears.
4. In the node tree, select the nodes that you want to see a graph for. To open several graphs at the same time, select node groups or multiple nodes. Click Next .
5. Select a Date Range that corresponds to the period you want to analyze.
6. Select a Granularity to specify how much detail you need.
7. *Optional.* Select the Periodically update data in graph checkbox to have your graph automatically refreshed. Clearing this check box can increase graph performance.
8. Click Show . A new window opens, which shows the graph for each node that you selected. Each graph appears on a separate tab in this window.

NOTE:
You can only show graphs from a service after the administrator associates a graph family or graph

category with the service or service type. (For more information, see Configure reports and graphs for services and Configure reports and graphs for service types .)

To show a graph from a message:

In the message browser, you can right-click a message, and then click Show Graph →Nodes... or Show Graph →Services... . The Show Graph wizard opens.

For some messages, the operator-initiated command shows a graph. To start an operator-initiated command, right-click the message, and then click Commands→Start→Operator Initiated . You can define operator-initiated commands that launch graphs in measurement threshold policies.

TIP:
To print a graph from the HP Performance Manager window, click Graphs→Show Print View . Your default Internet browser opens, with a view of the graph for you to print using the browser's print command.

Related Topics:

- Specify console General properties
- Add operator-initiated commands to a policy rule

HP Reporter Integration

HP Reporter is a flexible management reporting solution for the distributed IT environment. The license to use HPOM includes a separate license key for HP Reporter. (The license key for HP Reporter is different to the license key for the HPOM management server.)

HPOM provides two report packages, which contain report templates for use with HP Reporter:

- The HP Operations Smart Plug-in for Microsoft Windows Operating System reports package provides reports on the performance of Windows and Internet Information Server.
- The HP Operations Reports for Windows reports package provides event and service reports. These provide short-, medium-, or long-term views of your IT environment. You can use the reports, for example, to analyse trends in the following ways:
  - Identify potential bottlenecks in your IT system, so that you can take action before problems appear
  - Make accurate predictions for future upgrades
  - Collect accurate information to measure service levels.

NOTE:
The HP Operations Reports for Windows support unplanned outages. The following reports provide information on unplanned outages:

- Node Outage Messages
- OutageMess Overview
- Service Outage Messages

Other reports exclude information on nodes and services that are in unplanned outage mode.

HPOM also enables you to access any of the other reports that HP Reporter generates, directly from the HPOM console.

You can install HP Reporter on the same computer as your HPOM management server , or on a different computer. For details of the versions supported for integration with HPOM, see the support matrix at HP Software Support Online. You may need to configure HP Reporter with details of the HPOM management server, and install the HPOM reports packages.

After you configure HP Reporter integration, the Reports snap-in appears in the console tree. HP Reporter adds your HPOM managed nodes and node groups to its database of discovered systems, and begins to gather data from them. HP Reporter uses the gathered data and report templates to generate reports.

HP Reporter generates some reports that aggregate data from all discovered nodes. It is also often useful to have a report that aggregates data from a particular subset of nodes. HPOM enables you to associate a report template with a node group using the console. HP Reporter then generates the reports for the node group, and any subgroups. However, HP Reporter runs jobs that gather data and generate reports according to a schedule. (This is by default once a night, but you can reschedule the jobs and start them on demand from the HP Reporter interface.) Therefore, after you make changes to nodes and node groups in the HPOM console, there is a delay before HP Reporter updates the reports to show the changes.

You can also associate reports with services, and service types using the HPOM console. This does not configure to HP Reporter to generate any new reports, but enables you to more easily access appropriate reports for those services.

NOTE:
HP Reporter also enables you to generate reports for individual nodes. However, you can access these only from the HP Reporter interface.

Related Topics:

- Configuring outage information
- Configure HP Reporter integration
- Show reports
- Node group reports and graphs properties
- Configure reports and graphs for services
- Configure reports and graphs for service types
- Choosing a Windows OS SPI report

Configure HP Reporter integration

HP Reporter software is not installed by default with HPOM. You can install HP Reporter on the same computer as your HPOM management server, or on a different computer. The HP Reporter installation media is included with the HPOM installation media bundle.

Configure HP Reporter with details of HPOM

After you install HP Reporter, you must configure HP Reporter with details of the HPOM management server. For more details, see the *HP Reporter Installation and Special Integrations Guide* (In the HP Reporter interface, click Help→Reporter Document Set or check the product manuals search page at HP Software Support Online)

After you configure HP Reporter with details of HPOM, the Reports snap-in appears in the HPOM console tree. You can configure whether operators can view reports by setting the Reports can be viewed check box in their user roles.

Install HPOM report packages

HPOM provides two report packages for use with HP Reporter. If you install the HPOM management server on a computer that already has HP Reporter installed, the HPOM installation installs the packages automatically. Otherwise, you can find the report package installers in the following locations on the HPOM installation media:

- `\Packages\FoundationSPIs\HPOvSpiWinosR-<`*version number*`>-WinNT4.0-release.msi`
- `\Packages\FoundationCore\HPOMServiceReports\HPOvMgrRpt-<`*version number*`>-WinNT4.0-release.msi`

The same packages are also available on the HP Operations Smart Plug-ins installation media. (However, in some versions, `HPOvMgrRpt-<`*version number*`>-WinNT4.0-release.msi` has the name `HP Operations Manager Reports for Windows.msi` .) For more information, see the *HP Operations Smart Plug-ins DVD Installation and Upgrade Guide* .

Reconfigure the HP Reporter server name and port in HPOM

HP Reporter automatically configures its server name and port on the HPOM management server. If these details later change, you can reconfigure HPOM to connect to a different server or port.

1. In the console tree, right-click Operations Manager , and then click Configure→Server... . The Server Configuration dialog box opens.
2. Click Namespaces , and then click HP Reporter Integration . A list of values appears.

3. *Optional.* Set the value of Port .

4. *Optional.* Set the value of Server name .

   NOTE:
   If you set an empty Server name , the Reports snap-in disappears from the console tree.

5. Click Apply .

NOTE:
The HPOM console requests reports directly from the HP Reporter server using Windows components. For each computer that runs an HPOM console, the Internet Options in Windows must be correctly configured to communicate with the HP Reporter web server.

Related Topics:

- Configure general information for user roles

Show reports

HPOM enables you to access the reports that HP Reporter generates, directly from the HPOM console. You can do this from the following starting points:

- the Reports snap-in
- a node group
- a service
- a message

You can configure whether reports appear in a new window or the details pane. To do this, open the Console Properties dialog and set the Show results in new window check box.

To show reports from the Reports snap-in

1. In the console tree, expand Reports . The console tree shows the available report families.
2. Expand a report family. The console tree shows the report categories.
3. Click a report category. The details pane shows the available reports. For each report, the list shows which templates are available for the report definition (all, group, and system).
4. Double-click a report. The report appears in a new window or the details pane. If there is more than one template for the report definition, the default report appears.

TIP:
Expand Temporary Reports in the console tree for a list of reports that you viewed recently. Click any reports that you want to review.

To show a report for a node group:

1. In the console tree, right-click a node group, and then click All Tasks → Show Report... The Show Report wizard opens.
2. The Show Report wizard shows a tree that contains all reports that are available for this node group. Expand the node group, report family, and report category for the report that you need.
3. Click the report in the tree, and then click Show . The report appears in a new window or the details pane.

NOTE:

You can only show a report for a node group after the administrator associates reports with the node group. (For more information, see Node group reports and graphs properties .) There is then a delay before the reports are available because HP Reporter generates reports according to a schedule.

To show a report from a service:

1. In the console tree, right-click the service, and then click All Tasks→Show Report... *or* in the service map, right-click the service, and then click Show Report... . The Show Report wizard opens.
2. The Show Report wizard shows a tree that contains a report family. Expand the report family and a report category for the report that you need.
3. Click the report in the tree, and then click Show . The report appears in a new window or the details pane.

NOTE:
You can only show reports from a service after the administrator associates a report family or report category with the service or service type. (For more information, see Configure reports and graphs for services and Configure reports and graphs for service types .)

To show a report for a message:

In the message browser, you can right-click a message, and then click Show Report→Nodes... or Show Report→Services... . The Show Report wizard opens.

TIP:
To print a report that is currently showing, press CTRL + P .

Related Topics:

- Specify console General properties

Configure service logging

The management server calculates the current status of each service, based on active messages that contain the service's ID. By default, the management server does not store service statuses in the database. However, the service reports that HPOM provides require data on how the status of services change over time. Therefore if you intend to use service reports, you must configure the service logger to store status changes of the services that you are interested in.

You configure the service logger using a tool , which you launch from the console. The service logger itself runs as a Windows service (OvServiceLogger), which checks service statuses every five minutes and logs changes.

To configure service logging

1. In the console tree, expand **Tools** → **HP Operations Manager Tools** → **Service Logging** .
2. In the details pane, double-click **ServiceLogger Service GUI** . The Service Logger Configuration dialog appears.
3. Select the check box of each service that you want service reports for. You can uniquely identifiy services using their service IDs. Service names are not necessarily unique.
4. Click **OK** .
5. Wait until HP Reporter next runs the jobs that gather data and generate reports. This is by default once a night.
6. In the console tree, expand **Reports** → **OVO / Windows History** → **OVO / Windows Services** .
7. Double-click a report. A new window opens, which shows the report.

Related Topics:

- Configure HP Reporter integration

HP NNM Adapters

There are two adapters that integrate HP Network Node Manager (NNM) with HP Operations Manager for Windows (HPOM).

- HP NNMi Adapter is for the HP Network Node Manager i (NNMi) integration. It integrates NNMi with HPOM for Windows, and forwards incidents from NNMi to HPOM for Windows using the Incident Web Service.
- HP NNM Adapter is for the older NNM 7.x integration. It forwards events from NNM 7.x to HPOM using opctrapi.

Related Topics:

- HP NNMi Adapter for HPOM for Windows
- HP NNM Adapter for HPOM for Windows

HP NNMi Adapter

HP NNMi Adapter automatically forwards incidents from HP Network Node Manager i Software (NNMi) into the HPOM for Windows active messages browser. It also provides easy access to the NNMi console from within HP Operations Manager (HPOM) for Windows.

The HP NNMi Adapter is installed automatically with HP Operations Manager for Windows, but you need to perform some configuration tasks before you can use it. See HP NNMi Adapter Configuration Tasks .

HP NNMi Adapter: Three Main Components

- HP NNMi–HPOM Integration Module

  The HP NNMi–HPOM integration module forwards incidents from NNMi to HPOM. It is installed and configured on the NNMi management server.

  Installation and configuration requirements for the HP NNMi–HPOM integration module are described in the following documents:

  - HP NNMi 8.03: *HP NNMi Software Deployment Guide* .
  - HP NNMi 8.10: *HP NNMi Software Deployment and Migration Guide* .

- HP Operations Manager Incident Web Service

  HP Operations Manager Incident Web Service comes with HPOM, and is installed automatically with the HPOM installation. HPOM Incident Web Service integrates NNMi with HPOM, and provides the means by which incidents forwarded from NNMi are received by HPOM.

  For configuration requirements, see HP NNMi Adapter Configuration Tasks .

- HP NNMi Web Tools

  HP NNMi Web Tools are integrated into HPOM. They can be used following configuration in the server configuration user interface.

  For more information, see HP NNMi Web Tools .

HP NNMi Adapter: Features

- Automatic Event Forwarding

  Automatic forwarding of incidents from NNMi into the HPOM for Windows message browser.

- Launching the NNMi Console

  You can launch the NNMi console in the context of an incident forwarded from NNMi and in the context of an NNMi node that is set up as a managed node in HPOM. It is not necessary to deploy an agent from the HPOM server to the NNMi management server.

- Launching HP NNMi Web Tools

  You can launch HP NNMi Web Tools from the HPOM user interface to assess the network status.

NNMi detects a network problem, processes and correlates it, and displays it in the NNMi incident browser. When enabled, you can configure the HP NNMi Adapter to forward incidents automatically to one or more HPOM servers. You can also configure filters in NNMi that enable incidents to be forwarded to HPOM only when they meet certain criteria.

You can see the forwarded incidents in the HPOM active messages browser. These messages in the HPOM browser are associated with the original incidents reported in NNMi. So from within HPOM you can launch the NNMi incident browser showing the original incident. Each NNMi incident has a unique identity, so that even where HPOM is consolidating events across multiple NNMi management server installations, you can trace a particular incident back to its origin in NNMi and investigate it.

You can access HP NNMi Web Tools from nodes in the HPOM console, and from the active and acknowledged message browsers. All HP NNMi Web Tools require a web browser supported by NNMi to be installed on HPOM consoles. Check the NNMi documentation for supported web browser versions.

Related Topics:

- HP NNMi Adapter Configuration Tasks
- HP NNMi Web Tools
- Synchronization of Incident Updates

HP NNMi Adapter Configuration Tasks

The HP NNMi Adapter is installed automatically with HP Operations Manager for Windows, along with HP Operations Manager Incident Web Service.

The installation process also automatically creates a Microsoft Internet Information Server (IIS) certificate. This certificate ensures secure access to the HPOM Incident Web Service, and is valid for 20 years.

The following configuration tasks are required to enable the HP NNMi Adapter:

1. On the NNMi management server, follow the configuration steps appropriate for your version of NNMi:
   - HP NNMi 8.03: view the configuration steps

Configuring the HP NNMi Server Name and Port

You need to specify the NNMi server name and port number in HPOM for Windows. This configuration information is needed to launch NNMi web views and forms with the URL tools configured in the HP NNMi Web Tools group. You do this by changing the Server Configuration settings.

