Orchestration Console Reference

Cloud Manager 2.1

May 9, 2012

Legal Notice

Contents

About This Guide

This *Cloud Manager Orchestration Console Reference* introduces the Orchestration Console, the basic administration environment for the Cloud Manager Orchestration Server. The guide provides an introductory overview of the Orchestration Console interface. The guide is organized as follows:

Audience

This book is intended for data center managers and IT or Operations administrators. It assumes that users of the product have the following background:

- General understanding of network operating environments and systems architecture.
- Knowledge of basic UNIX shell commands and text editors.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to www.novell.com/documentation/feedback.html (http://www.novell.com/documentation/feedback.html) and enter your comments there.

Additional Documentation

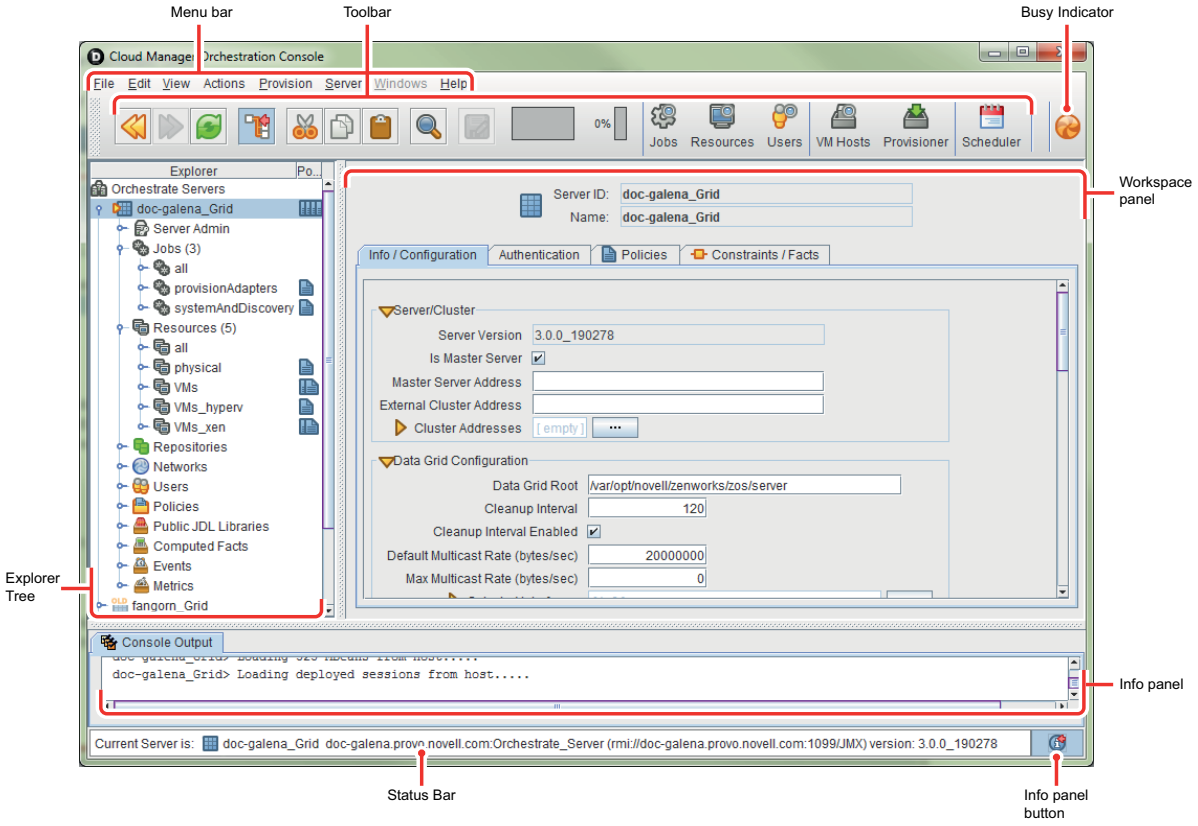
For other NetIQ Cloud Manager 2.1 documentation, see the NetIQ Cloud Manager 2.1 documentation site (http://www.novell.com/documentation/cloudmanager2).

1 Interface Layout of the Orchestration Console

Both the grid administrator and the job developer need to have access to and use the Cloud Manager Orchestration Console. The administrator needs to use the console to perform any management functions, such as creating user accounts and managing Orchestration Server activities. The developer uses the console to access the JDL editor for creating or modifying jobs and policies.

The following figure shows the general areas on the console interface that are referred to in this guide.

Figure 1-1 *The Cloud Manager Orchestration Console*



The following list describes the functional areas of the main Orchestration Console display.

- **Operations Menu bar:** The Operations Menu bar provides operations categorized under menus such as *File*, *Edit*, *View*, *Grid*, *Server*, *Windows*, and *Help*.
  - The *File* menu lets you save any changes you have made or exit the console.

- The *Edit* menu lets you cut, copy, and paste items and choose general and server preferences for the console.
- The *View* menu lets you manipulate the display of the different components of the console and refresh the Explorer and Workspace panels.
- The *Actions* menu lets you launch specific tools that create and delete users or user groups, computing resources, jobs, policies, and computed facts.
- The *Provision* menu lets you discover VM hosts and Repositories, VM Images, disks, and Repository capacities. you can also start and stop VMs and VM hosts, resync a VM's state or a VM hosts' state, reset the state of all VMs in the grid or cancel that resync, or can perform a rediscovery of all VMs in the grid.
- The *Server* menu lets you start a local server, log in to the server, create and display logs for logged-in servers, log out of the server, and shut down the server.
- The *Windows* menu lets you select console windows to display when you have more than one console window open. You can open the Explorer panel and the two tabs of the Info panel (*<Orchestrator> Log* and *Console Output*) in their own windows by right-clicking the tab and choosing *Open in window* in the pop-up menu.
- The *Help* menu provides access to the About box for the console. It also provides a link to Cloud Manager documentation on the Web.

- **Console toolbar:** The Console toolbar has buttons for executing common tasks. The basic tasks are *Go Back*, *Go Forward*, *Refresh the view*, *Hide or Show the Explorer Panel*, *Cut*, *Copy*, *Paste*, *Save changes in workspace view*, and *Open the Find Dialog*.

  The toolbar also includes buttons that open monitoring views for *Jobs*, *Resources*, and *Users*.

  To the far right of the toolbar, a pinwheel icon indicates when the console is busy.
- **Explorer tree:** Sometimes referred to as the "Tree view," this panel displays a hierarchical tree that lets you navigate to different objects; you can click items in the tree to see their details. For example, you can display computing resources for a selected grid. When you click *Computing Resources* in the tree, its details appear in the Workspace panel with a list of active computing resources. You can edit the *Computing Resource* attributes in the workspace panel.
- **Workspace panel:** This panel displays a detailed view for an item you select in the Explorer tree. For example, if you select a computing resource under physical in the Explorer tree, the Workspace panel view changes to and "Admin view" to show the details for that resource. You can edit the properties of Grid object in the views displayed in the Workspace panel.

  The views in the panel change as you select different monitoring tools (*Jobs*, *Resources*, *Users*, *VM Hosts*, *Provisioner*) or the *Scheduler* tool in the Main toolbar.
- **Info panel:** The Info panel displays a variety of information, such as validation and error messages, log files, and query results. You can display or hide the Info panel by clicking the *Info panel* icon in the Status bar.
- **Status bar:** The status bar displays general identity information about the Orchestration Server where you are logged in.

For information about launching the console and using it for the first time, see "Launching the Orchestration Console and Logging in to the Orchestration Server"in the *NetIQ Cloud Manager 2.1 Orchestration Installation Guide*.

For more information about the user interface of the Orchestration Console, see Chapter 2, "Orchestration Console Menus and Tools," on page 13.

2 Orchestration Console Menus and Tools

A number of operations are available from the Cloud Manager Orchestration Console and can be accessed from its menu bar and toolbar.

2.1 Operations Menu Bar

The Operations Menu Bar in the Orchestration Console provides options that help you to create and administer objects in the Explorer Tree.

2.1.1 File

The File menu (Alt+F) provides keyboard and mouse accessible methods for users to save changes or to exit the application.

Save

The Save operation provides a keyboard (File > Ctrl+S) and mouse-accessible method for users to save any changes made in the visible view.

Exit

The exit operation provides a keyboard (File > Alt+X) and mouse-accessible method for users to close all server connections and to exit the Orchestration Console application.

2.1.2 Edit

The Edit menu (Alt+E) provides keyboard and mouse-accessible methods for users to save changes or to exit the application.

Undo Addition

The Undo operation provides a mouse-accessible method for users to undo the action they have just performed in the Orchestration Console. The operation can also be executed from the keyboard (Ctrl+Z).

Redo

The Redo operation provides a mouse-accessible method for users to redo the action they have just performed in the Orchestration Console. The operation can also be executed from the keyboard (Ctrl+Y).

Cut

The Cut operation provides a mouse-accessible method for users to cut the selected object and move it to the clipboard. The operation can also be executed from the keyboard (Ctrl+X).

Copy

The Copy operation provides a mouse-accessible method for users to copy the selected object to the clipboard. The operation can also be executed from the keyboard (Ctrl+C).

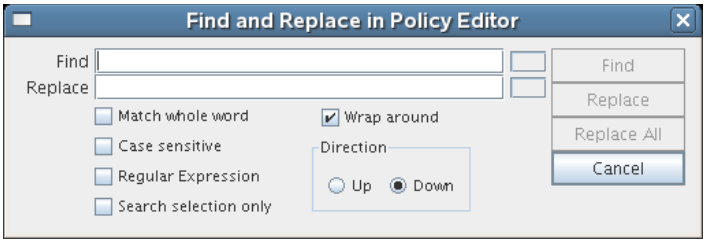
Paste

The Paste operation provides a mouse-accessible method for users to paste the contents of the clipboard to the desired location. The operation can also be executed from the keyboard (Ctrl+V).

Find

The Exit operation provides a mouse-accessible method for users to open the Find and Replace dialog box, where they can search for and replace (if necessary) editable strings located in logs and editing views (for example, the Policy Editor).

Figure 2-1 *The Find and Replace Dialog Box Invoked From the Policy Editor*



The operation can also be executed from the keyboard (Ctrl+F).

Find Next

The Find Next operation provides a mouse-accessible method for users to find the next occurrence of the string they previously searched for. The operation can also be executed from the keyboard (F3).

Find Previous

The Find Previous operation provides a mouse-accessible method for users to find the previous occurrence of the string they searched for. The operation can also be executed from the keyboard (Shift+F3).

Enter Find String

The Enter Find String operation provides a mouse-accessible method for users to load the text of the string they want to search for. The operation can also be executed from the keyboard (Ctrl+E).

Load Text

The Load Text operation provides a method for users to load text from an existing file into the open, editable view. When selected, the operation opens a browse dialog box where the file can be selected.

Save Text

The Save Text operation provides a method for users to save text in an editable, active view to a file. When selected, the operation opens a Save dialog box where you can browse to a network location where you want to save the file. By default, the file is named according to the view and the context within which you are viewing it. You can change the name of the file when you save it.

Preferences

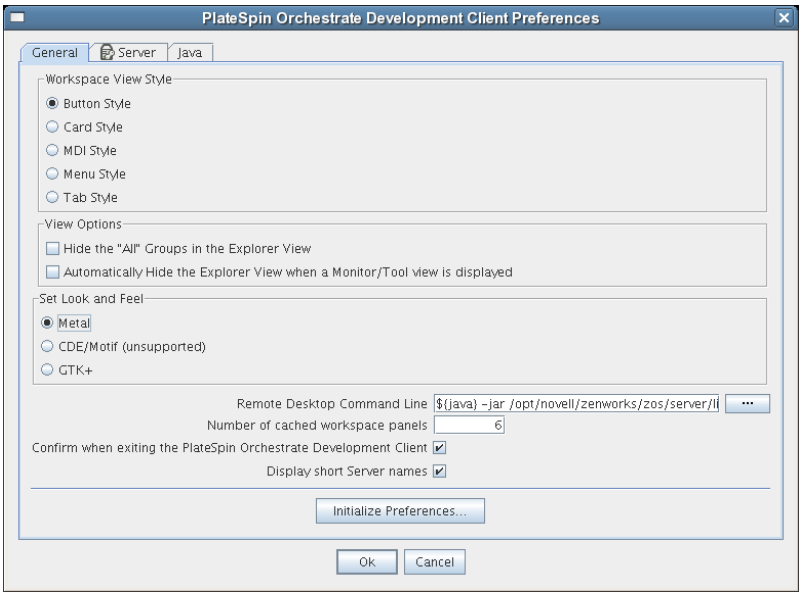
The Preferences operation provides a method for users to change the preferences for the Orchestration Console display. When selected, the operation opens the Orchestration Console Preferences dialog box.

The dialog box has three tabbed pages

- "General Page" on page 16
- "Server Page" on page 17
- "Java Properties Page" on page 17

General Page

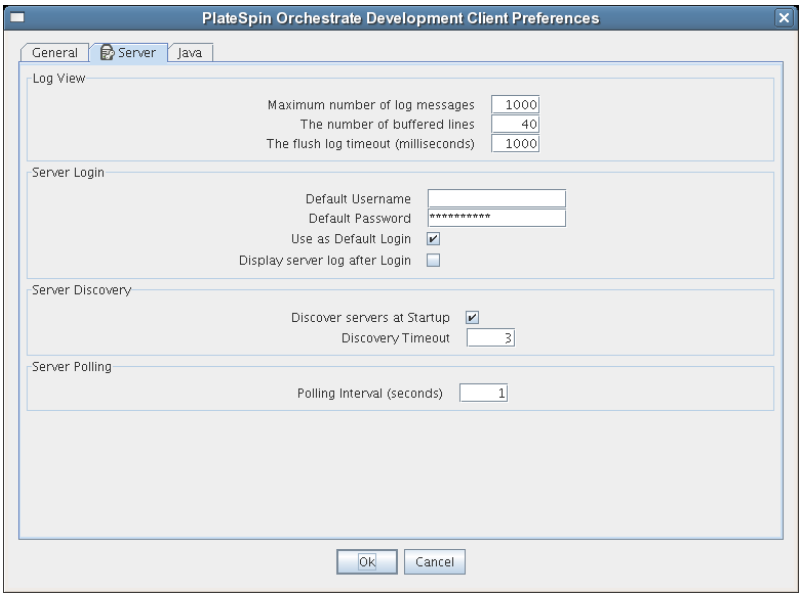
Figure 2-2 *General Page of the Orchestration Console Preferences*



Preference settings that you can change on this page are self-explanatory. If you click *Initialize Preferences*, the preference settings (except *Look and Feel* settings) are returned to installation values.

Server Page

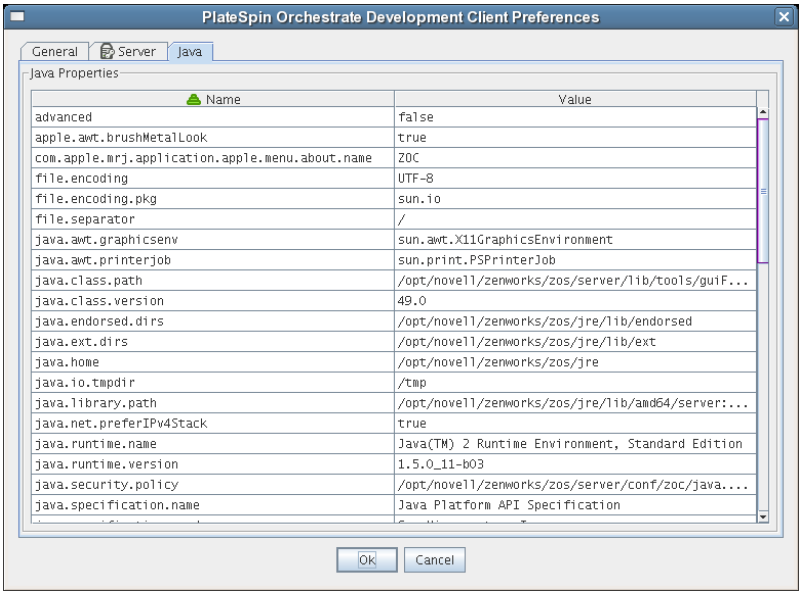
Figure 2-3 *Server Page of the Orchestration Console Preferences*



Preference settings that you can change on this page are self-explanatory.

Java Properties Page

Figure 2-4 *The Java Properties Page of the Orchestration Console Preferences*



This page lists the Java property names and values that are used to render the Orchestration Console interface in Java Swing. The list is for your information only.

2.1.3 View

The *View* menu includes various operations that let you manipulate the Orchestration Console display of the various Orchestration component views. The function of the options under this menu are self-explanatory, and are a compilation of view operations that are also available from the Operations toolbar.

For more information about the View operations, see Section 2.2, "Orchestration Console Toolbar," on page 23.

2.1.4 Actions

The multiple operations listed as options under the *Actions* menu provide a quick way for you to perform operations that can also be performed (generally by right-clicking an object) in the Explorer View.

For example, if you select a *Create* option from the Actions menu, the Create dialog box remains open after you create each object. You can repeatedly create new objects in the dialog box, pressing *OK* or *Create* after each is created. Similarly, in the dialog boxes of some operations in the Actions menu, you can select many objects and delete them at the same time.

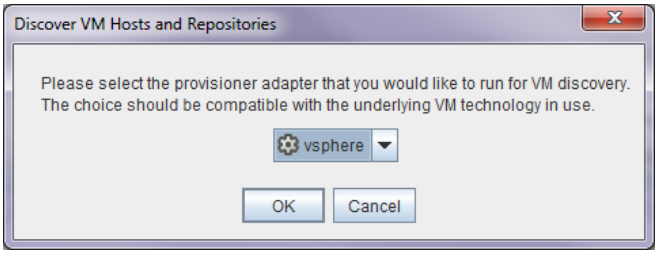
2.1.5 Provision

The *Provision* menu includes provisioning actions that you can perform on multiple resources simultaneously. Many of the operations listed in the menu include some of the provisioning actions that you can execute by right-clicking a single VM or VM Host object in the Explorer Tree.

Discover VM Hosts & Repositories

When you select this option, the Discover VM Hosts and Repositories dialog box is displayed.

Figure 2-5 *VM Discovery Dialog Box*

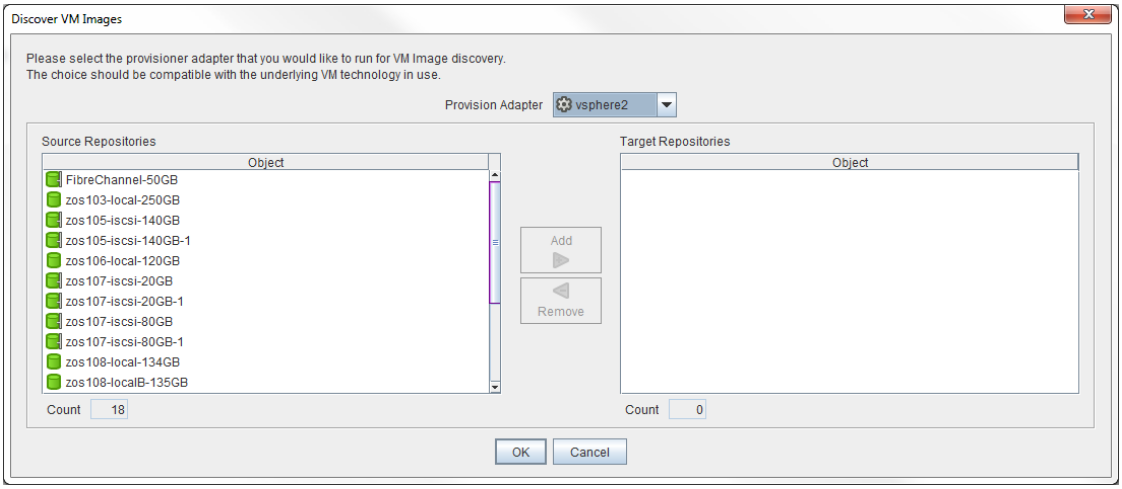


You use this dialog box to select a provisioning adapter (hyperv, vsphere, kvm, xen, xenserver) that discovers all VM host machines where the Cloud Manager Orchestration Agent is installed and creates objects in the model. The provisioning adapter also discovers the VM Repositories where VM hosts reside.

Discover VM Images

When you select the *Discover VM Images* menu option, the Discover VM Images dialog box is displayed.

Figure 2-6 *Discover VM Images Dialog Box*

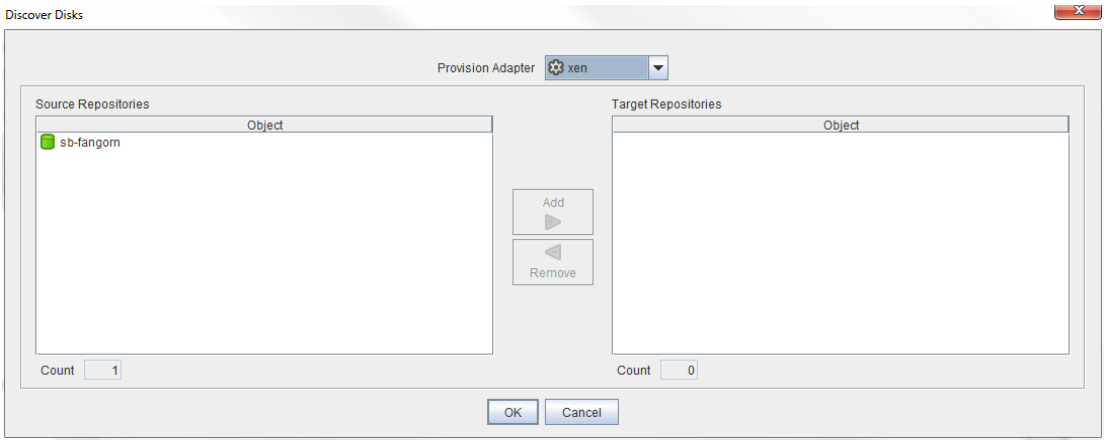


You use this dialog box to select a provisioning adapter (hyperv, kvm, vsphere2, xen, or xenserver) that discovers all of its associated VM images and creates objects in the model.

Discover Disks

When you select the *Discover Disks* menu option, the Discover Disks dialog box is displayed.

Figure 2-7 *Discover Disks Dialog Box*

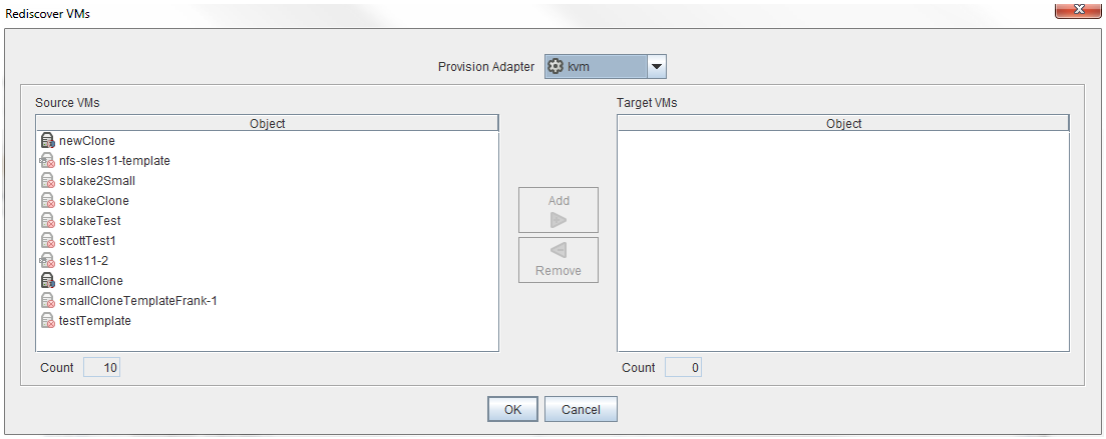


Using this dialog box, you can select a provisioning adapter that discovers all of its associated source repositories and the vDisks on those repositories.

Rediscover VMs

When you select the Discover Disks menu option, the Discover Disks dialog box is displayed.

Figure 2-8 *Rediscover VMs Dialog Box*



You use this dialog box to select a provisioning adapter that rediscovers the configurations of all of its associated VMs. This can occur when the native hypervisor is used to reconfigure a VM. Using this operation, you can reconfigure the Orchestration Server facts to match the configuration of the hypervisor facts.

A VM is not listed in the dialog box if the Cloud Manager Orchestration Server cannot rediscover its facts.

Other Provisioning Operations

The other operations listed in the menu are self-explanatory. Remember that all VMs

- *Start VM Hosts*
- *Shutdown VM Hosts*
- *Shutdown VMs*
- *Resync VMs' State*
- *Resync VM Host's State*
- *Reset State of all VMs*
- *Cancel Resync of all VMs*

2.1.6 Server

The *Server* menu lets you start a local server, log in to the server, create and display logs for logged in servers, log out from the server, and shut down a server.

- "Select Server" on page 21
- "Discover Servers" on page 21
- "Shutdown Server" on page 21
- "Login" on page 21
- "Logout" on page 21
- "Display Log" on page 21
- "Create Custom Log" on page 22

Select Server

The *Select Server* operation lets you select one of the Orchestration Servers in your grid to log in to. When you select a server, you are required to log in. This operation accomplishes the same thing as selecting a server object from the Explorer Tree.

Discover Servers

The *Discover Servers* operation lets you launch the discovery process for servers. This is the same process that initiates (if so chosen in your server preferences) when the Orchestration Console starts.

Shutdown Server

The *Shutdown Server* operation lets you shut down the current, logged-in Orchestration Server. The shutdown dialog box also lets you create a snapshot of the server state when you shut down.

Figure 2-9 *The Server Shutdown Dialog Box*

Login

The *Login* operation lets you establish a remote connection to another Orchestration Server. The server IP address is required for the login. When you enter the IP address, you need to provide the username and password for the server where you are logging on.

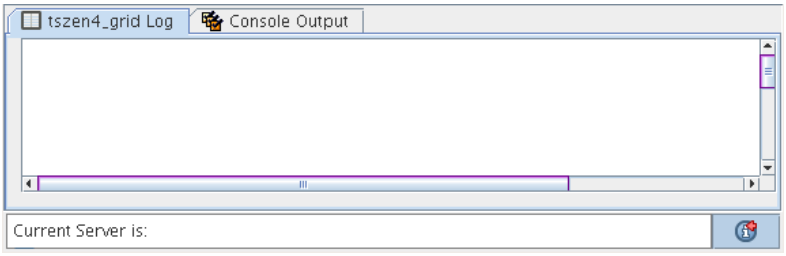
Logout

The *Logout* operation lets you log out of the current, logged-inn Orchestration Server without exiting the Orchestration Console. Logging out removes the server's nodes from the Explorer Tree and its workspace views.

Display Log

The *Display Log* operation displays the default server log for the current, logged-inn Orchestration Server. The display is in the Information window located at the bottom of the Orchestration Console. The server log file is also located by default in the `/var/opt/novell/zenworks/zos/server/logs` directory.

Figure 2-10 *Server Log Opened in Information Window of the Orchestration Console*



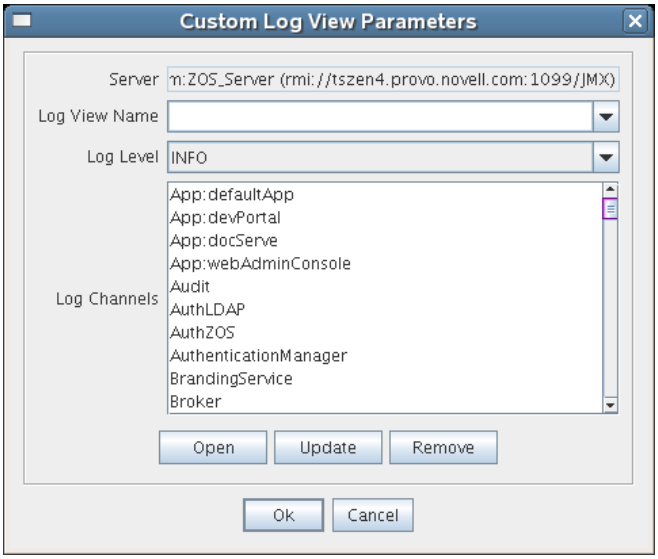
When a log is displayed, you can right-click its tab to further direct the actions of the display. You can pause logging in the window, copy the log to the clipboard, clear its contents, undock the log display as a new window, or remove it from the Information window.

If you right-click on the log display, all of the default editing capabilities of the Orchestration Console are available for your use inside the window. For more information, see Section 2.1.2, "Edit," on page 14.

Create Custom Log

The *Create Custom Log* operation opens the Custom Log View Parameters dialog box.

Figure 2-11 *The Custom Log View Parameters Dialog Box*



By enabling a custom log, you can monitor various components of the Orchestration Server. For example, you can view debugging information for the Audit facility. You can create, update, or remove a log view from the dialog box. You can open a custom view in the Information window by selecting *Open* in the dialog box.

Log View Name: Provide the name of the log view. This is displayed on a tab in the log display panel.

NOTE: You can enter only alphanumeric characters and spaces in the *Log View Name* field.

Log Level: From the drop-down list, select the minimum log level for the log view. The log messages included in the custom view will be of this level and above.

Log Channels: A log channel provides log information specific to an Orchestration Server component or facility, such as the Audit facility.

When the custom view is displayed, you can right-click its tab to further direct the actions of the display. You can pause logging in the window, copy the log to the clipboard, clear its contents, undock the log display as a new window, or remove it from the Information window.

If you right-click on the log display, all of the default editing capabilities of the Orchestration Console are available for your use inside the window. For more information, see Section 2.1.2, "Edit," on page 14.

2.1.7 Windows

When you right-click various views and panels in the Orchestration Console, you can select the *Open in Window* option to open these views and panels in separate windows. This allows you the perspective you sometimes need when working with Orchestration Server objects in conjunction with one another. The *Windows* menu lets you toggle between the various views or panels that are open. You can also choose to *Show All*, *Hide All*, or *Close All* of these windows.

When a window is open, its fields and selectable dialog boxes remain functional so that you can perform object operations or text editing as you would when these views or panels are docked normally to the Orchestration Console.

2.1.8 Help

From the *Help* menu, you can access a link to the online Cloud Manager documentation (available in `.html` or `.pdf` format) or you can open the About box for the product, where you can view its version number, its license expiration date, and a list of its current management pack capabilities (for example, the Virtual Machine Management capability).

2.2 Orchestration Console Toolbar

The Orchestration Console Toolbar includes several buttons that let you perform command tasks in the Orchestration Console workspace views and the Explorer Tree. The table below lists the functions of these buttons.

Table 2-1 *Tool Buttons from the Orchestration Console Toolbar*

Button	Tool Name	Tool Function
	Back	Go back to the previous workspace view seen.
	Forward	Go forward to the next workspace view.
	Refresh	Refresh the Explorer and Workspace views.

Button	Tool Name	Tool Function
	Open/Hide Explorer	Open the Explorer Tree in a window Hide the Explorer window
	Cut	Cut the selected object from the workspace and copy it to the clipboard
	Copy	Copy the selected object to the clipboard while keeping the original in place
	Paste	Paste the contents of the clipboard
	Find and Replace	Open the Find dialog box
	Save	Save changes (in the workspace views or in the Explorer)
	Resource Usage Meter	(Not an active button) visual indication of resource usage. Mouse over for a listing of Active Resources, Busy resources and Available Resources, right-click to stop the meter
none (blank area)	Bookmark Toolbox	Click and drag any object from the Explorer tree into this area to create a bookmark to jump to that object's view. Right-click the bookmark to select options to open and show the object or to remove it from the toolbox. Right-click to remove all objects when some are not visible.
	Busy Indicator	This pinwheel shape appears to rotate when the Server is busy performing an operation.

3 The Orchestration Server and the Server Admin Objects

The Orchestration Console lets you visualize the object model maintained by the Orchestration Server and that the server uses to make resource provisioning decisions. The left pane of the Orchestration Console displays a hierarchical tree known as the Explorer Tree or the Explorer View. This tree lets you navigate to different objects to see their details. Each object in the Explorer Tree is referred to as a "Grid object." These objects can also be associated with one or more containers called Groups. When you navigate to these objects, you can edit their attributes and view more detail about their configurations.

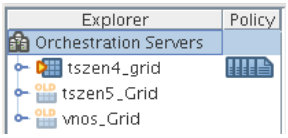
This section includes information about the following objects that you can manage in the Explorer Tree:

3.1 Orchestration Server Object

The highest object in the Explorer Tree is the Orchestration Server Object, sometimes called the Grid Server object because it represents the Orchestration Server acting as the holding place for all of the information used to manage objects for a single computing grid.

The Orchestration Console is version aware. When the console launches or when server discovery is manually run, the console recognizes both current Orchestration Server installations and old installations of discovered servers and displays their icons accordingly. This visual cue helps you to recognize when older Orchestration Servers need to be upgraded.

Figure 3-1 *Current and "Old" Server Objects*



The tool tip for an Orchestration Server lists its RMI configuration, its IP address, the directory location where the server instance was installed, and its exact version number.

The icons to the right of a current Orchestration Server represent its policies, either those added by default upon server install and configuration, or those added later. A drop-down menu of all associated policies is opened when you right-click a policy icon. From there, you can select a policy to open in the Policy Editor. For more information about policies, see Section 12.1, "Policy Object," on page 153.

When selected, the Server object exposes four tabs where you can further configure its attributes. Further information about these tabs is available in the following sections:

- Section 3.1.1, "The Orchestration Server Info/Configuration Page," on page 26
- Section 3.1.2, "Orchestration Server Authentication Page," on page 34
- Section 3.1.3, "Orchestration Server Policies Page," on page 38
- Section 3.1.4, "Orchestration Server Constraints/Facts Page," on page 38

3.1.1 The Orchestration Server Info/Configuration Page

The page that opens under the *Info/Configuration* tab includes several collapsible sections on the page where you can configure the general information and attributes of the server.

Server/Cluster Panel

If you are using this server in a High Availability environment, the information in this section is populated as a result of the configuration you managed during the High Availability installation. The following items are included in the section:

- Server Version
- Is Master Server
- Master Server Address
- External Cluster Address
- Cluster Addresses

Server Version: A non-editable field that lists the version of this server in the form `<major>.<minor>.<point>.<build_number>`. This is the data for the `matrix.version` fact.

Is Master Server: A check box that is selected if the server is the Master Server in a High Availability cluster configuration.

Master Server Address: Set this value when the Orchestration Server participates in a High Availability cluster.

External Cluster Address: Set this value when the Orchestration Server participates in a High Availability cluster.

Cluster Addresses: Shows the hostname or IP addresses associated with an Orchestration Server when it is in a High Availability configuration.

The  button opens the Attribute Element Values dialog box, where you can add, remove, or reorder addresses (element values) in an array of address choices.

For more information about using Cloud Manager Orchestration components in a High Availability environment, see the *NetIQ Cloud Manager 2.1 Orchestration Server High Availability Configuration Guide*.

Data Grid Configuration Panel

This section of the *Info/Configuration* tab allows for advanced configuration of datagrid related tuning parameters. The properties on the page and their descriptions are listed below.

- Data Grid Root
- Cleanup Interval
- Cleanup Interval Enabled
- Default Multicast Rate
- Selected Interfaces
- Available Interfaces
- Total Packets Sent
- Total Packets Resent
- Total Resend Rate
- Current Packets Sent
- Current Packets Resent
- Current Resend Rate
- Current File Size
- Current Bytes Sent
- Current Percent Complete
- Skipped (Sparse) Bytes
- Current Receiver Count
- Current File Name

Data Grid Root: The location of the Orchestration Server datagrid in the file system. For example, you might change this location to use a different file system mount point (recommended when there is considerable datagrid I/O).

Cleanup Interval: The interval at which the Orchestration Server scans User job history files on the datagrid. Job history files older than the owning user's job history retention time limit (`user.datagrid.maxhistory`) are deleted.

Cleanup Interval Enabled: Select this check box to set a flag to enable periodic job history cleanup checking. Deselect it to disable the checking.

Default Multicast Rate: Sets the default data rate in bytes per second for multicast operations in which the client has not explicitly set a rate for a particular file transfer.