1. In the Toolbar, click Action . From the drop-down list, click Configure , and then click Server .
   The Server Configuration dialog box opens.
2. In the Namespace field, choose HP NNMi Adapter from the drop-down list.
3. Double-click NNMi server hostname to edit the hostname of the NNMi server.
4. Double-click NNMi server port to edit the port number of the NNMi server.
   Related Topics:

   - HP NNMi Web Tools: By Node
   - HP NNMi Web Tools: General
   - HP NNMi Web Tools

Configuring the NNMi Server Nodes

You need to configure the NNMi managed nodes in HPOM for Windows. This identifies the nodes from which incidents originate (emitting nodes), and allows HPOM for Windows to receive incidents, forwarded by NNMi, from these emitting nodes.

NOTE:
If you do not set up the emitting nodes of forwarded incidents in the HPOM database, then all incidents forwarded from the NNMi server will be discarded by the HPOM management server.

You can configure the NNMi managed nodes in two ways:

- As managed nodes .

  NOTE:
  For the HP NNM 7.x integration, it was a requirement to deploy an agent from the HPOM management server to the NNM management server. For the new HP NNMi–HPOM integration, it is no longer essential to do so.

- As an external node .

  One external node can be set up to catch all incidents forwarded from the NNMi nodes, eliminating the need to configure each system in HPOM for Windows as a separate managed node.

Related Topics:

- Configure managed nodes
- Configuring external nodes
- Installation and Configuration Tasks

Enabling HP NNMi Web Tools in the By Node Sub-Group

You must start HP NNMi Web Tools in the sub-group By Node in the context of a node.

Before you can use these tools in the context of a node, you need to associate the node with the By Node tools sub-group. This association should be done for all nodes that are managed by NNMi and that have been configured in HPOM for Windows to receive incidents forwarded by NNMi. You can also associate a group of nodes with the tools in the By Node sub-group (for example, by grouping all NNMi nodes in a single node group in HPOM for Windows). In this way, all nodes in this node group are associated with the tool group.

To associate a node with the HP NNMi Web Tools in the By Node sub-group:

1. In the Toolbar, click **Action** , click **Configure** , and then click **Nodes** .

   The **Configure Managed Nodes** dialog box opens.

2. Right-click the node or node group, then select **Properties** .

3. In the **Node Properties** dialog box, click the **Tools** tab. Expand the **HP NNMi Web Tools** group, select the **By Node** check box, and then click **OK** .

Related Topics:

- Configuring the HP NNMi Server Name and Port
- HP NNMi Web Tools: By Node
- HP NNMi Web Tools

HP NNMi Web Tools

There are a number of tools, in the HP NNMi Web Tools group, that are integrated into HPOM for Windows. These tools are divided into three application sub-groups.

Sub-Group Name	Description
General	Tools in the General sub-group require that the NNMi server name and port number are correctly configured in the HP NNMi Adapter section of the general server configuration user interface. To start a tool in this sub-group, double-click it.
By Incident	You must start tools in the By Incident sub-group in the context of a forwarded NNMi incident. All the information required (incident identity, source NNMi server name, and port number) is contained in the custom message attributes in the message forwarded to the HPOM for Windows message browser.
By Node	Tools in the By Node sub-group require that the NNMi server name and port number are correctly configured in the HP NNMi Adapter section of the general server configuration user interface. You must start tools in this sub-group in the context of a node, and the node must first be associated with the By Node sub-group in the HP NNMi Web Tools group to enable the tools.

Related Topics:

- HP NNMi Web Tools: General
- HP NNMi Web Tools: By Incident
- HP NNMi Web Tools: By Node
- Configuring the NNMi Server Name and Port
- Enabling HP NNMi Web Tools in the By Node Group

HP NNMi Web Tools: By Incident

The available HP NNMi Web Tools in the By Incident sub-group are listed below. See the *HP Network Node Manager Online Help* for more details about these applications.

Tools in the By Incident Sub-Group

HP NNMi Web Tool	Action Performed
Layer 2 Neighbors to related NNMi node	Launches a Troubleshooting View in web browser, showing the Layer 2 Neighbors of the node from which the corresponding NNMi incident originated.
Layer 3 Neighbors to related NNMi node	Launches a Troubleshooting View in web browser, showing the Layer 3 Neighbors of the node from which the corresponding NNMi incident originated.
Show related NNMi incident	Launches an NNMi Incident Form, corresponding to a selected message, in a web browser.
Show related NNMi node	Launches a Node Form in a web browser, showing the NNMi setup information for the node from which the corresponding NNMi incident originated.

Launching the Tools in the By Incident Sub-Group

You must start tools in the sub-group By Incident in the context of a forwarded NNMi incident. A message forwarded by NNMi contains custom message attributes about the identity of the incident, the NNMi server name, and the server port number. To run the tools:

1. Right-click a message, forwarded by NNMi, in the HPOM for Windows message browser.
2. Select Launch Tool , then select Message .
3. In the Select the Tool to Execute dialog box, expand the HP NNMi Web Tools group, select the tool in the By Incident sub-group that you want to execute, and then click Launch .

Related Topics:

- HP NNMi Web Tools

HP NNMi Web Tools: By Node

The available HP NNMi Web Tools in the By Node sub-group are listed below. See the *HP Network Node Manager Online Help* for more details about these applications.

Tools in the By Node Sub-Group

HP NNMi Web Tool	Action Performed
Ping node	Launches the ping command and shows the real-time results of the ping from the NNMi server to a selected node in a web browser.
Show Layer 2 Neighbors	Launches a Troubleshooting View in a web browser, showing the Layer 2 Neighbors of a selected node.
Show Layer 3 Neighbors	Launches a Troubleshooting View in a web browser, showing the Layer 3 Neighbors of a selected node.
Show node communication configuration	Launches the real-time results of the ICMP and SNMP configuration report in a web browser, showing the communication configuration of a selected node.
Show node configuration poll	Launches the configuration poll of a selected node, showing the real-time results of a node's configuration in a web browser.
Show node information	Launches a Node Form in a web browser, giving details about the selected node for troubleshooting purposes.
Show node status poll	Launches the real-time check and results of a selected node's status in a web browser.
Traceroute to node	Launches the real-time results of a Trace Route from the NNMi server to a selected node in a web browser.

Launching the Tools in the By Node Sub-Group

You must start tools in the By Node sub-group in the context of a node.

To run the tools, follow these steps:

1. Make sure that the NNMi server name and port number are configured correctly in the Server Configuration user interface.

   For more information, see the section Configure the HP NNMi Server Name and Port .

2. Make sure that the selected node is associated with the By Node tool sub-group.

   For more information, see the section Enabling HP NNMi Web Tools in the By Node Sub-Group .

3. Right-click the node, select All Tasks , and then select Launch Tool .

   The Select the Tool to Execute dialog box opens.

4. In the Select the Tool to Execute dialog box, expand the HP NNMi Web Tools group, and double-click the tool in the By Node sub-group that you want to execute.

Related Topics:

- Configure the HP NNMi Server Name and Port
- Enabling HP NNMi Web Tools in the By Node Sub-Group
- HP NNMi Web Tools: By Incident
- HP NNMi Web Tools: General
- HP NNMi Web Tools

HP NNMi Web Tools: General

Tools in the General sub-group require that the NNMi server name and port number are configured correctly in the HP NNMi Adapter section of the general server configuration user interface.

HP NNMi Web Tool	Action Performed
Show my incidents	Launches the My Open Incidents view in a web browser.
Show NNMi console	Launches the NNMi console in a web browser.
Show NNMi server status	Launches a report of the current status of all NNMi server processes and services in a web browser.
Show open root cause incidents	Launches the Open Root Cause Incidents view in a web browser.
Show sign in audit log	Displays the current configuration for a node in a web browser (tracks log on and log out activity for each user account).

Launching the Tools in the General Sub-Group

1. Make sure that the NNMi server name and port number are configured correctly in the Server Configuration user interface.

   For more information, see the section Configure the HP NNMi Server Name and Port .
2. Double-click the tool you want to start.

Related Topics:

- Configuring the HP NNMi Server Name and Port
- HP NNMi Web Tools: By Incident
- HP NNMi Web Tools: By Node
- HP NNMi Web Tools

Synchronization of Incident Updates

When configured to do so, NNMi forwards incidents to one or more HPOM servers. NNMi will acknowledge an incident to one or more HPOM installations if that incident's lifecycle state changes to closed . NNMi will unacknowledge an incident to one or more HPOM installations if that incident's lifecycle state changes from closed .

Updates to these forwarded incidents are sent from the HPOM management server back to the NNMi management server to synchronize the lifecycle state of the incident.

Incident lifecycle state changes are synchronized from NNMi to HPOM and back to NNMi as follows:

Trigger	Result
In HPOM, the message is acknowledged.	In NNMi, the corresponding incident's lifecycle state is set to Closed .
In HPOM, the message is unacknowledged.	In NNMi, the corresponding incident's lifecycle state is set to Registered .
In NNMi, incident's lifecycle state is set to Closed .	In HPOM, the corresponding message is acknowledged.
In NNMi, the incident's lifecycle state is changed from Closed to any other state.	In HPOM, the corresponding message is unacknowledged.

Related Topics:

- HP NNMi Adapter Configuration Tasks

HP NNM Adapter for HPOM for Windows

HP NNM Adapter is a product that integrates NNM 7.x with HP Operations Manager for Windows. HP NNM Adapter automatically forwards events from NNM into the HP Operations active messages browser, and integrates HP NNM Web Tools into HP Operations Manager for Windows. NNM Web Tools are available to all nodes that are managed by both HPOM for Windows and NNM for Windows. Features include:

- New discovery mechanism: NNM folder appears in the Configure Managed Nodes dialog box.
- HP NNM Web Tools are available to all nodes managed by NNM and HPOM for Windows.
- Automatic forwarding of events from NNM into the HPOM for Windows message browser.

You can access HP NNM Web Tools from Service Views and nodes in the console tree and from the active and acknowledged message browsers. All HP NNM Web Tools require Internet Explorer supported by NNM to be installed on HPOM consoles. Some tools also require a Java Plug-in supported by NNM. Check the NNM documentation for supported Internet Explorer and Java Plug-in versions.

The tools and their functions are described in the expandable list shown below.

View the list of NNM tools integrated into HP Operations Manager for Windows

Related Topics:

- HP NNM Adapter Utilities

Automatic Update

Automatic update automatically identifies nodes that come online, as well as nodes that are not monitored by NNM anymore on regular intervals.

An HP NNM Adapter component, the NgNnmUpdater, controls the update time interval. The default update interval is one day, but you can change this using the appropriate registry key.

NOTE:
If you prefer, you can start an update manually using the Updater.exe tool described in the "HP NNM Adapter Utilities" help topic.

New Nodes Discovered by HP NNM

If the automatic update finds nodes in HPOM for Windows that are also known to NNM, but they are not members of the NNM Managed Nodes group, the update process adds those nodes to the NNM Managed Nodes group. Because HP NNM Web Tools are assigned to this node group, the nodes in this group inherit the tools. HP NNM Web Tools are also assigned to services hosted on those nodes.

Disconnected Nodes

Nodes can become disconnected and are no longer present in NNM's node discovery list. This usually means that the disconnected node has an SNMP or other network-related problem. During the automatic update, these nodes are reported as Disconnected with the following message:

Automatic Update from HP NNM: Node is disconnected. Cannot be discovered by HP NNM. Node may have SNMP or other network related problem.

Disconnected nodes are removed from the NNM Managed Nodes group and HP NNM Web Tools are unassigned from services hosted on them.

The way in which disconnected nodes are handled depends on a organization's processes and preferences. With this in mind, the HP NNM Adapter does not take any automatic action on disconnected nodes. With the right user role permissions, an operator can manually disconnect a node using the operator-selectable action Remove node from HP Operations Manager for Windows . If you do not want to manage this node anymore, you can easily remove it. If you want to keep it, you can disregard the disconnection message without initiating any other actions.

NOTE:
Internally, node removal in HP Operations Manager is a complex process. The HP NNM Adapter uses the

same process as HP Operations Manager and displays this message:

Deleting this node will remove it, and the services hosted on it, from the management server inventory. Policies and packages on this managed node will not be automatically removed, but inventory about the node will be removed from the management server database. You should remove policies and packages before removing the node.

Are you sure you want to remove the node <node>?

You should answer yes only after all policies and packages are removed from the node to be deleted.

Changing the update interval for NgNnmUpdater

You can configure the update interval using the NgNnmUpdater Update Interval registry key by completing these steps:

1. Run the registry editor on the HPOM for Windows computer with this command:

   Start → Run → regedit → OK

2. In the registry editor, go to this location:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-
   Packard\OpenView\NNMAdapter\NgNnmUpdater\Repl\UpdateInterval
   ```

3. Double-click the property labeled UpdateInterval to open the Edit DWORD Value dialog box.
4. Type a new update interval in seconds.
5. Click OK to confirm the change and close the dialog box. The new update interval takes effect on the next regular update or as triggered by a user.

Related Topics:

- HP NNM Adapter Utilities

Update the node list

The node list is updated with the current NNM database data at regular intervals and stored internally. When you expand the HP NNM entry in the Discovered Nodes list, nodes are retrieved from this data and not from the environment itself. This reduces response time at the time the request is made, but is also means that the available information may not be absolutely up-to-date.

An HP NNM Adapter component, the NgNnmDwProvider, controls the time interval and the number of threads used to collect the node data from NNM for Windows. You can configure the update interval and the number of threads with the appropriate registry keys. The default update value is one hour. The default number of threads is 40.

NOTE:
If you prefer, you can start an update manually using the DwRefresh.exe tool described in the "HP NNM Adapter Utilities" help topic.

When updating starts, the NgNnmDwProvider creates many check threads which update the NgNnmDwProvider. Each check thread takes one node from NNM for Windows; a greater number of check threads means a shorter time to update NgNnmDwProvider. Updating with one check thread requires about twice the time required when updating with two check threads. However the difference between 40 and 80 check threads may be as low as 10%.

The NgNnmDwProvider can operate in two modes:

- Gathers information about nodes only from NNM

  NNM recognizes an operating system on the node only if SNMP is installed on it. If you filter nodes based on operating system equals Windows, only nodes with both Windows and SNMP installed on them will be found.
- Gathers additional information (including that from NNM) directly from nodes using WinNet calls (detects Windows nodes and OS version). To keep network traffic to a minimum, information about nodes is cached, so each node is contacted only once.

  If you filter nodes based on o perating system equals Windows, all nodes with Windows are found, whether they have SNMP installed on them or not. This is the default mode.

  NOTE:
  NgNnmDwProvider mode does not affect filtering based on operating system other than Windows 2000.

To change the number of check threads

1. Run the registry editor on the NNM for Windows computer with this command:

   Start → Run → regedit → OK
2. In the Registry Editor, go to this location:

   `HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\OpenView\NNMAdapter\NgNnmUpdater\Repl\CheckThreads`
3. Double-click the property labeled `CheckThreads` to open the Edit DWORD Value dialog box for `CheckThreads` .
4. Enter a new value for the number of check threads to be used.
5. Click OK to confirm the change and close the dialog box.

The new check threads value takes effect after you restart the NGNnmDwProvider service.

To change the update interval for NgNnmDwProvider

1. Run the registry editor on the NNM for Windows computer with this command:

   Start → Run → regedit → OK
2. In the Registry Editor, go to this location:

   `HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\OpenView\NNMAdapter\NgNnmUpdater\Repl\UpdateInterval`
3. Double-click the property labeled `UpdateInterval` to open the Edit DWORD Value dialog box for `UpdateInterval` .
4. Enter the new update interval in seconds.
5. Click OK to confirm the change and close the dialog box.

The new update interval takes effect after you restart the NgNnmDwProvider service.

To change the mode of operation

1. Run the Registry Editor on the NNM for Windows computer with this command:

   Start → Run → regedit → OK
2. In the Registry Editor, go to this location:

   `HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\OpenView\NNMAdapter\NgNnmUpdater\Repl\AdditionalChecking`

3. Double-click the property labeled `AdditionalChecking` to open the Edit DWORD Value dialog box for `AdditionalChecking` .
4. Type 0 to gather data only from NNM; type 1 to perform additional checking on nodes.
5. Click OK to confirm the change and close the dialog box.
6. Stop the NgNnmDwProvider service.
7. Delete <HP BTO product base dir>\Installed Packages\{c9322d6f-d88c-11d3-98e3-o80009ef5c3b}\data\NgNnmPreviousNodes.bin file
8. Start the NgNnmProvider service.

Related Topics:

- HP NNM Adapter Utilities

Message-service assignments

Messages generated by the SNMP policies NNM-SNMP-NonNorm and NNM-SNMP-All are assigned to different services based on message context.

Messages related to NNM itself are assigned to the Network Node Manager service.

Network-related messages, such as Segment critical, Network critical, are assigned to the Network Infrastructure service.

System-specific messages, such as Node Down, Node Up, are assigned to the Systems Infrastructure service hosted on the node the message originates from.

If you manually add a component to the Systems Infrastructure service, you must set the service ID to `SystemServices` . SNMP system-specific messages are assigned to services with the SystemServices@@<nodeID> Service ID.

Messages originating from the HP NNM Adapter, forwarded by the NNMAdap-FwdAllLogEntries policy are assigned to the NNM Adapter service.

HP NNM Adapter utilities

The following utilities are installed with the HP NNM Adapter:

ConfNnmTools.vbs

This utility configures the NNM web server and enables the Customer Views tool in HP Operations Manager. If Customer Views is installed before the HP NNM Adapter is installed, the HP NNM Adapter installation process automatically configures Customer Views as a web tool in NNM.

To be used if Customer Views is installed after the HP NNM Adapter installation or if the NNM web server configuration needs to be changed. For example, location or port: secure -insecure. This utility is located on the HP Operations Manager server system in this directory:

```
%OvInstallDir%\NnmAdapter\bin
```

The default location is:

```
C:\Program Files\HP\HP BTO Software\NnmAdapter\bin
```

DwRefresh.exe

This utility starts NgNnmDwProvider refresh on demand, so that you can manually start the update process that occurs at regular intervals in the NgNnmProvider. The default update interval is one hour. The DwRefresh.exe utility is useful for actualizing data in NgNnmDwProvider before it does an auto update.

When the HP NNM entry in the Discovered Nodes list is expanded, nodes are retrieved from the NgNnmProvider, which means that the node list may be out-of-date by as much as the update interval. For example, if the last regular update occurred at 13:00, then the next update should occur at 14:00. However, if DwRefresh.exe is started at 13:20, then the update occurs immediately and the next update will occur at 14:00. DwRefresh.exe starts the updating process asynchronously (it does not wait for the previous update process to finish.)

The DwRefresh.exe utility is located on the HP Operations Manager server system and on the NNM server system. On the NNM server system, the file is located here:

```
<HP BTO product base dir>\Installed Packages\{c9322d6f-d88c-11d3-98e3-080009ef5cb}\bin
```

On the HP Operations server system, the file is located here:

```
C:\Program Files\HP\HP BTO Software\NnmAdapter\bin
```

Updater.exe

This utility controls the NgNnmUpdater service. To view the current status of the updater service (NgNnmUpdater), type `updater status` . Output includes the following information:

- NgNnmUpdater service updating status (Idle, Updating)
- Update interval
- Last update attempt (successful or unsucessful)
- Last successful update finish.
- When the next update begins.

Use `updater update` to start the update process manually. This is the same update process that occurs at regular intervals. The default update interval is one day. The `Updater.exe` utility is located on the HP Operations server system in this directory:

```
%OvInstallDir%\NnmAdapter\bin
```

The default installation location is:

```
C:\Program Files\HP\HP BTO Software\NnmAdapter\bin
```

Solving problems with HP NNM Adapter

Common problems, prerequisites, and workarounds include the following:

1. Problem: In order for the Network Presenter web tool to work, the NNM GUI must be running on the NNM server. If Network Presenter is started and the NNM GUI is not running, the following error message displays:

   ```
   Cannot find an ovw on host <nnm server> with map named default using session ID
   <sessionId>.
   ```

   Solution: To solve this problem, first start the NNM GUI on the NNM server before starting the Network Presenter.

2. Problem: HP NNM Adapter tools cannot be executed on a remote console. When starting HP NNM Adapter tools (HP NNM Web Tools) on a remote console logged in as a user with no administrative privileges, the following error message will appear on the local computer: "CompLoader: regstring execution failed."

   Solution: To avoid this problem you can do one of the following:

   On a remote console machine grant local administrative privileges to the interactive user accessing the remote console.

   Log in as the HP OVE Administrator or the HP OVE Operator with local administrative privileges and start any NNM web tool. After one successful NNM web tool execution, HP NNM Web Tools can be executed by any HP OVE Administrator or HP OVE Operator.

   Manually register the NnmNextGenProxy component: Log in as a user with local administrative privileges (on a remote console machine)

   At the command prompt, execute:
   ```
   regsvr32 \\<HP Operations server machine>\NnmUtils\NnmNextGenProxy.dll
   ```

3. Problem: If the HP NNM Adapter installation was aborted, reinstallation might fail or seem to take a very long time.

   Solution: Do a manual cleanup before reinstalling HP NNM Adapter.

   NOTE: Use this method ONLY if you are an experienced systems administrator.

HP Operations server	NNM server
1. execute <HP BTO product base dir>\NNM Adapter Package\Support\NNMAClean.vbs at command prompt: cscript NNMAClean.vbs	

2. stop NgNnmUpdater service (if exists) Note: kill NgNnmUpdater process if service cannot be stopped	
3. stop NgNnmAdapterProvider service (if exists) Note: kill NgNnmAdapterProvider process if service cannot be stopped	
4. check if "NgNnmEventsMan.exe" processes is running; kill it if it does	
	5. stop NgNnmDwProvider service (if exists) Note: kill NgNnmDwProvider process if service cannot be stopped
6. make sure that no MMC console (for example, Computer Management, HPOM console) is opened	make sure that no MMC console (for example, Computer Management) is opened
7. unregister NgNnmUpdater service: - open command prompt in <HP BTO product base dir>\NNM Adapter Package\bin - execute: "NgNnmUpdater /UnregServer" NOTE: if NgNnmUpdater.exe does not exist skip this step	
8. unregister NgNnmAdapterProvider service: - open command prompt in <HP BTO product base dir>\NNM Adapter Package\bin - execute: "NgNnmAdapterProvider /UnregServer" NOTE: if NgNnmAdapterProvider.exe does not exist skip this step	
	9. unregister NgNnmDwProvider service: - open command prompt in <HP BTO product base dir>\Installed Packages\{c9322d6f-d88c-11d3-98e3-080009ef5c3b}\bin - execute: "NgNnmDwProvider /UnregServer" NOTE: if NgNnmDwProvider.exe does not exist skip this step

10. restart HP Operations agent: - execute: "opcagt -stop" followed by "opcagt -start"	
	11. restart HP Operations agent: execute "opcagt -stop" followed by "opcagt -start"
12. delete "<HP BTO product base dir>\NNM Adapter Package" folder	
	13. delete "<HP BTO product base dir>\Installed Packages\{c9322d6f-d88c-11d3-98e3-080009ef5c3b}" folder
	14. delete "<HP BTO product base dir>\Installed Packages\temp\{c9322d6f-d88c-11d3-98e3-080009ef5c3b}" folder
15. In the HPOM console, uninstall and delete the HP NNM Adapter policies from the server and remove the HP NNM Adapter policy group. In "Policy Management" -->"Policies grouped by type" -->"Open Message Interface," uninstall and delete the NNM Adapter policy from the server. In "Policy Management" --> "Policies grouped by type" -->"SNMP Interceptor," uninstall and delete the NNM-SNMP-All and NNM-SNMP-NonNorm" policies from the server. Delete the policy group "Policy Management&quot -->"Policy Groups" --> "NNM Adapter."	
16. REBOOT machine	REBOOT machine

On cluster system repeat clean up procedure for HP Operations server on each node in cluster except for step 14. which can be executed only on one (active) node. Clean up procedure must be performed at least on one active node. On other nodes clean up procedure can be performed either in active or passive state.

Installation Overview and Prerequisites for the HP NNM Adapter

The steps required to install the HP NNM Adapter software are:

1. Check that all prerequisites specified below are met.
2. Configure the web server on the NNM for Windows system.
3. Install the HP NNM Adapter on the HPOM and NNM systems.

Prerequisites for the HP NNM Adapter

Before installing the HP NNM Adapter, make sure that the following prerequisites are met.

- HP Network Node Manager for Windows is installed and configured, including the installation of the NNM Web Interface.
- HPOM for Windows is installed and configured.
- The interactive user (the person who installs the HP NNM Adapter) must have administrative rights on both HPOM management server and NNM server.
- The HP NNM Adapter fully supports only configurations where the HPOM management server, NNM server, and managed nodes belong to the same DNS domain. If a node belongs to a different DNS domain than the HPOM management server and HP NNM server, it might not be recognized as an NNM node and would not be added to the HP NNM Managed Nodes node group. If such a node is added using NNM discovery, this problem does not arise, even if the node belongs to a different DNS domain.
- Installation of both NNM and HPOM for Windows on the same computer is supported, but not recommended.
- 20 MB disk space on the HPOM for Windows management server to install components necessary for the integration.
- 20 MB disk space on the NNM server to install components necessary for the integration.

NOTE:
The HP NNM Adapter is not supported with NNM running on a domain controller. The adapter attempts to add a user, which is not possible on a domain controller.

Related Topics:

- HP NNM Adapter Features
- Installing and Configuring the Web Server on the NNM for Windows System
- Installing the HP NNM Adapter on a Non-Cluster Installation
- Installing the HP NNM Adapter in a Cluster Environment

- Removing the HP NNM Adapter

HP NNM Adapter Features

Installing the HP NNM Adapter adds the following features to HPOM:

- Adds the NNM server system on the node group root if the NNM server is not a managed node in HPOM.
- Creates these three services:
  - HP Network Node Manager (under Services\Applications)
  - HP NNM Adapter (under Services\Applications)
  - Network Infrastructure (under Services)
- Creates the tool group HP NNM Web Tools , which contains HP NNM Web Tools. These tools are associated with the HP NNM Managed Nodes node group. All nodes that are discovered as NNM nodes are automatically associated with these tools indirectly through the HP NNM Managed Nodes group. (The tools are inherited from the HP NNM Managed Nodes node group.) All services hosted on nodes discovered as NNM nodes are associated with these tools.
- Creates the policy group NNM Policies , which contains the following policies. These allow messages and events to be sent between HPOM and NNM:
  - NNM Adapter
    This policy is deployed on the management server during HP NNM Adapter installation and is used internally by the HP NNM Adapter.
  - NNMAdap-FwdAllLogEntries
    This event log policy is deployed on the HP Network Node Manager server and the HPOM management server during HP NNM Adapter installation. The policy forwards all HP NNM Adapter events logs to the HPOM console.
  - NNM-SNMP-NonNorm
    This SNMP policy is deployed on the NNM server during HP NNM Adapter installation. It forwards NNM events with severities other than “Normal” and normal correlated NNM events to the HPOM for Windows console.
  - NNM-SNMP-All policy
    This SNMP policy forwards all NNM events to the HPOM for Windows console. It is provided as a policy. While you could deploy the NNM-SNMP-All policy directly to the NNM server, it is not recommended because this policy forwards all NNM events, including normal messages, to the HPOM for Windows console. This would produce large numbers of messages in the console.

NOTE:
When you want to deploy a new SNMP policy, either NNM-SNMP-All or a policy you have created from

this policy, you should first remove the old policy NNM-SNMP-NonNorm to prevent duplicate messages from appearing in the browser.

Related Topics:

- Installation Requirements for HP NNM Adapter
- Installing and Configuring the Web Server on the NNM for Windows System
- Installing the HP NNM Adapter on a Non-Cluster Installation
- Installing the HP NNM Adapter in a Cluster Environment
- Removing the HP NNM Adapter

Installing and Configuring the Web Server on the NNM for Windows System

Several of the NNM features are web-based and require a web server to be installed on the same system where NNM is installed. Check the NNM documentation for details about requirements related to NNM web functionality (Readme.html → Supported Configurations).

NOTE:
Microsoft Internet Explorer 6.0 does not provide Java support but does offer a download option. If an NNM web tool that requires the Java Virtual Machine (JVM) is started and the VM is not installed on the system, you have the option to download the Java VM. The Java VM can also be downloaded from the Microsoft web site. HP NNM Web Tools from HPOM 8.00 can only be run with JPI 1.4.2.

Verifying the web server installation

To verify that a web server is installed on Windows 2000, Windows XP, or Windows 2003, make sure that you have your Windows 2000 or 2003 operating system installation media, and complete the following steps:

1. From the Start menu, select Settings → Control Panel .
2. In the Control Panel , double-click Add/Remove Programs .
3. In the Add/Remove Programs dialog box, select the Add/Remove Windows Components button, which displays the Windows Components wizard.
4. Scroll to Internet Information Services . If this box is already selected, then the web server is installed. If the web server is not listed, follow these steps:
   a. Select the Internet Information Services (IIS) check box.
   b. Click Next and wait while the wizard configures components.
   c. When prompted, insert the Windows operating system media and click OK .
   d. In the Windows Components wizard, click Finish and remove the installation media.
   e. e In the Add/Remove Programs dialog box, click Close .

Verifying the web server configuration

To verify web server configuration, from the browser try to open an NNM for Windows Web GUI URL such as:

```
http://<server_name>/OVCgi/nnmRptPresenter.exe
```

If the URL opens without error messages, the web browser configuration is correct. correctly.

Related Topics:

- HP NNM Adapter Features
- Installation Requirements for HP NNM Adapter
- Installing the HP NNM Adapter on a Non-Cluster Installation
- Installing the HP NNM Adapter in a Cluster Environment
- Removing the HP NNM Adapter

Installing the HP NNM Adapter on a Non-Cluster Installation

The HP Operations Manager for Windows NNM Adapter is installed using an installation wizard that guides you through the procedure and prompts for information you must specify. Install the HP Network Node Manager (NNM) Adapter only after the HPOM installation is complete.

To install the HP NNM Adapter, complete the following steps. Read the instructions before you begin so that you are familiar with the information you must supply during installation.

1. Start the installation process.

   To start the installation process, select Start → All Programs → HP → HP Operations Manager → NNM Adapter . The Welcome page opens.

   Click Next in the Welcome page.
2. Read the software license agreement.

   Read the license agreement statement and select I accept the terms in the license agreement if you agree to the terms, and click Next . If you do not accept the agreement, you cannot proceed with the installation.
3. Specify the NNM server name.

   Specify the NetBios name of the system where you have installed HP Network Node Manager for Windows. This is the name that Windows uses for system names in Network Neighborhood. When this dialog box first appears, NNM Server Name is automatically filled with the local NetBios name if Network Node Manager is detected on the system.

   Click Next to go to the next screen. Before you are taken to the next screen, the installation verifies access to the specified NNM for Windows server. When verification is successful and access to the NNM for Windows server is possible, the next screen appears. If access verification fails, one of the following error messages may appear:

   - Computer is not found - you are not allowed to continue.
   - The user under which setup runs does not have administrative rights on computer <...> - you are not allowed to continue.
   - Access to computer is denied - you can leave the wizard with Abort , try to change the name of computer with Retry , or select Ignore and continue to the next screen.

   Verification may fail for a number of reasons:

   - The computer is currently not available
   - It is the wrong type of computer

- You do not have administrative rights on the specified computer

   Check for the cause of the failure, correct, and retry.

4. Specify the NNM web server address.

   Specify the address of the NNM web server. The web server name usually takes one of the following forms:

   - Regular server: `http://server.company.com`
   - Secure server: `https://server.company.com`
   - Server on custom port: `http://server.company.com:8000`

   Click Next to go to the next screen.

   Access to the web server is tested when you press Next , but access is not essential for the next step to begin.

   If the web server is not found at the required location, a warning message displays; you can leave the wizard with Abort , try to change the name with Retry , or select Ignore and continue to the next screen.

5. Specify the HP NNM Adapter user.

   Specify the Windows user account and password for the account to be used to run the HP NNM Adapter components on the HPOM server. All entries on this screen are required and must be valid.

   When this dialog first appears, default values are displayed as follows:

   - Domain installation:

     Domain = the same domain that was specified during HPOM installation.
     User = the same user that was specified during HPOM installation.

   - Workgroup installation:

     Domain = local computer NetBios name.
     User = the same user that was specified during HPOM installation.

   The specified user account must have at least local administrator rights on the system where HPOM is installed. If you do not use the HP-OVE-User user, make sure that the chosen user has access rights to both the NNM for Windows server and the HPOM server. The user right `Logon as a Service` must be set for the specified user on the HPOM server.

   For workgroup installations. In the Domain box, specify the local computer NetBios name. In the User box, specify the same account that HPOM was installed with. Before continuing with the installation, create a local user account on the NNM server that is identical to the account used to install HPOM (same user name and password) and add it to the local Administrators group on the NNM server.

   Click Next to proceed to the next page.

6. Specify the agent communication type.

Select the communication type for the HP Operations agent on the NNM for Windows server. If an agent is already installed on the NNM for Windows server, the installation wizard automatically detects the communicaiton type.

Click Next to proceed to the next page.

7. Start the final installation phase.

   In the Ready to Install the Program page, click Install to start the final installation phase.

   All required information is now specified and the installation begins its final phase. During the installation of the HP NNM Adapter component on the NNM server system, the installation checks to see if a valid installation is present, and if not, displays a message and closes the installation.

   NOTE:
   For workgroup installations only: The DCOM settings cannot be applied automatically during the installation process.
   As a result, the following error message appears in a pop-up window:

   "Could not set DCOM permissions for "NgNnmDwProvider" on the NNM server. Please set these permissions manually before continuing this installation. You need to add start and access permissions for local administrators, the SYSTEM account and the user running the NNM adapter, for example the HP-OVE-USER. This user also has to be added to the local admin group or to the DCOM Security Limits for Access Permissions and Launch Permissions."

   Set the DCOM permissions manually before you continue the installation.

   This phase may take up to one hour for large environments and cannot be cancelled once started.

8. Complete the installation.

   When the InstallShield Wizard Completed page opens, click Finish to close the wizard.

   All HPOM nodes that are also found in NNM are associated with HP NNM Web Tools.

Related Topics:

- HP NNM Adapter Features
- Installation Requirements for HP NNM Adapter
- Installing and Configuring the Web Server on the NNM for Windows System
- Installing the HP NNM Adapter in a Cluster Environment
- Removing the HP NNM Adapter

Removing the HP NNM Adapter

Requesting the removal of the HP NNM Adapter removes components from both the NNM server and the HPOM server.

NOTE:
The following NNM Adapter policies are automatically installed when the NNM Adapter is installed:

- On the HPOM management server system
  - NNM Adapter
  - NNMAdap-FwdAllLogEntries
- On the NNM server system
  - NNM-SNMP-NonNorm
  - NNMAdap-FwdAllLogEntries

If you have deployed these NNM Adapter policies on managed nodes, remove them form the managed nodes first before removing the NNM Adapter.

For more information about removing policies from a node, see the section Remove policy from node .

Non-Cluster Installations

Remove the HP NNM Adapter using one of the following methods:

- Use the Windows **Add/Remove Programs** dialog box:
  a. From the Windows **Control Panel** , open **Add/Remove Programs** .
  b. Select the entry for **HP OpenView Operations NNM Adapter** .
  c. Click **Remove** .
  d. Select **Yes** in the message box that asks you to confirm the removal and to start the removal process.
- Use the installation wizard:
  a. To start the removal process, select **Start** → **All Programs** → **HP** → **HP Operations Manager** → **NNM Adapter** . The **Welcome** page opens.

     Click **Next** in the **Welcome** page.
  b. In the **Remove the Program** page, click **Remove** .

c. When the InstallShield Wizard Completed page opens, click Finish to close the wizard.

Cluster Installations

Before being able to remove the HP NNM Adapter from a cluster node, the following prerequisites have to be met:

- The HP NNM Adapter software on the cluster node installed first must be removed last. All other nodes can be handled in any order.
- The node must be the owner of HPOM resource group when removing HP NNM Adapter from the node which was installed first, otherwise it must not be the owner of the HPOM resource group. Use the Microsoft Cluster Administrator to move the group if necessary.

Remove the HP NNM Adapter using one of the following methods:

- Use the Windows Add/Remove Programs dialog box:
  a. From the Windows Control Panel , open Add/Remove Programs .
  b. Select the entry for HP OpenView Operations NNM Adapter .
  c. Click Remove .
  d. Select Yes in the message box that asks you to confirm the removal and to start the removal process.
- Use the installation wizard:
  a. To start the removal process, select Start → All Programs → HP → HP Operations Manager → NNM Adapter . The Welcome page opens.

     Click Next in the Welcome page.
  b. In the Remove the Program page, click Remove .
  c. When the InstallShield Wizard Completed page opens, click Finish to close the wizard.

NOTE:
HP NNM Adapter components are removed from the NNM server only when the HP NNM Adapter is removed from the last node (the node on which HP NNM Adapter was installed first).

The following sections describe the actions that are taken on the NNM server and the HPOM server.

HP Network Node Manager Server

The removal process performs the following tasks on the NNM server:

- NNM deployment package is removed from the NNM server:

  - Registry keys are deleted.
  - Self-registering files are unregistered (COM components).
  - Additional files (COM components, configuration files) are removed.
- SNMP policy is removed from the NNM server.
- Event log policy is removed from the NNM server.

HP Operations Manager for Windows Server

Removing the HP NNM Adapter product performs the following tasks on the HPOM server:

- The HP NNM Managed Nodes node group is cleared and associations between managed nodes and the HP NNM Web Tools are removed. Associations between services hosted on nodes contained in the NNM Managed Nodes group and HP NNM Web Tools are removed.
- Three services are deleted:
  - Network Node Manager (under Services\Applications)
  - NNM Adapter (under Services\Applications)
  - Network Infrastructure (under Services)
- OpenMessage policy is removed from the HP Operations management server.
- Event log policy is removed from the HP Operations management server.
- Self-registering files are unregistered.
- Additional server files are deleted.
- HP NNM Adapter registry entries are removed.

Post-Removal Tasks

To fully clean up the HP Operations and NNM management server systems, complete the following tasks after removing the HP NNM Adapter.

- Restart both systems.
- Remove log files on NNM and HPOM management server systems.

Related Topics:

- HP NNM Adapter Features
- Installation Requirements for HP NNM Adapter
- Installing and Configuring the Web Server on the NNM for Windows System

- Installing the HP NNM Adapter on a Non-Cluster Installation
- Installing the HP NNM Adapter in a Cluster Environment

Installing the HP NNM Adapter in a Cluster Environment

The HP Operations Manager for Windows NNM Adapter is installed using an installation wizard that guides you through the procedure and prompts for information you must specify. Install the HP Network Node Manager (NNM) Adapter only after the HPOM installation is complete.

NOTE:
Do not start the HP NNM Adapter installation on different nodes in the same cluster in parallel. Install one node first and after the installation finished successfully start with the next node. Repeat that until all designated nodes are installed.

To install the HP NNM Adapter, complete the following steps. Read the instructions before you begin so that you are familiar with the information you must supply during installation.

Requirements

Ensure that the following cluster-specific requirements are met:

- HPOM must be installed on cluster system.
- NNM server must be installed on node which is not member of HPOM cluster.
- The interactive user must have administrative rights on each cluster node and also on the NNM server node.

Installing the HP NNM Adapter on the First Cluster Node

To install the HP NNM Adapter on the first cluster nodes, complete the following steps:

1. Select the first cluster node.

   Select a cluster node and designate this system as the first cluster node to be installed. This node must be the owner of the HPOM cluster resource group. Use the Microsoft Cluster Administrator to move the group if necessary. When removing the HP NNM Adapter from a cluster environment, it is important that the software is removed from the first installed node last.

2. Install the HP NNM Adapter on the first cluster node.

   Follow the installation steps listed in Installing the HP NNM Adapter on a non-cluster installation to install the HP NNM Adapter on the first cluster node.

3. Install the HP NNM Adapter on all other cluster nodes.

   Install the HP NNM Adapter software on all other cluster nodes. For further instructions, see Installing the HP NNM Adapter on subsequent cluster nodes .

Related Topics:

- HP NNM Adapter Features
- Installation Requirements for HP NNM Adapter
- Installing and Configuring the Web Server on the NNM for Windows System
- Installing the HP NNM Adapter on a Non-Cluster Installation
- Removing the HP NNM Adapter

Installing the HP NNM Adapter on Subsequent Cluster Nodes

To install the HP NNM Adapter software on subsequent cluster nodes, complete the following steps:

1. Make sure no other HP NNM Adapter installation is running in the cluster.

   To install HP NNM Adapter on a subsequent cluster node, no other HP NNM Adapter installation is allowed to run in the same cluster. Finish all running HP NNM Adapter installations before starting it on a new node.

2. Make sure the first cluster node is running and accessible.

   To be able to install HP NNM Adapter on a subsequent cluster node, the cluster node installed first needs to be running and be accessible from the subsequent node.

3. Make sure the HPOM resource group is not active on current node.

   Make sure the HPOM resource group is not active on current node. Use the Microsoft Cluster Administrator to move the group to some other node if necessary.

4. Install the HP NNM Adapter on the cluster node.

   Follow the installation steps listed in Installing the HP NNM Adapter on a non-cluster installation to install the HP NNM Adapter on the cluster node.

5. Complete the installation.

   The installation finishes when the completion screen displays. This screen displays a list of other cluster nodes on which HP NNM Adapter should also be installed to complete the HP NNM Adapter cluster installation. Click Finish to conclude the installation.

Related Topics:

- HP NNM Adapter Features
- Installation Requirements for HP NNM Adapter
- Installing and Configuring the Web Server on the NNM for Windows System
- Installing the HP NNM Adapter on a Non-Cluster Installation
- Installing the HP NNM Adapter in a Cluster Environment
- Removing the HP NNM Adapter

HP BAC Adapter for HPOM for Windows

You can collect performance and availability data from an existing HPOM management server and view the data in HP Business Availability Center applications.

The purpose of the HP BAC Adapter is to connect to the HPOM message infrastructure, to receive events from HPOM, and to forward these events to the HP SiteScope system.

Related Topics:

- Installing the HP BAC Adapter
- Configuring the HP BAC Adapter
- Tuning the HP BAC Adapter
- Starting and stopping the HP BAC Adapter
- Uninstalling the HP BAC Adapter

Installing the HP BAC Adapter

The HP Business Availability Center Adapter (HP BAC Adapter) is installed using an installation wizard that guides you through the procedure and prompts for information you must specify. Install the HP BAC Adapter only after the HPOM installation is complete.

Read the instructions before you begin so that you are familiar with the information you must supply during installation.

NOTE:
If HPOM is installed in a cluster environment, you must Install the HP BAC Adapter on each cluster node separately.

To install the HP BAC Adapter

1. To start the installation process, select:
   Start → All Programs → HP → HP Operations Manager → BAC Adapter

   The Welcome page opens.
2. Click Next to go to the License Agreement page.
3. Read the license agreement statement and click I accept the terms in the license agreement . Then click Next to continue.
4. In the Destination Folder page, click Next .
5. In the Ready to Install the Program page, click Install to begin the installation.
6. When the Installer Completed page opens, click Finish to close the wizard.

Related Topics:

- Configuring the HP BAC Adapter
- Tuning the HP BAC Adapter
- Starting and stopping the HP BAC Adapter
- Uninstalling the HP BAC Adapter

Configuring the HP BAC Adapter

Once installed, the HP Business Availability Center Adapter (HP BAC Adapter) must be configured on the HPOM management server before it can be used.

NOTE:
If the HP BAC Adapter is installed in a cluster environment, you must configure the HP BAC Adapter on each cluster node separately. All configuration settings on all cluster nodes must be identical.

To configure the HP BAC Adapter

1. Configure the hostname or IP address of the HP SiteScope system on which the HPOM Event Monitor is installed:

   ```
   ovconfchg -ns opc.bac -set TargetHost <hostname>
   ```

2. Configure the port if you are using a port other than the default (9000):

   ```
   ovconfchg -ns opc.bac -set TargetHost <hostname> -set TargetPort <port>
   ```

   NOTE:
   If you change this setting, make sure to update the HPOM Event Monitor.

3. Configure the HPOM Event Monitor on the HP SiteScope system as described in the HP BAC documentation.

Related Topics:

- Installing the HP BAC Adapter
- Tuning the HP BAC Adapter
- Starting and stopping the HP BAC Adapter
- Uninstalling the HP BAC Adapter

Tuning the HP BAC Adapter

You can tune the HP Business Availability Center Adapter (HP BAC Adapter) by running utilities from the command line on the HPOM management server.

To check the current settings

To check the current settings, type:

```
ovconfget opc.bac
```

To change a parameter

To change a parameter, type:

```
ovconfchg -ns opc.bac -set <variable name> <value>
```

Values for <variable name> and <value> a listed in the following table:

Variable Name	Default Value	Description
TargetHost	<empty>	Host name of the HP SiteScope receiver. No connection is attempted if this is empty.
TargetPort	9000	Port number of the HP SiteScope receiver. No connection is attempted if this is 0.
CacheMax	1000	Maximum number of messages stored in cache memory to avoid database lookups.
CacheKeep	500	If cache size reaches CacheMax, only the most-recently-used messages in CacheKeep are kept in the cache. All others are removed from the cache.
Connection Timeout	300	If no new messages or message changes are transmitted to the HP SiteScope receiver, the connection is closed after this number of seconds.
MinWaitTime	15	If the connecting to the HP SiteScope receiver failed, the HPOM BAC Adapter waits this many seconds the first time after connection failure before retrying to connect. The wait time is doubled after each retry, up to MaxWaitTime.
MaxWaitTime	120	Maximum number of seconds to wait after connection failures before retry. When doubling the wait time after connection failures exceeds MaxWaitTime, the wait time is no longer doubled and MaxWaitTime is used instead.
MaxQueueLen	1000	If the connection to the HP SiteScope receiver has been lost and new messages or message changes come in, these messages and message changes are buffered in a memory queue. If the number of entries in that queue reaches MaxQueueLen, the oldest entries are removed from the queue.

NodeKeepTime	900	The HPOM BAC Adapter looks up IP addresses from hostnames. In addition, HPOM for Windows hostnames also need to be looked up from the HPOM database. These IP addresses and hostnames are stored in a memory cache. Since hostnames and IP addresses of systems can be changed, entries in that cache are invalidated (and afterwards looked up again) after NodeKeepTime seconds.

Related Topics:

- Installing the HP BAC Adapter
- Configuring the HP BAC Adapter
- Starting and stopping the HP BAC Adapter
- Uninstalling the HP BAC Adapter

Starting and stopping the HP BAC Adapter

The HP Business Availability Center Adapter (HP BAC Adapter) service is automatically started after it is installed.

If the HP BAC Adapter disconnects from HP SiteScope during operation, it tries to reconnect to the HP SiteScope at regular intervals. In the meantime, events are stored within the HP BAC Adapter.

If the HP BAC Adapter terminates from HP SiteScope during operation, the events not yet sent to HP SiteScope are lost.

NOTE:
Since the HP BAC Adapter is linked with HPOM API libraries, it might be necessary to stop the HP BAC Adapter before installing HPOM patches, and start it after the patch installation.

To start or stop the HP BAC Adapter

The HP BAC Adapter runs as a Windows service.

1. On the HPOM management server, click Start → Settings → Control Panel → Administrative Tools → Services .
2. Right-click the service `HP OpenView Operations Message Forwarder to BAC` and select Start or Stop

To view HP BAC Adapter log messages

The HP BAC Adapter writes log messages into the log file `%OvDataDir%\log\System.txt` .

Log file entries use the process name opc2bac for messages logged by the HP BAC Adapter.

Related Topics:

- Installing the HP BAC Adapter
- Configuring the HP BAC Adapter
- Tuning the HP BAC Adapter
- Uninstalling the HP BAC Adapter

Uninstalling the HP BAC Adapter

If you must remove the HP Business Availability Center Adapter (HP BAC Adapter) files from the HPOM management server, perform the following procedure.

NOTE:
If the HP BAC Adapter is installed in a cluster environment, you must remove the HP BAC Adapter from each cluster node separately.

To remove the HP BAC Adapter

1. On the HPOM management server, click Start → Settings → Control Panel → Add/Remove Programs.
2. Remove the `HP OpenView Operations, BAC Integration` program.

Related Topics:

- Installing the HP BAC Adapter
- Configuring the HP BAC Adapter
- Tuning the HP BAC Adapter
- Starting and stopping the HP BAC Adapter

HP SiteScope Adapter for HPOM for Windows

HP SiteScope Adapter integrates HP SiteScope servers, monitors, and monitor groups with HP Operations Manager (HPOM). The HP SiteScope Adapter recognizes HP SiteScope monitors and monitor groups in the monitoring environment by utilizing the service discovery technology native to HPOM. These monitors and monitor groups are made available to the HPOM console for presentation in service map form. HP SiteScope collects data from monitoring targets (for example, servers, application software) using agentless data collection. Using the collected data, HP SiteScope can send alerts to HPOM by means of the HP SiteScope Adapter.

HP SiteScope Adapter components are intended to run on the HP SiteScope server set up as an HPOM managed node. This includes the adapter setup application, the HP SiteScope alert forwarder script, and the HP SiteScope alert text template.

After installing the HP SiteScope Adapter on the HPOM management server, HP SiteScope Adapter tools and policies are available from the HPOM console.

The HP SiteScope Adapter provides a variety of tools in two major tool groups:

- SiteScope Integration tools for execution on the selected managed nodes, for example, for configuring the HP SiteScope alert forwarding application, launching the HP SiteScope user interface from the HPOM console, and starting or stopping the HP SiteScope service.
- Monitor Group tools for running on services in HPOM that have been created for HP SiteScope monitors or monitor groups.

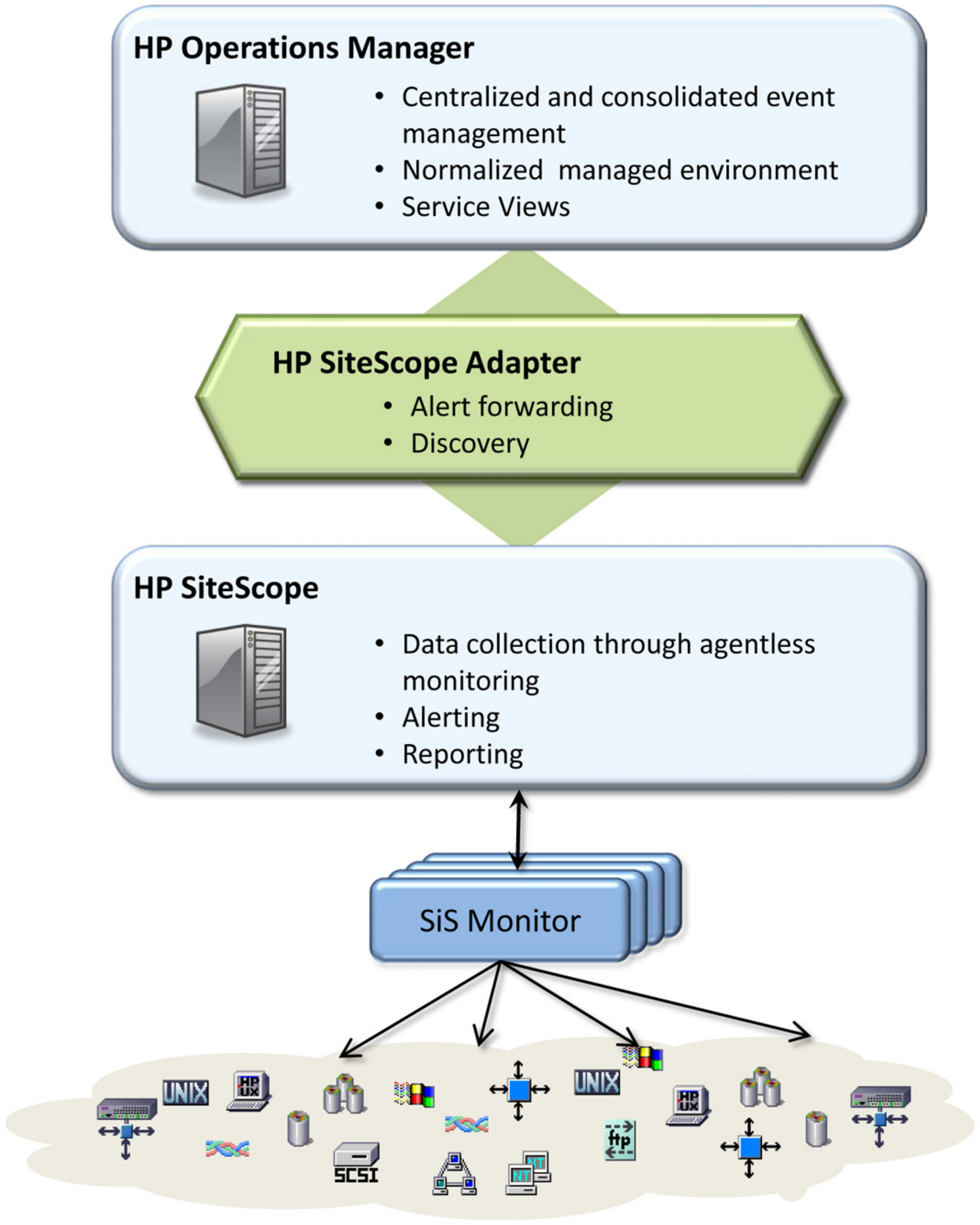
The HP SiteScope Adapter provides a variety of policies contained in three policy groups:

- SiteScope Monitor Alert policies for the HP SiteScope alert forwarding application, that transform HP SiteScope alerts to HPOM messages.
- SiteScope Monitor Config Discovery policies to assign and deploy policies to HP SiteScope servers running as HPOM managed nodes for which service discovery should be executed.
- SiteScope Server Health policies for monitoring essential HP SiteScope processes.

Following installation of the HP SiteScope Adapter, you need to deploy the necessary policies to the managed nodes.

When you have deployed the HP SiteScope Adapter to the HP SiteScope server system, you can implement HP SiteScope alerts for monitors or monitor groups, to forward the alerts to HPOM using the HP SiteScope Adapter.

The following diagram provides a simplified illustration of how the HP SiteScope Adapter fits into a typical monitoring environment.


HP Operations Manager
• Centralized and consolidated event management
• Normalized managed environment
• Service Views
HP SiteScope Adapter
• Alert forwarding
• Discovery
HP SiteScope
• Data collection through agentless monitoring
• Alerting
• Reporting
SiS Monitor
UNIX
HP UX
SCSI
NT
ftp

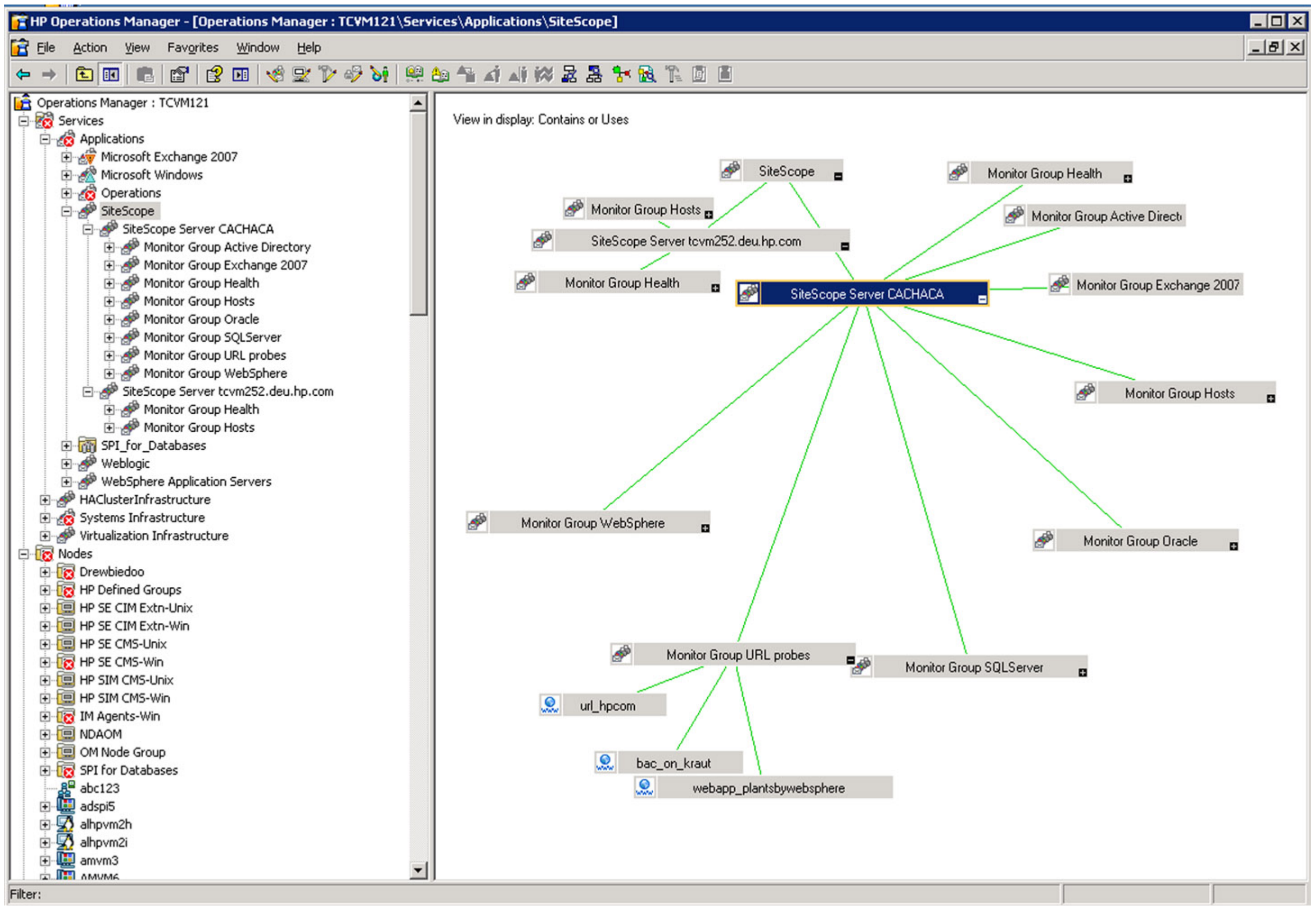
Related Topics:

- Service Discovery
- Tools
- SiteScope Integration Policies
- Implementing HP SiteScope Adapter Alerts

Service Discovery

The HP SiteScope Adapter takes advantage of the embedded service discovery and modeling technology native to HPOM. This allows for automatic and dynamic mapping of service models - including the ongoing maintenance and updating of the service map and its dependencies. The service discovery component of the HP SiteScope Adapter runs hourly on each HP SiteScope server managed node to which it has been deployed. HP SiteScope monitor groups, and monitors within each monitor group, are recognized by the discovery process and made available to the HPOM console for presentation in service map form.

The following HPOM example shows the service map representation of an HP SiteScope server with several monitors and monitor groups.



Related Topics:

- Troubleshooting the HP SiteScope Adapter

Tools

The HP SiteScope Adapter provides the following tools for execution from the HPOM console.

SiteScope Integration tool group

The following tools are available from the SiteScope Integration tools group. You can run the tools in the SiteScope Integration group directly for a selected node.

- Configure SiS2OM Adapter: Launches a script delivered by HP SiteScope Adapter on the managed node that copies relevant HP SiteScope Adapter components to SiteScope-specific directories. The script also prepares a configuration file used by the adapter during the process of forwarding alerts. Do not run the Configure SiS2OM Adapter tool until deployment of the HP SiteScope Adapter from HPOM for Windows has copied files to the managed node, as described in Creating the HP SiteScope Adapter Configuration .
- Configure SiteScope Directory (Unix): The default directory location for installation of HP SiteScope on Solaris or Linux is /opt/SiteScope. However, it is possible to install HP SiteScope at a non-default location. When deploying the HP SiteScope Adapter to Solaris or Linux managed nodes on which HP SiteScope is installed in a non-default directory, you must be run this tool prior to running the Configure SiS2OM Adapter tool.
- SiteScope Configuration: Launches the HP SiteScope interface on the system the HPOM for Windows console is running on. It opens HP SiteScope with the default top level view. This tool is run by right-clicking on a service in HPOM for Windows that has been created for an HP SiteScope server or monitor group.
- Start SiteScope: Starts the HP SiteScope service on the selected nodes.
- Stop SiteScope: Stops the HP SiteScope service on the selected nodes.
- Unconfigure SiS2OM Adapter: Removes all SiteScope Adapter components from SiteScope-specific directories that have previously been copied there during the configuration phase.

Monitor Group Tools group

The following tools are available from the Monitor Group Tools tool group. The tools in the Monitor Group Tools group must be run from within the service map context.

- View Group Dashboard: Run this tool by right-clicking on a service in HPOM that has been created for an HP SiteScope monitor group. The tool starts the HP SiteScope user interface on the system the HPOM console is running on. It opens HP SiteScope at the corresponding HP SiteScope monitor group view.

- View Monitor Dashboard: Run this tool by right-clicking on a service in HPOM that has been created for an HP SiteScope monitor. (You cannot run this tool on a SiteScope monitor group.) The tool starts the HP SiteScope user interface in a web browser on the system the HPOM console is running on. It opens HP SiteScope at the corresponding HP SiteScope Monitor view.

Related Topics:

- HP SiteScope Adapter for HPOM for Windows

SiteScope Integration Policies

The HP SiteScope Adapter ships with a set of predefined policies arranged in policy groups. There are policies available for the alert forwarding application, service discovery and monitoring of essential HP SiteScope services. This section describes the different policy sub-groups in the SiteScope Integration policy group, and their intended applications.

For a list of default message attributes set by the HP SiteScope Adapter and the related SiteScope monitor alert forwarding policies, see Message Attributes .

For a list of predefined policy condition variables that you can use for customizations, see Variables in HP SiteScope Adapter Monitor Alert Policy Conditions .

For details about how to implement meaningful alert forwarding, see Implementing HP SiteScope Adapter Alerts.

SiteScope Monitor Alerts

The HP SiteScope Adapter contains a variety of Open Message Interface policies that transform HP SiteScope alerts to qualified HPOM events, necessary for HP SiteScope monitor alert forwarding.

These Open Message Interface policies are contained in the SiteScope Monitor Alerts policy group, which contains the following policy sub-groups:

- per monitor class
- development + test
- advanced mapping for OMi
- adapter 1.x conform

NOTE:
Note that for all policies that generate events that are mapped to HPOM node configuration items, these specific items must be available on the HPOM management server to make the events visible.

SiteScope Monitor Class Policy Group: per monitor class

The per monitor class policy group contains predefined HPOM policies for each HP SiteScope monitor class. For every SiteScope monitor of a certain class to which alerts are to be forwarded, you can deploy the corresponding policy . This enables you to make very fine adjustments on event level with the benefit of policy manageability. Additionally, the policies map the out-going events to HPOM managed nodes that are the monitoring targets of the HP SiteScope monitors. All events originating from these policies have message keys to enable severity-based auto-acknowledgment of events coming from a single monitor, and duplicate counting in the HPOM message browser.