Max Multicast Rate: The maximum data rate (in bytes per second) that a client can specify for a multicast file transfer.

Selected Interfaces: The interfaces on which multicast file transfers are to be sent. This allows an administrator to limit multicast traffic to specific interfaces (that is, the interfaces where the agents are connected). You can add or delete interfaces by clicking the ··· button.

Available Interfaces: Lists the network interfaces that are available on the local machine for multicasting.

The property is read-only and is provided for your information.

Multicast Metrics Subpanel: This panel lets you monitor multicast data transfer, including:

- **Total Packets Sent:** The total number of multicast data packets sent by the file multicaster since the last reset of the counters.
- **Total Packets Resent:** The total number of multicast packets resent because of errors since the last counter reset.
- **Total Resend Rate:** The total packet resend rate as a percentage since the last counter reset.
- **Current Packets Sent:** The total number of multicast packets sent during the current or most recent multicast file transfer.
- **Current Packets Resent:** The total number of multicast packets resent because of errors, corruption, or loss during the current or most recent multicast file transfer.
- **Current Resend Rate:** The packet resend rate as a percentage of packets sent since the start of the current or most recent multicast file transfer.
- **Current File Size:** The file size in bytes for the current or most recent multicast file transfer.
- **Current Bytes Sent:** The number of bytes sent so far in the current or most recent multicast file transfer.
- **Current Percent Complete:** The completion percentage of the current or most recent multicast file transfer.
- **Skipped (Sparse) Bytes:** The number of bytes skipped because of long runs of zeros. These "holes" are skipped in order to reduce file transfer time for large sparse files like VM images.
- **Current Receiver Count:** The number of recipient agents for the current or most recent multicast file transfer.
- **Current File Name:** The name of the file transferred in the current or most recent multicast file transfer.

The data list includes a check box that is selected if the current multicast transfer is finished. It also includes a *Reset Stats* button that you can click to clear all of the metrics in order to begin monitoring multicast statistics from a new point in time.

Security/TLS Configuration Panel

This section lets you configure TLS or SSL data encryption for both user and agent connections. There are four different levels of encryption that can be set for both users and nodes. These are described below. The properties in this section also let you configure the TCP/IP socket listener address and port for TLS connections.

- TLS On Agent
- Forbid TLS for agents
- Allow TLS on the agents; default to falling back to unencrypted
- Allow TLS on the agents; default to TLS encrypted if not configured encrypted
- Make TLS mandatory on the agents
- TLS On Client
- Forbid TLS for clients

- Allow TLS on clients; default to falling back to unencrypted
- Allow TLS on agents; default to TLS encrypted if not configured encrypted
- Make TLS mandatory on the clients
- TLS Address

TLS On Agent: Allows the encryption level to be set to one of four values, as described (in order of security level) below:

- **Forbid TLS for agents:** Only unencrypted connections are allowed for nodes (that is, agents) authenticating to this server. If the agent attempts to initiate encrypted communication, the connection attempt is rejected. This is the least secure of the encryption levels and is only recommended for installations where encryption is forbidden because of legal or policy restrictions, or where the performance benefits of disabling encryption outweigh security concerns.
- **Allow TLS on the agents: default to falling back to unencrypted:** Specifies that the server defaults to unencrypted communication, but the agent can optionally enable encryption.

  This is the default setting for the Orchestration Server. More secure installations might require a setting to one of the higher levels below.
- **Allow TLS on the agents; default to TLS encrypted if not configured encrypted:** The server defaults to using encryption, but the agent can optionally disable encryption.
- **Make TLS mandatory on the agents:** The Orchestration Server rejects any connections that do not establish TLS encryption. This is the most secure encryption level because it ensures that all message communication between the node (that is, an agent) and the server is protected from tampering or interception.

TLS On Client: This setting allows the encryption level to be set to one of four values, as described (in order of security level) below.

- **Forbid TLS for clients:** Only unencrypted connections are allowed for users of this server. If the user or client attempts to initiate encrypted communication, the connection attempt is rejected. This is the least secure of the encryption levels and is only recommended for installations where encryption is forbidden because of legal or policy restrictions, or where the performance benefits of disabling encryption outweigh security concerns.
- **Allow TLS on clients; default to falling back to unencrypted:** This level specifies that the server defaults to unencrypted communication, but that the user can optionally enable encryption.

  This is the default setting for the Orchestration Server. More secure installations might require a setting to one of the higher levels below.
- **Allow TLS on agents; default to TLS encrypted if not configured encrypted:** The server defaults to using encryption, but the user can optionally disable encryption.
- **Make TLS mandatory on the clients:** The Orchestration Server rejects any connections that do not establish TLS encryption. This is the most secure encryption level because it ensures that all message communication between the user's client programs and the server is protected from tampering or interception.

TLS Address: The port number and optional bind address for incoming encrypted connections from users and nodes. The format is `hostname:port`. For example, `10.10.10.10:8101` causes the server to accept only TLS connections on the address `10.10.10.10` on port `8101`. If "*" is used as the hostname, then the Orchestration Server listens on all available network interfaces. The default is `*:8101`, which causes the Orchestration Server to listen for encrypted sessions on all available interfaces on the system.

Agent/User Session Configuration Panel

When nodes (agents) and users log on to the Orchestration Server, they establish a session context that is used to manage the state of the messaging connection between client and server. This session can be revoked by the administrator, and it can also expire if the connection exceeds its maximum lifetime or idle timeout.

- Agent Session Lifetime
- Agent Session Timeout
- Socket Keeps Agent Sessions Alive
- User Session Lifetime
- User Session Timeout
- Socket Keeps User Sessions Alive

Agent Session Lifetime: The maximum number of seconds that an agent's session can last before the agent is disconnected and must re-authenticate with the server. A value of `-1` means "forever."

Agent Session Timeout: The idle timeout for agents. If an agent connection remains idle with no message traffic in either direction for this time period (in seconds), the session times out, and the agent is disconnected and must reauthenticate when it is ready to communicate with the server again.

Socket Keeps Agent Sessions Alive: Select this check box to set a flag that causes the server and agent to maintain a keepalive ping in order to detect hung/stalled network connections. This allows the agent to recover from hung connections and to transparently reconnect with the server.

User Session Lifetime: The maximum number of seconds that a user's session can last before the user is required to re-authenticate with the server. A value of `-1` means "forever."

User Session Timeout: The idle timeout (in seconds) for user sessions. If a user's session encounters no message traffic or requests in either direction for this amount of time, any connection with user software is closed and the session expires. At this point, the user must re-authenticate.

Socket Keeps User Sessions Alive: Select this check box to set a flag that causes the server and user client to maintain a keepalive ping in order to detect hung/stalled network connections. This allows the agent to recover from hung connections and to transparently reconnect with the server. This setting applies only in situations where you are using custom user client software or certain subcommands of the zos command line utility to maintain a persistent connection.

Audit Database Configuration Panel

This section of the Info/Configuration page lets you configure the connection to a relational database that uses a deployed JDBC driver and connection properties. The PostgreSQL driver is deployed by default.

- JDBC Driver Name
- JDBC Library
- JDBC Connection URL
- Database Username
- Database Password
- Is Connected

- Connect
- Disconnect
- Clear Queue

JDBC Driver Name: Specifies the Java class for the driver.

JDBC Library: Specifies the deployed library that contains the driver.

JDBC Connection URL: Specifies the driver-dependent connection string.

Database Username: Specifies the username for database authentication.

Database Password: Specifies the password to be used for database authentication.

Is Connected: Indicates that the driver is successfully connected.

Connect (button): Click to connect through the current connection settings.

Disconnect (button): Click to disconnect the current connection.

Clear Queue (button): Clear queued records that have not yet been written to the database.

Sentinel Server Configuration Panel

This section of the Info/Configuration page lets you configure the values needed to connect to a deployed Novell Sentinel Event Source Server, where logging events from the Orchestration Server are collected, parsed, and mapped for prioritization and subsequent administrator analysis.

For information about setting up a Sentinel Collector Server in your Orchestration Server environment, see "Integrating the Orchestration Server with a Sentinel Collector" in the *NetIQ Cloud Manager 2.1 Orchestration Administrator Reference*.

The following fields are available in the *Sentinel Server Configuration* panel:

- Server Hostname
- Server Port Number
- Is Connected
- Log Channels
- Connect
- Disconnect
- Configure

Server Hostname: Specify the hostname of the Sentinel Event Source Server where log messages are to be sent.

Server Port Number: Specify the port number on the Sentinel Event Source Server where the Orchestration Server should make its SSL connection.

Is Connected: Selected when the connection between the Orchestration Server and the Sentinel Event Source Server is established.

Log Channels: Lists the log channels from which log messages are to sent to the Sentinel server.

Connect (button): Click to connect to the Sentinel Event Source Server. When the SSL connection is made, the Orchestration Server begins to send its log messages to Sentinel.

Disconnect (button): Click to disconnect the Orchestration Server from the Sentinel server. When the connection ends, log messages are no longer sent to the Sentinel server.

Configure (button): Click to open the Sentinel Log Parameters dialog box. In this dialog box, you can map a log level to one or more log channels. These log channels send log messages to the Sentinel server.

For more information about Orchestration Server log levels, see "Orchestration Server Log Levels Mapped to Sentinel Log Levels" in the *NetIQ Cloud Manager 2.1 Orchestration Administrator Reference*.

NOTE: To select multiple log channels, press Ctrl while selecting the log channel options you want. You can select only one log level at a time for mapping log channels.

The following table shows the log channels you can choose and the Orchestration Server actions that prompt sending a log message through this channel.

Table 3-1 *Log Channels and the Occasions for Sending Messages Through Each*

Log Channel Name in the Orchestration Console (Sentinel Server Configuration Panel)	When Are Messages Sent to This Channel?
ActionStatusManager	◆ When the status of a Grid action is updated.
Audit	◆ When the Grid interacts with the audit database.
AuthLDAP *AuthZOS* *AuthenticationManager*	◆ On grid-wide authentication events.

Broker	◆ On job execution.
	  ◆ start
	  ◆ cancel

Event Manager *JobManager* *NodeManager* *UserManager* *repositoryManager* *vbridgeManager* *vdiskManager* *vnicManager*	◆ When a Grid object of the corresponding type is created, deleted, or its health changes to a bad state.
GroupManager	◆ When a member is added/removed in a Group.
JobScheduler	◆ When the job schedule or the job trigger deployment/ undeployment.
MBeanServer	◆ When internal Grid Resources are updated.
PolicyManager	◆ On policy creation/deletion. ◆ On policy association/disassociation with any Grid object.
Sentinel	◆ When the Grid interacts with a Novell Sentinel server.
SessionManager	◆ On user or Resource login/logout.

Log Channel Name in the Orchestration Console (Sentinel Server Configuration Panel)	When Are Messages Sent to This Channel?
VmManager	◆ When actions are performed on VMs (provision, migrate, shutdown, clone, etc.). This could be initiated automatically or manually. ◆ When authorization fails during VM operation. ◆ When provisioning job fails.
computedFact	◆ When computed facts are created or updated or deleted.
deployer/computedFact *deployer/event* *deployer/facility* *deployer/jdlLibrary* *deployer/job* *deployer/library* *deployer/metric* *deployer/policy* *deployer/properties* *deployer/schedule* *deployer/service* *deployer/trigger* *deployer/xml*	◆ When a corresponding resource is deployed to or undeployed from the Grid.

Job Limits Panel

The facts in this section of the page are used in the default constraints to help protect the Orchestration Server from denial-of-service attacks or badly written jobs that might otherwise get stuck in the server queue, consume resources, and cause adverse server performance.

The following fields are available in the *job.limits* panel:

- max.active.jobs
- max.queued.jobs
- job.finishing.timeout

max.active.jobs: Sets a global default limit on the number of active jobs.

The Orchestration Server uses this value in the `start` constraint and does not allow more than this number of jobs (including child jobs) to be actively running at the same time. Jobs that exceed this number might be queued. See `max.queued.jobs`.

max.queued.jobs: Sets a global default limit on the number of queued jobs.

This value is similar to `max.active.jobs` but it is used in the `accept` constraint to limit the number of jobs in a queue waiting to be started. Therefore, the maximum jobs that can be present on an Orchestration Server is `max.active.jobs` + `max.queued.jobs`. New jobs are not accepted by the server if they exceed this total.

job.finishing.timeout: Sets a global default limit on the timeout for job completion.

This value represents the number of seconds that the Orchestration Server allows a job to execute its job_cancelled_event() (if defined) before forcibly canceling the job. This prevents jobs from potentially hanging during cancellation.

3.1.2 Orchestration Server Authentication Page

The *Authentication* tab opens a page with several collapsible sections where you can configure various methods for authenticating both users and resources to the Orchestration Server.

- "Resources Panel" on page 34
- "Users Panel" on page 34
- "Authentication Page" on page 37

Resources Panel

The resources in a Orchestration Server grid are actually Orchestration Agents that authenticate or "register" with the Orchestration Server.

Auto Register Agents: Select this check box if you want the Orchestration Server to automatically register agents when they first connect to the Orchestration Server.

Auto Upgrade Agents: This check box is already selected if you chose to enable the automatic upgrade of Orchestration Agents that communicate with your Orchestration Server. If you did not select this option during upgrade configuration, the check box is not selected.

If you select this check box, the associated Orchestration Agents are upgraded at intervals over a period of approximately five minutes. The upgrade happens without administrator approval.

If you deselect this check box at any time before or during the automatic upgrade, the upgrade process stops. Any agents that are not upgraded continue to be identified as "OLD" and are not useable with the newly upgraded server.

Users Panel

Only authenticated users can log into the Orchestration Server. As an administrator, you can configure this authentication to use an internal user database or to externally authenticate users through an LDAP server.

Auto Register Users: Select this check box if you want the Orchestration Server to automatically register users when they first connect to the Orchestration Server.

Enable LDAP Subpanel

Depending on the selections you make in this subpanel, the following settings are displayed:

- Enable LDAP
- Administrators
- Server Type
- Settings Subpanel
  - Directory Name
  - Servers

- Advanced
  - SSL
  - Start TLS
  - Query Account
  - Query Password
- Generic Settings Subpanel
  - Base Domain Name
  - User Attribute
  - User Filter
  - User Prefix
  - Group Attribute
  - Group Filter
  - Group Prefix
  - Group DNA Attribute
  - Nested DNA Attribute

Enable LDAP (Check Box): Select this check box if you want the Orchestration Server to authenticate users externally by using an LDAP server. Additional LDAP-related configuration fields are displayed when you select the check box:

Administrators: The Administrators list specifies the group names whose membership includes Orchestration Server administrators as returned by the specified authentication provider. You can add groups to this list by clicking the ... button to open an array editor dialog box, which allows groups to be added, removed, and reordered. A group must be in the format `<provider>:<group|groupnocase>:<groupname>`, where the `<provider>` is either ZOS or LDAP. For example, adding `LDAP:groupnocase:XyZ` allows users reported by the LDAP server as members of a group xyz, or XYZ, xYz, etc. to authenticate as an administrator. To enforce to case-sensitive matching, use `LDAP:group:XyZ` instead. Non-case-sensitive matching is needed for Active Directory servers.

Server Type: This drop-down list lets you specify which authentication provider you want to use: *Active Directory Service* or *Generic LDAP Directory Service*.

- If you select *Active Directory Service*, specify the values in the *Settings* subpanel only.
- If you select *Generic LDAP Directory Service*, specify the values in the *Settings* subpanel (except *Advanced* settings) and the values in the *Generic Settings* subpanel.

Settings Subpanel: Set the values in this subpanel for the ADS authentication provider.

- **Directory Name:** The name of the Active Directory Service server.
- **Servers:** A list of strings containing `server:port` entries for a list of servers to be used.

  Each entry can be of one of three forms:

  - `<hostname>`
  - `<hostname>:<port>`
  - `<hostname>:<port>:<sslport>`

  In all cases, `<hostname>` is a resolvable DNS name or an IP address. If SSL or TLS is in use, the hostname must exactly match the name on the ADS server SSL certificate.

You can modify this list by clicking the ··· button to open an Attribute Element Values dialog box, where you can add, remove, or change the order of server names.

- **Advanced:** The settings in this subpanel are for more selective ADS authentication.
  - **SSL:** If the accompanying *Start TLS* check box is not selected and if the ADS server's SSL certificate has been installed on the Orchestration Server JVM, this option securely connects to the ADS server through SSL encryption.

    The older LDAP protocol (`ldaps://`) is used for the connection.
  - **Start TLS:** Selecting this option immediately promotes the connection to SSL encryption by bypassing the older protocol in favor of the LDAPv3 Start TLS extended operation on the non-SSL LDAP port. To use this option, the ADS server's SSL certificate must be installed on the JVM of the Orchestration Server.
  - **Query Account:** The account name that is to be used for querying group information on authenticated users.
  - **Query Password:** The clear text password used to authenticate the query account on the LDAP server.

Generic Settings Subpanel: When you select *Generic LDAP Directory Service* as the Server Type, the following additional fields are displayed:

- **Base Domain Name:** The Root DN of the LDAP server's directory tree. This must be obtained by the administrator, and is usually in the form of `dc=adsroot,dc=novell,dc=com`.
- **User Attribute:** The attribute on a user's entry that identifies his or her login account name. For ADS servers, this attribute is sAMAccountName.
- **User Filter:** The name of the filter to be used in the lookup for the user's LDAP distinguished name.

  For ADS, this prefix is `cn=Users`.
- **User Prefix:** The prefix used to define the LDAP subtree within the BaseDN tree that contains user accounts. If you leave this property blank, the Orchestration Server uses the BaseDN.
- **Group Attribute:** Specifies the attribute of a group entry describing the login name of that group.
- **Group Filter:** A filter to be used in the lookup for group memberships on some LDAP schemas. The filter can use either ${USER_NAME} or ${USER_DN} to substitute that value. For example: memberUid=${USER_NAME}.
- **Group Prefix:** The prefix used to define the LDAP subtree within the BaseDN tree that contains group accounts.

  This field is not used for Active Directory authentication.
- **Group DNA Attribute:** The directory root where all queries for a user's group memberships (stored as a list of "member of" attributes on the user's entry on an ADS server) are to occur.
- **Nested DNA Attribute:** The attribute of a group entry where subgroups can be queried.

Authentication Page

As a data center administrator, you often have to provide credentials and certificates as you interact with the different hypervisor technologies, such as the Amazon EC2 or vSphere technologies. The Orchestration Server lets you store this data in a centralized, secure (no clear text passwords are accessible) location in its Credential Manager.

NOTE: The Cloud Manager Orchestration Server uses Triple DES password-based encryption in its Credential Manager to encrypt stored credentials and certificates.

The Credential Manager requires information from the Authentication page of the Orchestration Server object in the Orchestration Console, The page includes the following sections:

- "Stored Credentials Panel" on page 37
- "Stored Certificates Panel" on page 37

Stored Credentials Panel

The Stored Credentials panel displays a list of names of credential sets that you have created. You can create additional credentials if you select *Add Credential* and fill in the following fields:

- **Name:** (Required) The name that you want to use to refer to this credential set.
- **User:** (Required) The username with rights to administer objects in this grid.
- **Secret/Password:** (Required) The password that authenticates the user.
- **Type:** (Optional) A user-defined string that lets similar credentials be put into a category or group. For example, you might have a "type" of credential for the amazon-ec2 provisioning adapter and another type for the vsphere provisioning adapter.

Stored Credentials Password: (Conditional) If you want to change the password element of your stored credentials, click *Change* and enter the new password.

This password is used to encrypt the stored passwords. By default the password is `CHANGE_THIS_PASSPHRASE`. We recommend that you select a new password to use for encrypting stored passwords.

Stored Certificates Panel

In order to trust certificates not signed by well-known certificate authorities, the Orchestration Server lets you store certificates that are trusted by Java.

NOTE: Public/Private key pairs can be stored as certificates. This is useful if you need to manage amazon- ec2 key pairs.

The Stored Certificates panel displays a list of stored certificates. These certificates are not mapped to anything other than the name or identifier that you assign. They are not stored in a trust store, but their PEM-encoded representation is encrypted and stored alongside the credentials referred to above. Trust stores are generated on demand and are available to the Orchestration Agents.

Currently, this functionality is used only by the Orchestration vsphere provisioning adapter.

You can create additional trust stores if you select *Add Certificate* and fill in the following fields:

Identifier: (Required) The name that you want to use to refer to this trust store.

Location: (Required) Where the certificate should be obtained. This can be either a file (one that you can browse to find on the local machine), or an HTTPS server.

Select *Browse* if you want to select an existing a PEM-encoded certificate file from the local machine.

If you want to provide the actual URL for the certificate, open the drop-down list, select *HTTPS*, then enter the URL. The HTTPS server address can be entered as:

```
https://your.server.name
```

or as

```
your.server.name
```

or as

```
https://your.server.name:<sslport>
```

With this address, the Orchestration Server retrieves the public server certificate from the server and then stores it in a secure location.

Group: (Optional) A user-defined string used for grouping related certificates. For example, you might have a grouping called "vsphere" when you are managing resources in a multiple-vSphere Server environment.

3.1.3 Orchestration Server Policies Page

The Policies tab opens a page that contains a policy viewer for each of the policies associated with the Server object.

NOTE: You can edit a policy by right-clicking a policy icon, selecting *Edit Policy*, and clicking the *Save* button.

3.1.4 Orchestration Server Constraints/Facts Page

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for the Server object. The Server object has an associated set of facts and constraints that define its properties. By building, deploying, and running jobs on the Orchestration Server, you can individually change the functionality of any system resource by managing an object's facts and constraints. The Orchestration Server assigns default values to each of the component facts, although they can be changed at any time by the administrator, unless they are read-only. Facts that have mode `r/o` have read-only values, which can be viewed by using the pencil icon, but changes cannot be made.

3.2 Server Admin Object

The Server Admin object lists the accessible Orchestration Servers and their deployed components. Clicking a deployed component displays information about that component's associated Deployment Session.

4 The Job Object

A job is deployed to the Orchestration Server to automate processes, such as coordinating VM provisioning, high-performance computing, or general data center management. Jobs consist of Job Development Language (JDL) scripts and might have one or more policies associated with them. Policies define job arguments and other facts that are used by the job.

Usually a job has logic that runs on the Orchestration Server itself and schedules work to run on one or more managed resources that are running the Orchestration Agent. The logic that is dispatched and run on the managed resources is called a joblet. A job might or might not define one or more joblets.

A JDL script is partitioned into a Job section and one or more Joblet sections. The joblet sections of the script describe most of the work of a job. The Orchestration Server dispatches joblets to resources in the grid where the work is done.

A Job object represents an individual job in the Explorer tree of the Orchestration Console. This object contains facts with attributes that are used for job and joblet control. Policies associated with the job also control the job. The Orchestration Console has an administrative ("admin") view in the Explorer Panel that lets you edit these objects.

This section includes information about a Job object that is visible in the Explorer view and the accompanying Admin view of the Orchestration Console:

4.1 Job Groups

Any group object displayed in the Explorer panel represents a collection of similar object types. Groups can also be created automatically, such as when a provisioning adapter discovers a local repository on a VM host. For example, the xen provisioning adapter, upon discovery of a VM host, automatically creates a local repository for that VM host and places the created repository in a xen30 repository group. You can also create groups manually in the Orchestration Console, either by clicking the *Actions* menu and choosing *Create Job Group* or by right-clicking a Job Group object (anywhere in the Job hierarchy) and selecting *New Job Group*.

4.2 The Job Info/Groups Page

The page that opens under the *Info/Configuration* tab of the Job admin view includes several collapsible sections on the page where you can configure the general information and attributes of the job.

- Section 4.2.1, "Info," on page 40
- Section 4.2.2, "Groups," on page 49

NOTE: Whenever you make changes to any Grid object, the write icon is superimposed on the objects icon  , signifying that the object has been altered. If you want to save the changes you have made, you need to click the *Save* button  on the Orchestration Console toolbar.

4.2.1 Info

The following fields on the Information panel provide facts for the Job object:

- "Show Inherited Fact Values Check Box" on page 40
- "Job Control Settings" on page 40
- "Joblet Control Settings" on page 44
- "Automatic Resource Provisioning Settings" on page 45
- "Resource Preemption Settings" on page 46
- "Job Counts" on page 47
- "Job History" on page 47

Show Inherited Fact Values Check Box

Select this check box to show facts with overridden values supplied through attached or inherited policies. These fact values are read only (non-editable).

Job Control Settings

The Job Control Settings panel on the Info/Groups page includes the following fields:

NOTE: Tool tip text is available when you mouse over any of these fields.

Description: Enter information in this box that describes the nature or purpose of this job.

In the Fact Editor, this fact is listed as `job.description`:

```
<fact name="job.description" value="" type="String" />
```

Enabled: This check box is selected by default. When it is selected (it has a value of true), the job is enabled and is ready to run.

In the Fact Editor, this fact is listed as `job.enabled`:

```
<fact name="job.enabled" value="true" type="Boolean" />
```

Job Visible to Users: This check box is selected by default. When it is selected (it has a value of true), the job can be viewed in the Orchestration Console, through the use of command line queries, or in the Orchestration Server Portal. Deselecting this check box does not keep the job from running.

In the Fact Editor, this is fact is listed as `job.visible`:

```
<fact name="job.visible" value="true" type="Boolean" />
```

JDL Debug Tracing: This check box is not selected by default. When it is selected (it has a value of true), the job log includes tracing information when job events are executed.

In the Fact Editor, this fact is listed as `job.tracing`:

```
<fact name="job.tracing" value="false" type="Boolean" />
```

Job Type: Lets you choose the job type that applies to this job. This setting is optional and is leveraged by the server in order to provide better quality completion time calculation for the job.

The job type options (completion time algorithms) include:

- **normal:** The default job type. If this job has joblets, the job is based on PSPACE progression algorithm. If it does not have joblets, it is based on historical wall time average.
- **workflow:** This job type does not offer a time algorithm to the server.
- **pspace:** If this job has joblets, the job is based on PSPACE progression. If it does not have joblets, do not offer a time algorithm.
- **fixedtime:** This job type directs the server to use a time algorithm based on historical wall time average.
- **fixedgcycles:** If this job has joblets, the job is based on average gcycles and current consumption rate. If it does not have joblets, the job is based on historical wall time average.

NOTE: You can change this setting at runtime to refine the calculation time as the job progresses. For example, the zosmake job might start out as type *normal*, but when all tasks have been submitted, you could change it to type *workflow* to allow its subjobs to drive the end time.

In the Fact Editor, the Job Type fact is listed as `job.jobtype`:

```
<fact name="job.jobtype" value="normal" type="String" />
```

Job Timeout: The amount of time (in seconds) after which the server can take action to cancel the whole job, including all joblets and subjobs. A value of `-1` indicates no timeout.

In the Fact Editor, this fact is listed as `job.timeout`:

```
<fact name="job.timeout" value="-1" type="Integer" />
```

Job Auto Terminate: This check box is selected by default. When it is selected (it has a value of true), the job ends when all child jobs and joblets are executed.

In the Fact Editor, this fact is listed as `job.autoterminate`.

```
<fact name="job.autoterminate" value="true" type="Boolean" />
```

Provision Adapter Hook Jobs: The name of a job that implements administrator-defined pre-provisioning or post-provisioning hooks.

NOTE: This fact is visible in the *Info*/*Groups* tab only when a provision adapter job is selected.

In the Fact Editor, this fact is listed as an array:

```
<fact name="job.paHooksVmJob">
  <array type="String">
  </array>
</fact>
```

You can edit this array by clicking the ... button to open the Attribute Element Values dialog box, where you can add or remove fact specifications to the array of element choices. For more information, see "Provisioning Adapter Hooks" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

Queue Type: Lets you choose the queue type that applies to this job. This setting is optional and is leveraged by the server to provide a better start time calculation for the job.

The queue type options (start time algorithms) include:

- **none:** The start time is always unknown for jobs that are queued.
- **pfifo:** Packet First In First Out. The start time implemented through policies. The server is directed to look at the job as having a finite number of active slots, so its start time depends on its position in the queue and the estimated end time of running jobs of this type. The FIFO queue for this queue reshuffles based on priority.
- **fifo:** First In First Out. The start time implemented through policies. The server is directed to look at the job as having a finite number of active slots, so its start time depends on its position in the queue (first-come, first-served) and the estimated end time of running jobs of this type. The FIFO queue for this job does not reshuffle based on priority.
- **lifo:** Last In First Out. The start time is implemented through policies. The server is directed to look at the job as having a finite number of active slots, so its start time depends on its position in the queue and the estimated end time of running jobs of this type. The queue for this job does not reshuffle based on priority.
- **fixedtime:** The start time is based on the historical average queue time. This can be explicitly overridden through setting the `job.history.queuetime.average` fact.

In the Fact Editor, this fact is listed as `job.queuetype`:

```
<fact name="job.queuetype" value="pfifo" type="String" />
```

Job Queued Timeout: The amount of time (in seconds) after which the server can take action to cancel a queued job, including all joblets and subjobs. A value of `-1` indicates no timeout.

In the Fact Editor, this fact is listed as `job.queuedtimeout`:

```
<fact name="job.queuedtimeout" value="-1" type="Integer" />
```

Resource Match Cache TTL: Specifies the job's willingness to allow resource matches to be cached if the Job Scheduler becomes too loaded. The value is the time (in seconds) to live (TTL) of the cache. Enter a value less than zero (<0) to disable caching.

In the Fact Editor, this fact is listed as `jopb.cacheresourcematches.ttl`:

```
<fact name="job.cacheresourcematches.ttl" value="30" type="Integer" />
```

Preemptible: This check box is not selected by default. When it is selected (it has a value of true), you set the job's ability to be preempted. This setting can be overridden by the job instance.

In the Fact Editor, this fact is listed as `job.preemptible`:

```
<fact name="job.preemptible" value="false" type="Boolean" />
```

Restartable: This check box is not selected by default. When it is selected (it has a value of true), you set the job's ability to be restarted when the server restarts. This setting can be overridden by the job instance.

In the Fact Editor, this fact is listed as `job.restartable`:

```
<fact name="job.restartable" value="false" type="Boolean" />
```

Absolute Max Joblets: Specifies the absolute maximum number of joblets that you want this job to schedule.

In the Fact Editor, this fact is listed as `job.joblet.max`:

```
<fact name="job.joblet.max" value="1000" type="Integer" />
```

Max Joblet Failures: Specifies the number of non-fatal joblet errors that you want this job to tolerate before the job fails completely. Set the value at `-1` to attempt to continue after errors.

In the Fact Editor, this fact is listed as `job.joblet.maxfailures`:

```
<fact name="job.joblet.maxfailures" value="0" type="Integer" />
```

Max Node Failures: Specifies the number of resource failures that you want this job to tolerate before the node is excluded from further joblet processing. Set the value at `-1` to specify that limited failures are acceptable.

In the Fact Editor, this fact is listed as `job.maxnodefailures`:

```
<fact name="job.maxnodefailures" value="2" type="Integer" />
```

Max Resources: Specifies the maximum number of resources that you want the job to use at one time. The Orchestration Server does not exceed the value set here. Set the value at `-1` to specify unlimited resources.

In the Fact Editor, this fact is listed as `job.maxresources`:

```
<fact name="job.maxresources" value="-1" type="Integer" />
```

Max Joblets Running: Specifies the maximum number of joblets that you want the job to have running at one time. The Orchestration Server does not exceed the value set here. Set the value at `-1` to specify unlimited joblets.

In the Fact Editor, this fact is listed as `job.joblet.maxrunning`:

```
<fact name="job.joblet.maxrunning" value="-1" type="Integer" />
```

Max Joblets Per Resource: Specifies the maximum number of joblets that you want the job to occupy on a resource. Set the value at `-1` to specify unlimited joblets.

In the Fact Editor, this fact is listed as `job.joblet.maxperresource`:

```
<fact name="job.joblet.maxperresource" value="-1" type="Integer" />
```

Resource Selection Ranking: Displays ranking specification used to select suitable resources. The element syntax is `fact/order` where order is either ascending or descending

In the Fact Editor, this fact is listed as an array:

```
<fact name="job.resources.rankby">
  <array>
    <string>resource.loadaverage/a</string>
    <string>resource.anything/a</string>
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open the Attribute Element Values dialog box, where you can add or remove fact specifications to the array of element choices.

Persist Facts on Completion: This check box is not selected by default. When it is selected (it has a value of true), you specify that the Grid objects that this job modifies are persistent after the job. This setting is available and applicable only in a High Availability setup.

In the Fact Editor, this fact is listed as `job.persistfactsonfinish`:

```
<fact name="job.persistfactsonfinish" value="false" type="Boolean" />
```

Joblet Control Settings

Joblet Timeout: Specifies the amount of time (in seconds) you want the Orchestration Server to wait until canceling the joblet. Set the value at -1 to specify no timeout.

In the Fact Editor, this fact is listed as `job.joblet.timeout`:

```
<fact name="job.joblet.timeout" value="-1" type="Integer" />
```

Max Joblet Retries: Specifies the number of joblet retries (of any type) to be attempted before the Orchestration Server considers the joblet as failed. A value of zero (0) specifies that the joblet should not be retried. A value of less than zero (<0) specifies that the joblet should be continually retried.

In the Fact Editor, this fact is listed as `job.joblet.maxretry`:

```
<fact name="job.joblet.maxretry" value="0" type="Integer" />
```

Retry Limit (Forced): Specifies the number of forced joblet retries (requested by the joblet to run on another resource) to be allowed before the Orchestration Server considers the joblet as failed. A value of zero (0) specifies that the joblet should not be retried. A value of less than zero (<0) specifies that the joblet should be continually retried. This value should never exceed the value in `job.joblet.maxretry`.

In the Fact Editor, this fact is listed as `job.joblet.retrylimit.forced`:

```
<fact name="job.joblet.retrylimit.forced" value="-1" type="Integer" />
```

Retry Limit (Unforced): Specifies the number of unforced joblet retries to be allowed before the Orchestration Server considers the joblet as failed. A value of zero (0) specifies that the joblet should not be retried. A value of less than zero (<0) specifies the joblet should be continually retried. This value should never exceed the value in `job.joblet.maxretry`.

In the Fact Editor, this fact is listed as `job.joblet.retrylimit.unforced`:

```
<fact name="job.joblet.retrylimit.unforced" value="-1" type="Integer" />
```

Retry Limit (Resource Disconnect): Specifies the number of joblet retries caused by an unexpected resource disconnect to be allowed before the Orchestration Server considers the joblet as failed. A value of zero (0) specifies that the joblet should not be retried. A value of less than zero (<0) specifies the joblet should be continually retried. This value should never exceed the value in `job.joblet.maxretry`.

In the Fact Editor, this fact is listed as `job.joblet.retrylimit.disconnect`:

```
<fact name="job.joblet.retrylimit.disconnect" value="-1" type="Integer" />
```

Retry Limit (Timeout): Specifies the number of joblet retries caused by a server-initiated joblet timeout to be allowed before the Orchestration Server considers the joblet as failed. A value of zero (0) specifies that the joblet should not be retried. A value of less than zero (<0) specifies that the joblet should be continually retried. This value should never exceed the value in `job.joblet.maxretry`.

In the Fact Editor, this fact is listed as `job.joblet.retrylimit.timeout`:

```
<fact name="job.joblet.retrylimit.timeout" value="-1" type="Integer" />
```

Immediately Retry Failed Joblet: This check box is not selected by default. When it is selected (it has a value of true), you specify that you want the system to immediately retry a joblet rather than waiting until all other joblets are either running or complete before retrying.

In the Fact Editor, this fact is listed as `job.joblet.immediateretry`:

```
<fact name="job.joblet.immediateretry" value="true" type="Boolean" />
```

Max Joblet Wait Time: Specifies the amount of time (in seconds) you want a resource to wait before being utilized by a joblet. A setting of -1 indicates no timeout.

In the Fact Editor, this fact is listed as `job.joblet.maxwaittime`:

```
<fact name="job.joblet.maxwaittime" value="-1" type="Integer" />
```

Joblet JDL Debug Tracing: This check box is not selected by default. When it is selected (it has a value of true), you specify that you want the joblet to include tracing information on the job log as it executes joblet events.

In the Fact Editor, this fact is listed as `job.joblet.tracing`:

```
<fact name="job.joblet.tracing" value="false" type="Boolean" />
```

Joblet Run Type: Use this list to select whether or not the file and executable operations that run in the joblet are in behalf of the job user.

- **RunAsJobUserFallingBackToNodeUser:** (The default setting.) If this option is selected, any joblet logic executes as the local user with the same name as the grid user. If a local user of a matching name is not available, the joblet logic runs as the same user who is running the Orchestration Agent (also known as the Node User). By default, the agent (Node User) is `root`.
- **RunOnlyAsJobUser:** If this option is selected, any joblet logic executes as the local user using the same name as the grid user (that is, the Orchestration Server user who matches the Orchestration Server username). If a local user of a matching name is not available, the joblet logic, and perhaps the job, fails. By default, the agent (Node User) is `root`.
- **RunOnlyAsNodeUser:** If this option is selected, any joblet logic runs as the same user who is running the Orchestration Agent (also known as the "Node User"). It does not run as the OS user whose username matches the Orchestration Server user name. By default, the agent (Node User) is `root`.

In the Fact Editor, this fact is listed as `job.joblet.runtype`:

```
<fact name="job.joblet.runtype" value="RunAsJobUserFallingBackToNodeUser"
type="String" />
```

Automatic Resource Provisioning Settings

Max Resource Provisions: Specifies the number of resources that can be automatically provisioned in behalf of this job. A setting of zero (0) turns off automatic provisioning behavior. A setting of -1 allows unlimited provisioning.

In the Fact Editor, this fact is listed as `job.provision.maxcount`:

```
<fact name="job.provision.maxcount" value="0" type="Integer" />
```

Max Pending Provisions: Specifies the number of resources that can be automatically provisioned at one time (that is, simultaneously) in behalf of this job. A setting of less than or equal to zero (<=0) turns off automatic provisioning behavior.

In the Fact Editor, this fact is listed as `job.provision.maxpending`:

```
<fact name="job.provision.maxpending" value="1" type="Integer" />
```

Max Resource Provision Failures: Specifies the maximum number of resource provisioning failures to be tolerated before excluding the node from future automatic provisioning. A setting of -1 indicates that unlimited failures are acceptable.

In the Fact Editor, this fact is listed as `job.provision.maxnodefailures`:

```
<fact name="job.provision.maxnodefailures" value="1" type="Integer" />
```

Provision Selection Ranking: Displays the ranking the specification used to select suitable resources to automatically provision. The element syntax is `fact/order` where the order is either ascending or descending.

In the Fact Editor, this fact is listed as an array:

```
<fact name="job.provision.rankby">
  <array type="String">
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open the Attribute Element Values dialog box, where you can add or remove fact specifications for the array of element choices.

Host Selection Strategy: Lets you choose the type of strategy you want to use in finding a host for any automatically provisioned resource. The choices include:

- **queue:** Directs the server to use the default affinity wait period defined by the resource before considering all possible hosts. The request is queued until a suitable resource becomes available or a requesting job finishes.
- **immediate:** Directs the server to immediately consider the affinity host before trying to find any matching resources and to fail if a suitable resource is not available.

In the Fact Editor, this fact is listed as `job.provision.hostselection`:

```
<fact name="job.provision.hostselection" value="immediate" type="String" />
```

Resource Preemption Settings

Job Selection Ranking: Displays the ranking specification used to select suitable jobs to automatically preempt on a resource. Element syntax is `fact/order` where the order is either ascending or descending.

In the Fact Editor, this fact is listed as an array:

```
<fact name="job.preemption.rankby">
  <array>
    <string>jobinstance.priority/a</string>
    <string>jobinstance.joblets.running/d</string>
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open the Attribute Element Values dialog box, where you can add or remove fact specifications for the array of element choices.

Job Counts

Total Instances: Displays the total number of job instances of this type that exist in the Cloud Manager Orchestration Server system.

In the Fact Editor, this fact is listed as `job.instances.total`:

```
<fact name="job.instances.total" value="0" type="Integer" />
```

Active Instances: Displays the total number of job instances of this type that are in a queued state in the Orchestration Server system.

In the Fact Editor, this fact is listed as `job.instances.active`:

```
<fact name="job.instances.active" value="0" type="Integer" />
```

Queued Instances: Displays the total number of job instances of this type that are active in the Orchestration Server system.

In the Fact Editor, this fact is listed as `job.instances.queued`:

```
<fact name="job.instances.queued" value="0" type="Integer" />
```

Job Accounting Group: Lets you select the Job Group whose statistics are updated by default when the job runs.

In the Fact Editor, this fact is listed as `job.accountinggroup`:

```
<fact name="job.accountinggroup" value="all" type="String" />
```

Job Resource Group: Lets you select the default Resource Group whose members and any of its resource policies are selected for this job.

In the Fact Editor, this fact is listed as `job.resourcegroup`:

```
<fact name="job.resourcegroup" value="all" type="String" />
```

Job History

Shared Instance Count: (Read only) Displays the total number of job instances, including those denied by "accept" constraints, of this job that have ever been initiated on this Orchestration Server system.

In the Fact Editor, this fact is listed as `job.history.jobcount`:

```
<fact name="job.history.jobcount" value="0" type="Integer" />
```

Completed Count: (Read only) Displays the total number of job instances, including those denied by "accept" constraints, of this job that have been canceled.

In the Fact Editor, this fact is listed as `job.history.jobcount.complete`:

```
<fact name="job.history.jobcount.complete" value="0" type="Integer" />
```

Cancelled Count: (Read only) Displays the total number of job instances, including those denied by "accept" constraints, of this job that have been completed.

In the Fact Editor, this fact is listed as `job.history.jobcount.cancelled`:

```
<fact name="job.history.jobcount.cancelled" value="0" type="Integer" />
```

Failed Count: (Read only) displays the total number of job instances of this type that have failed.

In the Fact Editor, this fact is listed as `job.history.jobcount.failed`:

```
<fact name="job.history.jobcount.failed" value="0" type="Integer" />
```

Total Cost: Displays the total cost of running this job. The amount is calculated since the job was deployed or last modified.

In the Fact Editor, this fact is listed as `job.history.cost.total`:

```
<fact name="job.history.cost.total" value="0.0000" type="Real" />
```

Average Cost: Displays the average cost of running this job. The amount is calculated since the job was deployed or last modified and is updated only if the job finishes successfully.

In the Fact Editor, this fact is listed as `job.history.cost.average`:

```
<fact name="job.history.gcycles.average" value="0" type="Integer" />
```

Total Runtime: Displays the total runtime (in seconds) since the job was deployed.

In the Fact Editor, this fact is listed as `job.history.runtime.total`:

```
<fact name="job.history.runtime.total" value="0" type="Integer" />
```

Average Runtime: Displays the average runtime (in seconds) since the job was deployed.

In the Fact Editor, this fact is listed as `job.history.runtime.average`:

```
<fact name="job.history.runtime.average" value="0" type="Integer" />
```

Total Execution Time: Displays the total combined resource wall time (in seconds) of all work performed on behalf of this job since the job was deployed.

In the Fact Editor, this fact is listed as `job.history.time.total`:

```
<fact name="job.history.time.total" value="0" type="Integer" />
```

Average Execution Time: Displays the average resource wall time (in seconds) of all work performed on behalf of this job since the job was deployed.

In the Fact Editor, this fact is listed as `job.history.time.average`:

```
<fact name="job.history.time.average" value="0" type="Integer" />
```

Total Grid Time: Displays the total amount of normalized grid time (in gcycles) consumed by this job since deployment.

In the Fact Editor, this fact is listed as `job.history.gcycles.total`:

```
<fact name="job.history.gcycles.total" value="0" type="Integer" />
```

NOTE: A gcycle can be thought of as a normalized second of compute time. It is a relative measure that is approximately a second of the real processing time of a 2Ghz Pentium class Intel processor.

Average Grid Time: Displays the average amount of normalized grid time (in gcycles, which is a normalized grid cycle) consumed by running this job. The value is updated only if the job finishes successfully.

In the Fact Editor, this fact is listed as `job.history.gcycles.average`:

```
<fact name="job.history.gcycles.average" value="0" type="Integer" />
```

Total Queue Time: Displays the total amount of time (in seconds) since deployment that the job has spent in a queued state.

In the Fact Editor, this fact is listed as `job.history.queuetime.total`:

```
<fact name="job.history.queuetime.total" value="0" type="Integer" />
```

Average Queue Time: Displays the average amount of wall time (in seconds) spent waiting for this job to start.

In the Fact Editor, this fact is listed as `job.history.queuetime.average`:

```
<fact name="job.history.queuetime.average" value="0" type="Integer" />
```

Average Sample Size: Displays the total number of points you want to use in the trailing average calculation for all historical averages.

In the Fact Editor, this fact is listed as `job.history.samplesize`:

```
<fact name="job.history.samplesize" value="2" type="Integer" />
```

NOTE: A trailing average is the mean average measured over the last *x* datapoints.

4.2.2 Groups

This section of the Info/Groups page lists the groups of Job objects in the grid. Click *Choose* to open the Job Group Selection dialog box. In this dialog box, you can choose which Job Groups to display in the Explorer Panel by selecting a group and then clicking *Add* or *Remove* to move it to or from the *Source Job Groups* list.

4.3 The Job Configuration Page

The *Job Configuration* tab of the Job admin view opens the Job Configuration page, which includes the LVM Discovery Settings panel. On this panel, you can configure settings that control the discovery of logical volume managers. The *Volume Group Patterns* fact (`job.volume_group_patterns`) provides a place for you to define regular expression patterns for Volume Group names. The Orchestration Server uses these patterns to create Repository objects.

In the Fact Editor, this fact is listed as an array:

```
<fact name="job.volume_group_patterns" description="Regular expression patterns
for Volume Group names to create a Repository Object for">
  <array>
    <string>NCM-*</string>
  </array>
</fact>
```

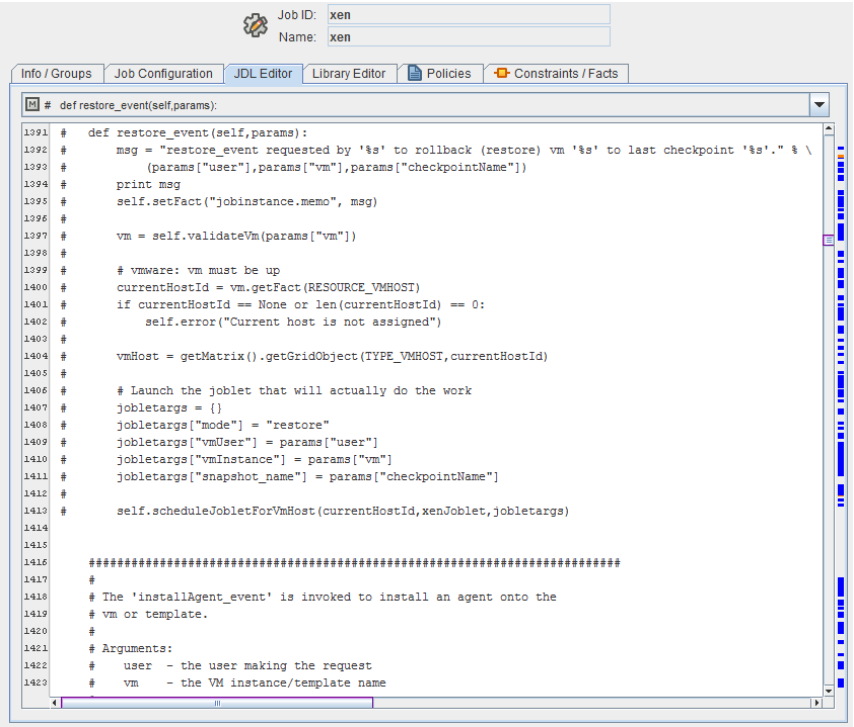
You can edit this array by clicking the ··· button to open the Attribute Element Values dialog box, where you can add or remove fact specifications to the array of element choices.

The admin view also includes a link to open and edit the configuration definition file.

4.4 The JDL Editor Page

The *JDL Editor* tab of the Job admin view opens an editor where you can inspect and modify the Job Description Language (JDL) code. This code consists of a Python script that contains the bits to control a job. The JDL code for each job includes commented documentation to explain the job's purpose and methods for implementation.

Figure 4-1 *The JDL Editor*

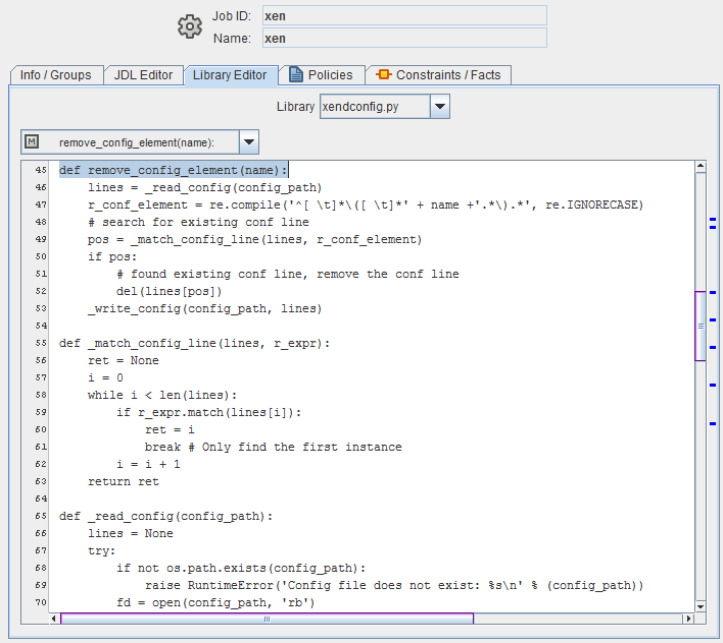


A drop-down list at the top of the editor includes the Java classes and their methods that are bookmarked in the code. Select any of these to go to the location in the code where they are invoked. Clickable colored blocks on the editor scroll bar perform a similar bookmarking function.

4.5 The Job Library Editor Page

The *Library Editor* tab of the Job admin view opens an editor where you can inspect and modify the different library scripts for a job. The scripts for each job include instructions to the Orchestration Server for handling job functions.

Figure 4-2 *The Job Library Editor*



There are two drop-down lists located at the top of the Library Editor view. The first list displays the different libraries for the job, and the second list displays the methods that are bookmarked in the code. Select a method in the second drop-down list to go to the location in the library code where that method is invoked. Clickable colored blocks on the editor scroll bar perform a similar bookmarking function.

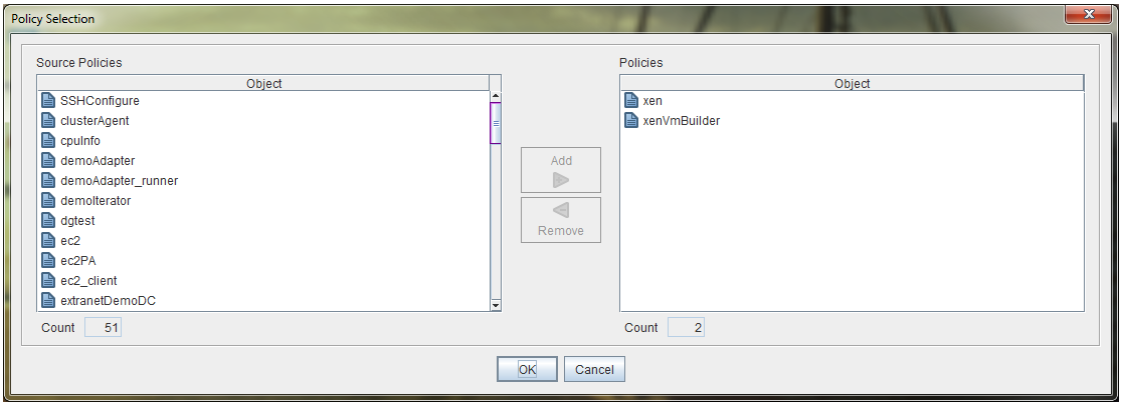
4.6 The Job Policies Page

The *Policies* tab of the Job admin view opens a page that contains a policy viewer for each of the policies associated with a Job Grid object.

You can modify a policy by using the Policy Grid object. For more information see Section 12.1, “Policy Object,” on page 153.

Click *Choose* in the admin view of the Policy viewer to launch a Policy Selection dialog box where you can add or remove individual policies to be applied to the selected Job Grid object.

Figure 4-3 *The Policy Selection Dialog Box*



4.7 The Job Constraints/Facts Page

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties. By changing the policy constraints and fact values for a job, you can change the behavior of the job and how the Orchestration Server allocates available system resources to it. The Orchestration Server assigns default values to each of the component facts. Facts with no mode shown can be changed at any time by the administrator. Facts with mode `r/o` have read-only values, which can be viewed by using the pencil icon, but changes cannot be made.

5 The Resource Object

A Resource object in the Explorer tree represents a fixed physical machine or a virtual machine (VM) that is managed by the Cloud Manager Orchestration Server. If a resource is running the Orchestration Agent, that resource can be scheduled for remote execution of a job.

This section includes information about a Resource object that is visible in the Explorer tree and the accompanying Admin view of the Orchestration Console:

5.1 Resource Groups

Any group object displayed in the Explorer tree represents a collection of similar object types. Groups can also be created automatically, such as when a provisioning adapter discovers a local repository on a VM host. For example, the xen provisioning adapter, upon discovery of a VM host, automatically creates a local repository for that VM host and places the created repository in a xen repository group. You can also create groups manually in the Orchestration Console, either by clicking the *Actions* menu and choosing *Create Resource Group* or by right-clicking a Resource Group object anywhere in the Resource hierarchy and selecting *New Resource Group*.

5.2 Resource Info/Groups Page

The page that opens under the *Info/Configuration* tab of the Resource admin view includes several collapsible sections on the page where you can configure the general information and attributes of the job.

NOTE: Whenever you make changes to any Grid object, the write icon is superimposed on the object’s icon, signifying that the object has been altered. If you want to save the changes you have made, you need to click the *Save* button on the Orchestration Console toolbar.

5.2.1 Info Panel

The following fields on the Information panel provide facts for the Resource object:

Show Inherited Fact Values Check Box

Select this check box to show facts with overridden values supplied through attached or inherited policies. These fact values are read only (non-editable).

Resource Information

The Job Control Settings panel on the Info/Groups page includes the following fields:

- Resource Type
- Resource Enabled
- Healthy
- Shutting Down
- Host Name
- Host Fully Qualified Name
- Password
- Host IP Address
- VNC IP Address
- VNC Port
- Billing Rate
- Bill For
- Power Factor
- Load Average
- CPU Load
- Joblet Slots
- Extra System Joblet Slots

- Joblets Active
- Became Idle On
- Total Joblets Started
- Total Completed Joblets
- Total Cancelled Joblets
- Total Failed Joblets
- Total Charge
- Total Wall Time
- Total Grid Time
- Sessions
- Provisionable Resource

NOTE: Tool tip text is available when you mouse over any of these fields.

Resource Type: Lets you choose the resource type. If you manually create a resource, you must select the appropriate type.

- **Fixed Physical:** The node is a physical, hardware-based computer.
- **VM:** The node is a virtual, software-based container that can run its own operating system and applications as if it were a physical computer.
- **VM Template:** The node is an image of a server that can be used to create and provision new virtual servers. The template includes virtual hardware components, a guest operating system, its configuration, and other software applications.

In the Fact Editor, the Resource Type fact is listed as `resource.type`:

```
<fact name="resource.type" value="Fixed Physical" type="String" />
```

Resource Enabled: This check box is selected by default. When it is selected (it has a value of true), the resource is enabled and allowed to accept work.

In the Fact Editor, this fact is listed as `resource.enabled`:

```
<fact name="resource.enabled" value="true" type="Boolean" />
```

Healthy: When this check box is selected (it has a value of true), the resource is considered to be in good health. You can set the health of the object by selecting or deselecting the health check box. Changing the value in this way has an immediate effect unless the value is overriden by an attached policy. For more information, see Appendix A, "Grid Object Health Monitoring," on page 187.

In the Fact Editor, this is fact is listed as `resource.health`:

```
<fact name="resource.health" value="true" type="Boolean" />
```

Shutting Down: (Read Only) When this check box is selected (it has a value of true), the node is attempting to shut down, pause, or suspend and does not accept new work.

In the Fact Editor, this fact is listed as `resource.shuttingdown`:

```
<fact name="resource.shuttingdown" value="false" type="Boolean" />
```

Host Name: The network hostname of the resource that is running the Orchestration Agent. The resource ID and the hostname are often the same, but this is not always the case.

In the Fact Editor, this fact is listed as `resource.hostname`:

```
<fact name="resource.hostname" value="foonode" type="String" />
```

Host Fully Qualified Name: The full network hostname of the resource that is running the Orchestration Agent.

In the Fact Editor, this fact is listed as `resource.hostname.full`:

```
<fact name="resource.hostname.full" value="foonode.division.company.com"
type="String" />
```

Password: The password you want the Orchestration Agent on this node to use for authentication to the Orchestration Server.

In the Fact Editor, this fact is listed as `resource.password`.

```
<fact name="resource.password" value="xxx" type="String" />
```

Host IP Address: The network IP address of the resource running the Orchestration Agent.

In the Fact Editor, this fact is listed as `resource.ip`:

```
<fact name="resource.ip" value="10.255.255.255" type="String" />
```

VNC IP Address: The IP address for a VNC session running on this resource.

In the Fact Editor, this fact is listed as `resource.vnc.ip`:

```
<fact name="resource.vnc.ip" value="" type="String" />
```

VNC Port: The port number for a VNC session running on this resource.

In the Fact Editor, this fact is listed as `resource.vnc.port`:

```
<fact name="resource.vnc.port" value="0" type="Integer" />
```

Billing Rate: The billing rate (in dollars per hour) that you want to charge for this resource running its assigned joblets.

In the Fact Editor, this fact is listed as `resource.billingrate`:

```
<fact name="resource.billingrate" value="1.0000" type="Real" />
```

Bill For: Lets you choose the time scale you want to bill for.

- **walltime:** The total time for the process to complete.
- **gcycles:** The normalized average of compute cycles.

In the Fact Editor, this fact is listed as `resource.billfor`:

```
<fact name="resource.billfor" value="walltime" type="String" />
```

Power Factor: (Read Only) The normalized power index of this machine relative to a 2.0 GHz Intel Pentium 4 machine.

In the Fact Editor, this fact is listed as `resource.powerfactor`:

```
<fact name="resource.powerfactor" value="1.0000" type="Real" />
```

Load Average: (Read Only) The load average on this resource as determined with the `uptime` command or other similar methods. The resource is polled every 30 seconds to determine the average.

In the Fact Editor, this fact is listed as `resource.loadaverage`:

```
<fact name="resource.loadaverage" value="0.0000" type="Real" />
```

CPU Load: (Read Only) The percentage of CPU utilization currently used by the resource.

In the Fact Editor, this fact is listed as `resource.cpuload`

```
<fact name="resource.cpuload" value="0" type="Integer" />
```

Joblet Slots: The number of joblets that this resource can run simultaneously.

In the Fact Editor, this fact is listed as `resource.joblets.slots`:

```
<fact name="resource.joblets.slots" value="1" type="Integer" />
```

Extra System Joblet Slots: The number of extra slots you want to be made available to privileged `system` joblets.

In the Fact Editor, this fact is listed as `resource.joblets.systemslots`:

```
<fact name="resource.joblets.systemslots" value="1" type="Integer" />
```

Joblets Active: (Read Only) The number of joblets that are currently active on this resource.

In the Fact Editor, this fact is listed as `resource.joblets.active`:

```
<fact name="resource.joblets.active" value="0" type="Integer" />
```

Became Idle On: (Read Only) The date and time when the resource became idle. The field displays `-1` if the resource is active.

In the Fact Editor, this fact is listed as `resource.becameidle`:

```
<fact name="resource.becameidle" value="7/23/11 5:02 PM" type="Date" />
```

Total Joblets Started: (Read Only) The total number of joblets that have run historically on this resource.

In the Fact Editor, this fact is listed as `resource.history.jobletcount`:

```
<fact name="resource.history.jobletcount" value="8" type="Integer" />
```

Total Completed Joblets: (Read Only) The total number of joblets that have completed historically on this resource.

In the Fact Editor, this fact is listed as `resource.history.jobletcount.completed`:

```
<fact name="resource.history.jobletcount.completed" value="8" type="Integer" />
```

Total Cancelled Joblets: (Read Only) The total number of joblets that have been canceled historically on this resource.

In the Fact Editor, this fact is listed as `resource.history.jobletcount.cancelled`:

```
<fact name="resource.history.jobletcount.cancelled" value="0" type="Integer" />
```

Total Failed Joblets: (Read Only) The total number of joblets that have failed historically on this resource.

In the Fact Editor, this fact is listed as `resource.history.jobletcount.failed`:

```
<fact name="resource.history.jobletcount.failed" value="0" type="Integer" />
```

Total Charge: (Read Only) The cost (in dollars) of all of the joblets run on this resource.

In the Fact Editor, this fact is listed as `resource.history.cost.total`:

```
<fact name="resource.history.cost.total" value="0.0088" type="Real" />
```

Total Wall Time: (Read Only) The total wall time (measured in seconds) that this resource has spent running joblets.

In the Fact Editor, this fact is listed as `resource.history.time.total`:

```
<fact name="resource.history.time.total" value="31" type="Integer" />
```

Total Grid Time: (Read Only) The amount of time (measured in gcycles, which is the normalized average of compute cycles) of all work performed on this resource.

In the Fact Editor, this fact is listed as `resource.history.gcycles.total`:

```
<fact name="resource.history.gcycles.total" value="31" type="Integer" />
```

Sessions: (Read Only) The number of current active sessions (that is, the resource instances with an active agent). The value will be either 1 or 0, unless the object is actually a resource template, in which case it might be greater than 1.

In the Fact Editor, this fact is listed as `resource.sessions`:

```
<fact name="resource.sessions" value="0" type="Integer" />
```

Provisionable Resource: This check box is not selected by default. When it is selected (it has a value of true), you specify that this resource is a provisionable type. Currently, only a VM resource and a VM template resource are provisionable.

In the Fact Editor, this fact is listed as `resource.provisionable`:

```
<fact name="resource.provisionable" value="false" type="Boolean" />
```

VM Host Info

The settings in this section of the Info/Groups page are available when the resource is a VM host.

- VM Host Containers
- VM Host Repositories

VM Host Containers: A list of VM host containers that are supported by this resource. The list is aggregated from the VM host containers.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.vmhosts">
  <array>
    <string>host1slesx_xen30</string>
  </array>
</fact>
```

You can edit this array by clicking the ... button to open an array editor, where you can add or remove VM host containers for the array of element choices.

VM Host Repositories: A list of VM host repositories visible to this resource. The list is aggregated from the VM host repositories.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.repositories">
  <array>
    <string>zos</string>
    <string>vmh3slesx</string>
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open an array editor, where you can add or remove VM host repositories to the array of element choices.

Virtual Machine Configuration

The settings in this section of the Info/Groups page are available when the resource is a VM:

- Under Construction
- VM Vendor
- VM UUID
- VM Version
- Default Storage Repository
- Virtual NICs
- Virtual NIC Networks
- Virtual Disks
- Virtual Disk Repositories
- Moveable Virtual Disk Repositories
- Unmoveable Virtual disk Repositories
- Storage Location in Repository
- VM Files
- Required VM Memory
- Required VM Memory Overhead
- Required Total VM Memory
- Host CPU Architecture
- Requires Host HVM Support
- Host CPU % Weight
- Host CPU Number
- Moveable Disk Size
- Allow VM Migration
- VM Host Ranking
- Construction Specification

For information about VM provisioning based on options exposed in the Orchestration Console, see "Using the Right-Click Menu for Provisioning Actions" in *NetIQ Cloud Manager 2.1 VM Orchestration Reference*.

Under Construction: This check box is not selected by default. When it is selected (it has a value of true), the VM is currently in the process of being created and cannot be provisioned.

In the Fact Editor, this fact is listed as `resource.vm.underconstruction`:

```
<fact name="resource.vm.underconstruction" value="false" type="Boolean" />
```

VM Vendor: The vendor name of the hypervisor that provides the virtual machine.

In the Fact Editor, this fact is listed as `resource.vm.vendor`:

```
<fact name="resource.vm.vendor" value="xen" type="String" />
```

VM UUID: The vendor and adapter-specific UUID of the resource. You should edit this value only if you are manually creating a Resource object.

In the Fact Editor, this fact is listed as `resource.vm.uuid`:

```
<fact name="resource.vm.uuid" value="237e9975-xxx15-yy1122-7c62-bf6d23d3a049"
type="String" />
```

VM Version: This fact is no longer used.

In the Fact Editor, this fact is listed as `resource.vm.version`:

```
<fact name="resource.vm.version" value="0" type="Integer" />
```

Default Storage Repository: Lets you choose the repository where the images of this VM disk and other configuration files are currently stored or where they will be stored.

In the Fact Editor, this fact is listed as `resource.vm.repository`:

```
<fact name="resource.vm.repository" value="vmh1slesx" type="String" />
```

Virtual NICs: The virtual network interface cards (VNICs) that make up this VM. The list is aggregated from the VNIC containers.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.vm.vnics">
  <array>
    <string>win2003_vnic1</string>
  </array>
</fact>
```

Virtual NIC Networks: The networks associated with the VM network interfaces.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.vm.networks">
  <array>
    <string>eth1</string>
  </array>
</fact>
```

Virtual Disks: The list of virtual disks that make up this VM.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.vm.vdisks">
  <array>
    <string>websrvr_vdisk1</string>
  </array>
</fact>
```

Virtual Disk Repositories: The repositories where the VM disk images are stored.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.vm.repositories">
  <array>
    <string>zos</string>
  </array>
</fact>
```

Moveable Virtual Disk Repositories: The repositories where the moveable VM disk images are stored.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.vm.repositories.moveable">
  <array>
    <string>zos</string>
  </array>
</fact>
```

Unmoveable Virtual Disk Repositories: The repositories where the unmoveable VM disk images are stored.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.vm.repositories.unmoveable">
  <array type="String">
  </array>
</fact>
```

Storage Location in Repository: The file system location (either absolute or relative to `repository.location`) of the VM files.

In the Fact Editor, this fact is listed as `resource.vm.basepath`:

```
<fact name="resource.vm.basepath" value="vm/websrvr" type="String" />
```

VM Files: The files that make up this VM.The dictionary key (String) represents the file type (adapter specific). The value is the file path either absolute or relative to `repository.location` of the `resource.vm.repository`.

In the Fact Editor, this fact is listed as a dictionary:

```
<fact name="resource.vm.files">
  <dictionary>
    <dictelement key="config">
      <string>/var/lib/xen/images/win2kbuild/config.xen</string>
    </dictelement>
  </dictionary>
</fact>
```

You can edit this array by clicking the ··· button to open an array editor, where you can add or remove ranking specifications for the array of element choices.

Required VM Memory: The amount (measured in MB) of virtual memory required for this VM image.

In the Fact Editor, this fact is listed as `resource.vm.memory`:

```
<fact name="resource.vm.memory" value="1024" type="Integer" />
```

Required VM Memory Overhead: The amount (measured in MB) of virtual memory overhead required for this VM image to provision.

In the Fact Editor, this fact is listed as `resource.vm.memory.overhead`:

```
<fact name="resource.vm.memory.overhead" value="70" type="Integer" />
```

Required Total VM Memory: The total amount (measured in MB) of virtual memory required for this VM image.

In the Fact Editor, this fact is listed as `resource.vm.memory.total`:

```
<fact name="resource.vm.memory" value="402" type="Integer" />
```

Host CPU Architecture: The type of CPU architecture required by this VM. You should edit these values only when you are manually creating a Resource object.

Possible types include:

- x86
- x86_64
- sparc
- ppc
- mips
- alpha

In the Fact Editor, this fact is listed as `resource.vm.cpu.architecture`:

```
<fact name="resource.vm.cpu.architecture" value="x86" type="String" />
```

Requires Host HVM Support: This check box is selected by default. When it is selected (it has a value of true), this VM requires host HVM support. The setting is required when you want to perform paravirtualization; otherwise, only full virtualization is possible.

In the Fact Editor, this fact is listed as `resource.vm.cpu.hvm`:

```
<fact name="resource.vm.cpu.hvm" value="true" type="Boolean" />
```

Host CPU % Weight: The CPU weight (as a percentage of the virtual processor runtime) that you can assign to the virtual processor associated with this VM.

A value of 1.0 represents normal weighting. Setting another VM to a weight of 2.0 means that it gets twice as much CPU runtime as this VM.

In the Fact Editor, this fact is listed as `resource.vm.cpu.weight`:

```
<fact name="resource.vm.cpu.weight" value="1.0000" type="Real" />
```

Host CPU Number: The number of virtual CPUs assigned to this VM.

In the Fact Editor, this fact is listed as `resource.vm.vcpu.number`:

```
<fact name="resource.vm.vcpu.number" value="1" type="Integer" />
```

Moveable Disk Size: The total size (measured in MB) of all the virtual moveable disks for this VM image.

In the Fact Editor, this fact is listed as `resourc.vm.vdisksize`:

```
<fact name="resource.vm.vdisksize" value="4096" type="Integer" />
```

Allow VM Migration: This check box is selected by default. When it is not selected (it has a value of false), Orchestration Server prevents migration of the VM to another potential VM host.

In the Fact Editor, this fact is listed as `resource.vm.migratable`:

```
<fact name="resource.vm.migratable" value="true" type="Boolean" />
```

VM Host Ranking: This list box includes the ranking specifications used to select suitable VM hosts. The element syntax is *fact*/*order*, where order is either `ascending` or `descending`.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.vm.vmhost.rankby">
  <array>
    <string>vmhost.vm.placement.score/a</string>
    <string>vmhost.loadindex.slots/a</string>
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open an array editor, where you can add or remove ranking specifications for the array of element choices. A trailing `/a` indicates an ascending sort order. A trailing `/d` indicates a descending sort order.

Construction Specification: The *VM Builder Specifications* field in this section displays a list of specifications that were used to build this VM. These specifications are interpreted by the provisioning adapter. You should edit these values by using the Cloud Manager VM Client.

In the Fact Editor, this fact is listed as a dictionary:

```
<fact name="resource.vm.spec">
  <dictionary>
    <dictelement key="ssd">
      <string>ddd</string>
    </dictelement>
  </dictionary>
</fact>
```

You can edit this array by clicking the ··· button to open an array editor dialog box, where you can add or remove builder specifications for the array of element choices.

Provisioning Information

The settings on this section of the Info/Groups panel are not available unless the resource you select is a VM.

- Provisioning Job
- Provisioned Instances
- Cloned Instances
- Instances
- Max Provisioned Instances
- Agent Shutdown Timeout
- Default Agent Idle Timeout
- Host Wait Timeout
- Preferred Host Wait
- Recommended Host
- Debug Provision Log
- Parent Template
- Current State
- Current Host
- Current Status

- Current Action
- Request Time
- Start Time
- Shutdown Time
- Host Wait Time
- Managing Job ID
- Automatic Provision
- Needs Resync

If the Resource object is a VM, it can be automatically personalized for provisioning with information you provide in one of the following subpanels of the *Provisioning Information* panel:

- "Linux Autoprep Config:" on page 67
- "Windows Sysprep Config" on page 68

Provisioning Job: Lets you select the name of the provisioning job that manages the life cycle of this resource.

In the Fact Editor, this fact is listed as `resource.provisioner.job`:

```
<fact name="resource.provisioner.job" value="xen" type="String" />
```

Provisioned Instances: The total count of operational instances and provisions in progress.

In the Fact Editor, this fact is listed as `resource.provisioner.count`:

```
<fact name="resource.provisioner.count" value="0" type="Integer" />
```

Cloned Instances: The total count of cloned instances of the template.

In the Fact Editor, this fact is listed as `resource.provisioner.instancecount`:

```
<fact name="resource.provisioner.instancecount" value="0" type="Integer" />
```

Instances: The IDs of the instances of this template resource (if applicable).