Development and Testing Policy Group: development + test

The development + test policy group consists of a single policy that is intended for development and test purposes. All incoming events from all HP SiteScope monitors and monitor groups equipped with alert actions are forwarded with no modification. The events are mapped to the HPOM managed node representing the HP SiteScope server. No acknowledgment is performed.

HP OMi-Specific Policy Group: advanced mapping for OMi

The advanced mapping for OMi policy group consists of a single policy that is intended to be used only when:

- HPOM is connected to HP Operations Manager i (HP OMi)
- and HP SiteScope 10.00 is already integrated with HP Business Availability Center (BAC) 8.xx.

In this case, the HP SiteScope Adapter can provide HP OMi with advanced information about the UCMDB configuration item (CI) being monitored by the SiteScope monitor. This information can be utilized by HP OMi to perform proper event mapping. The policy forwards all alerts coming from all HP SiteScope monitors and monitor groups that are equipped with alert actions. Events are mapped to the respective service element and automatically acknowledged by monitor source and severity level. The events are further mapped to the HPOM node configuration element representing the HP SiteScope monitoring target.

Backward-Compatible Policy Group: adapter 1.x conform

The HP SiteScope Adapter incorporates a special policy that enables backward compatibility with the previous version of HP SiteScope Adapter. The adapter 1.x conform policy sub-group consists of a single policy responsible for forwarding alerts from the HP SiteScope Adapter in a similar way to the previous adapter version. The policy conditions only distinguish between normal and non-normal incoming events from all SiteScope alerts forwarded by the adapter. All events from HP SiteScope monitors and monitor groups that are equipped with alert actions are forwarded to the HPOM management server, and matched to the HPOM node representing the HP SiteScope server. The events are equipped with message keys and are mapped to the respective service element and auto-acknowledged by source and severity level on a per-monitor basis. The event text consists only of the SiteScope monitor run result.

SiteScope Monitor Config Discovery

The SiteScope Monitor Config Discovery policy group contains the Discover SiteScope policy sub-group. Use this policy to assign policies to nodes or node groups running an HP SiteScope server for which discovery should be executed. After assigning the policies, you add the node or node group to the list of distribution targets, and then deploy the policies.

SiteScope Server Health

The SiteScope Server Health policy group contains two distinct HPOM Service/Process Monitoring policies for monitoring critical HP SiteScope server processes:

- SiteScope Server UNIX: for deployment on HP SiteScope servers with a UNIX operating system, to monitor how the SiteScope Server Engine is running.
- SiteScope Server Windows: for deployment on HP SiteScope Servers with a Windows operating system, to check that the SiteScope Tomcat WebServer process is running.

If the SiteScope Engine or Tomcat WebServer process stops, an HPOM event is generated and sent to the active message browser. You can run the included action to restart the HP SiteScope server processes. Alternatively, you can do this manually. As soon as the policies determine that the processes are running again, another HPOM event is sent to the HPOM acknowledged message browser, and any previous events from these policies are automatically acknowledged in the active message browser.

Related Topics:

- Implementing HP SiteScope Adapter alerts

Installing the HP SiteScope Adapter

Full implementation of the HP SiteScope Adapter requires the following steps:

1. Install the HP SiteScope Adapter on the HPOM management server.

   To install the HP SiteScope Adapter in a clustered HPOM environment, special installation steps are necessary. For more information, see Installing the HP SiteScope Adapter in a Cluster Environment .
2. Deploy the HP SiteScope Adapter to managed nodes.
3. Configure HP SiteScope to use the script alerts.

The HP SiteScope Adapter is installed using an installation wizard that guides you through the procedure and prompts for information you must specify. Install the HP SiteScope Adapter only after the HPOM installation is complete.

Read the instructions before you begin so that you are familiar with the information you must supply during installation.

To install the HP SiteScope Adapter

1. To start the installation process, select:

   Start → All Programs → HP → HP Operations Manager → SiteScope Adapter

   The Welcome page opens.
2. Click Next to go to the License Agreement page.
3. Read the license agreement statement and click I accept the terms in the license agreement . Then click Next to continue.
4. In the Destination Folder page, click Next .
5. In the Ready to Install the Program page, click Install to begin the installation.
6. When the Installer Completed page opens, click Finish to close the wizard.

Following successful installation of the HP SiteScope Adapter on the HPOM management server system, you will see a SiteScope Integration policy group in the Policy groups folder, and also a SiteScope Integration tools group in the Tools folder.

To determine the HP SiteScope version

After installation, it may be necessary to determine the version of HP SiteScope Adapter installed on the

HPOM server for support, patch installation, or other reasons.

1. Log on to the HPOM for Windows server system, using an account with Administrator permissions.
2. Run the Control Panel.
3. Select Add or Remove Programs .
4. Select HPOM SiteScope Adapter .
5. Select Click here for support information.

Related Topics:

- Deploying the HP SiteScope Adapter
- Implementing HP SiteScope Adapter alerts
- Installing the HP SiteScope Adapter in a cluster environment
- Uninstalling the HP SiteScope Adapter

Installing the HP SiteScope Adapter in a Cluster Environment

To install the HP SiteScope Adapter in a cluster environment, complete the following steps.

To install the HP SiteScope Adapter in a cluster environment

1. Select a cluster node and designate this system as the first cluster node to be installed. This node must be the owner of the HPOM cluster resource group. Use the Microsoft Cluster Administrator to move the group if necessary. When removing the HP SiteScope Adapter from a cluster environment, it is important that the software is removed from the first installed node last.
2. Install the HP SiteScope Adapter on the first cluster node as described in Installing the HP SiteScope Adapter .
3. Move the HPOM resource group to the next cluster node. Use the Microsoft Cluster Administrator to switch the group.
4. On the now active cluster node, install the HP SiteScope Adapter software as described in Installing the HP SiteScope Adapter .
5. Repeat steps 2 to 4 on all subsequent cluster nodes.

Related Topics:

- Installing the HP SiteScope Adapter
- Deploying the HP SiteScope Adapter
- Implementing HP SiteScope Adapter Alerts
- Uninstalling the HP SiteScope Adapter

Uninstalling the HP SiteScope Adapter

To remove the HP SiteScope Adapter files from the HPOM management server, perform the following procedures.

To remove the HP SiteScope Adapter on the management server

1. Uninstall the HP SiteScope Adapter policies from the managed nodes.

   In the console tree, right-click the policy group SiteScope and select All Tasks → Uninstall from .

2. Set the HP SiteScope alert actions not to call the HP SiteScope Adapter alert forwarding application.

3. Remove HP SiteScope Adapter alert forwarding components from HP SiteScope-specific directories on the managed nodes.

   For this task, run the HPOM tool Unconfigure SiS2OM Adapter.

4. Uninstall the HP SiteScope Adapter on the management server.
   a. Log on to the HPOM server system, using an account with Administrator permissions.
   b. Run the Control Panel.
   c. Select Add or Remove Programs .
   d. Right-click HPOM SiteScope Adapter .
   e. Select Remove.

   This removes all HP SiteScope Adapter installation components from the HPOM for Windows management server.

5. Delete the HP SiteScope Adapter tools and policies.

   After the uninstallation, delete the HP SiteScope Adapter tools and policy groups manually in the HPOM console.

To remove the HP SiteScope Adapter from the HP SiteScope servers

1. Detach the HP SiteScope monitors.

   After HP SiteScope Adapter has been removed from the HPOM server system, it is recommended that HP SiteScope monitors on managed nodes be detached from the SendOVO script alerts. This will prevent undefined messages from being sent to the message browser on the HPOM server system.

2. Delete the script alerts.

   To completely remove the HP SiteScope Adapter from the managed nodes, delete the script alerts from the HP SiteScope scripts directory on the HP SiteScope servers.

To remove the HP SiteScope Adapter in cluster environments

Before being able to remove the HP SiteScope Adapter from a cluster node, the following prerequisites have to be met:

- The HP SiteScope Adapter software on the cluster node installed first must be removed last. All other nodes can be handled in any order.
- The node must be the owner of HPOM resource group when removing the HP SiteScope Adapter. Use the Microsoft Cluster Administrator to move the group if necessary.

Related Topics:

- Installing the HP SiteScope Adapter
- Installing the HP SiteScope Adapter in a Cluster Environment
- Tools
- SiteScope Integration Policies and Message Attributes

Upgrading the HP SiteScope Adapter

If you are upgrading from an earlier version of the HP SiteScope Adapter (lower than version 2.00), perform an uninstallation as described in the appropriate product documentation. After the uninstallation, install HP SiteScope Adapter 2.00.

Binaries are replaced during installation. If you have not customized policies, no further action is required, regardless of whether you want to use the new features of HP SiteScope Adapter version 2.00, or continue to use the existing policies.

If you have customized policies, consider the following when upgrading:

- If you do not want to use the new features of HP SiteScope Adapter version 2.00, you can continue to use the existing policies. In this case, you must configure the new SiteScope Alert Action to run in a mode compatible with earlier versions of the adapter (see Backward Compatible Policy Group: adapter 1.x conform).
- If you want to use the new features of HP SiteScope Adapter version 2.00, you need to apply the customizations to the new policies manually.

  Related Topics:

  - Deploying the HP SiteScope Adapter
  - Implementing HP SiteScope Adapter alerts
  - Installing the HP SiteScope Adapter in a Cluster Environment
  - Uninstalling the HP SiteScope Adapter

Deploying the HP SiteScope Adapter

Following installation of the SiteScope Adapter on the HPOM management server, you must deploy the policies associated with HP SiteScope Adapter to HPOM managed nodes. The following prerequisites must be satisfied prior to deploying the HP SiteScope Adapter policies:

- HP SiteScope servers must be available as HPOM managed nodes.
- HPOM agents must be up and running on the HPOM managed nodes. The agent software version must be greater than or equal to versions listed in the support matrices listed at HP Software Support Online.

NOTE:
The HP SiteScope Adapter makes use of the Operations agent perl, which is installed on UNIX nodes in the following location:
`/opt/OV/nonOV/perl/a/bin/perl`
The user account the SiteScope server is executed in must have executable rights on the HPOM agent perl.

- For Linux HPOM managed nodes only, a valid shell must be accessible for the agent user, as described in the following note:

NOTE:
Linux HPOM managed nodes only: a shell must be accessible through `/usr/bin/sh` for the HPOM agent user. Create a symbolic link to a valid shell.

To deploy the HP SiteScope Adapter to managed nodes, you need to:

1. Deploy agent instrumentation.
2. Create the HP SiteScope Adapter configuration by running the Configure SiS2OM Adapter tool.
3. Deploy policies for matching SiteScope monitor classes.