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.provisioner.instances">
  <array type="String">
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open an array editor, where you can add or remove instance IDs to the array of choices.

Max Provisioned Instances: (For VM templates only) The maximum allowed number of instances of this provisionable resource.

In the Fact Editor, this fact is listed as `resource.provisioner.maxinstances`:

```
<fact name="resource.provisioner.maxinstances" value="1" type="Integer" />
```

Agent Shutdown Timeout: The maximum amount of time, measured in seconds, allowed for this VM to shut down and for the Orchestration Agent to disconnect from the Orchestration Server.

In the Fact Editor, this fact is listed as `resource.provisioner.timeout.shutdown`:

```
<fact name="resource.provisioner.timeout.shutdown" value="" type="Integer" />
```

Default Agent Idle Timeout: The maximum amount of time, measured in seconds, allowed for a VM instance to be idle before relaxing the reservation policy or shutting down. Behavior depends on the mode, and can be overridden by the provisioning request.

In the Fact Editor, this fact is listed as `resource.provisioner.timeout.idle`:

```
<fact name="resource.provisioner.timeout.idle" value="" type="Integer" />
```

Host Wait Timeout: The maximum amount of time, measured in seconds, to wait for a suitable host before timing out. A value of less than zero (<0), means that the VM waits indefinitely.

In the Fact Editor, this fact is listed as `resource.provisioner.host.maxwait`:

```
<fact name="resource.provisioner.host.maxwait" value="-1" type="Integer" />
```

Preferred Host Wait: The amount of time, measured in seconds, after which some VM Host constraints (for example, whether to move the disk image) are lifted to increase the available pool of hosts. A value of less than zero (<0), means that the VM resource waits indefinitely.

In the Fact Editor, this fact is listed as `resource.provisioner.host.preferredwait`:

```
<<fact name="resource.provisioner.host.preferredwait" value="0" type="Integer" />
```

Recommended Host: The names of VM hosts that you can choose to associate with this VM resource image. You might specify this host when you want a quick VM startup or if you want to change hosts because the original host was suspended. When combined with the `resource.provisioner.host.preferredwait` fact, this fact can lock a VM to one host.

In the Fact Editor, this fact is listed as `resource.provisioner.recommendedhost`:

```
<fact name="resource.provisioner.recommendedhost" value="" type="String" />
```

Debug Provision Log: This check box is not selected by default. When it is selected (it has a value of true), the debug log level in the provisioner is enabled.

In the Fact Editor, this fact is listed as `resource.provisioner.debug`:

```
<fact name="resource.provisioner.debug" value="false" type="Boolean" />
```

Parent Template: The ID of the template resource from which this instance was created.This is only applicable if the template was copied from another template.

In the Fact Editor, this fact is listed as `resource.provision.template`:

```
<fact name="resource.provision.template" value="" type="String" />
```

Current State: The current state of this provisioned instance. The different states include:

- down
- suspended
- up
- paused
- unknown (when an administrative action is in process)

In the Fact Editor, this fact is listed as `resource.provision.state`:

```
<fact name="resource.provision.state" value="down" type="String" />
```

Current Host: The ID of the VM host that is currently housing this provisioned resource.

In the Fact Editor, this fact is listed as `resource.provision.vmhost`:

```
<fact name="resource.provision.vmhost" value="vmh6sles_xen" type="String" />
```

Current Status: (Read Only) The current descriptive status of the provisioned resource.

In the Fact Editor, this fact is listed as `resource.provision.status`:

```
<fact name="resource.provision.status" value="Undefined" type="String" />
```

Current Action: (Read Only) The management action currently in progress on this provisioned resource.

In the Fact Editor, this fact is listed as `resource.provision.currentaction`:

```
<fact name="resource.provision.currentaction" value="" type="String" />
```

Request Time: (Read Only) The time when the last provision or other administrative action was requested.

In the Fact Editor, this fact is listed as `resource.provision.time.request`:

```
<fact name="resource.provision.time.request" value="8/24/11 4:36 PM" type="Date" /
>
```

Start Time: (Read Only) The time when the resource was last successfully provisioned.

In the Fact Editor, this fact is listed as `resource.provision.time.start`:

```
<fact name="resource.provision.time.start" value="12/31/69 4:59 PM" type="Date" />
```

Shutdown Time: (Read Only) The time when the resource was last shut down.

In the Fact Editor, this fact is listed as `resource.provision.time.shutdown`:

```
<fact name="resource.provision.time.shutdown" value="12/31/69 4:59 PM" type="Date"
/>
```

Host Wait Time: (Read Only) The amount of time, measured in seconds, that this resource has been waiting for or did wait for a suitable host.

In the Fact Editor, this fact is listed as `resource.provision.time.hostwait`:

```
<fact name="resource.provision.time.hostwait" value="0" type="Integer" />
```

Managing Job ID: (Read Only) The current or last Job ID that performed a provisioning action on this resource. This is useful when viewing the job log to monitor specific provisioning actions.

In the Fact Editor, this fact is listed as `resource.provision.jobid`:

```
<fact name="resource.provision.jobid" value="system.xen.74239" type="String" />
```

Automatic Provision: (Read Only) This check box is not selected by default. When it is selected (it has a value of true), the resource was cloned or provisioned automatically and will be shut down or destroyed automatically.

In the Fact Editor, this fact is listed as `resource.provision.automatic`:

```
<fact name="resource.provision.automatic" value="false" type="Boolean" />
```

Needs Resync: This check box is not selected by default. When it is selected (it has a value of true), you specify that the provisioned resource's state needs to be resynchronized by using the associated provisioning technology at the next opportunity.

In the Fact Editor, this fact is listed as `resource.provision.resync`:

```
<fact name="resource.provision.resync" value="false" type="Boolean" />
```

Linux Autoprep Config:

NOTE: This section displays when a Linux VM is selected.

If any of the fields in this section are blank (that is, undefined), click *Define* to install a fact editor that you can use to define the value.

This section includes the following settings:

- Linux Computer Name
- Linux Domain

The section also includes a subpanel where you can set Network Autoprep Configuration information.

Linux Computer Name: This value specifies the host name of a new VM. Enter "*" to indicate that the VM ID is to be used rather than the hostname you specify.

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.linuxglobal.ComputerName`:

```
<fact name="resource.provisioner.autoprep.linuxglobal.ComputerName" value="afd"
type="String" />
```

Linux Domain: This value specifies the domain to which the new VM belongs.

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.linuxglobal.Domain`:

```
<fact name="resource.provisioner.autoprep.linuxglobal.Domain" value=""
type="String" />
```

Network Autoprep Config

This section includes the following settings:

- DNS Server IP Addresses
- DNS Suffixes
- Gateway IP Addresses

DNS Server IP Addresses: This field displays a list of DNS server IP addresses for name lookup. This is only for cloning/personalize actions.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.provisioner.autoprep.DNSServers">
  <array>
    <string>0.0.00.200</string>
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open an array editor, where you can add or remove a server IP address or change its order in the array of element choices.

DNS Suffixes: The list of suffixes to append to a name for lookup.This is only for cloning/personalize actions.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.provisioner.autoprep.DNSSuffixes">
  <array>
    <string>afjkdl</string>
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open an array editor. In this dialog box you can add or remove a suffix or change its order in the array of element choices.

Gateway IP Addresses: The list of Internet gateways available to this VM. This is only for cloning/personalize actions.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.provisioner.autoprep.Gateways">
  <array>
    <string>afdasadfs</string>
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open an array editor. In this dialog box, you can add or remove the IP address or change its order in the array of element choices.

Windows Sysprep Config

NOTE: This section displays when a Windows VM is selected.

If any of the fields in this section are blank (that is, undefined), click *Define* to install a fact editor that you can use to define the value.

The section includes the following settings/facts:

- Change SID
- Delete Accounts
- Admin Password
- Admin Password Plaintext
- Timezone
- Autologon
- Autologon Count
- Fullname
- Org Name
- Computer Name
- Product ID
- Run Once Command
- Workgroup
- Domain
- Domain Admin
- Domain Admin Password
- Domain Admin Password Plaintext
- Machine Object OU
- Machine Password

- Machine Password Plaintext
- License File Automode
- License File Autousers

The section also includes a subpanel where you can set Network Sysprep Configuration information.

Change SID: The Windows Security ID. If the value is true, sysprep generates a new Security ID.

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.options.changeSID`:

```
<fact name="resource.provisioner.autoprep.options.changeSID" value="false"
type="Boolean" />
```

Delete Accounts: If it is set to true, this fact removes all accounts from the destination VM. If it is false, it retains existing accounts from the source VM.

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.options.deleteAccounts`:

```
<fact name="resource.provisioner.autoprep.options.deleteAccounts" value="true"
type="Boolean" />
```

Admin Password: This field displays the IP address for the adapter.

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.sysprep.GuiUnattended.AdminPassword.value`:

```
<fact
name="resource.provisioner.autoprep.sysprep.GuiUnattended.AdminPassword.value"
value="klvm" type="String" />
```

Admin Password Plaintext: This field displays the subnet mask for the adapter.

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.sysprep.GuiUnattended.AdminPassword.plainText`:

```
<fact
name="resource.provisioner.autoprep.sysprep.GuiUnattended.AdminPassword.plainText"
value="false" type="Boolean" />
```

Timezone: The time zone of the new VM. See Microsoft [GUI Unattended] (Sysprep) product documentation (http://technet.microsoft.com/en-us/library/cc772783%28WS.10%29.aspx). (Scroll to the *TimeZone* heading on that page.)

If you do not specify a value for this fact, the default value is 004 (Pacific Standard Time).

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.sysprep.GuiUnattended.TimeZone`:

```
<fact name="resource.provisioner.autoprep.sysprep.GuiUnattended.TimeZone"
value="10" type="String" />
```

Autologon: If the value is true, the VM automatically logs into the Administrator account using Admin Password. If it is false, logon is prompted.

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.sysprep.GuiUnattended.AutoLogon`:

```
<fact name="resource.provisioner.autoprep.sysprep.GuiUnattended.AutoLogon"
value="true" type="Boolean" />
```

Autologon Count: The limit count for the VM to auto log on with the Administrator account. *AutoLogon* must be True.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.GuiUnattended.AutoLogonCount`:

```
<fact name="resource.provisioner.autoprep.sysprep.GuiUnattended.AutoLogonCount"
value="2" type="Integer" />
```

Fullname: The user's full name.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.UserData.FullName`:

```
<fact name="resource.provisioner.autoprep.sysprep.UserData.FullName" value="adfkl"
type="String" />
```

Org Name: The organization name.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.UserData.OrgName`:

```
<fact name="resource.provisioner.autoprep.sysprep.UserData.OrgName" value="Novell"
type="String" />>
```

Computer Name: The VM's new host name. An asterisk (*) means to generate a name based on the source VM name.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.UserData.ComputerName`:

```
<fact name="resource.provisioner.autoprep.sysprep.UserData.ComputerName"
value="docdev1" type="String" />
```

Product ID: The Windows product key.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.UserData.ProductID`:

```
<fact name="resource.provisioner.autoprep.sysprep.UserData.ProductID"
value="jklaieuqa4354" type="String" />
```

Run Once Command: A list of commands that run the first time a user logs on after the new VM is created. Commands are scheduled using the `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce` registry key.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.GuiRunOnce.Command`:

```
<fact name="resource.provisioner.autoprep.sysprep.GuiRunOnce.Command"
value="purge" type="String" />>
```

Workgroup: Windows workgroup name. If the VM is joining a domain, use `JoinDomain`.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.Identification.JoinWorkgroup`:

```
<fact name="resource.provisioner.autoprep.sysprep.Identification.JoinWorkgroup"
value="prod" type="String" />
```

Domain: Windows domain name. If the VM is joining a workgroup, use `JoinWorkgroup`. For joining a domain, `DomainAdmin` and `DomainAdminPassword` must be defined.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.Identification.JoinDomain`:

```
<fact name="resource.provisioner.autoprep.sysprep.Identification.JoinDomain"
value="test" type="String" />>
```

Domain Admin: The Windows domain administrator name.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.Identification.DomainAdmin`:

```
<fact name="resource.provisioner.autoprep.sysprep.Identification.DomainAdmin"
value="admin" type="String" />
```

Domain Admin Password: The Windows domain administrator account password.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.Identification.DomainAdminPassword.value`:

```
<fact
name="resource.provisioner.autoprep.sysprep.Identification.DomainAdminPassword.val
ue" value="cleanwindow" type="String" />
```

Domain Admin Password Plaintext: Select the check box if `DomainAdminPassword` is in plain text.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.Identification.DomainAdminPassword.plainText`
:

```
<fact
name="resource.provisioner.autoprep.sysprep.Identification.DomainAdminPassword.pla
inText" value="false" type="Boolean" />
```

Machine Object OU: Provide the organizational unit (OU) of the Windows Active Directory machine.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.Identification.MachineObjectOU`

```
<fact name="resource.provisioner.autoprep.sysprep.Identification.MachineObjectOU"
value="dd" type="String" />
```

Machine Password: Provide the account password for the Windows Active Directory machine.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.Identification.MachinePassword.value`

```
<fact
name="resource.provisioner.autoprep.sysprep.Identification.MachinePassword.value"
value="fad" type="String" />
```

Machine Password Plaintext: Select the check box if `MachinePassword` is in plain text.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.Identification.MachinePassword.plainText`

```
<fact
name="resource.provisioner.autoprep.sysprep.Identification.MachinePassword.plainTe
xt" value="true" type="Boolean" />
```

License File Automode: Enter either PerServer or PerSeat. If you enter PerServer, `AutoUsers` must be set.

In the Fact Editor, this fact is listed as
`resource.provisioner.autoprep.sysprep.LicenseFilePrintData.AutoMode`:

```
<fact name="resource.provisioner.autoprep.sysprep.LicenseFilePrintData.AutoMode"
value="PerSeat" type="String" />
```

License File Autousers: The number of client licenses. Use this setting only if `AutoMode` is PerServer.

In the Fact Editor, this fact is listed as `resource.provisioner.autoprep.sysprep.LicenseFilePrintData.AutoUsers`:

```
<fact name="resource.provisioner.autoprep.sysprep.LicenseFilePrintData.AutoUsers"
value="33" type="Integer" />
```

Network Sysprep Config

This section includes the following settings:

- DNS Suffixes

DNS Suffixes: The list of suffixes to append to a name for lookup.This is only for cloning/ personalize actions.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.provisioner.autoprep.DNSSuffixes">
  <array>
    <string>afjkdl</string>
  </array>
</fact>
```

You can edit this array by clicking the ... button to open an array editor, where you can add or remove a suffix or change its order in the array of element choices.

OS Information

This section includes the following settings:

- OS Name
- OS Version
- OS Version String
- OS Architecture
- OS Family
- OS Type
- OS Vendor
- OS Vendor Version
- OS Vendor String
- OS File Path Separator

OS Name: (Read Only) The name of the resource operating system.

In the Fact Editor, this fact is listed as `resource.os.name`:

```
<fact name="resource.os.name" value="Windows" type="String" />
```

OS Version: (Read Only) The version number of the resource operating system. For a VM resource, this fact remains undefined.

In the Fact Editor, this fact is listed as `resource.os.version`:

```
<fact name="resource.os.version.string" value="Microsoft Windows XP [Version
5.1.2600]" type="String" />
```

OS Version String: (Read Only) The operating system vendor full identification string (requires the osInfo system job).

In the Fact Editor, this fact is listed as `resource.os.version.string`:

```
<fact name="resource.os.vendor.string" value="Microsoft Windows XP [Version
5.1.2600]" type="String" />
```

OS Architecture: The operating system architecture (for example, *x86*, *amd64*, *i386*, or *sparc*).

In the Fact Editor, this fact is listed as `resource.os.arch`:

```
<fact name="resource.os.arch" value="i386" type="String" />
```

OS Family: The operating system family name (for example, *windows*, *linux*, *solaris*, *unix*, *aix*, *mac*) of the resource, if known.

In the Fact Editor, this fact is listed as `resource.os.family`:

```
<fact name="resource.os.family" value="linux" type="String" />
```

OS Type: This drop-down list lets you select the unique string identifier for each OS release, for example, *sles11*.

In the Fact Editor, this fact is listed as `resource.os.type`:

```
<fact name="resource.os.type" value="sles10" type="String" />
```

OS Vendor: (Read Only) The operating system vendor. SuSE for SUSE Linux Enterprise Server or SUSE Linux Enterprise Desktop.

In the Fact Editor, this fact is listed as `resource.os.vendor`:

```
<fact name="resource.os.vendor" value="SuSE" type="String" />
```

OS Vendor Version: (Read Only) This field displays the vendor-defined version for the operating system. Ffor example, *11* for SUSE Linux Enterprise Server 11.

In the Fact Editor, this fact is listed as `resource.os.vendor.version`:

```
<fact name="resource.os.vendor.version" value="10" type="String" />
```

OS Vendor String: (Read Only) This field displays the full identification for the operating system that is supplied by the vendor. The osinfo system job must run for this value to be displayed.

In the Fact Editor, this fact is listed as `resource.os.vendor.string`:

```
<fact name="resource.os.vendor.string" value="Welcome to SUSE Linux Enterprise
Server 11 (i586) - Kernel  (\l)." type="String" />
```

OS File Path Separator: (Read Only) The resource operating system file separator.

In the Fact Editor, this fact is listed as `resource.os.file.separator`:

```
<fact name="resource.os.file.separator" value="/" type="String" />
```

CPU Information

This section includes the following settings:

- Number of CPUs
- CPU Speed (Mhz)

- CPU Vendor
- CPU Model
- CPU Architecture
- CPU HVM Support

Number of CPUs: (Read only) The number of CPUs available for this resource to use. For a VM resource, this fact remains undefined.

In the Fact Editor, this fact is listed as `resource.cpu.number`:

```
<fact name="resource.cpu.number" value="2" type="Integer" />
```

CPU Speed (Mhz): (Read only) The processor speed measured in Mhz. The cpuinfo job must run for this value to be displayed. For a VM resource, this fact remains undefined.

In the Fact Editor, this fact is listed as `resource.cpu.mhz`:

```
<fact name="resource.cpu.mhz" value="2594" type="Integer" />
```

CPU Vendor: (Read only) The name of the CPU vendor. The cpuinfo system job must run for this value to be displayed. For a VM resource, this fact remains undefined.

In the Fact Editor, this fact is listed as `resource.cpu.vendor`:

```
<fact name="resource.cpu.vendor" value="GenuineIntel" type="String" />
```

CPU Model: (Read only) The full vendor model number of the CPU. The cpuinfo system job must run for this value to be displayed. For a VM resource, this fact remains undefined.

In the Fact Editor, this fact is listed as `resource.cpu.model`:

```
<fact name="resource.cpu.model" value="Intel(R) Pentium(R) 4 CPU 2.60GHz"
type="String" />
```

CPU Architecture: The CPU architecture (for example, *x86*, *x86_64*, *sparc*) of this resource. For a VM resource, this fact remains undefined.

In the Fact Editor, this fact is listed as `resource.cpu.architecture`:

```
<fact name="resource.cpu.architecture" value="x86" type="String" />
```

CPU HVM Support: This field is marked true if the CPU has hardware virtualization support.

In the Fact Editor, this fact is listed as `resource.cpu.hvm`:

```
<fact name="resource.cpu.hvm" value="false" type="Boolean" />
```

Memory Information

This section includes the following settings:

- Virtual Memory (Mb)
- Virtual Available
- Physical Memory (Mb)
- Physical Available
- Swap Memory (Mb)
- Swap Available

Virtual Memory (Mb): (Read only) The total amount of virtual memory, measured in MB, on the resource. The memInfo system job must run for this value to be displayed

In the Fact Editor, this fact is listed as `resource.memory.virtual.total`:

```
<fact name="resource.memory.virtual.total" value="4060" type="Integer" />
```

Virtual Available: (Read only) The amount of available virtual memory, measured in MB, on the resource. The memInfo system job must run for this value to be displayed.

In the Fact Editor, this fact is listed as `resource.memory.virtual.available`:

```
<fact name="resource.memory.virtual.available" value="19951" type="Integer" />
```

Physical Memory (Mb): (Read only) The total amount of physical memory, measured in MB, on the resource. The memInfo system job must run for this value to be displayed

In the Fact Editor, this fact is listed as `resource.memory.physical.total`:

```
<fact name="resource.memory.physical.total" value="3889" type="Integer" />
```

Physical Available: (Read only) The amount of available physical memory, measured in MB, on the resource. The memInfo system job must run for this value to be displayed.

In the Fact Editor, this fact is listed as `resource.memory.physical.available`:

```
<fact name="resource.memory.physical.available" value="3565" type="Integer" />
```

Swap Memory (Mb): (Read only) The total amount of configured swap space, measured in MB, on the resource. The memInfo system job must run for this value to be displayed.

In the Fact Editor, this fact is listed as `resource.memory.swap.total`:

```
<fact name="resource.memory.swap.total" value="16386" type="Integer" />
```

Swap Available: (Read only) The total amount of free swap space, measured in MB, on the resource. The memInfo system job must run for this value to be displayed.

In the Fact Editor, this fact is listed as `resource.memory.swap.available`:

```
<fact name="resource.memory.swap.available" value="16386" type="Integer" />
```

Disk/Network Information

The facts in the *Disk/Network Information* section of the Info/Groups page are not currently functional and are not supported.

Agent Information

This section includes the following settings:

- Agent Version
- Agent Install Dir
- Agent Java Version
- Agent Java Runtime
- Agent Java Vendor
- Agent Java Home Dir

- Available Agent Memory
- Enhanced Exec Available
- Clustered Agent

Agent Version: (Read only) The Orchestration Agent version and build number that is installed on this resource. The string uses the following syntax:

major.minor.point_build

In the Fact Editor, this fact is listed as `resource.agent.version`:

```
<fact name="resource.agent.version" value="2.0.2_70917" type="String" />
```

Agent Install Dir: (Read only) The name of the home directory of the Orchestration Agent installation files.

In the Fact Editor, this fact is listed as `resource.agent.home`:

```
<fact name="resource.agent.home" value="/opt/novell/zenworks/zos/agent"
type="String" />
```

Agent Java Version: (Read only) The version of the Java JVM currently in use by the Orchestration Agent installed on this resource.

In the Fact Editor, this fact is listed as `resource.agent.jvm.version`:

```
<fact name="resource.agent.jvm.version" value="1.5.0_17" type="String" />
```

Agent Java Runtime: (Read only) The version of the Java JVM runtime currently in use by the Orchestration Agent installed on this resource.

In the Fact Editor, this fact is listed as `resource.agent.jvm.runtime`:

```
<fact name="resource.agent.jvm.runtime" value="1.5.0_17-b04" type="String" />
```

Agent Java Vendor: (Read only) The name of the vendor of the Java JVM currently in use by the Orchestration Agent installed on this resource.

In the Fact Editor, this fact is listed as `resource.agent.jvm.vendor`:

```
<fact name="resource.agent.jvm.vendor" value="Sun Microsystems Inc." type="String"
/>
```

Agent Java Home Dir: (Read only) The path to the home directory of the Java JVM currently in use by the Orchestration Agent installed on this resource.

In the Fact Editor, this fact is listed as `resource.agent.jvm.home`:

```
<fact name="resource.agent.jvm.home" value="/opt/novell/zenworks/zos/agent/jre"
type="String" />
```

Available Agent Memory: (Read only) The amount of memory, measured in MB, available to the Orchestration Agent installed on this resource.

In the Fact Editor, this fact is listed as `resource.agent.jvm.memory`:

```
<fact name="resource.agent.jvm.memory" value="127" type="Integer" />
```

Enhanced Exec Available: This check box is selected by default. When it is selected (it has a value of true), it indicates that the Orchestration Agent installed on this resource is able to use enhanced exec features, as opposed to unsupported agent installs, such as AIX.

In the Fact Editor, this fact is listed as `resource.agent.exec.installed`:

```
<fact name="resource.agent.exec.installed" value="true" type="Boolean" />
```

Clustered Agent: This check box is not selected by default. When you select it (it has a value of true), you specify that the agent is "clustered" on this VM resource. This means that it converts duplicate logins to failover logins.

In the Fact Editor, this fact is listed as `resource.agent.clustered`:

```
<fact name="resource.agent.clustered" value="false" type="Boolean" />
```

Agent Configuration

This section includes the following settings:

- Gmond Port
- Datagrid Cache TTL
- Datagrid Cleanup Interval
- Exec Daemon Timeout
- Exec As Agent User Only
- Cleanup After Joblets
- Use Enhanced Exec
- Log Level
- Debug Logging

Gmond Port: The port that the agent uses for gmond. Port 8649 is the default port. A setting of zero (0) or less means that the value is not read.

In the Fact Editor, this fact is listed as `esource.agent.config.gmond.port`:

```
<fact name="resource.agent.config.gmond.port" value="8649" type="Integer" />
```

Datagrid Cache TTL: The amount of time (measured in minutes) that inactive files should remain in the agent's datagrid cache. A setting of zero (0) turns off the cache.

In the Fact Editor, this fact is listed as `jresource.agent.config.datagrid.cache.lifetime`:

```
<fact name="resource.agent.config.datagrid.cache.lifetime" value="1440"
type="Integer" />
```

Datagrid Cleanup Interval: The amount of time (measured in minutes) that the Orchestration Server should wait between cleanup sweeps of the agent's datagrid cache.

In the Fact Editor, this fact is listed as `resource.agent.config.datagrid.cache.cleanupinterval`:

```
<fact name="resource.agent.config.datagrid.cache.cleanupinterval" value="60"
type="Integer" />
```

Exec Daemon Timeout: The amount of time (measured in seconds) that the enhanced exec daemon is to remain running. A setting of zero (0) specifies that the daemon is to remain running. The exec daemon is the non-Java component of the agent that is responsible for executing commands remotely.

In the Fact Editor, this fact is listed as `resource.agent.config.exec.daemon.timeout`:

```
<fact name="resource.agent.config.exec.daemon.timeout" value="300" type="Integer"
/>
```

Exec As Agent User Only: This check box is selected by default. When you select it (it has a value of true), you specify that the agent is to always run executables as the Agent User only. Selecting this check box overrides any job fact settings for the `job.joblet.runtype` fact.

In the Fact Editor, this fact is listed as `resource.agent.config.exec.asagentuseronly`:

```
<fact name="resource.agent.config.exec.asagentuseronly" value="true"
type="Boolean" />
```

Cleanup After Joblets: This check box is not selected by default. When you select it (it has a value of true), you specify that the agent on this resource is to clean up temporary directories created for each joblet. You can deselect this check box for debugging purposes; when you select it again, the cleanup process starts again, deleting temporary directories that were created while the setting was deactivated.

In the Fact Editor, this fact is listed as `resource.agent.config.joblet.cleanup`:

```
<fact name="resource.agent.config.joblet.cleanup" value="true" type="Boolean" />
```

Use Enhanced Exec: This check box is not selected by default. When you select it (it has a value of true), you specify that the agent on this resource is to use the enhanced exec feature of the agent, which is available for supported agent installations. Marking this fact as false causes the enhanced exec feature to not be used.

In the Fact Editor, this fact is listed as `resource.agent.config.exec.enhancedused`:

```
<fact name="resource.agent.config.exec.enhancedused" value="true" type="Boolean" /
>
```

Log Level: Lets you choose the level of agent logging in terms of the amount of detail (that is, the verbosity) you want to include in the agent log. The choices include:

- quiet
- normal
- verbose

In the Fact Editor, this fact is listed as `resource.agent.config.loglevel`:

```
<fact name="resource.agent.config.loglevel" value="normal" type="String" />
```

Debug Logging: This check box is not selected by default. When you select it (it has a value of true), you activate the debug function in the agent log, which is additive to the log level.

In the Fact Editor, this fact is listed as `resource.agent.config.logdebug`:

```
<fact name="resource.agent.config.logdebug" value="false" type="Boolean" />
```

Installed Components

Applications: A list of the names of applications (including the full version name) that are installed on this resource. This is useful for constraining joblets to run only on a resource with a particular application installed.

In the Fact Editor, this fact is listed as an array:

```
<fact name="resource.installed.apps">
  <array>
    <string>man-pages-2.39-0.9</string>
    <string>xorg-x11-fonts-scalable-6.9.0-50.45</string>
    <string>cifs-mount-3.0.24-2.23</string>
    <string>gdbm-1.8.3-243.2</string>
    <string>libaio-0.3.104-14.2</string>
    <string>libnl-1.0-18.4</string>
...
  </array>
</fact>
```

You can edit this array by clicking the ... button to open an array editor, where you can add or remove the application name or change its order in the array of element choices.

5.2.2 Groups Panel

This section of the Info/Groups page lists the groups of Resource objects in the grid. Click *Choose* to open the Resource Group Selection dialog box. In this dialog box, you can choose which Resource Groups to display in the Explorer Panel by selecting a group and then clicking *Add* or *Remove* to move it to or from the Source Resource Groups list.

5.3 Provision Info Page

The *Provision Info* tab is displayed only for VM resource objects selected in the Explorer tree. The read-only fields displayed at the top of the Provision Info page summarize information related to the provisioning of the resource, whether that resource is a VM or VM template.

The page has several subtabs that open other pages that display further information about the VM:

- **Show Log:** A log that includes historical details about the history of the provisioning of the VM.
- **Host Assignment Log:** A log of VM host assignment constraint errors of the last migration or provision action.
- **Autoprep Data:** A list of autoprep/sysprep fact overrides used to prepare this VM instance. For more information about autoprep, see "Understanding and Configuring Autoprep" in the *NetIQ Cloud Manager 2.1 Orchestration Installation Guide*.
- **Policy Debugger:** Information to help you debug the policy and its resulting constraints and facts that are associated with this VM. For more information, see Chapter 14, "The Policy Debugger," on page 179.
- **Action History:** A log that includes historical details about the history of the provisioning of the VM. For more information, see "Provisioning Actions and History" in the *NetIQ Cloud Manager 2.1 VM Orchestration Reference*.

5.4 Resource Log Page

Open the *Resource Log* tab to view the contents of the log file for this resource. You can click *Refresh* to update the content. You can also select *Debug Logging* to activate the debug feature as part of the logging or change the *Log Level* to the level of detail that you want.

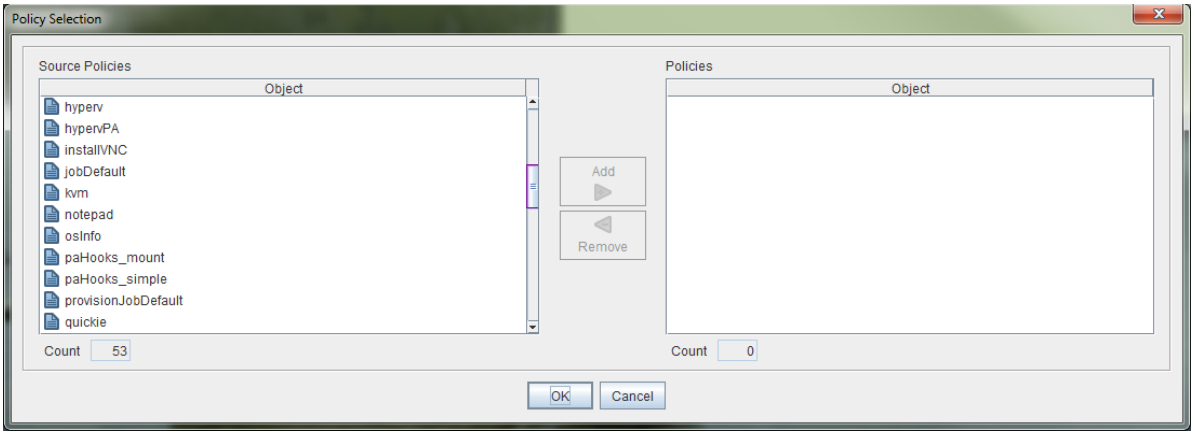
5.5 Resource Policies Page

The *Policies* tab of the Resource admin view opens a page that contains a policy viewer for each of the policies associated with a Resource Grid object.

You can modify a policy by using the Policy Grid object. For more information, see Section 12.1, "Policy Object," on page 153.

Click *Choose* in the admin view of the Policy viewer to launch a Policy Selection dialog box where you can add or remove individual policies to be applied to the selected Resource Grid object.

Figure 5-1 *The Policy Selection Dialog Box*



5.6 Resource Health Debugger Page

The Health Debugger is a common Admin view in the Orchestration Console for most Grid objects. For information about this tool, see Health Debugger in Appendix A, "Grid Object Health Monitoring," on page 187.

5.7 Resource Constraints/Facts Page

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties. By changing the policy constraints and fact values for a job, you can change the behavior of the job and how the Orchestration Server allocates available system resources to it. The Orchestration Server assigns default values to each of the component facts. Facts with no mode specified can be changed at any time by the administrator.Facts with mode `r/o` have read-only values, which can be viewed by using the edit pencil icon, but changes cannot be made.

For information about renaming Resource objects using the Fact Editor on this page, see Section 5.8, "Resource Object Naming and Renaming," on page 80.

5.8 Resource Object Naming and Renaming

The Orchestration Server Resource Grid object type can include resources of various types, including physical machines, virtual machines (VMs), and VM templates, all of which are modeled differently in the Orchestration Console because of their varying roles in the Orchestration Server system. Some

resource names are generated by the Orchestration Server system and can therefore receive generic, arbitrary names such as `mysql-1`, `mysql-2`, and so on. Resources you name at installation time or creation time might also change in their purpose or facilities.

As the quantity of these Resource objects grows in your grid, you might find it helpful or necessary to rename them, assigning more meaningful, intuitive names to suit the purpose of the object. The object's "display name" is visible in the Orchestration Console interface, the VM Client interface, the Server Portal, and in optional zos and zosadmin commands.

NOTE: Resource object groups (that is, the folders that contain these Resource objects) can also be renamed. Objects such as jobs, events, and users cannot be renamed.

A Resource object's name is stored in the `${objectType}.displayname` fact, which exists on every Grid object type, even those objects that cannot be renamed.

You can rename a Resource object in the Orchestration Console by using one of three methods:

- Right-click the Resource object in the Explorer tree, then select *Rename* to allow editing of the display name.
- Triple-click the Resource object in the Explorer tree to allow editing of the display name.
- In the Constraints/Facts page, select the resource object `.displayname` fact and then open the Fact Editor to enter a new value for that fact.

As you use one of these methods, you will notice that the fact value is prepopulated with the `${objectType}.id` fact. This functions as the name value for the object name until you decide to change it.

NOTE: Even after being renamed, the Resource object retains its associated resource ID in the `.id` fact. This is not editable.

For more information about making the Resource object display names visible from the zos or zosadmin command line, see the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

6 The VM Host Object

A VM host represents a VM host technology or hypervisor (for example, Xen, Hyper-V, and so on) either installed on a physical resource or accessed by it (in the case of VMware). VM host objects can be used when making provisioning decisions for a resource.

The Cloud Manager Orchestration Server also supports the discovery of VMware vSphere clusters used for high availability in a VMware environment or managed by the VMware Distributed Resource Scheduler (DRS) after an Orchestration Agent has been deployed into such an environment. In this scenario, the Orchestration Server also allows you to determine when actions have taken place outside of the Orchestration environment, such as when the DRS moves a VM to an alternate host in the cluster or when an administrator moves a VM into a different resource pool (see "Setting Up Orchestration to Accommodate VMware DRS Clustering and Updates" in the *NetIQ Cloud Manager 2.1 Orchestration Installation Guide*).

Although the VM host and the VM host Cluster are regarded as two different types of VM host object, and have differing icons, the discovered clusters are represented in the Explorer tree of the Orchestration Console as VM host objects.

NOTE: The Orchestration Console interface (that is, the fields in the admin view) for a VM host and a VM host Cluster are nearly identical. Facts unique to the VM host Cluster are listed in Section 6.7, "Unique VM Host Cluster Facts," on page 90.

This section includes the following information:

6.1 Info Page

The page that opens under the *Info* tab includes several collapsible sections on the page where you can configure the general information and attributes of the VM host.

- Section 6.1.3, “Provisioning Adapter Config Panel,” on page 87
- Section 6.1.4, “Guest VM Monitor Information Panel,” on page 87

NOTE: Whenever you make changes to any Grid object, the write icon  is superimposed on the object's icon, signifying that the object has been altered. If you want to save the changes you have made, you need to click the Save button  on the Orchestration Console toolbar.

6.1.1 Show Inherited Fact Values Check Box

Select this check box to show facts with overridden values supplied through attached or inherited policies. These fact values are read only (non-editable), although you can use the Policy Editor to modify the policy values themselves if you want to.

6.1.2 VM Host Information Panel

The *VM Host Information* panel on the Info page includes the following fields:

- Physical Resource
- VM Host Type
- VmHost Cluster
- Enabled
- Online
- Healthy
- Shutting Down
- Location
- Supports VM Migration
- Supports H/W HVM
- Accounting Group
- Max Hosted VMs
- Max Hosted vCPUs
- Max Virtual Memory
- Repositories
- Available VM Resource Groups
- Managing Job
- Needs Resync

NOTE: Tooltip text is available when you mouse over any of these fields.

Physical Resource: (Read Only) The name of the resource that houses this VM host container.

In the Fact Editor, this fact is listed as `vmhost.resource`:

```
<fact name="vmhost.resource" value="vmh7sles" type="String" />
```

VM Host Type: This field displays a read-only fact, as discovered by the Orchestration Server. It identifies the VM host as a regular VM host (*vmhost*) or as a VMware cluster (*vmhostcluster*).

In the Fact Editor, this fact is listed as `vmhost.type`:

```
<fact name="vmhost.type" value="vmhostcluster" type="String" />
```

VmHost Cluster: (Conditional) This field displays a read-only fact, as discovered by the Orchestration Server. It identifies the VM host cluster to which this VM host belongs. The field is displayed on the information page only when the VM host is a member of a cluster.

In the fact editor, this fact is listed as `vmhost.cluster`:

```
<fact name="vmhost.cluster" value="esx35_cluster_vsphere" type="String" />
```

Enabled: This check box is selected by default. When it is selected (it has a value of true), the VM host is enabled, which means that VM instances can be provisioned on it.

In the Fact Editor, this fact is listed as `vmhost.enabled`:

```
<fact name="vmhost.enabled" value="true" type="Boolean" />
```

Online: When this check box is selected (it has a value of true), the agent on the physical resource is online.

In the Fact Editor, this fact is listed as `vmhost.online`:

```
<fact name="vmhost.online" value="true" type="Boolean" />
```

Healthy: This check box is selected by default. When it is selected (it has a value of true), the VM host is designated as being in good health. You can set the health of the object by selecting or deselecting the health check box. Changing the value in this way has an immediate effect unless the value is overriden by an attached policy. For more information, see Appendix A, “Grid Object Health Monitoring,” on page 187

In the Fact Editor, this is fact is listed as `vmhost.health`:

```
<fact name="vmhost.health" value="false" type="Boolean" />
```

Shutting Down: When this check box is selected (it has a value of true), the VM host is attempting to shut down and does not accept provisioning requests.

In the Fact Editor, this fact is listed as `vmhost.shuttingdown`:

```
<fact name="vmhost.shuttingdown" value="false" type="Boolean" />
```

Location: For the vsphere provisioning adapter, this is the `ManagedObjectReference` path for this VM host. For other provisioning adapters, this is an optional description of the physical location of the VM host.

In the Fact Editor, this fact is listed as `vmhost.location`:

```
<fact name="vmhost.location" value="" type="String" />
```

Supports VM Migration: When this check box is selected (it has a value of true), the VM host can support VM migration. The state of this fact can also depend on the migration capabilities of the provisioning adapter used to provision the VM.

In the Fact Editor, this fact is listed as `vmhost.migration`:

```
<fact name="vmhost.migration" value="true" type="Boolean" />
```

Supports H/W HVM: When this check box is selected (it has a value of true), the hypervisor on the VM host can support hardware virtualization.

In the Fact Editor, this fact is listed as `vmhost.hvm`:

```
<fact name="vmhost.hvm" value="false" type="Boolean" />
```

Accounting Group: The default VM host group that you want to be adjusted for VM tracking statistics.

In the Fact Editor, this fact is listed as `vmhost.accountinggroup`:

```
<fact name="vmhost.accountinggroup" value="all" type="String" />
```

Max Hosted VMs: The maximum number of VM instances allowed on this VM host.

In the Fact Editor, this fact is listed as `vmhost.maxvmslots`:

```
<fact name="vmhost.maxvmslots" value="8" type="Integer" />
```

Max Hosted vCPUs: The maximum number of virtual CPUs that this VM host can support.

In the Fact Editor, this fact is listed as `vmhost.vcpu.max`:

```
<fact name="vmhost.vcpu.max" value="8" type="Integer" />
```

For more information about vCPU slots, see Section 6.8, "vCPU Slots for VM Hosts," on page 93.

Max Virtual Memory: The amount of memory (measured in MB) available to hosted VMs.

In the Fact Editor, this fact is listed as `vmhost.memory.max`:

```
<fact name="vmhost.memory.max" value="1000" type="Integer" />
```

Repositories: The list of repositories (VM disk stores) that are visible to this VM host.

In the Fact Editor, this fact is listed as an array:

```
<fact name="vmhost.repositories">
  <array type="String">
  </array>
</fact>
```

You can edit this array by clicking the ... button to open the Choose Grid Objects dialog box, where you can add, remove, or edit repositories in an array of repository choices.

Available VM Resource Groups: This field displays a list of resource groups containing VMs that are allowed to run on this VM host.

In the Fact Editor, this fact is listed as an array:

```
<fact name="vmhost.vm.available.groups">
  <array type="String">
  </array>
</fact>
```

You can edit this array by clicking the ... button to open the Choose Grid Objects dialog box, where you can add, remove, or edit the resource groups (element values) in an array of choices.

Managing Job: The ID of a running job that manages VM operations on this VM host. When this field is completed, the VM Manager prevents other jobs from initiating provisioning actions. The fact is cleared when the managing job ends.

In the Fact Editor, this fact is listed as `vmhost.controllingjob`:

```
<fact name="vmhost.controllingjob" value="" type="String" />
```

Needs Resync: When this check box is selected (it has a value of true), you specify that, at the next opportunity, this VM host is to be probed to resynchronize all the VMs that are managed here.

In the Fact Editor, this fact is listed as `vmhost.resync`:

```
<fact name="vmhost.resync" value="false" type="Boolean" />
```

6.1.3 Provisioning Adapter Config Panel

- Adapter Job Name
- Username
- Password

Adapter Job Name: The name of the provisioning adapter job that manages VM discovery on this host. Do not change this value unless you have implemented your own discovery job.

In the Fact Editor, this fact is listed as `vmhost.provisioner.job`:

```
<fact name="vmhost.provisioner.job" value="vsphere" type="String" />
```

Username: (Optional) The username required for provisioning on the VM host.

In the Fact Editor, this fact is listed as `vmhost.provisioner.username`:

```
<fact name="vmhost.provisioner.username" value="" type="String" />
```

Password: (Optional) The password required for provisioning on the VM host.

In the Fact Editor, this fact is listed as `vmhost.provisioner.password`:

```
<fact name="vmhost.provisioner.password" value="" type="String" />
```

6.1.4 Guest VM Monitor Information Panel

- Current VM Count
- Available vCPUs
- Available Virtual Memory
- VM Image Counts
- Running VM Instances
- Load Index (Slots)
- Load Index (Memory)

Current VM Count: (Read Only) The current number of active VM instances.

In the Fact Editor, this fact is listed as `vmhost.vm.count`:

```
<fact name="vmhost.vm.count" value="0" type="Integer" />
```

Available vCPUs: The number of vCPUs available on this VM host.

In the Fact Editor, this fact is listed as `vmhost.vcpu.available`:

```
<fact name="vmhost.vcpu.available" value="8" type="Integer" />
```

For more information, see Section 6.8, “vCPU Slots for VM Hosts,” on page 93.

Available Virtual Memory: The amount of memory (measured in MB) available to new VMs.

In the Fact Editor, this fact is listed as `vmhost.memory.available`:

```
<fact name="vmhost.memory.available" value="1000" type="Integer" />
```

VM Image Counts: The dictionary of running instance counts for each running VM template.

In the Fact Editor, this fact is listed as a dictionary:

```
<fact name="vmhost.vm.templatecounts">
  <dictionary>
    <dictelement key="ads">
      <time>12:00 AM</time>
    </dictelement>
  </dictionary>
</fact>
```

You can edit the dictionary elements by clicking the ··· button to open the VM Image Counts dialog box, then adding or removing the names in the dictionary.

Running VM Instances: (Read Only) A list of active VM instances.

In the Fact Editor, this fact is listed as an array:

```
<fact name="vmhost.vm.instanceids">
  <array type="String">
  </array>
</fact>
```

Load Index (Slots): (Read Only) The current loading index of resource slots, which is a ratio of the active hosted VMs to the specified maximum number of VMs allowed on this host. Each provision VM takes up one slot. For more information, see *Max Hosted VMs*.

In the Fact Editor, this fact is listed as `vmhost.loadindex.slots`:

```
<fact name="vmhost.loadindex.slots" value="0.1250" type="Real" />
```

Load Index (Memory): (Read Only) The current loading index for memory, which is a ratio of the virtual memory consumed on this VM host to the specified maximum amount of memory allocated to this host.

In the Fact Editor, this fact is listed as `vmhost.loadindex.virtualmemory`:

```
<fact name="vmhost.loadindex.virtualmemory" value="0.0000" type="Real" />
```

6.2 Policies Page

The *Policies* tab opens a page that contains a policy viewer for each of the policies associated with a Grid object.

NOTE: You can edit a policy by right-clicking a policy icon and selecting *Edit Policy*. Remember to click the Save button when your changes are complete.

6.3 Health Debugger Page

The Health Debugger is a common Admin view in the Orchestration Console for most Grid objects. For information about this tool, see Health Debugger in Appendix A, "Grid Object Health Monitoring," on page 187.

6.4 Constraints/Facts Page

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties. By building, deploying, and running jobs on the Orchestration Server, you can individually change the functionality of any system resources by managing an object's facts and constraints. The Orchestration Server assigns default values to each of the component facts. Facts with no mode listed can be changed at any time by the administrator. Facts with mode `r/o` have read-only values, which can be viewed by using the pencil icon, but changes cannot be made.

For information about using the Fact Editor on this page to rename VM host objects, see Section 6.6, "VM Host Object Naming and Renaming," on page 89

6.5 Action History Page

The *Action History* tab is displayed in the administrative view of the Repository object. When you select the *Action History* tab, a table displays on the Action History page with a list of the history for all VM provisioning actions performed on this Grid object.

The Orchestration Server must be connected to an audit database for the *Include Audit Database* check box to be available. If the *Include Audit Database* check box is selected in this view, the action status is not polled. Click the Refresh button in the toolbar to retrieve and display fresh data.

For more details about the information listed on the Action History page, see "Action History in Admin Views of the Orchestration Console" in the *NetIQ Cloud Manager 2.1 VM Orchestration Reference*.

6.6 VM Host Object Naming and Renaming

Some VM host names (or VM host Cluster names) are generated by the Orchestration Server and can therefore receive generic, arbitrary names such as `host2_demoAdapter`, `host3_demoAdapter`, and so on. VM hosts you name at creation time might also change later in their purpose or facilities.

As the quantity of these VM host objects grows in your grid, you might find it helpful or necessary to rename them, assigning more meaningful, intuitive names to suit the purpose of the object. The object's "display name" is visible in the Orchestration Console interface, the VM Client interface, and in optional zos and zosadmin commands.

NOTE: Resource object groups (that is, the folders that contain these VM host objects) can also be renamed. Objects such as jobs, events, and users cannot be renamed.

A VM host object's name is stored in the `${objectType}.displayname` fact, which exists on every Grid object type, even those objects that cannot be renamed.

You can rename a VM host object (or a VM host Cluster object) in the Orchestration Console by using one of three methods:

- Right-click the VM host object in the Explorer tree, then select *Rename* to allow editing of the display name.
- Triple-click the VM host object in the Explorer tree to allow editing of the display name.
- In the Constraints/Facts page, select the VM host object `.displayname` fact and then open the Fact Editor to enter a new value for that fact.

As you use one of these methods, you will notice that the fact value is prepopulated with the `${objectType}.id` fact. This functions as the name value for the object name until you decide to change it.

NOTE: Even after being renamed, the VM host object retains its associated resource ID in the `.id` fact. This is not editable.

For more information about making the Resource object display names visible from the zos or zosadmin command line, see the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

6.7 Unique VM Host Cluster Facts

There are several VM host cluster-related facts that are not found in a regular VM host object. This section contains detail about those facts.

6.7.1 Orchestration Server Facts in the VM Host Cluster Object

Any vSphere clusters discovered by the Orchestration Server are listed in the Orchestration Console as members of a convenience group (for example, a group named `clusters_vsphere`). The following table lists the read-only, cluster-related facts in a VM host Cluster object.

Table 6-1 *Cluster-Related Facts in a Cluster Object*

Fact Name	Type	Description
`vmhost.cluster.vmhosts`	String[]	This string array lists all of the VM hosts that are members of this cluster.
`vmhost.location`	String	The data center Managed Object Reference (MOR) path to the cluster.

Fact Name	Type	Description
vmhost.vphere.cluster.das.admission_control_enabled	Boolean	If this fact value is true, the VM host cluster has high availability configuration to use admission control. **NOTE**: High availability was originally named “Dynamic Availability Service” (DAS) in VMware. This is the origin of the .das prefix.
vmhost.vphere.cluster.das.enabled	Boolean	If this fact value is true, high availability is turned on in this cluster.
vmhost.vphere.cluster.drs.allow_behavior_override	Boolean	Whether VMs can specify their own placement behavior.
vmhost.vphere.cluster.drs.default_behavior	String	Specifies manual, partially automated, or fully automated VM placement.
vmhost.vphere.cluster.drs.enabled	Boolean	If this fact value is true, DRS is turned on in this cluster.
vmhost.vphere.pools	Dictionary	The mapping of the pool MOR path to dictionaries of pool configuration values.
vmhost.type	String	The “type” differentiation for clusters vs. VM hosts. The value is either vmhost or vmhostcluster.

6.7.2 Orchestration Server Facts in a VM Host Residing in a Cluster

The following table lists the read-only, cluster-related facts in a virtual or physical machine with an installed Orchestration Agent and residing in a cluster.

Table 6-2 *Cluster-Related Facts in a VM Host Residing in a vSphere Cluster*

Fact Name	Type	Description
`vmhost.cluster`	String[]	The ID of the cluster that contains this host.
`vmhost.type`	String	The “type” differentiation for clusters vs. VM hosts. Tthe value is either `vmhost` or `vmhostcluster`.

6.7.3 Orchestration Server Facts in VMs Hosted in Clusters

The following table lists the read-only, cluster-related facts in a VM hosted in vSphere clusters.

Table 6-3 *Cluster-Related Facts in VMs Hosted in Clusters*

Fact Name	Type	Description
`resource.provision.vmhost`	String	The ID of the VM host cluster or VM host that contains this VM. When provisioning or migrating to a cluster, this value is initially the cluster ID. On subsequent resync or discovery or when using the VsphereUpdate daemon, this is set to the value of the selected VM host within the cluster.
`resource.vm.pool`	String	The MOR path of the resource pool that contains this VM. This value is also a key in the `vmhost.vsphere.pools` dictionary.
`resource.vm.vmhost.location`	String	The MOR path of the VM host cluster that contains this VM. This is always the cluster if the VM host is in a cluster; otherwise, it is the MOR path of the VM host itself.
`resource.vm.vsphere.cpu.limit`	Integer	The maximum amount (in MHz) of CPU resources to be used by this VM.
`resource.vm.vsphere.cpu.reservation`	Integer	The minimum amount (in MHz) of CPU resources guaranteed to this VM.
`resource.vm.vsphere.cpu.shares.level`	String	The relative amount (assigned a value of low, normal, high, or custom) of CPU allocated for this VM.

Fact Name	Type	Description
resource.vm.vsphere.cpu.shares.custom	Integer	The custom amount of relative CPU for this VM. This fact is valid only when the value for the resource.vm.vsphere.cpu.shares.level fact is “custom.”
resource.vm.vsphere.memory.limit	Integer	The maximum amount (measured in MB) of memory resources to be used by this VM.
resource.vm.vsphere.memory.reservation	Integer	The minimum amount (measured in MB) of memory resources guaranteed to this VM.
resource.vm.vsphere.memory.shares.level	String	The relative amount (assigned a value of low, normal, high, or custom) of memory specified for this VM.
resource.vm.vsphere.memory.shares.custom	Integer	The custom amount of relative memory for this VM. This fact is valid only when the value for the resource.vm.vsphere.memory.shares.level fact is “custom.”

6.8 vCPU Slots for VM Hosts

In the Orchestration Server, a vCPU represents a logical CPU. It provides a way to set up limits for allocating CPUs on VM hosts. These limits let you specify how many vCPUs should be hosted by each VM host so that you can control how much CPU processing power is available. If all vCPUs are in use, a subsequent provision can be denied or made to wait. This lets you ensure the quality of service you want to maintain in the data center.

When a VM is provisioned, the Orchestration Server runs a constraint check on every suitable VM host to determine if the number of available vCPUs on the VM host is sufficient for a VM. If a VM host with sufficient available vCPUs is not available, the provisioning request waits until one becomes available or (depending on VM facts) the request is denied.

Using vCPU facts differs from using the existing slot fact (vmhost.maxvmslots). The maxvmslots fact provides basic control of the number of VMs allocated to a VM host, which is useful for limiting VMs because of license restrictions or for generally limiting the VMs being managed. The vCPU facts are similar to the memory limit facts, giving you more control to avoid overloading a VM host and letting you ensure quality of service.

This section includes information about how the vCPU facts are used in the Orchestration Server.

- Section 6.8.1, “Configuring vCPUs on VM Hosts,” on page 94
- Section 6.8.2, “Configuring vCPUs on VM Host Clusters,” on page 94
- Section 6.8.3, “Configuring vCPUs on VMs,” on page 95

6.8.1 Configuring vCPUs on VM Hosts

There are two vCPU facts displayed on the VM host Info page in the Orchestration Console:

- **Max Hosted vCPUs:** This value (Integer) represents the maximum number of vCPUs that the VM host can support. The fact name is `vmhost.vcpu.max`.
- **Available vCPUs:** This value (Integer) represents the number of virtual CPUs available on this host. The fact name is `vmhost.vcpu.available`.

The `vmhost.vcpu.available` value changes when a VM is provisioned or shut down on that VM host. If the `vmhost.vcpu.max` fact is set to -1 (unlimited), the `vmhost.vcpu.available` value changes to -1 (unlimited). When it is set to unlimited, no counting occurs, so the Orchestration Server does not check vCPU limits.

When a VM host object is created (during discovery), the `vmhost.vcpu.max` value is set to the number of physical cores multiplied by a factor of 4. For example, on a Xen VM host that has eight physical cores, the Orchestration VM host discovery for Xen creates a VM host object with a maximum vCPU of 32. The factor value of 4 represents partitioning a physical core to four vCPUs, which represents 25% capacity.

You can change this default value by creating a policy that sets the `vmhost.vcpu.max` fact value and associates the policy either to the VM host or to a Resource Group of VM hosts. In the preceding Xen VM host example, if you wanted to partition the eight physical cores on a VM host to 50% capacity, you would set the maximum vCPUs to 16 by creating the following policy and then associating it to the VM host or to a Resource Group of VM hosts:

```
<policy>
   <vmhost>
       <fact name="vcpu.max" type="Integer" value="16" />
  </vmhost>
</policy>
```

If you want the Orchestration Server to allocate vCPUs without checking limits, you can set the maximum vCPUs to -1, which indicates an unlimited number. You would create the following policy to make that configuration setting:

```
<policy>
   <vmhost>
       <fact name="vcpu.max" type="Integer" value="-1 />
  </vmhost>
</policy>
```

6.8.2 Configuring vCPUs on VM Host Clusters

Because the VM host Cluster object represents a set of VM hosts, both `vmhost.vcpu.available` and `vmhost.vcpu.max` facts are sums of the underlying VM host objects. If all of the underlying VM hosts have their `vmhost.vcpu.max` values set to -1 (unlimited), then the corresponding `vmhost.vcpu.max` fact in the VM host Cluster is -1 (unlimited). This also means that no counting or checking occur for vCPU limits. This is used in scenarios where you rely on the underlying hypervisor to account for vCPUs rather than the Orchestration Server. For example, if you have set up vSphere with DRS and clustering, you do not need the server to do any checking.

NOTE: The Orchestration Server vsphere provisioning adapter sets the `vmhost.vcpu.max` fact value to -1 (unlimited) for DRS-enabled VM host clusters.

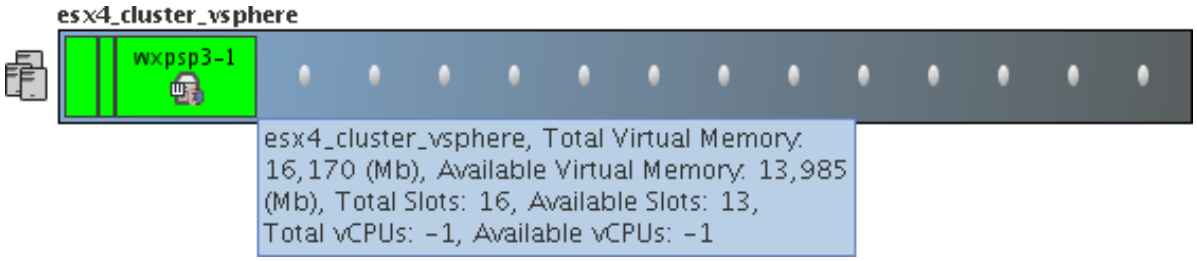
6.8.3 Configuring vCPUs on VMs

An existing vCPU fact (`resource.vm.vcpu`) specifies the number of vCPUs for a VM. This fact is set on VM image discovery. You can view it in the Orchestration Console on the VM object Admin view on the Info/Groups page. The Virtual Machine Configuration pane on this page has a *Host CPU Number* field where this fact is set.

In the VM Hosts Monitor view, the tooltip for a VM host displays the following:

`..Total vCPUs: xx, Available vCPUs: xx`

Figure 6-1 *Tool Tip Text for VM Host vCPU Slots*



These values change when a VM is provisioned or shut down. For example, if the `vmhost.vcpu.max` fact is set to -1 (unlimited), then the tooltip shows "-1" for both total and available.

7 The Virtual Disk Object

This section includes the following information:

7.1 Understanding the Virtual Disk Object

A virtual disk (vDisk) represents a VM’s view of its storage devices, which could include any type of physical disk (such as a file-backed disk image, an ISO image file, a physical hard drive, a CD/DVD device, or a block device) associated to a VM. The vDisk objects are discovered, along with their associated VM, when a Discover VM Images job is run on a repository.

The vDisk is modeled as a Grid object, located as a subordinate to the VM Grid object in the Explorer tree of the Orchestration Console. In the Explorer Tree, a vDisk is given the form *vmname*`_vdisk<`*n*`>` where <*n*> represents the numerical order in which this vDisk was discovered, with 1 appended to the name of the first vDisk discovered or created. For example, `suse11_vdisk1` would be the name of the first disk discovered for a VM with the Grid ID `suse11`. Each additional vDisk is incremented by one, so the second vDisk in this example would be named `suse11_vdisk2`.

This section includes the following information:

7.1.1 Creating or Deleting a vDisk in the Orchestration Console

This section includes the following information:

Creating and Configuring a Virtual Disk

You might want to manually create a vDisk in the following scenarios:

- When you want to create a "blank" disk image file for the VM. In this scenario, the disk image does not actually reside on the local file system, but a disk image of the specified size (measured in MB) should be created at the location specified for use by the VM. This is essentially a blank file, until it is used by the VM.
- When the Orchestration Server might not have discovered the vDisk objects correctly, such as omitting a disk that should exist. You need to manually correct the incorrect discovery.
- A VM that already exists needs to have patches applied to it. The patches are delivered through an ISO file, which was not configured to be attached to the VM. This configuration lets the administrator configure the VM with access to the ISO disk image, then apply the patches, and then later delete the vDisk object, returning the VM to its original configuration.

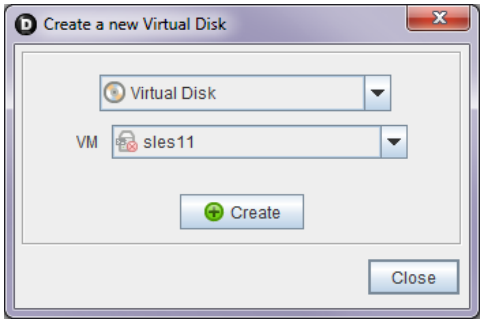
  You need to manually add the vDisk, select the *Save Config* action in the Orchestration Console, then apply the patches to the running VM. Later, you shut down the VM, delete the vDisk object from the Orchestration Server, then select the *Save Config* action again.

  The scenario includes configuring the VM to use the existing ISO file (that is, creating the vDisk object, then selecting *Save Config*), and then deconfiguring the VM to no longer use the ISO file (that is, deleting the vDisk object, then selecting *Save Config*).

  In this scenario, only the vDisk object from the Orchestration Server is deleted, not the ISO file.

To create a virtual disk in the Orchestration Console, you can either right-click the VM where you want to create the vDisk, then select *Create Virtual Disk* (if you do this, you can skip to Step 4 below) or you can use the following procedure from the Orchestration Console menu:

1. In the Orchestration Console main menu, select *Actions* > *Create Virtual Disk* to display the Create a New Virtual Disk dialog box.



2. In the *VM* drop-down list, select the name of the VM where you want to add a vDisk, then click *Create*.
3. When you have created all of the vDisks you need, click *Close*.
4. Select a newly created vDisk object in the Explorer tree to view the Info/Groups page of the admin view.
5. On the Info/Groups page, configure the following settings:

   Type: Specify the vDisk type as the VM host sees it.

   Description: Describe the vDisk with any text that you choose

   Healthy: Designates the health state of the vDisk. Do not configure.

Moveable: Specifies whether the disk image can be copied (relocated) with the VM when the VM is moved (relocated) to another repository. For more information, see "Moveable" later in this section.

Mode: Specifies the mode of the vDisk as made available and supported by the provisioning adapter:

- r = read only
- w = read/write

VM: Specifies the name of the VM that uses this vDisk.

Repository: The repository where this disk location path resides.This setting is important because the Orchestration Server uses it to find a suitable VM host for provisioning, building, or migration actions. The value of this setting can be *none*, which informs the server to ignore this vDisk when locating a suitable VM host.

Physical Disk: The name of the pDisk that this vDisk is associated with.

Location: The path (location) to the disk image.

- If you specify a location to a disk that already exists, the existing disk file is used and the VM configuration is modified (according to the value in `vdisk.location` fact) to use this existing disk.
- If you specify a path to a disk that does not exist (that is, if the value in `vdisk.location` is invalid), the action fails and an empty disk image file of the specified size is created. An error in the action status or job log is created.
- For a vDisk created for a Hyper-V VM, you need to provide the complete path of that vDisk file.

  To form the path, you need to know the repository path where the VM currently resides, the vDisk name, which is the name you give it plus the `.vhd` extension. For example, the syntax would be

  `<value_of_the_repository.preferredpath_fact>\<your_vhd_filename>.vhd`

  NOTE: Make sure that the `.vhd` file you designate in this field doesn't already exist in the path.

Size: The size (measured in MB) of the disk image. Do not configure.

Sparse Disk: Designates whether the vDisk file is a sparse file. Do not configure.

Actual Size: The actual sparse size (measured in MB) of the vDisk file. Do not configure.

6 Click the Save button in the toolbar to save the fact changes you made.

7 In the Explorer tree, right-click the VM object where you added the vDisk, then select *Save Config* to apply the changes to the VM's configuration.

Creating a Sparse Virtual Disk

Sparse disk creation is supported by the Xen, vSphere, and KVM hypervisors. If you want to create your vDisk as a sparse file, you can use the procedure for "Creating and Configuring a Virtual Disk" on page 98 (see Step 5 on page 98 in particular).

You need to set the Sparse Disk fact to true, specify the amount of space (in MB) for the disk in the Size fact, and also specify the path to the repository where the sparse vDisk resides. Make sure that you perform the *Save Config* action to apply the vDisk changes to the VM's configuration.

Deleting a Virtual Disk

You might want to manually delete a vDisk in at least two scenarios:

- When the Orchestration Server might not have discovered the vDisk objects correctly, such as adding a disk that should not exist. The administrator needs to manually correct the incorrect discovery.
- A VM that already exists needs to have patches applied to it. The patches are delivered through an ISO file, which was not configured to be attached to the VM. This configuration lets the administrator configure the VM with access to the ISO disk image, then apply the patches, then later delete the vDisk object, returning the VM to its original configuration.

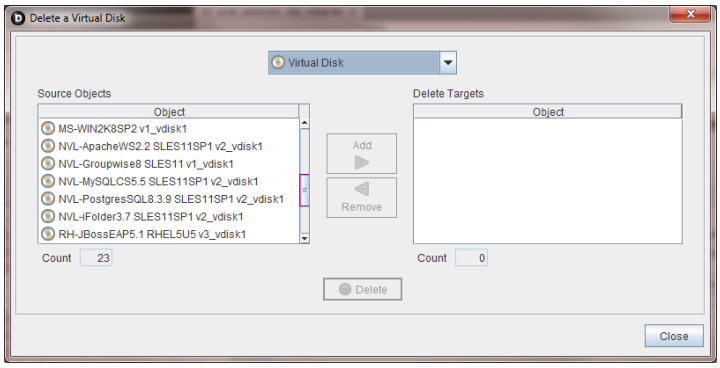
  The administrator needs to manually add the vDisk, run the Save Config command from the Orchestration Console, then apply the patches to the running VM. Later, the administrator shuts down the VM, deletes the vDisk object from the server, then performs the *Save Config* action again.

  The scenario includes configuring the VM to use the existing ISO file (that is, creating the vDisk object and selecting the *Save Config* action), then deconfiguring the VM to no longer use the ISO file (that is, deleting the vDisk object, then selecting the *Save Config* action).

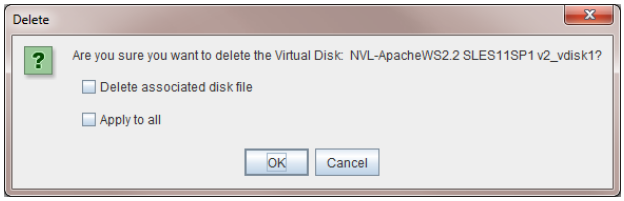
  In this scenario, only the vDisk object from the server is deleted, not the ISO file.

To delete a virtual disk, you can either right-click the vDisk object in the Explorer, then select *Delete* (if you do this, you can skip to Step 4, below), or you can use the following procedure from the Orchestration Console:

1. In the Orchestration Console, select *Actions* > *Delete* > *Delete Virtual Disk* to display the Delete a Virtual Disk dialog box.



2. In the *Source Objects* list, select the name of the vDisk (hold down the Ctrl key to select multiple objects), then click *Add* to move these objects to the *Delete Targets* list.
3. When you have selected all of the vDisks you want to delete, click *Delete* to display the Delete dialog box.



4 In the dialog box, select *Apply to all* to delete all of the vDisk objects in the *Delete Targets* list, select *Delete associated disk file* if you want to delete the files associated with the vDisks you have selected, click *OK*, then click *Close*.

5 In the Explorer tree, right-click the VM object where you deleted the vDisk, then select *Save Config* to apply the changes to the VM's configuration.

NOTE: The *Save Config* action rewrites the configuration file for the VM (for example, `config.xen` for the xen provisioning adapter), but it does not delete any vDisk files on the file system. In this case, manual deletion of the vDisk file is required.

To delete a VM and its backing files, use the *Delete/Destroy Resource* action.

7.1.2 Sharing Virtual Disks Among VM Hosts

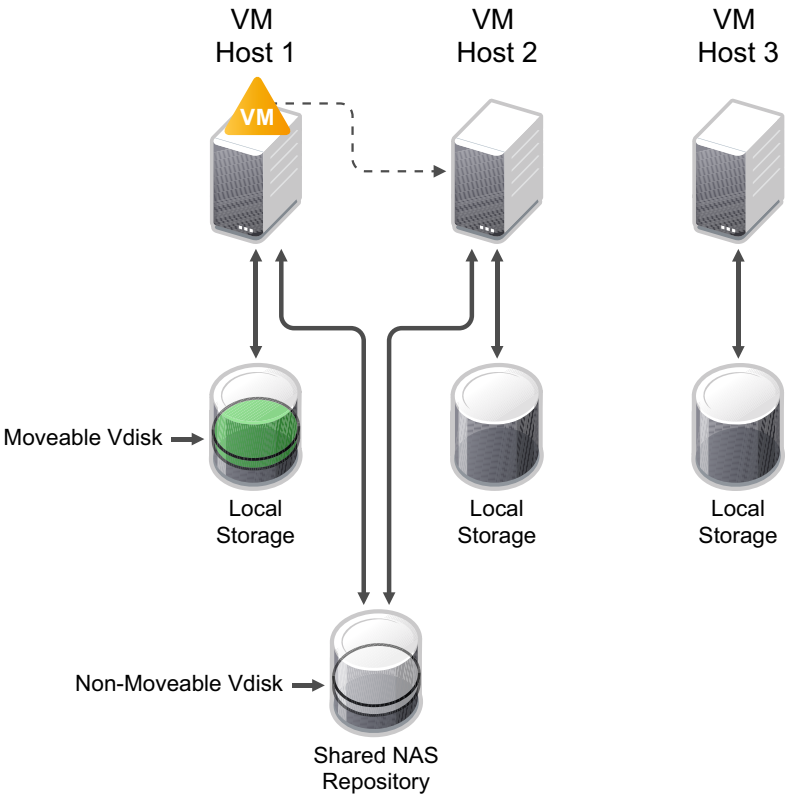
For a VM to be provisionable by other VM hosts, all of a VM's vDisks must be visible in the same way that the VM's default repository (`resource.vm.repository`) is visible to VM hosts. If a VM has multiple vDisks and each vDisk has a different associated repository, these repositories must also be visible from a potential VM host.

7.1.3 Moving Virtual Disks

When you move a VM to a new repository (see the *Move Disk Image* action for each hypervisor at "Orchestration Provisioning Adapter Information"), all of that VM's moveable vDisk images (see "Moveable:" on page 104) are moved with it to be co-located in the same repository. The Orchestration Server uses the aggregated size of each moveable vDisk to determine if the designated repository has enough space for all of the disk images. vDisks that are marked as not moveable stay in place and are not used in the calculation for the VM disk size.

The following illustration further explains this concept:

Figure 7-1 *Example of Moving Virtual Disks with the VM*



- VM host 1, VM host 2, and VM host 3 all have their own local storage repositories.
- VM host 1 has a vDisk located on it. It is designated as a moveable vDisk.
- VM host 1 and VM host 2 are also connected to a shared NAS storage repository.
- The local repository connected to VM host 1 has a vDisk located on it. It is designated as a moveable vDisk.
- The shared NAS repository has a vDisk located on it. It is designated as a non-moveable vDisk.

NOTE: Shared repositories are not created on discovery. They must be manually created and the sharing (visibility) configured.

- VM host 1 has a VM located on it.
- VM host 3 cannot communicate with the NAS repository; its `vmhost.repositories` fact does not include the NAS repository in the array, so that repository is not visible to VM host 3.

If you want to move the VM from VM host 1 to another VM host, the server manifests the following behavior:

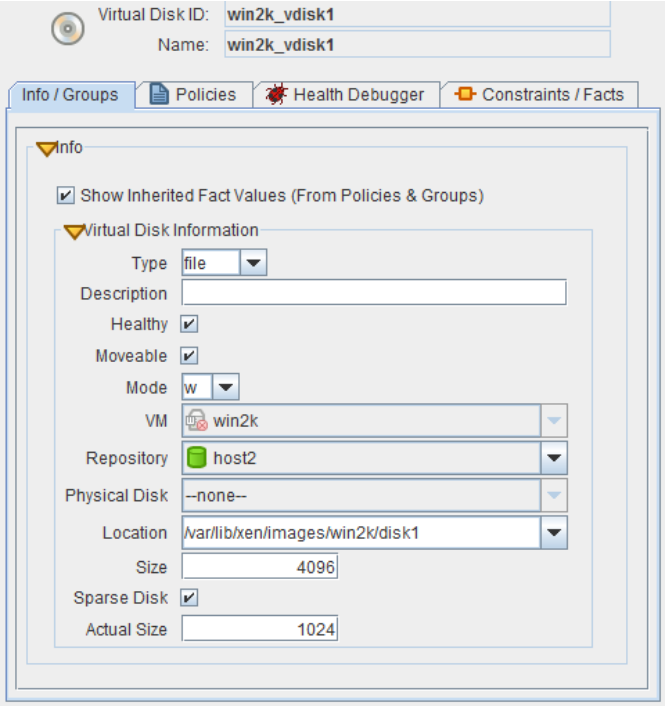
- The vDisk sizes used by the VM (on local storage and shared storage) are aggregated and compared to free space available on the repositories.
- The only vDisk that is allowed to move is the moveable disk. This disk would be copied to either the shared NAS repository or the local storage on VM host 2.
- VM host 3 is not considered because it does not have access to the non-moveable disk on the NAS repository.

7.2 Viewing Virtual Disk Configuration in the Orchestration Console

You can visually expose a vDisk Grid object in the Orchestration Console in two ways:

- In the Explorer Tree, select a VM Resource object, select the *Info/Groups* tab in the admin view to open the VM Info Groups page, then scroll to the *Virtual Machine Configuration* panel on that page. Right-click the vDisk icon in that panel to display the four tabs in the Virtual Disk admin view.
- In the Explorer Tree, click the expand/collapse icon of a VM Grid object, identify the vDisk icon, then select the icon to display the four tabs in the Virtual Disk admin view.

Figure 7-2 *The Virtual Disk Info/Groups Page*



The page that opens under the *Info* tab includes fields where you can configure the general information and attributes (facts) of the vDisk.

NOTE: Whenever you make changes to vDisk object facts, the write icon is superimposed on the object's icon, signifying that the object has been changed. If you want to save the changes you have made, click the Save button on the Orchestration Console toolbar.

This section includes the following additional information:

- Section 7.2.1, "Virtual Disk Information Panel," on page 104
- Section 7.2.2, "Virtual Disk Policies Tab," on page 106
- Section 7.2.3, "Virtual Disk Health Debugger Tab," on page 106
- Section 7.2.4, "Virtual Disk Constraints/Facts Tab," on page 106
- Section 7.2.5, "Virtual Disk Object Naming and Renaming," on page 107

7.2.1 Virtual Disk Information Panel

The Virtual Disk Information panel on the Info page includes the following fields:

NOTE: Tool tip text is displayed when you mouse over any of these fields.

Type: This drop-down list lets you select one of the vDisk types as the VM host sees it:

- **file:** Specifies this vDisk as a file-backed disk.
- **block:** Specifies this vDisk as a block device.

In the Fact Editor, this fact is listed as `vdisk.type`:

```
<fact name="vdisk.type" value="file" type="String" />
```

Description: Describes the vDisk with any text that you choose.

In the Fact Editor, this fact is listed as `vdisk.description`:

```
<fact name="vdisk.description" value="" type="String" />
```

For a vDisk discovered and managed by the xen provisioning adapter or the kvm provisioning adapter, this field is usually blank. Because this is a free form field, you can enter any text you want here. For a vDisk discovered and managed by the vsphere provisioning adapter, this field is populated with a display name label obtained by vSphere and mapped to this vDisk by the Orchestration Server.

Healthy: For a vDisk managed by any supported hypervisor, this check box is selected by default, which designates the vDisk as being in good health.

NOTE: We recommend that you do not change the *Healthy* value from its default.

In the Fact Editor, this is fact is listed as `vdisk.health`:

```
<fact name="vdisk.health" value="true" type="Boolean" />
```

Moveable: When this check box is selected (its value is true), the vDisk is moveable, which means that the disk image can be copied to a different repository when the VM moves.

In the Fact Editor, this fact is listed as `vdisk.moveable`:

```
<fact name="vdisk.moveable" value="true" type="Boolean" />
```

If *Moveable* is not selected, the disk image must stay at its current location because it cannot be copied or moved. By default upon discovery, if the server sees that this vDisk is an ISO image, the fact is set to false because it is assumed that the administrator doesn't want to copy ISO images from one location to another.

Whenever you want to prevent a vDisk from being moved, you can deselect this check box.

The vDisk is not deleted during a VM *Delete*/*Destroy* action if this check box is deselected.

NOTE: In Cloud Manager 2.0, block type disks (that is, pDisks) are not moveable, even if you change this setting.

Mode: Specifies the mode of the vDisk as made available and supported by the provisioning adapter:

- r = read only
- w = read/write

In the Fact Editor, this fact is listed as `vdisk.mode`:

```
<fact name="vdisk.mode" value="w" type="String" />
```

VM: (Read Only) Specifies the name of the VM that uses this vDisk.

In the Fact Editor, this fact is listed as `vdisk.vm`:

```
<fact name="vdisk.vm" value="mysql" type="String" />
```

This is a fact junction referencing the associated VM. Conversely, the `resource.vm.vdisks` fact visible from the VM Grid object is a fact junction showing the associated vDisks associated with the VM.

Repository: The storage location containing the vDisk image on the VM host.

Changing this fact after discovery only corrects a possible incorrectly discovered fact. Changing the storage location does not move the vDisk.

In general terms, a *block* type Repository represents a container for physical devices you can use for VM disks. For the xen and kvm provisioning adapters, a block type Repository represents a Volume Group on a VM host. If you select this type, the physical disks (Logical Volumes) on that VM host are listed.

In the Fact Editor, this fact is listed as `vdisk.repository`:

```
<fact name="vdisk.repository" value="zos" type="String" />
```

Physical Disk: Indicates the name of the pDisk to which this vDisk is associated.

In the Fact Editor, this fact is listed as `vdisk.pdisk`:

You can edit this fact only if a *block* type Repository is selected for the `vdisk.repository` fact. When you make that selection, a drop-down list of Physical Disks becomes available for this field.

```
<fact name="vdisk.pdisk" value="" type="String" />
```

Although there might be several pDisks available in the block Repository, you can select only one.

Location: For file-backed disks, this fact represents the file system path to the vDisk image in the specified repository.

For example, a vDisk located on an NFS repository datastore would show the URI to the NFS share with the path to the disk appended to it.

For *block* type disks, this fact contains the URI to the block device, for example `/dev/hdc`, which could represent a CD/DVD tray on a VM host.

In the Fact Editor, this fact is listed as `vdisk.location`:

```
<fact name="vdisk.location" value="/var/lib/xen/images/mysql/disk1" type="String"
/>
```

Size: The size (measured in MB) of this vDisk image.

In the Fact Editor, this fact is listed as `vdisk.size`:

```
<fact name="vdisk.size" value="2048" type="Integer" />
```

The disk size value for each moveable vDisk on a VM is aggregated by the Orchestration Server into the `resource.vm.vdisksize` fact, which is used to determine if the VM can relocate from one repository to another, given that the new repository has enough free space to store the VM.

Sparse Disk: When this check box is selected (its value is true), this is a sparse (thin) backed vDisk.

In the Fact Editor, this fact is listed as `vdisk.sparse`:

```
<fact name="vdisk.sparse" value="true" type="Boolean" />
```

Actual Size: The actual (sparse) size (measured in MB) of this virtual disk.

In the Fact Editor, this fact is listed as `vdisk.size.actual`:

```
<fact name="vdisk.size.actual" value="1024" type="Integer" />
```

7.2.2 Virtual Disk Policies Tab

The *Policies* tab opens a page that contains a policy viewer for each of the policies associated with a Grid object.

NOTE: You can edit a policy by right-clicking a policy icon, selecting *Edit Policy*, and clicking the Save icon.

7.2.3 Virtual Disk Health Debugger Tab

The Health Debugger is a common admin view in the Orchestration Console for most Grid objects. For information about this tool, see "Health Debugger" on page 189.

7.2.4 Virtual Disk Constraints/Facts Tab

To support constraining a VM's provisioning actions based on more than one disk's repository (that is, more than just `resource.vm.repository`), the vDisk can be referenced in constraints. The vDisk constraints are used to assign VM hosts during actions such as provisioning, building, or migrating. You can write constraints against attributes of disks (such as the repository where the vDisk resides) and against the available VM host repositories.

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties. In essence, by building, deploying, and running jobs on the Orchestration Server, you can individually change the functionality of any and all system resources by managing an object's facts and constraints.

The Orchestration Server assigns default values to each of the component facts. The following table lists the possible fact modes and their function:

Table 7-1 *vDisk Fact Modes Displayed in the Constraints/Facts Tab*

vDisk Mode Type	Mode Function
blank (no mode displayed)	read/write, not deleteable
`del`	read/write, deleteable
`dynamic`	read/write, not deleteable
`dynamic, r/o`	read only, not writable
`r/o`	read only

IMPORTANT: Several custom facts for the vDisk object can be added at discovery time, and vary according to the VM technology that manages the respective vDisk. These facts are documented in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

For more information about using the Fact Editor on this page to rename the Virtual Disk object, see Section 7.2.5, "Virtual Disk Object Naming and Renaming," on page 107.

7.2.5 Virtual Disk Object Naming and Renaming

Some resource names are generated by the Orchestration system and can therefore have generic, arbitrary names such as `mysql-vdisk1`, `mysql-vdisk2`, and so on. A Virtual Disk (vDisk) you name at creation time might also change later in its purpose or facilities.

The object's display name is visible in the Orchestration Console interface, the VM Client interface, and in optional zos and zosadmin commands. As the number of these vDisk objects grows in your grid, you might find it helpful or necessary to rename them, assigning more meaningful, intuitive names to suit the purpose of the object.

NOTE: Resource object groups (that is, the folders that contain these vDisk objects) can also be renamed. Objects such as jobs, events, and users cannot be renamed.

A vDisk object's name is stored in the `${objectType}.displayname` fact, which exists on every Grid object type, even those objects that cannot be renamed.

You can rename a vDisk object in the Orchestration Console by using one of three methods:

- Right-click the vDisk object in the Explorer tree, then select *Rename* to allow editing of the display name.
- Triple-click the vDisk object in the Explorer tree to allow editing of the display name.
- In the Constraints/Facts page, select the vDisk object `.displayname` fact and then open the Fact Editor to specify a new value for the fact.

As you use one of these methods, notice that the fact value is prepopulated with the `${objectType}.id` fact. This functions as the name value for the object name until you decide to change it.

NOTE: Even after being renamed, the vDisk object retains its associated resource ID in the `.id` fact. This is not editable.

For more information about making the Resource object display names visible from the zos or zosadmin command line, see the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

7.3 Managing Block Devices as VM Virtual Disks

Cloud Manager Orchestration Server can use a block device (an addressable, physical location) for storage on Xen, KVM, VMware vSphere, or Microsoft Hyper-V managed hosts, but it manages such devices as physical disks (pDisks) attached to VM virtual disks (vDisks) to provide better performance compared to a file-backed virtual disk.

A vDisk represents a VM's view of its storage devices, but a pDisk represents a VM Host's view of its physical storage devices allocated for VM usage.

Managing block devices as pDisks also saves the additional setup and management that normal block devices require.

IMPORTANT: Block devices defined in the Amazon EC2 environment are not discovered or supported by the Orchestration Server in NetIQ Cloud Manager.

This section includes information to help you set up this kind of block device support for the xen and kvm provisioniong adapters in the Cloud Manager Orchestration Server.

- Section 7.3.1, "Prerequisites to Configure on Xen and KVM Hosts Before Setting Up Block Device Support," on page 108
- Section 7.3.2, "How Block Device Support Works," on page 109
- Section 7.3.3, "Viewing the Physical Disk Configuration in the Orchestration Console," on page 111

7.3.1 Prerequisites to Configure on Xen and KVM Hosts Before Setting Up Block Device Support

Before you can set up block device support in the Orchestration Server, you need to create the block device on a host that is visible to the supported hypervisor. The block device you set up on that host must be configured with Logical Volume Management (LVM), creating Logical Volumes and Volume Groups. LVM allows flexibility in administering the underlying device, such as making it easeir to move objects, create snapshots, and back up data.

Before you create Volume Groups, you should plan with the Cloud Manager model in mind: an LVM Logical Volume maps to an Orchestration pDisk and an LVM Volume Group maps to an Orchestration Server Repository object of type "block." The Orchestration Server leverages the Volume Group Name to identify a shared repository, which could map to a shared iSCSI target. The Volume Group Name must be the same on each VM host that accesses this repository (a shared iSCSI target).

Additional Prerequisite for KVM Hosts

In addition to the LVM prerequisite mentioned above, if you are managing VMs with the KVM hypervisor, you must also use the Virtual Machine Manager to create a Logical Volume type *Storage Pool* on each VM host.

7.3.2 How Block Device Support Works

The following information is included in this section:

Creating New Block Type Repositories

When you run the *Discover Hosts and Repositories* action by using the appropriate the Orchestration Server provisioning adapter job, the Orchestration Server discovers the block device on the VM host and creates a "block" type Repository Grid object for it in the Explorer tree  . The repository is to be used only as a container for physical disks.

NOTE: For Xen hosts, you need to configure a setting on the xen provisioning adapter job that facilitates the discovery of multiple Volume Groups previously created for the Xen environment. You can find this setting, called *Volume Group Patterns*, in the *Job Configuration* panel of the xen job. The default expression is *NCM-**. You can add regular expression patterns (for example, `*XenPAVolGrp*`) for Volume Group names that are used in the creation of new Repository objects.

You do not need a naming pattern for the storage pools managed by the KVM hypervisor because they are explicitly created.

For more information about repository Grid objects, see Chapter 10, "The Repository Object," on page 131.

Discovering pDisks

After the initial VM host and Repository discovery action, some block devices (considered to be a "physical disk" or "pDisk" in the the Orchestration Server model) might be discovered already attached to a VM, while some are not yet discovered and added under the block repository container. Further discovery requires a *Discover Disks* action on the new block Repository Grid object. The action launches a provisioning adapter job that discovers the physical disks available for the repository. When this job finishes, new pDisk objects  are created in the Explorer tree under the block Repository object.

You can also create a pDisk in a block Repository by right-clicking the block Repository object and selecting the *Create Physical Disk* action. If you create a pDisk without using the discovery, you need to provide information (that is, facts) about it for the Orchestration Server. For more information, see Section 7.3.3, "Viewing the Physical Disk Configuration in the Orchestration Console," on page 111. If you create this object without having a corresponding Logical Volume on the host system, you must create it there so that block device support for this pDisk works in the Orchestration Server.

Attaching an Available pDisk to a VM

After you discover a pDisk (that is, a Logical Volume) using *Discover Disks* on the block type Repository, that pDisk is not associated with any VM (workload) that Cloud Manager can access. You need to create this association by attaching the pDisk to a vDisk on an available VM.

1. In the Explorer tree of the Cloud Manager Orchestration Console, select a VM to which you want to attach a pDisk.
2. Right-click the VM object and select *Create Virtual Disk* to create a new vDisk object to be associated with the VM.
3. Attach the pDisk to the vDisk:
   - **3a** In the admin view of the vDisk object, select the *Info*/*Groups* tab to open the *Info* page for the object.
   - **3b** Select the *Repository* drop-down menu to display the list of available repositories, then select the appropriate block type repository from the list.
   - **3c** Select the *Physical Disk* drop-down menu to display the list of available pDisks, then select the appropriate pDisk form the list. The default value, *none*, does not select any physical disk.
   - **3d** Click the *Save* icon  on the toolbar to save the fact changes you have made.

     When you perform this action, the vDisk facts are automatically populated with the corresponding values from the pDisk object. These values become read-only, because they are retrieved from the pDisk.
   - **3e** In the Explorer tree, right-click the VM object associated with the new vDisk you created, then select the *Save Config* action to reconfigure this VM.

     NOTE: Remember that when you save the configuration, you only update the object model in the Orchestration Server; you have not affected the VM in the hypervisor. To do so, you must perform the *Save Config* action.

Sharing Block Devices Between KVM or Xen VMs

You can use the Orchestration Server to share block devices between VMs residing on either a Xen host or on a KVM host. If you configure this sharing, however, you must be careful to avoid data corruption.

You can safeguard the data on the underlying block device being shared by using the hypervisor or operating system tools to configure it (for example, during logical volume creation) as "read-only." Another alternative is to install a cluster aware file system like OCFS2 on the block device if that device must be writeable.

To protect data on the block device using the server, you have two alternatives:

- Modify the `vdisk.mode` fact to a *readonly* value, select *Save* on the console toolbar to save this change, then right-click the VM object in the Explorer Tree and select *Save Config* to modify the configuration file.
- Set a maximum number of pDisks that can be associated to a vDisk by using the `pdisk.vdisks.max` fact. This counter is respected by the provisioning action; the VM does not start if the maximum value is reached.

Limitations of Block Device Support in This Release

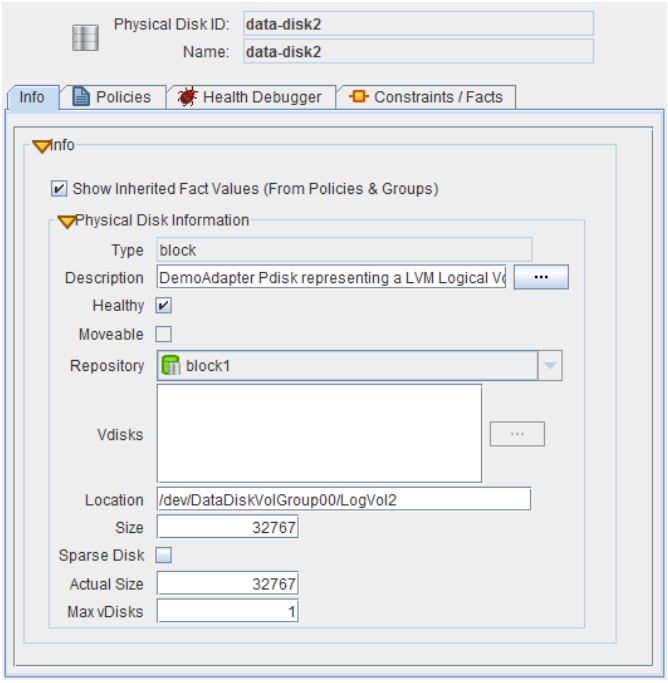
Block device support for this release has the following limitations:

- Orchestration Services does not allow the creation or deletion of pDisks. In other words, no LUN or volume management is possible from the Orchestration Console.
- Although a VM move is supported (block device disks are marked as unmoveable), cloning a VM that includes the pDisk is not supported if the `pdisk.vdisks.max fact` is set to 1. Cloning works if the fact is set to a value greater than 1.
- Not all variations of possible block-based data disks can be discovered on a VM host.

7.3.3 Viewing the Physical Disk Configuration in the Orchestration Console

You can visually expose a pDisk Grid object in the Explorer Tree of the Orchestration Console by selecting the expand/collapse icon of a block Repository object, identifying the pDisk icon, then selecting the icon to display the four tabs in the Virtual Disk admin view.

Figure 7-3 *The Physical Disk Info/Groups Page*



The page that opens under the *Info* tab includes fields where you can configure the general information and attributes (facts) of the pDisk.

NOTE: Whenever you make changes to pDisk object facts, the write icon is superimposed on the object's icon, signifying that the object has been changed. If you want to save the changes you have made, click the *Save* icon on the Orchestration Console toolbar.

This section includes the following additional information:

- "The Physical Disk Info Panel" on page 112
- "The Physical Disk Policies Tab" on page 113

The Physical Disk Info Panel

The Physical Disk Info panel on the Info page includes the following fields:

NOTE: Tool tip text is displayed when you mouse over any of these fields.

Show Inherited Fact Values: Select this check box to show facts with overridden values supplied through attached and/or inherited policies. Such fact values are read only (non-editable).

Type: This field is read-only. It specifies this pDisk as a block device, which is the pDisk type as the VM host sees it:

In the Fact Editor, this fact is listed as `pDisk.type`:

```
<fact name="pDisk.type" value="block" type="String" />
```

Description: Describes the pDisk with any text that you choose.

In the Fact Editor, this fact is listed as `pDisk.description`:

```
<fact name="pdisk.description" value="" type="String" />
```

For a pDisk discovered and managed by the xen provisioning adapter, this field is usually blank. Because this is a free form field, you can enter any text you want here. For a pDisk discovered and managed by the vSphere provisioning adapter, this field is populated with a display name label obtained by vSphere and mapped to this pDisk by the Orchestration Server..

Healthy: For a pDisk managed on a Xen host, this check box is selected by default, which designates the pDisk as being in good health.

NOTE: We recommend that you do not change the *Healthy* value from its default.

In the Fact Editor, this is fact is listed as `pDisk.health`:

```
<fact name="pdisk.health" value="true" type="Boolean" />
```

Moveable: This fact is marked false by default. Do not change this fact.

In the Fact Editor, this fact is listed as `pDisk.moveable`:

```
<fact name="pdisk.moveable" value="false" type="Boolean" />
```

Repository: The storage location (a Volume Group) containing the pDisk image on the VM host. The field is pre-populated with a block type repository name. It cannot be edited.

In the Fact Editor, this fact is listed as `pDisk.repository`:

```
<fact name="pdisk.repository" value="xen3vg" type="String" />
```

Vdisks: The list of virtual disks that are associated with this physical disk.

In the Fact Editor, this fact is listed as an array:

```
<fact name="pdisk.vdisks">
  <array type="String">
  </array>
</fact>
```

This list box is populated when you attach the pDisk to a VM.

Location: For block type disks, this fact contains the URI to the block device, for example `/mnt/shared/iscsi/`, which could represent an iSCSI share on a VM host.

In the Fact Editor, this fact is listed as `pdisk.location`:

```
<fact name="pdisk.location" value="/dev/xen3vg/xen3lv2" type="String" />
```

Size: The size (measured in MB) of this pDisk image.

In the Fact Editor, this fact is listed as `pdisk.size`:

```
<fact name="pdisk.size" value="32767" type="Integer" />
```

Sparse Disk: This check box is marked true if the image is a sparse (thin) backed vDisk.

In the Fact Editor, this fact is listed as `pdisk.sparse`:

```
<fact name="pdisk.sparse" value="false" type="Boolean" />
```

Actual Size: The actual (sparse) size (measured in MB) of this virtual disk.

In the Fact Editor, this fact is listed as `pdisk.size.actual`:

```
<fact name="pdisk.size.actual" value="8192" type="Integer" />
```

Max vDisks: This fact represents the maximum number of vDisks that are allowed concurrent access to this pDisk. Its value is set to 1 by default. To allow multiple instances of the pDisk, you need to set this value accordingly. A value of -1 indicates that an unlimited number of vDisks have access.

One practical application of this setting is for specifying the pDisks in an OCFS2 environment. If you are using OCFS2, change this value of this fact to match the number of hosts, with each host representing an OCFS2 node.

In the Fact Editor, this fact is listed as `pdisk.vdisks.max`:

```
<fact name="pdisk.vdisks.max" value="1" type="Integer" />
```

In order to allow multiple instances of the pDisk, this value has to be set to some positive number other than 1.

The Physical Disk Policies Tab

The *Policies* tab opens a page that contains a policy viewer for each of the policies associated with a Grid object.

NOTE: You can edit a policy by right-clicking a policy icon, selecting *Edit Policy*, and clicking the *Save* icon.

The Physical Disk Health Debugger Tab

The Health Debugger is a common Admin view in the Orchestration Console for most Grid objects. For information about this tool, see Chapter A.3, “Health Debugger,” on page 189.

The Physical Disk Constraints/Facts Tab

To support constraining a VM's provision actions based on more than one disk's repository (that is, more than just `resource.vm.repository`), the pDisk can be referenced in constraints. The pDisk constraints are used to assign VM hosts during actions such as provisioning, building, or migrating. You can write constraints against attributes of disks (such as the repository where the pDisk resides) and against the available VM host repositories.

The Constraints/Facts tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties. In essence, by building, deploying, and running jobs on the Orchestration Server, you can individually change the functionality of any and all system resources by managing an object's facts and constraints. The Orchestration Server assigns default values to each of the component facts, although they can be changed at any time by the administrator, unless they are read-only. Facts with mode `r/o` have read-only values, which can be viewed (using the edit pencil icon) but changes cannot be made.

IMPORTANT: Several custom facts for the pDisk object can be added at discovery time, and they vary according to the VM technology that manages the respective pDisk. These facts are documented in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

For more information about using the Fact Editor on this page to rename the Physical Disk object, see Section 7.2.5, "Virtual Disk Object Naming and Renaming," on page 107.

Physical Disk Object Naming and Renaming

Some resource names are generated by the Orchestration Server and can therefore have generic, arbitrary names such as `mysql-pDisk1`, `mysql-pDisk2`, and so on. A Physical Disk (pDisk) you name at creation time might also change later in its purpose or facilities.

The object's display name is visible in the Orchestration Console interface and in optional zos and zosadmin commands. As the number of these pDisk objects grows in your grid, you might find it helpful or necessary to rename them, assigning more meaningful, intuitive names to suit the purpose of the object.

NOTE: Resource object groups (that is, the folders that contain these pDisk objects) can also be renamed. Objects such as jobs, events, and users cannot be renamed.

A pDisk object's name is stored in the `${objectType}.displayname` fact, which exists on every Grid object type, even those objects that cannot be renamed.

You can rename a pDisk object in the Orchestration Console by using one of three methods:

- Right-click the pDisk object in the Explorer tree, then select *Rename* to allow editing of the display name.
- Triple-click the pDisk object in the Explorer tree to allow editing of the display name.
- In the Constraints/Facts page, select the pDisk object `.displayname` fact and then open the Fact Editor to specify a new value for the fact.

As you use one of these methods, notice that the fact value is prepopulated with the `${objectType}.id` fact. This functions as the name value for the object name until you decide to change it.

NOTE: Even after being renamed, the pDisk object retains its associated resource ID in the `.id` fact. This is not editable.

For more information about making the Resource object display names visible from the zos or zosadmin command line, see the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

8 The Virtual NIC Object

This section includes the following information:

8.1 Understanding the Virtual NIC Object

A virtual network interface card (vNIC) represents the configuration of a VM connected to a network. A VM can be configured to have multiple vNICs. When a VM is provisioned, each of its associated vNICs can be attached to a virtual network bridge in order to gain connectivity to a specified network. The vNIC objects are discovered, along with their associated VM, when a Discover VM Images job has been run on a repository.

The vNIC is modeled as a Grid object, located as a subordinate to the VM Grid object in the Explorer tree of the Orchestration Console. A vNIC is given the form of *vmname*_vnic<*n*> where <*n*> is appended to indicate the order of discovery or creation of the vNIC. For example, `redhat_vnic1` would be the name of the first NIC discovered for a VM with the Grid ID `redhat`. Each additional vNIC is incremented by one, so the second vNIC in this example would be named `redhat_vnic2`.

8.1.1 The Purpose of the Virtual NIC

A vNIC represents the network interface configuration for a virtual machine. A vNIC is linked to a network by connecting to a virtual network bridge (vBridge). A group of vBridge objects is represented as a Network group in the Explorer Tree. By convention, during VM host discovery, any vBridges that are configured with the same name are assumed to be part of the same network.

For more information, see Chapter 9, “The Network Group and its Virtual Bridge Objects,” on page 125.

8.1.2 Creating or Deleting a vNIC in the Orchestration Console

Although a vNIC is generally discovered on a VM, you can also manually create or delete a vNIC. This section includes the following information:

Creating a Virtual NIC

You can manually create a vNIC anytime you want to give a VM access to a network configured on the VM host.

To create a vNIC, you can select the VM Grid object in the Explorer tree, then right-click and select *Create Virtual NIC*. You can also use the following alternate method:

1. In the Orchestration Console main menu, select *Actions* > *Create* > *Create Virtual NIC* to display the Create a New Virtual NIC dialog box.
2. In the *VM* drop-down list, select the name of the VM to which you want to add a vNIC, then click *Create*.
3. When you have created all of the vNICs you need, click *Close*.
4. Select the newly created vNIC object in the Explorer Tree to view the Info/Groups page of the admin view.
5. On the Info/Groups page, configure the following settings:

   MAC Address: The MAC Address assigned to this vNIC. If this field is left empty, or if it contains an asterisk (*), a MAC address is autogenerated for this vNIC.

   Network: The network (vBridge group) that should be used when provisioning the VM. When a VM is provisioned, any vBridge contained within the specified network group can be used for attaching the VNIC to the network.

   Although the vNIC can be formally created at this point, you can also configure Autoprep or Sysprep facts used to prepare a personalized version of the VM that can be provisioned later.
6. Click the *Save* button to save the fact changes you have made.
7. In the Explorer tree, right-click the VM object to which you added the vNIC, then select *Save Config* to apply the changes to the VM's configuration.

   IMPORTANT: If you do not run the *Save Config* action after configuring the vNIC, any vNICs that were added to the grid or vNIC settings that were modified could be lost.

Creating a Virtual NIC for a Hyper-V VM

1. Right-click the VM for which you want to create a vNIC, select *Create Virtual NIC*.
2. In the Explorer tree, select the vNIC object you created in the previous step, then from the Admin view, select the *Constraints/Facts* tab to open the Constraints/Facts page.
3. In the Constraints/Facts page, click the *Add a fact* button to open the Add Fact dialog box.
4. In the dialog box, add a custom fact for each of the items in the table below.

Fact name	Type	Value and Description
vnic.type	String	SyntheticEthernetPort or EmulatedEthernetPort SyntheticEthernetPort is the default vNIC type. Use the EmulatedEthernetPort type to perform a network-based installation of the guest OS or when integration services are not installed in the guest OS.
vnic.mac	String	Provide a valid MAC address.
vnic.static_mac_address	Boolean	True or false. When this fact is set to true, the provided MAC address (vnic.mac) is used to set for this adapter; otherwise, it is dynamically set by the Hyper-V system.
vnic.network	String	Copy the value of the group.id fact from the networks to which you want to attach the vNIC.

5 In the Explorer tree, right-click the VM object to which you added the vNIC, then select *Save Config* to apply the changes to the VM's configuration.

IMPORTANT: If you do not run the *Save Config* action after configuring the vNIC, any vNICs that were added to the grid or vNIC settings that were modified could be lost.

Deleting a Virtual NIC

Although it is uncommon, you might want to manually delete a vNIC when you no longer want the VM to have access to a specified network on the VM host. For example, if the initial need for connecting the VM to a network no longer exists or if the network is going to become private and the VM should not have access, you would delete the virtual NIC that allows connectivity.

To delete a virtual NIC, you can select the vNIC object in the Explorer tree, then right-click and select *Delete*, or you can use the following procedure:

1 In the Orchestration Console main menu, select *Actions* > *Delete* > *Delete Virtual NIC* to display the Create a New Virtual NIC dialog box.

2 In the *Source Objects* list, select the name of the vNIC (hold down the Ctrl key to select multiple vNICs), then click *Add* to move these objects to the *Delete Targets* list.

3 When you have selected all of the vNICs you want to delete, click *Delete* to display the Delete dialog box.

4 In the dialog box, select *Apply to all* to delete all of the vNIC objects in the *Delete Targets* list, click *OK*, then click *Close*.

5 In the Explorer tree, right-click the VM object where you deleted the vNIC, then select *Save Config* to apply the changes to the VM's configuration.

NOTE: You must run the *Save Config* action to confirm the deletion of the vNIC. If you do not run this action, the vNIC is not deleted from the VM's configuration.

8.2 Viewing the Virtual NIC Configuration in the Orchestration Console

You can visually expose a vNIC Grid object  in the Orchestration Console in two ways:

- In the Explorer Tree, select a VM Resource object, then select the *Info/Groups* tab in the Admin View to open the VM Info Groups page, then scroll to the *Virtual Machine Configuration* panel on that page. You can right-click the vNIC icon in that panel to display the four tabs in the Virtual Disk Admin view.
- In the Explorer Tree, click the expand/collapse icon of a VM Grid object, identify the vNIC icon, then select the icon to display the four tabs in the Virtual NIC Admin view.

The page that opens under the *Info/Groups* tab includes fields where you can configure the general information and attributes (facts) of the vNIC.

NOTE: Whenever you make changes to vNIC object facts, the write icon is superimposed on the object's icon  , signifying that the object has been altered. If you want to save the changes you have made, you need to click the *Save* button  on the Orchestration Console toolbar.

This section includes the following additional information:

8.2.1 Virtual NIC Info Panel

The Virtual NIC Info panel on the Info/Groups page includes the following sections:

Show Inherited Fact Values Check Box

Select this check box to show facts with overridden values supplied through attached and inherited policies. These fact values are read only (non-editable).

Virtual NIC Information

The Virtual NIC Information panel on the Info page includes the following fields:

NOTE: Tooltip text is available when you mouse over any of these fields.

Description: A free-form field you can use to add any description about this vNIC.

In the Fact Editor, this fact is listed as `vnic.description`:

```
<fact name="vnic.description" value="" type="String" />
```

Healthy: In most cases, this check box is selected by default, which designates the vNIC as being in good health.

We recommend that you do not change the *Healthy* value from its default.

In the Fact Editor, this is fact is listed as `vnic.health`:

```
<fact name="vnic.health" value="true" type="Boolean" />
```

MAC Address: Specifies the MAC address assigned to this vNIC. An empty string implies an auto-generated MAC address, as does an asterisk (*).

When the VM appears on the network, this will be its MAC identifier. A MAC address must be unique on the network to avoid routing conflicts.

There are some situations when you might want to define a static MAC address. For example, if a VM uses DHCP, you might want the DHCP service on the network to give the VM a static address. When the VM boots up and attempts to get an IP address, it contacts the DHCP server, which sees its statically-defined MAC address and then provides the same IP address (not a new one) each time the VM boots up. In this way, the VM is consistently configured with the same IP address. You might also want to define a static MAC address for audit trails or other security reasons.

In the Fact Editor, this fact is listed as `vnic.mac`:

```
<fact name="vnic.mac" value="" type="String" />
```

vBridge: Specifies the name of the actual host bridge used by this vNIC. When the VM is not running, this field is blank. When a VM is provisioned, a vBridge is chosen for this vNIC based on the available VM hosts and the specified network group. When the VM is running, the associated vBridge is identified in this field. For more information, see Chapter 9, "The Network Group and its Virtual Bridge Objects," on page 125.

In the Fact Editor, this read-only fact is listed as `vnic.vbridge`:

```
<fact name="vnic.vbridge" value="" type="String" />
```

VM: Specifies the name of the VM resource that uses this vNIC.

In the Fact Editor, this read-only fact is listed as `vnic.vm`:

```
<fact name="vnic.vm" value="" type="String" />
```

This is a fact junction referencing the associated VM. Conversely, the `resource.vm.vnics` fact visible from the VM Grid object is a fact junction showing the associated vNICs associated with the VM. A vNIC cannot be shared between two VMs. Each VM has its own vNIC objects.

Network: The network (vBridge group) that should be used by this vNIC when the VM is provisioned. This fact is used in combination with the VM host placement constraints to choose a suitable vBridge at provision time.

In the Fact Editor, this fact is listed as `vnic.network`:

```
<fact name="vnic.network" value="eth1" type="String" />
```

Autoprep/Sysprep Network Adapter

VMs can be prepared for provisioning by configuring the facts in this panel. Click *Define* for each field if the value has not been previously configured.

NOTE: When you change any of the settings in this panel, you need to right-click the VM and select *Personalize* for the changes to take effect.

- **MAC Address:** The MAC address of the interface. Specify an asterisk (*) to generate a new MAC address. If the value is not set, the existing `vnic.mac` is used.

  IMPORTANT: An unset *MAC Address* fact generates a new MAC address. This is contrary to the current tooltip text.

  In the Fact Editor, this fact is listed as `vnic.provisioner.autoprep.MACAddress`:

  ```
  <fact name="vnic.provisioner.autoprep.MACAddress" value="" type="String" />
  ```

- **Use DHCP:** When this check box is selected (it has a value of true), the VM is configured to retrieve its network settings from a DHCP server. If the check box is not selected (it has value of false), you should make sure that the IP address, subnet mask, and gateway address facts are defined.

  In the Fact Editor, this fact is listed as `vnic.provisioner.autoprep.UseDHCP`:

  ```
  <fact name="vnic.provisioner.autoprep.UseDHCP" value="false" type="Boolean" />
  ```

- **IP Address:** The IP address for the adapter.

  In the Fact Editor, this fact is listed as `vnic.provisioner.autoprep.IPAddress`:

  ```
  <fact name="vnic.provisioner.autoprep.IPAddress" value="" type="String" />
  ```

- **Subnet Mask:** The subnet mask for this adapter.

  In the Fact Editor, this fact is listed as `vnic.provisioner.autoprep.subnetMask`:

  ```
  <fact name="vnic.provisioner.autoprep.subnetMask" value="" type="String" />
  ```

- **Gateway IP Addresses:** (Windows sysprep only) A list of the gateway IP addresses available to the interface.

  In the Fact Editor, this fact is listed as an array:

  ```
  <fact name="vnic.provisioner.autoprep.Gateways">
    <array type="String">
    </array>
  </fact>
  ```

  You can edit this array by clicking the ··· button to open an array editor, where you can add or remove the IP address or change its order in the array of element choices.

- **DNS from DHCP:** (Optional. SUSE VM only) When this check box is selected (it has a value of true), the SUSE VM is configured to retrieve its DNS server settings from DHCP.

  In the Fact Editor, this fact is listed as `vnic.provisioner.autoprep.DNSFromDHCP`:

  ```
  <fact name="vnic.provisioner.autoprep.DNSFromDHCP" value="false"
  type="Boolean" />
  ```

- **DNS Server IP Addresses:** (Windows VM only) The adapter's list of DNS servers used for name lookup.

  In the Fact Editor, this fact is listed as an array:

  ```
  <fact name="vnic.provisioner.autoprep.DNSServers">
    <array type="String">
    </array>
  </fact>
  ```

- **DNS Domain:** (Windows VM only) The adapter's DNS domain name.

  In the Fact Editor, this fact is listed as `vnic.provisioner.autoprep.DNSDomain`:

  ```
  <fact name="vnic.provisioner.autoprep.DNSDomain" value="" type="String" />
  ```

- **Primary WINS Server:** (Windows VM only) The name of the adapter's primary WINS server.

  In the Fact Editor, this fact is listed as `vnic.provisioner.autoprep.primaryWINS`:

  ```
  <fact name="vnic.provisioner.autoprep.primaryWINS" value="" type="String" />
  ```

- **Secondary WINS Server:** (Windows VM only) The name of the adapter's secondary WINS server.

  In the Fact Editor, this fact is listed as `vnic.provisioner.autoprep.secondaryWINS`:

  ```
  <fact name="vnic.provisioner.autoprep.secondaryWINS" value="" type="String" />
  ```

- **NetBIOS:** (Windows VM only) The NetBIOS options for this VM. Options include:
  - *EnableNetBIOSviaDhcp*
  - *EnableNetBIOS*
  - *DisableNetBIOS*

  In the Fact Editor this fact is listed as `vnic.provisioner.autoprep.netBIOS`:

  ```
  <fact name="vnic.provisioner.autoprep.netBIOS" value="" type="String" />
  ```

8.2.2 Virtual NIC Policies Tab

The *Policies* tab opens a page that contains a policy viewer for each of the policies associated with a Grid object.

NOTE: You can edit a policy by right-clicking a policy icon, selecting *Edit Policy* and clicking the *Save* button.

8.2.3 Virtual NIC Health Debugger Tab

The Health Debugger is a common Admin view in the Orchestration Console for most Grid objects. For information about this tool, see "Health Debugger" on page 189.

8.2.4 Virtual NIC Constraints/Facts Tab

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties. In essence, by building, deploying, and running jobs on the Orchestration Server, you can individually change the functionality of any system resources by managing an object's facts and constraints. The Orchestration Server assigns default values to each of the component facts. Facts with no displayed mode can be changed at any time by the administrator. Facts with mode `r/o` have read-only values, which can be viewed by using the pencil button, but changes cannot be made.

For information about using the Fact Editor on this page to rename the Virtual NIC object, see Section 8.2.5, "Virtual NIC Object Naming and Renaming," on page 124.

8.2.5 Virtual NIC Object Naming and Renaming

Some resource names are generated by the Orchestration Server and can therefore receive generic, arbitrary names such as `mysql-vnic1`, `mysql-vnic2`, and so on. A Virtual NIC (vNIC) you name at creation time might also change later in its purpose or facilities.

The object's display name is visible in the Orchestration Console, the VM Client interface, and in optional zos and zosadmin commands.As the number of these vNIC objects grows in your grid, you might find it helpful or necessary to rename them, assigning more meaningful, intuitive names to suit the purpose of the object.

NOTE: Resource object groups (that is, the folders that contain these vNIC objects) can also be renamed. Objects such as jobs, events, and users cannot be renamed.

A vNIC object's name is stored in the `${objectType}.displayname` fact, which exists on every Grid object type, even those objects that cannot be renamed.

You can rename a vNIC object in the Orchestration Console by using one of three methods:

- Right-click the vNIC object in the Explorer tree, then select *Rename* to allow editing of the display name.
- Triple-click the vNIC object in the Explorer tree to allow editing of the display name.
- In the Constraints/Facts page, select the vNIC object `.displayname` fact, then open the Fact Editor to enter a new value for that fact.

As you begin to use one of these methods, you will notice that the fact value is prepopulated with the `${objectType}.id` fact. This functions as the name value for the object name until you decide to change it.

NOTE: Even after being renamed, the vNIC object retains its associated resource ID in the `.id` fact. This is not editable.

For more information about making the Resource object display names visible from the zos or zosadmin command line, see the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

9 The Network Group and its Virtual Bridge Objects

This section includes the following information:

9.1 Understanding the Network Group and Virtual Bridge Objects

9.1.1 Virtual Bridge Object

In the Cloud Manager Orchestration Server, a group of Virtual Bridge (vBridge) objects is called a “network.” It represents the networks (actually, LANs and VLANs) that are available to that VM host, and can be shared across multiple VM hosts. Network groups are created automatically during VM and VM host discovery as their virtual networking settings are determined. By default, the server automatically groups vBridges with the same name into a corresponding network with the same name, assuming that vBridges with similar names usually refer to the same network. After discovery, you can freely reassign the vBridges to other networks or multiple networks, depending on your organization’s physical network topology.

VLAN Fact

A Virtual LAN (VLAN) provides a “logical” LAN topology for a group of machines that does not need to depend on the switch hardware to which the machines (virtual or physical) are directly connected. Modern “smart” network switches can keep track of a VLAN ID on network traffic passing though them. Using this ID, the switches transparently route such traffic to the hosts configured to use the VLAN specified by the ID.

Using VLANs can reduce the costs of an organization’s physical network infrastructure. For example, if your organization requires multiple subnets or multi-homed hosts, you won’t need to buy new equipment for each new subnet or install multiple physical NICs on the physical machines; this can be done with VLANs using the same common physical network layer. VLANs allow greater flexibility in managing the network topology.

Where possible, the Orchestration Server discovers that a VLAN ID is in use on a network. The Virtual LAN Identifier fact is found on a Network object:

```
<fact name="group.vlanid" value="" type="String" />
```

The fact is populated with a positive integers value upon discovery of an already-existing VM host configuration. The server can track a VLAN ID for each Network object to allow correct management of individual VLANs as full-fledged networks.

The server applies a graphic overlay to the Network icon  to signify that the network was discovered as a VLAN. In cases where VLAN discovery is not accomplished through automation, you can also set the VLAN fact value manually.

9.1.2 The Purpose of the Virtual Bridge

The vBridge is discovered, along with its associated VM, when a discovery job runs on a VM host. A virtual bridge (vBridge) acts as a "virtual" Ethernet segment contained entirely within the software of a VM host. The virtual NIC (vNIC) devices on that host's VMs can each be assigned to one of the VM host's vBridges as if they were physically connected to a LAN.

Virtual bridges can also be associated with one or more physical NICs to combine the virtual LANs on these hosts into one common virtual LAN. This combined LAN is referred to as a "network" in the Orchestration Console. Association of virtual bridges is also frequently done to include a "virtual" LAN on a VM host in the organization's overall physical LAN topology so the VMs can access other systems in the organization as if they were directly connected to the switches.

The following points might help you understand the relationship of these objects:

- Associating a vNIC to a vBridge is like plugging a physical NIC into an Ethernet switch.
- Associating a vBridge with a physical NIC is like connecting two physical switches together using their uplink ports.
- A vBridge can become part of a vLAN by associating it with a physical NIC device that itself is configured as a VLAN.
- A VM host can have multiple vBridges, each of which can be connected to separate physical networks (for example, through 802.1Q VLAN tagging).
- When a VM is provisioned, a virtual NIC must be connected to a virtual bridge in order for the virtual NIC to be usable.

In the Explorer Tree, a vBridge is given the form *vmhostname*_eth*n* where *n* represents the numerical order in which this vBridge was discovered on the VM host, with 0 being appended to the name of the first vBridge discovered or created. For example, `host1_eth0` might be the name of the first bridge. Each additional vBridge is incremented by one, so the second vBridge in this example would be named `host1_eth1`.

9.1.3 Creating or Deleting a vBridge in the Orchestration Console

This section includes the following information:

- "Creating a Virtual Bridge" on page 127
- "Deleting a Virtual Bridge" on page 127

Creating a Virtual Bridge

You might want to manually create a vBridge if, for some reason, the discovery process did not find one of the vBridges in a network. Creating a new vBridge in the Orchestration Console does not "add" a physical bridge to the network, it only helps to model a physical bridge that was not previously discovered.

To create a vBridge, you can select the Network object in the Explorer tree, then right-click and select *New Virtual Bridge*. You can also use the following alternate method:

1. From the Orchestration Console main menu, select *Actions* > *Create* > *Create Virtual Bridge* to display the Create a New Virtual Bridge dialog box.
2. In the *Source Groups* list, select the Network object where you want to add a vBridge, then click *Add* to move it to the *Target Groups* list.
3. In the *New Virtual Bridge Name* field, specify the name you want to use to identify this vBridge. The data you enter here is completely free form. When the Orchestration Server discovers and names vBridges, it associates them by name as the default practice. This is done for convenience in locating the objects. A vBridge can be named anything, provided that it is a legal name for a vBridge on the VM host's operating system.

   It is not necessary to add the `<vmhostname>_` prefix on the name. This prefix is automatically prepended when the new object is created. For example, if you select bridge name `sales` on VM host `vmh1`, the actual object is created as `vmh1_sales`.
4. Click *Create*, then click *Close*.
5. In the Explorer tree, expand the Network group where you created the new vBridge object.
6. Select the new vBridge object to open its Info/Groups admin view.
7. Configure the settings described in Section 9.2.1, "Virtual Bridge Info/Groups Tab," on page 129.
8. Click the *Save* button to save the fact changes you have made.

Deleting a Virtual Bridge

Although it is uncommon, you might want to manually delete a vBridge when you no longer want the VM to have access to a specified network on the VM host. For example, if the initial need for connecting the VM to a network no longer exists or the network is going to become private and the VM should not have access you would delete the vBridge that allows connectivity.

To delete a virtual bridge, you can select the vBridge object in the Explorer tree, then right-click and select *Delete*, or you can use the following procedure:

1. In the Orchestration Console main menu, select *Actions* > *Delete* > *Delete Virtual Bridge* to display the Create a New Virtual Bridge dialog box.
2. In the *Source Objects* list, select the name of the vBridge (hold down the Ctrl key to select multiple), then click *Add* to move these objects to the *Delete Targets* list.
3. When you have selected all of the vBridges you want to delete, click *Delete* to display the Delete dialog box.
4. In the dialog box, select *Apply to all* to delete all of the vBridge objects in the *Delete Targets* list, click *OK*, then click *Close*.

9.1.4 Virtual Bridge Object Naming and Renaming

Some resource names are generated by the Orchestration Server and can therefore receive generic, arbitrary names such as `host1-eth1`, `host2-eth1`, and so on. A Virtual Bridge (vBridge) you name at creation time might also change later in its purpose or facilities.

The object's display name is visible in the Orchestration Console, in the VM Client interface, and in optional zos and zosadmin commands. As the number of these vBridge objects grows in your grid, you might find it helpful or necessary to rename them, assigning more meaningful, intuitive names to suit the purpose of the object.

NOTE: A Network object (that is, the group that contains these vBridge objects) can also be renamed. Objects such as jobs, events, and users cannot be renamed.

A vBridge object's name is stored in the `${objectType}.displayname` fact, which exists on every Grid object type, even those objects that cannot be renamed.

You can rename a vBridge object in the Orchestration Console by using one of three methods:

- Right-click the vBridge object in the Explorer tree, then select *Rename* to allow editing of the display name.
- Triple-click the vBridge object in the Explorer tree to allow editing of the display name.
- In the Constraints/Facts page, select the vBridge object `.displayname` fact and then open the Fact Editor to enter a new value for that fact.

As you use one of these methods, you will notice that the fact value is prepopulated with the `${objectType}.id` fact. This functions as the name value for the object name until you decide to change it.

NOTE: Even after being renamed, the vBridge object retains its associated resource ID in the `.id` fact. This is not editable.

For more information about making the Resource object display names visible from the zos or zosadmin command line, see the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

9.2 Viewing the Virtual Bridge Configuration in the Orchestration Console

You can visually expose a vBridge Grid object  in the Explorer tree by selecting a Network object to expand it, then selecting the vBridge you want to view in the Info/Groups page.

The page that opens under the *Info/Groups* tab includes fields where you can configure the general information and attributes (facts) of the vBridge.

NOTE: Whenever you make changes to vBridge object facts, the write icon is superimposed on the object's icon, signifying that the object has been altered. If you want to save the changes you have made, you need to click the *Save* button  on the Orchestration Console toolbar.

9.2.1 Virtual Bridge Info/Groups Tab

The page that opens under the *Info/Groups* tab includes two collapsible sections where you can configure the general information and attributes of the vBridge.

Virtual Bridge Info Panel

The *Info* panel on the *Info/Groups* page includes the following information:

Show Inherited Fact Values Check Box

Select this check box to show facts with overridden values supplied through attached and inherited policies. Such fact values are read only (non-editable).

Network Information

The *Network Information* panel on the Info/Groups page for the vBridge Grid object includes the following fields:

NOTE: Tooltip text is available when you mouse over any of these fields.

Description: Enter a description of the vBridge Grid object.

In the Fact Editor, this fact is listed as `vmhost.resource`:

```
<fact name="vbridge.description" value="" type="String" />
```

Vbridge Enabled: This check box is selected by default. When it is selected (it has a value of true), the vBridge is enabled and Virtual NICs can be attached to it.

In the Fact Editor, this fact is listed as `vbridge.enabled`:

```
<fact name="vbridge.enabled" value="true" type="Boolean" />
```

Healthy: This check box is selected by default. When it is selected (it has a value of true), the virtual bridge is designated as being in good health. You can set the health of the object by selecting or deselecting the health check box. Changing the value in this way has an immediate effect unless the value is overridden by an attached policy (this follows the normal rules of policy inheritance). For more information, see Appendix A, "Grid Object Health Monitoring," on page 187.

In the Fact Editor, this is fact is listed as `vbridge.health`:

```
<fact name="vbridge.health" value="true" type="Boolean" />
```

Attached Virtual NICs: This list box lists the Virtual NICs that are attached to this vBridge.

In the Fact Editor, this fact is listed as an array:

```
<fact name="vbridge.vnics">
  <array type="String">
  </array>
</fact>
```

The list includes vNICs from currently running VMs only. When a VM is not running, the Orchestration Server does not consider its vNICS as "currently attached." This is because vNICs are configured to associate with Network objects, and the attached vBridge might change, depending on which host the VM has been provisioned. The list can also change dynamically if the VM is migrated to another host (and that host's vBridge) on the same network.

Virtual Bridge Groups Panel

This section of the *Info/Groups* page lists the groups of vBridge objects (called "Networks") in the grid to which this vBridge is associated. Click *Choose* to open the Network Selection dialog box. In this dialog box, you can choose which networks to display in the Explorer tree by selecting a group and then clicking *Add* or *Remove* to move it to or from the *Source Networks* list.

9.2.2 Virtual Bridge Policies Tab

The *Policies* tab opens a page that contains a policy viewer for each of the policies associated with a Grid object. Click *Choose* to associate an existing policy with a vBridge.

NOTE: You can edit a policy by right-clicking a policy icon, selecting *Edit Policy*, and clicking the *Save* button.

9.2.3 Virtual Bridge Health Debugger Tab

The Health Debugger is a common Admin view in the Orchestration Console for most Grid objects. For information about this tool, see Appendix A.3, "Health Debugger," on page 189.

9.2.4 Virtual Bridge Constraints/Facts Tab

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties. In essence, by building, deploying, and running jobs on the Orchestration Server, you can individually change the functionality of any system resources by managing an object's facts and constraints. The Orchestration Server assigns default values to each of the component facts. Facts with no mode can be changed at any time by the administrator. Facts with mode `r/o` have read-only values, which can be viewed by using the edit pencil button, but changes cannot be made.

10 The Repository Object

Repositories are storage areas for VM image files and VM template files.

If a VM's files are stored on a particular host server, that VM must be run from that host server. If a VM's files are stored on a shared repository, that VM can be run on any host server that has access to the shared repository.

Host servers can have multiple repositories associated with them, and some repository types can be associated with multiple host servers as shared repositories. A host server can be associated with repositories stored locally on its server and with shared repositories stored on other machines.

The default size for all repositories is unlimited. To control disk space usage, you can change this default. For more information, see Capacity (MB): in "Repository Information Subpanel" on page 132.

The repository groups and their constituent repository objects are displayed in the Explorer panel and the accompanying Repository Admin View of the Cloud Manager Orchestration Console.

10.1 Right-Click Menu Actions on the Repository Object

The Repository object displayed in the Explorer tree has three available actions in the right-click menu:

- **Discover VM Images:** Use this action on a newly discovered or newly created repository to populate it with the VMs residing in the mapped datastore location.
- **Discover Disks:** Use this action to discover ISO disks stored in this repository.
- **Delete:** Use this action to delete the Repository object from the Explorer tree in the Orchestration Console.

  Before you delete the Repository object, you need to delete all the VMs contained in that repository; otherwise, an error message is displayed.

- **Rename:** Use this action to rename the Repository object. For more information, see Section 10.8, "Repository Object Naming and Renaming," on page 140.

10.2 Repository Groups

Any group object displayed in the Explorer panel represents a collection of similar object types. Groups can also be created automatically, such as when a provisioning adapter discovers a local repository on a VM host. For example, the xen provisioning adapter, upon discovery of a VM host, automatically creates a local repository for that VM host and places the created repository in a Xen repository group. You can also create groups manually in the Orchestration Console, either by clicking the *Actions* menu and choosing *Create Repository Group* or by right-clicking the Repository object (anywhere in the Repository hierarchy) and selecting *New Repository Group*.

10.3 Repository Info/Groups Tab

The page that opens under the *Info/Configuration* tab includes several collapsible sections on the page where you can configure the general information and attributes of the repository.

NOTE: Whenever you make changes to any Grid object, the write icon is superimposed on the object's icon, signifying that the object has been altered. If you want to save the changes you have made, you need to click the *Save* button on the Orchestration Console toolbar.

10.3.1 Info Panel

The following fields on the Information panel provide facts for the Repository object:

"Show Inherited Fact Values" Check Box

Select this check box to show facts with overridden values supplied through attached or inherited policies. These fact values are read only (non-editable).

Repository Information Subpanel

The Repository Information panel on the Info/Groups page includes the following fields:

NOTE: Tool tip text is available when you mouse over any of these fields.

Description: The nature or purpose of this repository.

In the Fact Editor, this fact is listed as `repository.description`:

```
<fact name="repository.description" value="" type="String" />
```

Repository Enabled: This check box is selected by default. When it is selected (it has a value of true), VMs can be moved to this repository or they can be provisioned from it.

In the Fact Editor, this fact is listed as `repository.enabled`:

```
<fact name="repository.enabled" value="true" type="Boolean" />
```

Healthy: This check box is selected by default. When it is selected (it has a value of true), the repository is designated as being in good health. You can set the health of the object by selecting or deselecting the health check box. Changing the value in this way has an immediate effect unless the value is overridden by an attached policy. For more information, see Appendix A, "Grid Object Health Monitoring," on page 187

In the Fact Editor, this is fact is listed as `repository.health`:

```
<fact name="repository.health" value="true" type="Boolean" />
```

Type: Select the repository type for this Repository object by selecting an option from the drop-down list.

In the Fact Editor, this fact is listed as `repository.type`:

```
<fact name="repository.type" value="local" type="String" />
```

The following table includes information about the various repository types:

Table 10-1 *Repository Types and Descriptions*

Repository Type	Description	Orchestration Console Icon
`local`	The default repository on a host server. Each host server starts with its own local repository, which has the same name as the server's Resource Grid object.	
`NAS` (Network Attached Storage)	Represents a NAS device connected to host servers (for example, NFS mount). This NAS device must be mounted and available on all host servers associated with this Repository Grid object.	
`SAN` (Storage Area Network)	Represents an iSCSI or Fibre Channel SAN. Currently supported only with the vsphere provisioning adapter.	
`datagrid`	The shared datagrid repository (named `zos`) is located on the Orchestration Server and is accessible to all host servers in the datagrid. By default, each host server has access to the `zos` datagrid repository.	
`virtual`	Represents an externally managed virtual disk (for example, Amazon EC2).	

IMPORTANT: If you have a vSphere environment with an iSCSI datastore based on an ESX 3.5 (or previous) host, the Orchestration Console incorrectly displays its type as *local* rather than *SAN*. This misrepresentation affects the accuracy of the VM host assignment (how the repositories are scored in the plan), and could possibly affect VM migration validation.

To work around this issue, set the type to *SAN*. You need to check that this setting is retained when another VM host or Repository discovery is executed.

SAN ID: (SAN repositories only) The SAN ID (the Virtual Fabric ID) for this repository.

In the Fact Editor, this fact is listed as `repository.id`:

```
<fact name="repository.id" value="test1" type="String" />
```

Root Location: The repository's logical root location. You can also think of this as the base location for all VM files and subdirectories contained within this repository.

In the Fact Editor, this fact is listed as `repository.location`:

```
<fact name="repository.location" value="/" type="String" />
```

The table below provides some examples you can consider as you enter a shared root path in this field. For more information, see "Best Practices for Entering Repository File Paths" on page 137.

Table 10-2 *Repository Types and Root Location Examples*

Repository Type	Root Location Examples
`local`	◆ `/` (`root`) ◆ `c:/vm`
`NAS` (Network Attached Storage)	This is the mount point that is assumed to be the same on every host server with a connection to this NAS. ◆ `/u` `/mnt/myshareddisk`
`SAN` (Storage Area Network)	Not required.
`datagrid`	`grid:///vms`
`virtual`	◆ `/` (`root`) ◆ `c:/vm`

VM Config Search Path: The relative path (from `repository.location`) to be used during discover of VM configuration files. This fact also implicitly includes the `resource.preferredpath` fact. For `xen30` repositories, the default path is `/etc/xen/vm`.

In the Fact Editor, this fact is listed as an array:

```
<fact name="repository.searchpath">
  <array>
    <string>/etc/xen/vm</string>
  </array>
</fact>
```

The ··· button opens the Attribute element values dialog box, where you can add, remove, or edit the path (element values) in an array of path choices.

The table below provides some examples you can consider as you enter a search path in this field.

Table 10-3 *Repository Types and VM Config Search Path Examples*

Repository Type	VM Config Search Path Examples
`local`	`/etc/xen/vm`
`NAS` (Network Attached Storage)	Each of these is the relative path from the location to search for VM configuration files. Null specifies to search the whole mount. ◆ *my_vms* ◆ *saved_vms* ◆ null (no path entry)
`SAN` (Storage Area Network)	Not required.
`datagrid`	`grid:///vms`
`virtual`	

Preferred Storage Path: The relative path (from `repository.location`) where you want the Orchestration Server to place the VM files after a move or a clone operation.

In the Fact Editor, this fact is listed as `repository.preferredpath:`

```
<fact name="repository.preferredpath" value="" type="String" />
```

IMPORTANT: If you use this field, do not include a leading forward slash (/) in the path. For more information, see "Best Practices for Entering Repository File Paths" on page 137.

Table 10-4 *Repository Types and Preferred Storage Path Examples*

Repository Type	Preferred Storage Path Examples
`local`	`var/lib/xen/images` for Xen VMs
`NAS` (Network Attached Storage)	*my_vms*
`SAN` (Storage Area Network)	Not required.
`datagrid`	`grid:///vms`
`virtual`	

Disk Discovery Path: The directories to search for disk image files

In the Fact Editor, this fact is listed as an array:.

```
<fact name="repository.disks.paths">
  <array type="String">
  </array>
</fact>
```

Disk Discovery Patterns: The patterns used to discover disk image files.

In the Fact Editor, this fact is listed as an array:

```
<fact name="repository.disks.patterns">
  <array>
    <string>*.iso</string>
  </array>
</fact>
```

Capacity (MB): The maximum amount (measured in MB) of storage space on the repository. The default (-1) designates an unlimited amount of space.

In the Fact Editor, this fact is listed as `repository.capacity`:

```
<fact name="repository.capacity" value="-1" type="Integer" />
```

Used Space (MB): The amount (measured in MB) of storage space used for VMs.

In the Fact Editor, this fact is listed as `repository.usedspace`:

```
<fact name="repository.usedspace" value="0" type="Integer" />
```

Free Space (MB): The amount (measured in MB) of storage space available to new VMs. The value is always set to `-1`, which designates an unlimited amount of space.

In the Fact Editor, this fact is listed as `repository.freespace`:

```
<fact name="repository.freespace" value="-1" type="Integer" />
```

Actual Used Space (MB): The actual amount (measured in MB) of storage space used for VMs.

In the Fact Editor, this fact is listed as `repository.usedspace.actual`:

```
<fact name="repository.usedspace.actual" value="0" type="Integer" />
```

Actual Free Space (MB): The actual amount (measured in MB) of storage space available to new VMs. A value of -1 means unlimited.

Efficiency: Enter an efficiency coefficient that the Orchestration Server uses to calculate the cost of moving VM disk images to and from the repository. This value is multiplied by the disk image size (in MB) to determine an efficiency score. A score of zero (0) means no cost (very efficient).

NOTE: A fact not visible in the UI (except in the Fact Editor) is `repository.capacity.set`, a Boolean flag used by provisioning adapters to indicate that the repository capacity has been discovered.

Stored VMs: The VM images stored in this repository. The list is aggregated from individual VM facts.

In the Fact Editor, this fact is listed as an array:

```
<fact name="repository.vmimages">
  <array type="String">
  </array>
</fact>
```

This fact designated as readonly and is not editable.

Compatible VM Hosts: The VM hosts capable of using this repository. The list is aggregated from individual VM facts.

In the Fact Editor, this fact is listed as an array:

```
<fact name="repository.vmhosts">
  <array>
    <string>tszen4_xen30</string>
  </array>
</fact>
```

You can edit this array by clicking the ... button to open the Attribute element values dialog box, where you can add, remove, or edit the VM host IDs (element values) in an array of VM host ID choices.

Accessed By Provision Adapters: The provisioning adapter jobs that are allowed access to VMs on this repository.

In the Fact Editor, this fact is listed as an array:

```
<fact name="repository.provisioner.jobs">
  <array>
    <string>xen30</string>
  </array>
</fact>
```

You can edit this array by clicking the ... button to open the Choose Grid Objects dialog box, where you can add or remove provisioning adapters for the array of provisioning adapter choices.

NOTE: In the Fact Editor, you edit the provisioning adapter array by using the *Attribute Element Values* dialog box.

SAN Adapter Configuration

SAN Adapter Vendor: (SAN repositories only) The name of the vendor of the SAN.This should be adapter specific, such as `iqn`, `npiv`, `emc`. An empty field (that is, no value in the string) indicates that bind/unbind is a no-op (no operation performed).

In the Fact Editor, this fact is listed as `repository.san.vendor`:

```
<fact name="repository.san.vendor" value="" type="String" />
```

SAN Transport: (SAN repositories only) From the drop-down list, select *iSCSI* or *Fibre Channel* to indicate the type of SAN transport this repository uses.

In the Fact Editor, this fact is listed as `repository.san.type`:

```
<fact name="repository.san.type" value="" type="String" />
```

10.3.2 Best Practices for Entering Repository File Paths

Use the following guidelines in scenarios where you need VM repositories.

- "Creating a Repository to Use with New VMs" on page 138
- "Creating a Repository to Use with Existing VMs" on page 138
- "Creating a Repository for Existing VMs with Shared Root Locations and Separate Configuration Directories" on page 138

Creating a Repository to Use with New VMs

If you are creating a repository for new VMs that you will eventually provision:

1. In the *Root Location* field, specify the location for the new repository.

   Example: `/vms_new`

2. In the *Preferred Storage Path* field, specify the path to your image file store (relative to the root location path). This becomes the path for VM configuration files and VM image files when you associate a VM with this repository.

   Example: `images` (no leading forward slash)

   Because the fields are concatenated, the provisioning adapter searches for the existing VM files in `/vms_new/images`.

Creating a Repository to Use with Existing VMs

Use this procedure when you already have VMs in your grid and a store for the VM configuration and disk image files already exists.

1. In the *Root Location* field, specify the shared location for this repository.

   Example: `/vms_new`

2. In the *VM Config Search Path* field, specify the search path to your existing configuration file store (relative to the root location path).

   Example: `old_config` (no leading forward slash)

   Because the fields are concatenated, the provisioning adapter searches for the existing VM configuration files in `/vms_new/old_config`.

3. In the *Preferred Storage Path* field, specify the path to your existing image file store (relative to the root location path). This also becomes the path for VM configuration files and VM image files when you associate a VM with this repository.

   Example: `all_images` (no leading forward slash)

   Because the fields are concatenated, the provisioning adapter searches for the existing VM files in `/vms_new/all_images`.

Creating a Repository for Existing VMs with Shared Root Locations and Separate Configuration Directories

Use this procedure when you want to create a repository for existing VMs that have a shared root path but separate configuration file directories such as `/vms_new/old_config1` and `/vms_new/old_config2`).

1. In the *Root Location* field, specify the shared location for this repository.

   Example: `/vms_new`

2. In the *VM Config Search Path* field, specify the search paths to your existing configuration file store (relative to the Root Location path).

   Example: Adjacent to the *VM Config Search Path* field, click ..., click *Add element*, enter `old_config1` (no leading forward slash), click *OK*, click *Add element* again, specify `old_config2` (no leading forward slash), then click *OK*.

Because the fields are concatenated, the provisioning adapter searches for the existing VM configuration files in the array consisting of `/vms_new/old_config1` and `/vms_new/old_config2`.

3 In the *Preferred Storage Path* field, specify the path to your existing image file store (relative to the root location path).This path also becomes the path for VM configuration files and VM image files after a move or clone when a VM has been associated with this repository.

Example: `all_images` (no leading forward slash)

Because the fields are concatenated, the provisioning adapter searches for the existing VM files in `/vms_new/all_images`.

10.3.3 Groups

This section of the Info/Groups page lists the groups of Repository objects in the grid. Click *Choose* to open the repository Group selection dialog box. In this dialog box, you can choose which Repository Groups to display in the Explorer Panel by selecting a group, then clicking *Add* or *Remove* to move it to or from the *Source Repository Groups* list.

10.4 Repository Policies Tab

The *Policies* tab opens a page that contains a policy viewer for each of the policies associated with a Grid object.

NOTE: You can edit a policy by right-clicking a policy icon, selecting *Edit Policy,* then clicking the *Save* button.

10.5 Repository Health Debugger Tab

The Health Debugger is a common Admin view in the Orchestration Console for most Grid objects. For information about this tool, see Appendix A.3, “Health Debugger,” on page 189.

10.6 Repository Constraints/Facts Tab

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties.

By building, deploying, and running jobs on the Orchestration Server, you can individually change the functionality of any system resources by managing an object’s facts and constraints. The Orchestration Server assigns default values to each of the component facts, although they can be changed at any time by the administrator, unless they are read-only. Facts with mode `r/o` have read-only values, which can be viewed by using the pencil icon, but changes cannot be made.

10.7 The Repository Action History Tab

The *Action History* tab is displayed in the administrative view of the Repository object. When you select the *Action History* tab, a table displays a list of the history for all actions performed on this Grid object.

The Orchestration Server must be connected to an audit database for the *Include Audit Database* check box to be available. If the *Include Audit Database* check box is selected in this view, the action status is not polled. Click the *Refresh* button in the toolbar to retrieve and display fresh data.

For more details about the information listed on the Action History page, see "Action History in Admin Views of the Orchestration Console" in the *NetIQ Cloud Manager 2.1 VM Orchestration Reference*.

10.8 Repository Object Naming and Renaming

Some resource names are generated by the Orchestration Server and can therefore have generic, arbitrary names such as `host1`, `host2`, `host3`, and so on. Repositories you name at creation time might also change later in their purpose or facilities.

The object's display name is visible in the Orchestration Console, the VM Client interface, and in optional zos and zosadmin commands. As the numbery of these Repository objects grows in your grid, you might find it helpful or necessary to rename them, assigning more meaningful, intuitive names to suit the purpose of the object.

NOTE: Repository object groups (that is, the folders that contain these Repository objects) can also be renamed. Objects such as jobs, events, and users cannot be renamed.

A Repository object's name is stored in the `${objectType}.displayname` fact, which exists on every Grid object type, even those objects that cannot be renamed.

You can rename a Repository object in the Orchestration Console using one of three methods:

- Right-click the Repository object in the Explorer tree, then select *Rename* to allow editing of the display name.
- Triple-click the Repository object in the Explorer tree to allow editing of the display name.
- In the Constraints/Facts page, select the Repository object `.displayname` fact and then open the Fact Editor to enter a new value for that fact.

As you use one of these methods, you will notice that the fact value is prepopulated with the `${objectType}.id` fact. This functions as the name value for the object name until you decide to change it.

NOTE: Even after being renamed, the Repository object retains its associated resource ID in the `.id` fact. This is not editable.

For more information about making the Resource object display names visible from the zos or zosadmin command line, see the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

10.9 Shared Storage for Disk Images

The Orchestration Server can discover disk files (such as ISOs) inside storage represented by a Repository Grid object, making it easier to specify a disk file that you want to attach to a VM.

This section includes the following information:

10.9.1 Setting Disk Discovery Facts

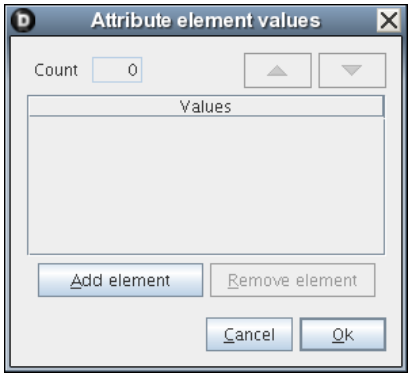
Before the Orchestration Server can discover disks, you need to set two facts in the Repository object. These facts are listed in the table below.

Table 10-5 *Required Facts for ISO Disk Discovery*

Fact Name	Purpose
`repository.disks.paths`	The directories to search for disk image files
`repository.disks.patterns`	The patterns used to discover disk image files (the default pattern is `*.iso`).

You can use the built-in array editor in the Orchestration Console to modify these facts. For example, to modify the `repository.disks.paths` fact,

1. In the Orchestration Console, select a Repository object that represents your shared storage.
2. Select the *Info/Groups* tab to open *Info* page in the admin view.
3. In the *Disk Discovery Path* field, click the array editor button ... to open the *Attribute element values* dialog box.



4 On the dialog box, click *Add element* to open the editor where you can add a string value to the `repository.disks.paths` fact. For example, you could enter `/data/isos` as a possible search directory for disk image files.

5 Click the *Save* icon on the main toolbar to save your fact changes.

A similar array editor is available for changing the values in the repository.disks.patterns fact found in the *Disk Discovery Patterns* field of the Repository *Info* page of the Orchestration Console.

10.9.2 Running the Discovery

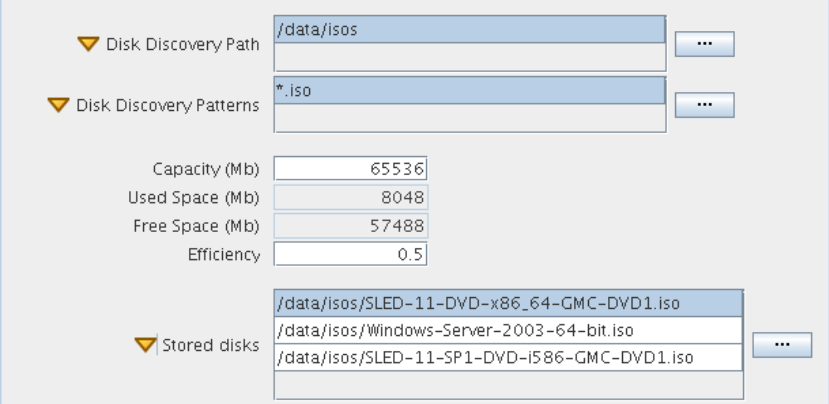
The *Discover Disks* action is included in the right-click menu of a Repository Grid object and in the *Provision* main menu of the Orchestration Console. The right-click action lets you discover disks on a single repository, while the action under the *Provision* menu lets you select multiple repositories for disk discovery.

When you select this action, a provisioning adapter function runs to search for and retrieve ISO disk filenames stored on the selected repository or repositories. If the selected provisioning adapter does not support the *Discover Disks* action, the option is dimmed in the menu.

NOTE: The Orchestration Server can discover any disk, but its default is to search for filenames with a `.iso` suffix.

After discovery, the disk filenames are stored in the `repository.disks` fact of the Repository object, located in the *Stored disks* field of the Repository's *Info/Groups* page in the admin view.

Figure 10-1 *The Disk Discovery Portion of the Info/Groups Page for the Repository Object*



This is an array of disks, each of which is a valid value for the `vdisk.location` vDisk fact and stored in this repository. You can edit the filenames of these disks or add new filenames as values in the array.

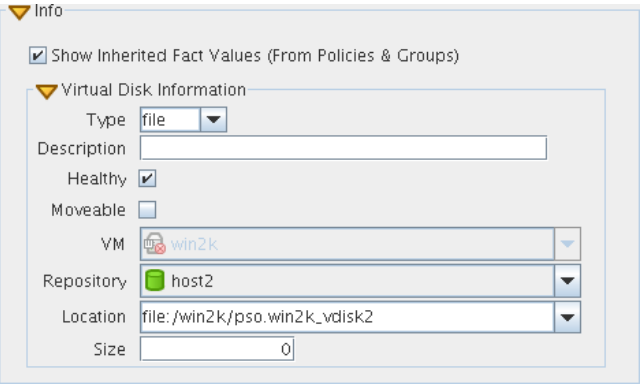
10.9.3 Sharing Disks Between VMs

To share ISOs between VMs, you would need to store the ISOs in a shared location that is accessible on all VM hosts where those VMs are to run. In such a scenario, you would set up a directory on all VM hosts where `/data/isos/` is a mount point to some SAN or NFS storage of ISO files. For example, an ISO file might have this path: `/data/isos/SLES-11-SP1-DVD-x86_64-DVD1.iso`.

10.9.4 Attaching a Discovered Disk to a VM

You can attach a discovered disk to a VM.

1. In the Orchestration Console, select the VM to which you want to attach a disk, then right-click to display the available actions for the VM.
2. Select *Create Virtual Disk* to create a new disk. The Info/Groups page for the disk is displayed in the admin view.



   The *Location* drop-down menu lists the ISO file names that were previously discovered in the Repository associated to this vDisk.
3. In the *Location* field (that is, the `vdisk.location` fact), select the disk that you want to attach to the VM. You can also manually edit the filename of the disk you want to attach.
4. Click the *Save* icon on the main toolbar to save your changes.
5. Commit the changes to the VM configuration.
   - **5a** (If the VM is running) In the Explorer tree, right-click the VM object, select *Apply Config*.
   - **5b** (If the VM is not running) In the Explorer tree, right-click the VM object, select *Save Config*.

     If the VM state or the provisioning adapter do not support either of these actions, the options are dimmed.

10.9.5 Using Attached Disks in the Guest OS

After the disk is attached to a VM, you might need to take additional steps for the disk to be usable.

Linux Guest OS: Use the `mount` command to make the disk visible to the guest OS. Use the `umount` command to unmount the disk prior to changing or deleting the underlying PSO vDisk object.

Windows Guest OS: If the disk is not immediately visible, you need to reboot your guest OS. In general, the you can assume that disk is visible if it is replacing an already-mounted disk that is visible in a CD-ROM device.

11 The User Object

A User object represents an individual account that is allowed to connect to the Cloud Manager Orchestration Server. Administrator users are also allowed to connect by using the `zosadmin` command line and the Orchestration Console user interfaces.

You can use the Orchestration Console user interface to manually create a User object. You can create objects automatically if authentication through LDAP or Active Directory is enabled, or optionally if autoregistration is configured.

The user object icon and the red square user object icon.

11.1 User Groups

Any group object displayed in the Explorer panel represents a collection of similar object types. Groups can also be created automatically, such as when a provisioning adapter discovers a local repository on a VM host. For example, the xen provisioning adapter, upon discovery of a VM host, automatically creates a local repository for that VM host and places the created repository in a Xen repository group. You can also create groups manually in the Orchestration Console, either by clicking the *Actions* menu and choosing *Create User Group* or by right clicking a User Group object (anywhere in the User hierarchy) and selecting *New User Group*.

11.2 User Info/Groups Tab

The page that opens under the *Info/Configuration* tab of the User admin view includes several collapsible sections on the page where you can configure the general information and attributes of the user.

NOTE: Whenever you make changes to any Grid object, the write icon is superimposed on the object’s icon, signifying that the object has been altered. If you want to save the changes you have made, you need to click the *Save* button on the Orchestration Console toolbar.

11.2.1 Info

The following fields on the Information panel provide facts for the User object:

Show Inherited Fact Values Check Box

Select this check box to show facts with overridden values supplied through attached or inherited policies. These fact values are read only (non-editable).

User Information

The User Information panel on the Info/Groups page includes the following fields:

NOTE: Tooltip text is available when you mouse over any of these fields.

Account Enabled: This check box is selected by default.When the check box is selected, the user is allowed to log in and run jobs.

In the Fact Editor, this fact is listed as `user.enabled`:

```
<fact name="user.enabled" value="true" type="Boolean" />
```

Online: When this check box is selected (it has a value of true), the user is currently logged in to the server.

In the Fact Editor, this fact is listed as `user.online`:

```
<fact name="user.online" value="false" type="Boolean" />
```

Healthy: This check box is selected by default. When it is selected (it has a value of true), the user is designated as being in good health. You can set the health of the object by selecting or deselecting the health check box. Changing the value in this way has an immediate effect unless the value is overridden by an attached policy. For more information, see Appendix A, "Grid Object Health Monitoring," on page 187

In the Fact Editor, this is fact is listed as `user.health`:

```
<fact name="user.health" value="true" type="Boolean" />
```

External Groups: A list of external groups (for example, LDAP) that this user belongs to.

In the Fact Editor, this fact is listed as an array:

```
<fact name="user.external.groups">
  <array type="String">
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open the Attribute element values dialog box, where you can add, remove, or edit the name and value for every user environment you want to use.

Personal Information

First Name: The user's first name.

In the Fact Editor, this fact is listed as `user.name.first`:

```
<fact name="user.name.first" value="" type="String" />
```

Last Name: The user's last name.

In the Fact Editor, this fact is listed as `user.name.last`:

```
<fact name="user.name.last" value="" type="String" />
```

Password: The user's hashed login password.

In the Fact Editor, this fact is listed as `user.password`:

```
<fact name="user.password" value="kNLj1_Fc96C3ajVXcqQEGZBRrbivgxhhzK3TKLpP"
type="String" />
```

IMPORTANT: The System User password should not be changed. If you try to change this password by using the Orchestration Console, the password does not actually change and no warning is posted that the change was unsuccessful. There is no indication of authentication failure until you try to log in with the "new" password.

EMail: The user's e-mail address.

In the Fact Editor, this fact is listed as `user.name.email`:

```
<fact name="user.name.email" value="" type="String" />
```

City: The name of the city where the user is located.

In the Fact Editor, this fact is listed as `user.location.city`:

```
<<fact name="user.location.city" value="" type="String" />
```

State: The name of the state or province where the user is located.

In the Fact Editor, this fact is listed as `user.location.state`:

```
<fact name="user.location.state" value="" type="String" />
```

Country: The name of the country where the user is located.

In the Fact Editor, this fact is listed as `user.location.country`:

```
<fact name="user.location.country" value="" type="String" />
```

Site: The name of the site (for example, a campus or building) where the user works.

In the Fact Editor, this fact is listed as `user.location.site`:

```
<fact name="user.location.site" value="" type="String" />
```

Environment: A list of default user environment variable names and values that the Orchestration Server sets when executing joblets remotely.

In the Fact Editor, this fact is listed as a dictionary:

```
<fact name="user.env">
  <dictionary>
    <dictelement key="dfadsafd">
      <string>safdaf</string>
    </dictelement>
  </dictionary>
</fact>
```

Job Information

Total Job Count: The total number of jobs that this user has historically initiated on this Orchestration Server.

In the Fact Editor, this fact is listed as `user.history.jobcount`:

```
<fact name="user.history.jobcount" value="0" type="Integer" />
```

Active Jobs: The number of top-level jobs run with this user account that are in an active state.

In the Fact Editor, this fact is listed as `user.jobs.active`:

```
<fact name="user.jobs.active" value="0" type="Integer" />
```

Queued Jobs: The number of top-level jobs run with this user account that are currently in a queued state.

In the Fact Editor, this fact is listed as `user.jobs.queued`:

```
<fact name="user.jobs.queued" value="0" type="Integer" />
```

Total Jobs: The total number of top-level jobs run by this user account.

In the Fact Editor, this fact is listed as `user.jobs.total`:

```
<fact name="user.jobs.total" value="0" type="Integer" />
```

Active Sessions: The number of currently active sessions (that is, connections) that the user has established with the Orchestration Server.

In the Fact Editor, this fact is listed as `user.sessions`:

```
<fact name="user.sessions" value="1" type="Integer" />
```

Accounting Information

Total Spending: The total cost of computing resources by this user.

In the Fact Editor, this fact is listed as `user.history.cost.total`:

```
<fact name="user.history.cost.total" value="0.0088" type="Real" />
```

Average Spending Rate: The computed moving average spending (in dollars per hour) over the last hour of activity for this user.

In the Fact Editor, this fact is listed as `user.account.spendrate`:

```
<fact name="user.account.spendrate" value="-0.0006" type="Real" />
```

Maximum Spending Rate: An amount (in dollars per hour) to be used by the Resource Scheduler to throttle the rate at which computing cycles are consumed by the user. A value of less than or equal to zero (<=0) turns the feature off.

In the Fact Editor, this fact is listed as `user.account.maxspendrate`:

```
<fact name="user.account.maxspendrate" value="0.0000" type="Real" />
```

Default Accounting Group: Lets you select the default User Group to be billed for work conducted by this user.

In the Fact Editor, this fact is listed as `user.accounting.group`:

```
<fact name="user.accountinggroup" value="all" type="String" />
```

Total Wall Time: The total amount of wall time (in seconds) consumed by this user.

In the Fact Editor, this fact is listed as `user.history.time.total`:

```
<fact name="user.history.time.total" value="31" type="Integer" />
```

Total Grid Time: The total amount of grid time (in gcycles, which is a normalized average of compute cycles) consumed by this user.

In the Fact Editor, this fact is listed as `user.history.gcycles.total`:

```
<fact name="user.history.gcycles.total" value="31" type="Integer" />
```

Job Control

Default Priority Value: A numerical representation of the default priority at which this user's job runs, with 1 being the lowest priority and 9 being the highest priority.

In the Fact Editor, this fact is listed as `user.priority.default`:

```
<fact name="user.priority.default" value="7" type="Integer" />
```

Default Priority: The string representation of the default priority at which this user can run a job. The value is matched to the integer value in `user.priority.default`.

In the Fact Editor, this fact is listed as `user.priority.default.string`:

```
<fact name="user.priority.default.string" value="high" type="String" />
```

Maximum Priority Value: A numerical representation of the maximum priority at which this user's job can run, with 1 being the lowest priority and 9 being the highest priority. Only the system user can run jobs at priority 10.

In the Fact Editor, this fact is listed as `user.priority.max`:

```
<fact name="user.priority.max" value="5" type="Integer" />
```

Datagrid Maximum History: The maximum number of job instance directories that should be kept in the datagrid for this user.

In the Fact Editor, this fact is listed as `user.datagrid.maxhistory`:

```
<fact name="user.datagrid.maxhistory" value="25" type="Integer" />
```

Job Preemption Enabled: Select this check box if you want to allow the user to preempt jobs that have a priority less than the priority of the running job instance.

In the Fact Editor, this fact is listed as `user.preemption.enabled`:

```
<fact name="user.preemption.enabled" value="false" type="Boolean" />
```

Max Preemption Priority: The highest job priority band from which this user is allowed to preempt resources. The value acts as a delta from the current job instance priority. The maximum preemptible priority is always less than or equal to `user.priority.max`.

In the Fact Editor, this fact is listed as `user.preemption.priority.delta`:

```
<fact name="user.preemption.priority.delta" value="0" type="Integer" />
```

Resources Stealing Enabled: Select this check box to allow the user to steal resources that are running jobs that have a priority less than the priority of the running job instance.

In the Fact Editor, this fact is listed as `user.stealing.enabled`:

```
<fact name="user.stealing.enabled" value="false" type="Boolean" />
```

Max Stealing Priority: The highest job priority band from which this user is allowed to steal resources. The value acts as a delta from the current job instance priority, and must be less than zero (<0).

In the Fact Editor, this fact is listed as `user.stealing.priority.delta`:

```
<fact name="user.stealing.priority.delta" value="-1" type="Integer" />
```

Privileged Job Groups: A list of Job Groups with jobs and joblets that this user is allowed to run on resources that have reached their slot maximum or that are provisioned resources that are reserved for another user or job.

In the Fact Editor, this fact is listed as an array:

```
<fact name="user.privilegedjobgroups">
  <array type="String">
  </array>
</fact>
```

You can edit this array by clicking the ··· button to open the Choose Grid Objects dialog box, where you can add or remove Job Groups in an array of choices. The Job Groups can be added to or removed from a list of Source Grid Objects to a list of Target Grid Objects (or vice versa).

Quota Information

Account Balance Remaining: The balance (measured in dollars) that remains available for this user since the last reset. You can use this value to implement quotas on your server.

In the Fact Editor, this fact is listed as `user.account.balance`:

```
<fact name="user.account.balance" value="0.0000" type="Real" />
```

Job Counter: The number of jobs this user has initiated since the last reset. You can use this value to implement quotas on your server.

In the Fact Editor, this fact is listed as `user.jobcount`:

```
<fact name="job.history.jobcount.complete" value="0" type="Integer" />
```

Time Remaining: The amount of wall time (measured in seconds) remaining for use by this user since last the reset. You can use this value to implement quotas on your server.

In the Fact Editor, this fact is listed as `user.account.time`:

```
<fact name="user.account.time" value="0" type="Integer" />
```

Grid Time Remaining: The amount of grid time (measured in gcycles) remaining for use by this user since last the reset. You can use this value to implement quotas in your grid.

In the Fact Editor, this fact is listed as `job.history.jobcount.complete`:

```
<fact name="user.account.gcycles" value="0" type="Integer" />
```

11.2.2 Groups

This section of the Info/Groups page lists the groups of User objects in the grid. Click *Choose* to open the User Group Selection dialog box. In this dialog box, you can choose which User Groups to display in the Explorer Panel by selecting a group and then clicking *Add* or *Remove* to move it to or from the *Source Job Groups* list.

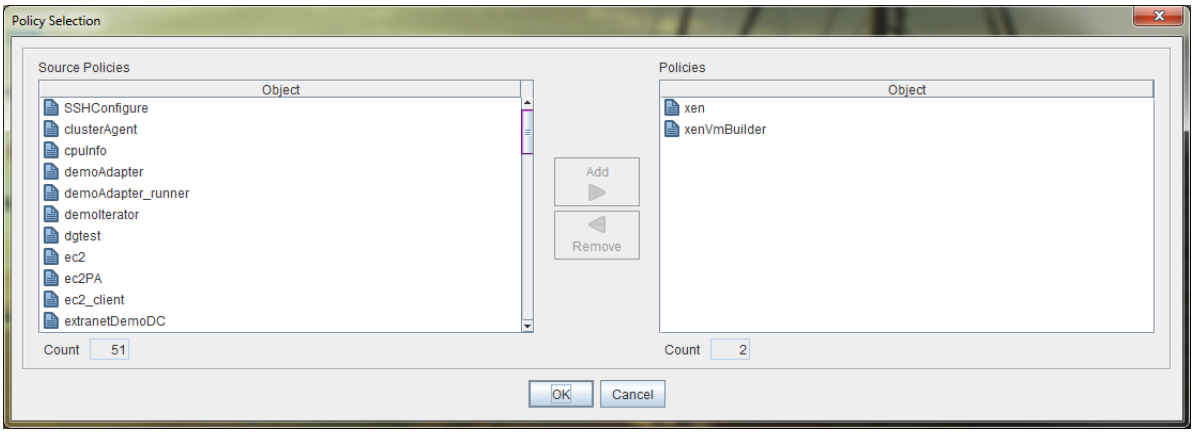
11.3 User Policies Tab

The *Policies* tab of the User admin view opens a page that contains a policy viewer for each of the policies associated with a User Grid object.

You can modify policies by using the Policy Grid object. For more information, see Section 12.1, "Policy Object," on page 153.

If you click *Choose* on the *Policy* tab, a Policy Selection dialog box is launched, where you can add or remove individual policies to be applied to the selected User Grid object.

Figure 11-1 *The Policy Selection Dialog Box*



11.4 User Health Debugger Tab

The Health Debugger is a common Admin view in the Orchestration Console for most Grid objects. For information about this tool, see Section A.3, "Health Debugger," on page 189.

11.5 User Constraints/Facts Tab

The *Constraints/Facts* tab opens a page that shows all of the effective constraints and facts for a Grid object. Each Grid object has an associated set of facts and constraints that define its properties.

By building, deploying, and running jobs on the Orchestration Server, you can individually change the functionality of any system resources by managing an object's facts and constraints. The Orchestration Server assigns default values to each of the component facts. Facts with no displayed mode can be changed at any time by the administrator. Facts with mode `r/o` have read-only values, which can be viewed by using the pencil button, but changes cannot be made.

11.6 The User Action History Tab

The *Action History* tab is displayed in the administrative view of the User object. When you select the *Action History* tab, a table displays a list of the history for provisioning actions performed on this Grid object (assuming that it is provisionable, for example, a VM or VM template).

The Orchestration Server must be connected to an audit database for the *Include Audit Database* check box to be available. If the *Include Audit Database* check box is selected in this view, the action status is not polled. Click the *Refresh* button in the toolbar to retrieve and display fresh data.

For more details about the information listed on the Action History page, see "Action History in Admin Views of the Orchestration Console" in the *NetIQ Cloud Manager 2.1 VM Orchestration Reference*.

12 Miscellaneous Objects Displayed in the Explorer Tree

The Explorer panel (tree) of the Orchestration Console displays some miscellaneous objects of importance:

12.1 Policy Object

XML is used to define Orchestration Server policies. A policy can be deployed to the server and associated with any grid object. The `policy` element is the root element for policies. Policies contain constraints and fact definitions for grid objects.

You can edit a policy by clicking its icon in the Explorer tree view, making your changes, then clicking the save icon. The *Where Used* tab of the Policy Editor lists the jobs where the selected policy is associated.

12.1.1 Policy Constraints

A policy can define a collection of constraints that are applied appropriately based on context. For example, a resource constraint can limit the selection of a resource to a subset based on resource group membership, or any number of other fact-based evaluations.

You can use `/opt/novell/zenworks/zos/server/examples/customNode.policy` on the Orchestration Server as an example policy file with constraints.

12.1.2 Policy Facts

The XML fact element defines a fact to be stored in the grid object’s fact namespace. The name, type and value of the fact are specified as attributes. For list or array fact types, the element tag defines list or array members. For dictionary fact types, the `dict` tag defines dictionary members.

You can see an example policy with an XML representation for all the fact types on the Orchestration Server at `/opt/novell/zenworks/zos/server/examples/allTypes.policy`.

12.2 Computed Fact Objects

Computed facts are used when you want to run JDL to generate the value for a fact. Although computed facts are not jobs, they use the same JDL syntax. You can see examples of computed facts on the Orchestration Server at `/opt/novell/zenworks/zos/server/examples/activejobs.cfact` and `/opt/novell/zenworks/zos/server/examples/repostiory.cfact`.

12.3 Event Objects

An Event object in the Explorer tree represents a user-described set of rules that can be associated with a schedule trigger or handled by long-running jobs written to respond to events.

For more information about using events, see "Using an Event Notification in a Job" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*, "Event Triggers" on page 161 of this guide, and Section A.2, "Health Events," on page 189 of this guide.

12.4 Metrics Objects

The Orchestratino Server Metrics Facility collects, aggregates, and allows simple fact-based retrieval of metric values by jobs and computed facts (via JDL), policy constraints, and Event triggers on a per-resource basis.

A Metrics object is deployed in the Explorer tree. Use the right-click menu to display the "deploy" and "undeploy" actions. Pre-defined `.metric` files are located in the `/opt/novell/zenworks/zos/server/components/metrics` folder, or you can create a new `.metric` file and paste in an XML file.

NOTE: You can also use the zosadmin deploy command to deploy a `.metric` file. For example:

```
zosadmin deploy load_one.metric
```

Metrics objects are listed by their deployment name, which may or may not be the same as the name of the actual metric. This potentially allows multiple, separately deployable, RRD definitions for a single "instantaneous" metric, with different aggregation periods defined.

For more information about using metrics, see Appendix C, "The Metrics Facility," on page 205 of this guide, and "Resource Metrics Facts" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

13 The Orchestration Server Job Scheduler

You can use the Job Scheduler in the Orchestration Server to automatically start deployed jobs on your grid by using either time or event triggers.

You can think of the functionality provided by the time triggers as being similar to a distributed cron system (in fact, time triggers can be described in cron syntax). This triggering, coupled with the job control functions in the Orchestration Server, allows for the sophisticated automation of routine data center tasks.

For example, suppose you want to periodically harvest a large log file in a coordinated way from a farm of several hundred machines. First, you could create an Orchestration Server job that uses the datagrid for file movement. The job control options specify that the job should run on not more than three machines at once and sweep across the entire grid. You would then create a schedule to run this job at the desired interval.

As another example, you could use the Job Scheduler to trigger a discovery job every time a new resource is added to the grid. In this case, the job developer writes the discovery job to discover and set facts about the resource. Next, you would create a schedule to run this job on the `RESOURCE_ONLINE` built-in trigger. In fact, this type of triggered job is currently used in the standard set of deployed discovery jobs to detect specific resource CPU and OS information.

Yet another example would be to run a job on server startup that sends a notification e-mail to an administrator.

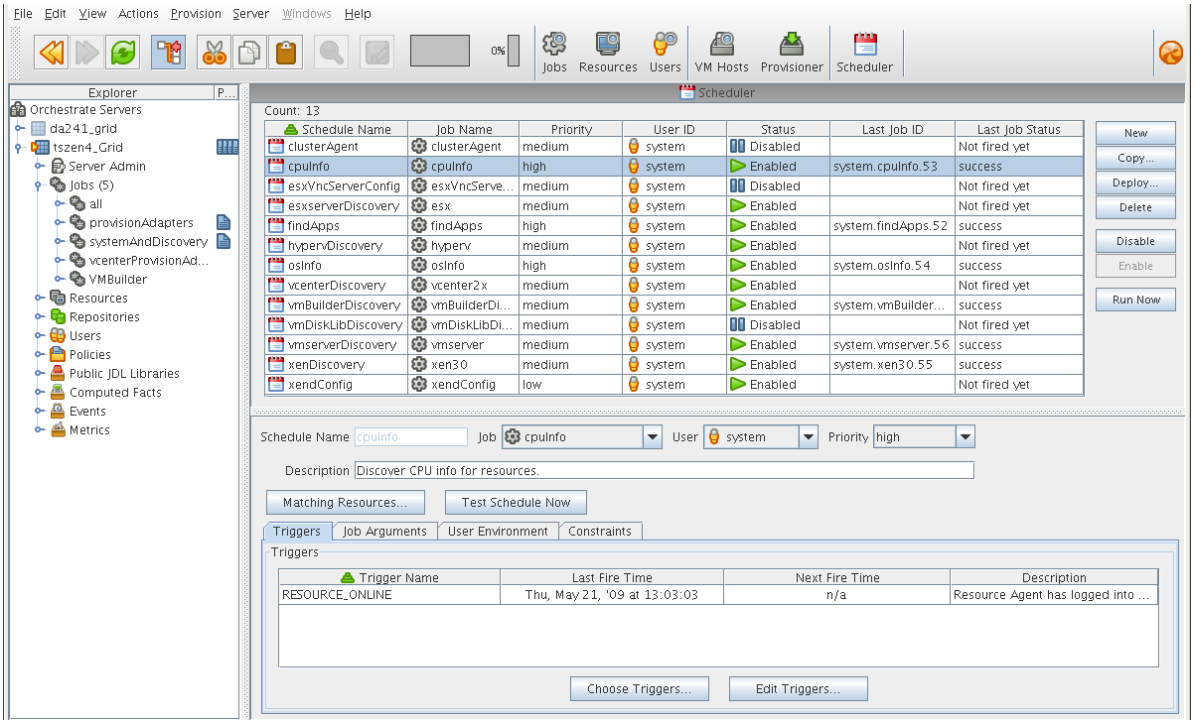
This section includes the following information:

13.1 Understanding the Job Scheduler View

This section includes information to help you understand the functions of the Job Scheduler and how to use it to launch Orchestration Server jobs.

Click *Scheduler* on the main toolbar of the Orchestration Console to open the Job Scheduler view.

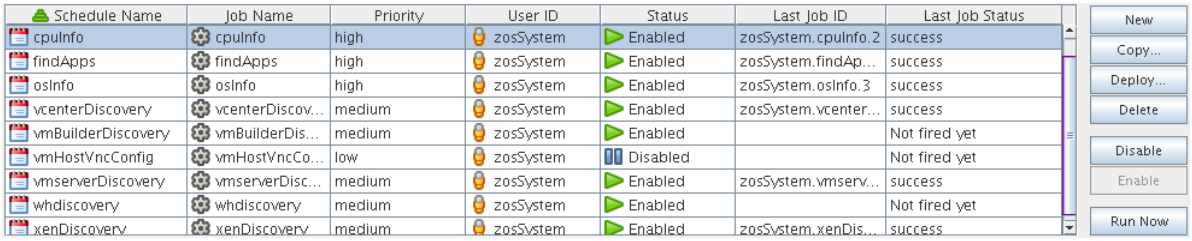
Figure 13-1 *Job Scheduler View of the Orchestration Console*



13.1.1 Navigating The Job Schedules Table

The Orchestration Server includes several predefined and predeployed discovery jobs that have predefined launch schedules. Among these jobs are the `cpuinfo`, `findapps`, `osinfo`, and other jobs, depending on the options (that is, the server profile) you chose and the configuration you used during the installation. After installation, these jobs are listed by name in a table in the Job Scheduler view.

Figure 13-2 *The Job Schedules Table in the Job Scheduler View*



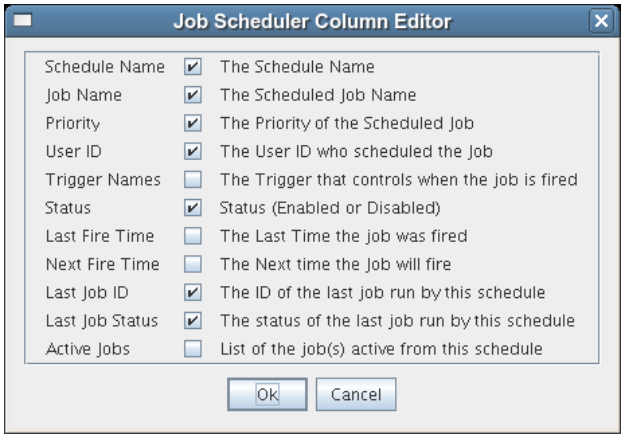
By default, the server uses schedule names that are similar to the job name so that schedules are easy to match (although this is not required). The schedules list shows all of the existing job schedules that accompany predefined jobs, along with the schedules that you create in the Job Scheduler.

NOTE: The Job Scheduler view is not a real-time monitor view of jobs, so if a job attribute (for example, Last Job Status or Last Fire Time) has changed, it might not be displayed until you click *Refresh*.

The Job Schedules Table has functionality that lets you decide how you want to display information about the job schedules:

- You can drag any column in the table to move it left or right in the table according to your preference.
- You can mouse over any column heading in the table to view tooltip text about the purpose of the data in that column.
- You can right-click any column heading in the table to open the Job Scheduler Column Editor dialog box.

Figure 13-3 *Job Scheduler Column Editor Dialog Box*



You can select any column heading in this dialog box to display it in the Job Schedules Table. The columns display the attributes of a previously configured job schedule. As the figure shows, this dialog box also includes text that clarifies the purpose of the data in each column.

In the Job Scheduler view, there are seven function buttons next to the Job Schedules Table (see Figure 13-2 on page 156) that let you take action on any schedule you select inside the table. (Only one schedule at a time can be selected.)

- **New:** Opens a dialog box where you can create a new schedule. When you create a new schedule, the Job Scheduler adds a new line to the Job Schedules Table. When the new line is added, you can use the Job Schedule Editor to edit the attributes of the schedule. A new schedule must be given a unique schedule name.

  The Job Scheduler forces a new schedule to be created in the *Disabled* state to prevent it from running while it is being defined. You click *Enable* when a job is ready to be used.
- **Copy:** Copies a schedule you have selected in the Job Schedules Table. Clicking this button opens a dialog box where you rename the copy. If you want to create a schedule that is similar but not identical to an existing schedule, use this button to save time in adding attributes to a job schedule configuration. A copy of a schedule must be given a unique schedule name.
- **Deploy:** Opens a dialog box where you can select a schedule (that is, a deployable `.sched` file) to deploy.
- **Delete:** Deletes the selected schedule from the Job Schedules Table. You cannot recover a deleted job schedule.

NOTE: Deleting a schedule that was deployed as part of a `.job` or `.sar` displays a confirmation dialog box. Deleting the schedule undeploys all contents of the `.job` or `.sar` that contains the schedule.

- **Disable:** Disables the selected schedule in the Job Schedules Table. The jobs associated with the schedule are not re-run, but any currently running instances of this job continue to run.
- **Enable:** Enables a disabled job schedule.
- **Run Now:** Forces the specified schedule to run immediately. This updates statistics such as *Last Fire Time*.

Removed Jobs or Users: Scheduler Behavior

If a job or a user is undeployed or removed from the Orchestration Server, the Job Schedules Table continues to list the schedule previously associated to that removed grid object, but the removed grid object no longer displays the icon that represents the object (job or user).

Figure 13-4 *User Object and Job Object Icons Are Not Displayed*

Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Last Job Status
CpuDiscovery	cpuInfo	high	zosSystem	Disabled		Not fired yet
clusterAgent	quickie	medium	zosSystem	Disabled		Not fired yet
findApps	findApps	high	zosSystem	Enabled	zosSystem.findAp...	success
osInfo	osInfo	medium	system	Enabled	zosSystem.osInfo.48	success
vcenterDiscovery	vcenterDiscov...	medium	zosSystem	Enabled	zosSystem.vcenter...	success
vmHostVncConfig	vmHostVncCo...	low	zosSystem	Disabled		Not fired yet
vmserverDiscovery	vmserverDisc...	medium	zosSystem	Enabled	zosSystem.vmserv...	success
xenDiscovery	xenDiscovery	medium	zosSystem	Enabled	zosSystem.xenDis...	success

In the preceding figure, the *CpuDiscovery* schedule displays no Job icon for the *cpuInfo* job in the schedules table. Even though the job has been undeployed, the schedule is still listed.

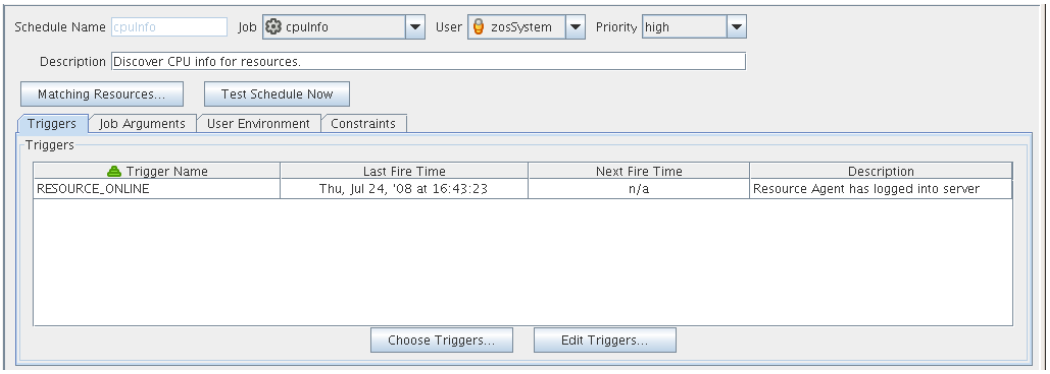
In the *osinfo* schedule, the *system* user has no User icon. That user has been removed from the server.

If you choose a new user or job to be associated with a schedule, a deleted or undeployed user or job is never displayed in the pop-up menu for that schedule again.

13.1.2 Creating or Modifying a Job Schedule

The Job Schedule Editor is located immediately below the Job Schedules Table in the Job Scheduler view.

Figure 13-5 *The Job Schedule Editor in the Job Scheduler View*



There are several times when you use this part of the Job Scheduler tool:

- When you create a new schedule by clicking *New*.

- When you modify the attributes of an existing schedule (available when you select a schedule in the table).
- When you create a copy of an existing schedule by clicking *Copy*.

The Job Schedule Editor lets you create or modify a job schedule by specifying its attributes.

You use the following controls and data when you create or modify a job schedule:

Schedule Name

When you create a new schedule, the unique name you specify is displayed in this field. If you select a schedule from the Job Schedules Table, the name of the schedule is displayed in this field. The field is not editable, because schedules cannot be renamed after they are created. (You can use a copy if this is required.)

Job

When you create a new schedule, you need to associate a deployed job with it. You can select the job you want to run from this drop-down list.

If you want to use a previously created schedule to run a different job, you can change the job here.

User

When you create a new schedule, you need to associate a user with it. The user represents the user for whom the job will run. The choice of user might affect the permissions, privileges and constraints of the job. You can select the user from this drop-down list.

If you want a different user to run a job on a previously created schedule, you can change the user here.

If you decide to change the user who runs the job, check the *Priority* field to make sure that the priority you want is selected.

Priority

When you create a new schedule and associate a job and a user with it, a list of possible run priorities becomes available in this drop-down list. The list of priorities varies, depending on the user that is specified in the previous field. In this field, you select the priority of the job that is to be run so that if other jobs are to start concurrently or are competing for resources, the Orchestration Server can determine which job takes priority.

Description

For predeployed jobs, this field contains a default description of what the job's schedule does. The field is editable, so you can enter a description of your own for job schedules that you create.

Matching