Related Topics:

- Implementing HP SiteScope Adapter Alerts

Deploying Agent Instrumentation

As the first step for deploying the HP SiteSCope Adapter to the managed nodes, you must deploy the HP SiteScope Adapter components to the HPOM managed nodes representing the HP SiteScope server. The HP SiteScope server configures an HPOM deployment category which you select in the Deploy Instrumentation dialog.

To deploy the SiteScope Adapter components to the HPOM managed nodes representing the HP SiteScope server, perform the following steps:

1. In the console tree pane of the HPOM console, right-click the HPOM managed node representing the HP SiteScope server to wish you wish to deploy the HP SiteSCope Adapter components.
2. From the menu, select All Tasks → Deploy instrumentation . The Deploy Instrumentation dialog box opens.
3. From the list of Instrumentation Files in the Deploy Instrumentation dialog box, select SiteScope .
4. Select OK to close the Deploy Instrumentation dialog box, and to deploy the instrumentation.

Related Topics:

- Deploying the HP SiteScope Adapter
- Creating the HP SiteScope Adapter Configuration
- Deploying Policies to HP SiteScope Servers
- Uninstalling the HP SiteScope Adapter

Creating the HP SiteScope Adapter Configuration

Follow the steps below to move the required files to the SiteScope directories and create an Adapter configuration. Note that in step 3 below there is some special handling for UNIX nodes if HP SiteScope is installed at a non-default directory location (the default is /opt/SiteScope). The steps below require that multiple selected UNIX nodes all have SiteScope installed at the same directory location. If multiple selected UNIX nodes have more than one SiteScope installation directory, the deployment will fail. In this situation, you need to execute the steps below multiple times, once for each unique UNIX SiteScope installation directory.

To create the SiteScope Adapter configuration, perform the following steps:

1. In the console tree pane of the HPOM console, open Tools .
2. In the console tree pane of the HPOM console, select the SiteScope Integration folder to highlight it.
3. If you are deploying to a Solaris or LINUX managed node on which HP SiteScope is installed at a non-default directory location:
    a. Double-click Configure SiteScope Directory (Unix) . The Select where to launch this tool window appears.
    b. In the Select where to launch this tool window, select the node that runs the HP SiteScope server.
    c. Select Launch . The Edit Parameters window opens.
    d. In the Parameters field of the Edit Parameters window, assign a value for the directory path to which the alerts should be copied and select Launch . The Tool Status window opens.
4. In the SiteScope Integration tool group, locate the tool Configure SiS2OM Adapter and double-click it.
5. In the Select where to launch this tool window, select the HPOM node on which you want to launch the tool.
6. Select Launch . The Tool Status window provides you with some information about the tool you just executed.
7. Select Close to close the Tool Status window.

Related Topics:

- Deploying the HP SiteScope Adapter
- Deploying Agent Instrumentation
- Deploying Policies to HP SiteScope Servers
- SiteScope Alert Text Templates

Deploying Policies to HP SiteScope Servers

Follow the steps below to deploy the HP SiteScope Adapter policies to managed nodes running HP SiteScope from the HPOM console.

Deploying Policies

1. In the console tree pane of the HPOM console, open Policy management → Policy groups → SiteScope Integration .
2. In the console tree pane, right-click SiteScope Monitor Config Discovery , then select All Tasks
3. Deploy the discovery policy to the managed node(s) running HP SiteScope. Select Deploy on . The Deploy policies on window opens. From the Deploy policies on window, select nodes running the HP SiteScope server.
4. Select OK . The discovery policy is deployed and the Deploy policies on window closes. The discovered SiteScope configuration should appear as an HPOM service tree.
5. Deploy additional policies for intercepting HP SiteScope alerts, monitoring the Tomcat WebServer process for HP SiteScope, and monitoring the HP SiteScope Engine process.
    a. Under SiteScope Integration in the console tree pane, locate the policy group SiteScope SiteScope Server Health .
    b. Depending on the HP SiteScope server platform, choose either SiteScope Server UNIX or SiteScope Server Windows . Right-click your selection, select All Tasks → Deploy On . The Deploy policies on window appears again.
    c. From the Deploy policies on window, select nodes running the HP SiteScope server and click OK . The additional policies are deployed and the Deploy policies on window closes.

Deploying Policies per Monitor Class

The per monitor class policy sub-group in the SiteScope Monitor Alerts policy group contains predefined HPOM policies for each HP SiteScope monitor class.

1. In the console tree pane of the HPOM console, open Policy management → Policy groups → SiteScope Integration → SiteSCope Monitor Alerts .
2. Select the policy sub-group per monitor class .
3. For each HP SiteScope monitor that should forward alerts to HPOM, determine the corresponding monitor class and choose the appropriate HPOM policy to deploy. In the details pane, right-click the

chosen HP SiteScope monitor, then select All Tasks → Deploy on . The Deploy policies on window opens.

View example screen

4. From the Deploy policies on window, select the deployment nodes.
5. Select OK to close the Deploy policies on window, and to deploy the policy.

   View example screen

   Events intercepted by these policies are mapped to HPOM node configuration items and service elements. Automatic acknowledgement is performed by source and event severity on a per-monitor basis. The policies are also equipped with custom message attributes (CMAs) that are used by HP OMi for CI resolution when HPOM is set up to work with HP OMi.

Related Topics:

- Deploying Agent Instrumentation
- Creating the HP SiteScope Adapter Configuration

Implementing HP SiteScope Adapter Alerts

This section describes the concepts behind alert forwarding, and provides a brief example showing implementation of HP SiteScope Adapter alerts with script actions and how to forward them to HPOM. Refer to HP SiteScope product documentation for further information on HP SiteScope configuration.

NOTE:
This release of the HP SiteScope Adapter does not support use of the HP SiteScope International Version setting on Solaris and Linux managed nodes. This setting is controlled from the HP SiteScope user interface within General Preferences. An incorrect setting can result in garbled text within presentation of service data from Solaris and Linux nodes.

Alert Forwarding Concepts

The HP SiteScope Adapter ships with a variety of HPOM policies for intercepting the alerts sent by the HP SiteScope Adapter alert forwarding application. The general alert forwarding workflow is described in this section.

View workflow

The SiteScope Server has a configured set of SiteScope monitors . You can configure a SiteScope monitor alert for each monitor or group of monitors. The alert is triggered if certain threshold conditions are met. The SiteScope monitor alerts have certain alert actions associated. The status of a SiteScope monitor execution triggers such an action. If the monitor execution triggers an alert, a SiteScope alert text file is written. The format of the SiteScope alert text file is determined by a SiteScope alert text template which is shipped with the HP SiteScope Adapter instrumentation.

In case of the HP SiteScope Adapter, each of the possible monitor statuses good , warning , error and unavailable are fetched in separate actions in one single alert. Each action is of type script . This means that if the alert action is triggered, a script is called by means of shell execution. The script invokes the SiS2OM Adapter Alert Forwarder . Due to the nature of the implementation, the adapter is called by a start-up wrapper script. The adapter gets its configuration from the start-up wrapper and, of course, from HP SiteScope itself.

The HP SiteScope Adapter reads the alert text file, and, based on information regarding monitoring target and relevant parameters, determines the HPOM event message text to send. It also normalizes the message text to remove problematic characters and to achieve a uniform message text format. It uses the opcmsg binary interface to pass the event to the HPOM agent message interceptor. The HPOM agent applies policies on the incoming event. If the given conditions are met, the agent forwards the event to the HPOM management server.

HPOM policies play a central role in successful alert forwarding. A number of policies are available to match events on a per-monitor class basis. If you want to use these policies, you need to deploy them according to the monitor class of the SiteScope monitors. You can also copy conditions between monitors.

The policies of the groups per monitor class and advanced mapping for OMi utilize a uniform format message text. A common set of policy condition variables is available in case you need to make customizations.

SiteScope Alerts with Alert Actions

Once you have deployed the HP SiteScope Adapter to the HP SiteScope server system, you can configure HP SiteScope alerts for monitors, or monitor groups, to forward the alerts to HPOM using the HP SiteScope Adapter. In HP SiteScope, you typically configure one alert for each monitor or monitor group of interest. The alert itself will contain one or more script actions based on the monitor status which reflects the severity of the event being forwarded. The following screenshot shows such an alert configuration.

▼ View example configuration screen

All actions trigger the HP SiteScope Adapter alert forwarding application in the same manner. During the adapter configuration, the necessary HP SiteScope Adapter files were placed in HP SiteScope-specific dierectories. Consequently, you can select the HP SiteScope Adapter alert forwarding application as a script in the Alert Action dialog box.

▼ View example Alert Action: Script dialog box

An important setting of the Alert Action dialog box is the Trigger Frequency. The trigger frequency setting in the above example triggers the action only when the monitor status changes. This prevents unnecessary event forwarding to HPOM, and reduces the processing load on the HP SiteScope server, the HPOM agent and the HPOM management server.

In the SiteScope Administration UI, you can copy the alert to other monitors and monitor groups.

Creating an HP SiteScope Alert

Using the HP SiteScope user interface, follow the steps below to create an HPOM forwarder alert for an individual HP SiteScope monitor or monitor group:

1. In the **Monitors** tree, select the monitor or monitor group for which the alert is to be created.
2. Right-click and select **New** → **Alert** from the menu, or select the **Alerts** tab.

   ▼ View example screen" dialog box
3. In the**New Alert** window, name the alert and create a new Alert Action by selecting **New Alert Action** .

   ▼ View example New Alert dialog box
4. In the Action Type dialog box, select **Script** as the Action Type.

   ▼ View example Action Type dialog box
5. In the Alert Action: Script dialog box, select `sis2om_alert.bat` (for HP SiteScope servers on Windows

systems) or sis2om_alert.sh (for HP SiteScope servers on UNIX systems) from the Script menu.

View example Alert Action: Script dialog box

6. Choose the appropriate template from the Template menu. For short event texts, when only a limited set of information is to be forwarded to HPOM, select the OM-SiSAlert template. If more information about the monitor execution is required, select the OM-SiSAlert_full template.
7. Select an appropriate schedule for the alert from the Schedule menu.
8. In the Status Trigger settings, select an appropriate alert category condition: error, warning, or good (reset).
9. In the Trigger Frequency settings, select a trigger frequency appropriate for the alert.

   View example Alert Action: Script dialog box
10. Select OK to save the new action.
11. Repeat the above steps as required to add alerts for error, warning, and good (reset) conditions.
12. Finally, select OK to save the alert. The alert can now be copied to different monitors or groups of monitors in the HP SiteScope user interface.

Related Topics:

- Deploying the HP SiteScope Adapter

HPOM Node Configuration Example

Most HP SiteScope Adapter policies attempt to generate HPOM events mapped to an HPOM node configuration element. The HP SiteScope Adapter alert forwarder application determines a valid monitoring target from the HP SiteScope alert text.

The HP SiteScope monitoring target is heavily dependent on the monitor class. For example, a URL monitor naturally has a URL as monitoring target, whereas a CPU monitor targets a host identified by a hostname. The different per-monitor class policies delivered with the adapter perform the actual transformation from HP SiteScope monitoring target to HPOM node target.

View a screenshot example, showing the SiS URL Monitor policy condition

The condition shown above extracts the URL from the incoming event and puts it into the variable which is used as the node parameter of the sent event. In this case, HPOM needs to have an external HPOM node set up with a node name pattern that exactly matches URL monitored. As an example, we are assuming the URL http://www.hp.com is monitored by HP SiteScope. The monitor run results are made available as HPOM events using the HP SiteScope Adapter and the policy shown in the above example.

View a screenshot example, showing the external node name pattern

Any further refinements of the mapping of the HP SiteScope monitoring target to the HPOM node are performed by the policies that come with the HP SiteScope Adapter. For example, if only the hostname part of a URL is required to be used as the node name in HPOM, the URL Monitor policy condition would extract it out of the complete URL using the HPOM policy pattern matching mechanism.

View a screenshot example, showing the policy condition that performs this task

Related Topics:

- Deploying the HP SiteScope Adapter
- Implementing HP SiteScope Adapter Alerts
- Alert Forwarding Concepts

Troubleshooting the HP SiteScope Adapter

This section contains troubleshooting tips focused on two areas:

- HPOM service discovery
  Service discovery for HP SiteScope Adapter consists of regularly scheduled collection of the status of HP SiteScope monitors and monitor groups, and providing this information to the HPOM server for presentation in service tree format.

  For details about how to verify that HP SiteScope is properly configured for service discovery, see Service Tree not Updated with HP SiteScope Monitor or Monitor Group Status .

  For information about log files, see Service Discovery Log Files .
- HP SiteScope Alert Forwarding

  For tips about troubleshooting alert forwarding problems, see No Alerts are Forwarded to the HPOM Management Server .

  Related Topics:

  - Service Tree not Updated with HP SiteScope Monitor or Monitor Group Status
  - Service Discovery Log Files
  - No Alerts are Forwarded to the HPOM Management Server

Service Tree Not Updated with HP SiteScope Monitor or Monitor Group Status

HP SiteScope Adapter service discovery runs once per hour. Therefore, it may take an hour for any changes to be reflected in the HPOM service tree.

SiteScope must be configured to provide configuration data for service discovery. Use the following steps to verify that SiteScope is properly configured for service discovery:

To verify HP SiteScope configuration

1. In the HP SiteScope user interface, click Preferences .
2. Click General Preferences .
3. In General Preferences , verify that Enable Configuration Files is selected.
4. If Enable Configuration Files is not selected, click Edit , then select Enable Configuration Files . Select OK .

Related Topics:

- Service Discovery Log Files

Service Discovery Log Files

Data in logs is available to assist in troubleshooting service discovery problems, as shown below.

Logs on the Management Server

Review the file `%OvShareDir%\logOvSVCDiscServer.log` for error messages.

Logs on Agents

Review the following log files:

- Windows Agents

  `%OvAgentDataDir%\log\javaAgent.log`
  `%OvAgentDataDir%\log\OvSvcDiscAgt.log`

- Solaris or Linux Agents

  `/var/opt/OV/log/javaAgent.log`
  `/var/opt/OV/log/OvSvcDiscAgt.log`

Manual Execution of Service Discovery

HP SiteScope Adapter gathers monitor and monitor group information from HP SiteScope with the tool sis_disc.pl. Please note that the application requires HPOM agent perl and must therefore be invoked using the provided startup script sis2om_perl.bat. This file has the same name on both Windows and UNIX managed nodes.

On Windows nodes, sis_disc.pl is stored at the following location:
`%OvAgentDataDir%\bin\instrumentation\`
On UNIX nodes, sis_disc.pl is stored at the following location:
`/var/opt/OV/bin/instrumentation/`

Logging from this tool can be done by running with the `-d` switch on the command line, as follows:
sis2om_perl.bat sis_disc.pl -d

A log file named `sis_disc.log` is created in the same directory as the `sis_disc.pl` file resides in. Review the log file for any error messages indicating errors encountered while reading the HP SiteScope configuration files.

Related Topics:

- Service Tree Not Updated with HP SiteScope Monitor or Monitor Group Status

No Alerts are Forwarded to the HPOM Management Server

If there are no alerts forwarded to the HPOM management server, check the following:

- Verify that the HP SiteScope Adapter components are correctly installed on the HP SiteScope server.
- Verify that the the matching SiS2OM policies are present and enabled on the HPOM agent policy inventory
- Verify that the start-up wrapper script `sis2om_alert.bat` (on HP Sitecope servers on Windows systems) or `sis2om_alert.sh` (on HP SiteScope servers on UNIX systems), that was created during adapter configuration, exists in the default location *`SiteScope_Installdir`* `/scripts` .
- Verify that the trigger settings of the alert actions are set appropriately. During development, it is appropriate to execute the action each time the trigger condition is met. In production environments, it is usually preferable to only execute the action once the status of the monitor changes.
- In the HP SiteScope user interface/Dashboard of the monitoring group, verify that the alert action has been performed successfully.
- Verify that there is a suitable HPOM node configuration item that the event can be mapped to. If the target set by the HP SiteScope Adapter alert forwarder application is unclear, the SiS2OM dev+test policy, located in the policy group development + test, forwards the events as they come from the forwarder, and include the target string. The policy uses this information to intercept the event, and to perform customizations if necessary.
- Verify that the incoming SiS2OM adapter events are processed by the correct policy. It is possible that other OpenMessage interface policies may intercept the incoming events. You may need to establish 'gatekeeper' conditions in existing policies (suppress on match condition).
- Enable the debug mode of the HP SiteScope Adapter by setting the DEBUG configuration item in the start-up wrapper to ON . This produces debug-relevant traces in the log file. The default location of the log file is specified in the start-up wrapper script: *`SiteScope_installdir`* `\logs\SiS2OM.log, where` *SiteScope_installdir* `is the directory where the The debug output also includes the complete opcmsg command line.`
- Use the SiS2OM dev+test policy, located in the policy group development + test, for more detailed information about the incoming events. The policy condition matches any events coming from the SiteScope Adapter alert forwarder application.

  Related Topics:

  - Alert Forwarding Concepts
  - Implementing HP SiteScope Adapter Alerts
  - HPOM Node Configuration Example

Reference Information for the HP SiteScope Adapter

This section contains reference information about the HP SiteScope Adapter:

- Message Attributes
- Variables in HP SiteScope Adapter Monitor Alert Policy Conditions
- HP SiteScope Alert Text Templates

Message Attributes

The default message attributes set by the HP SiteScope Adapter and the related HP SiteScope Monitor alert forwarding policies are as follows:

Message Attribute	Value	Description
Message Group	SiS Monitoring	The Message Group attribu statically set to 'SiS Monit
Application	SiteScope	The Message Application a statically set to 'SiteScope
Object	SiS_<SiS_MonitorClass>	The Message Object attribu SiteScope Monitor Class na with the string 'SiS_'
Severity	<normal|warning|critical>	The Message Severity is se to the SiteScope Monitor S 'good', 'warning' and 'error
Service ID	SiteScopeMonitor:<SiS_MonitorGroupName>:<SiS_GroupMonitorID>\ @@<SiS_Server_OM_PNN>	The Service ID is and color string consisting of:
		SiteScopeMonitor
		<SiS_MonitorGroupName>
		<SiS_GroupMonitorID>
		<SiS_Server_OM_PNN>

Message Key	<SiS_Server_DNSName>:<SiS_MoniTarget>:<SiS_MoniGroupName>:\ <SiS_MonitorName>:<$MSG_SEV>	The Message Key is a colo string assembled as follow	
		<SiS_Server_OM_PNN>	
		<SiS_MoniTarget>	
		<SiS_MoniGroupName>	
		<SiS_MonitorName>	
		<$MSG_SEV>	
		All Message Texts from the SiteScope Monitor Alert po	

		the same format as shown parameters substituted by processing are:
Message Text	Monitor <SiS_MonitorName> of type <SiS_ MonitorClass> for <SiS_MoniTarget> reported <SiS_MoniState>	<SiS_MonitorName>
		<SiS_ MonitorClass>
		<SiS_MoniTarget>
		<SiS_MoniState>

Related Topics:

- Variables in HP SiteScope Adapter Monitor Alert Policy Conditions
- HP SiteScope Alert Text Templates
- SiteScope Integration Policies

Variables in HP SiteScope Adapter Monitor Alert Policy Conditions

The policies of the groups per monitor class and advanced mapping for OMi utilize a uniform format message text. A common set of policy condition variables is available and is listed in the following table in case you need to make customizations. The variables are listed in the following table.

Policy Condtion Variable	Description
<sismoniname>	name of the HP SiteScope monitor that triggered the alert action
<sismonigroup>	name of the HP SiteScope monitor that triggered the alert action
<sismoniclass>	HP SiteScope Monitor Class name of the monitor that triggered the alert action
<sismonistatus>	HP SiteScope monitor execution status
<sismonitarget>	target of the HP SiteScope monitor that triggered the alert action
<sisserver>	HPOM primary pode pame of the managed node representing the HP SiteScope server
<sismonstate>	results string of the HP SiteScope monitor execution
<remains>	all information in the customizable area of the alert text template is put into this variable

SiteScope Alert Text Templates

HP SiteScope alerts are written to log files. The format of the HP SiteScope alert log files is defined by alert text templates. An alert text template is a text file containing static strings and HP SiteScope template variables. When an alert action is triggered, HP SiteScope substitutes the variables with runtime values and writes the log file.

The HP SiteScope Adapter requires that the templates have a designated header containing certain specific information. When configuring the adapter, two alert text templates are copied to the file *`SiteScope_installdir/`* `templates.script` , where *`SiteScope_installdir`* is the name of the directory where HP SiteScope is installed. The template `HPOM-SiSAlert` creates short event texts. If more information is required, the template `HPOM-SiSAlert_full` can be used. The following table shows the short event alert text template and gives an explanation of the template variables used.

Template Variable	Explanation
OM SiS Alert Template	Template identification
Monitor=<name>	SiteScope monitor name
Group=<groupID>	SiteScope monitor group
Class=<_class>	SiteScope monitor class
InternalID=<_internalId>	Internal monitor ID
Time=<time>	Alert time stamp
Severity=<category>	Alert severity
Target=<_server>;<_host>;<_hostname>;<_url>; \ <_database>;<_targetMachineName>;<_machine>; \ <_pdhMachine>;<remoteMachineName>	Used for target identification
BACMoniID=<bacMonitorID>	UCMDB ID of the monitor CI
BACSessionID=<bacSessionID>	UCMDB ID of the SiteScope profile
ServerURL=<sitescopeurl>	Server URL
no more details selected	Customizable area

Special text or other variables can be put into the customizable area and can therefore be made visible in HPOM. The customizable area can span multiple lines.

Related Topics:

- Implementing HP SiteScope Adapter Alerts

We appreciate your feedback

If an email client is configured on this computer, open an email window by clicking on the “Comments” bookmark (at left).

Otherwise, copy the information below to a web mail client, and send this email to ovdoc-asm@hp.com.

Product name: HP Operations Manager for Windows

Version: 8.16

Document title: PDF version of the Online Help

Feedback: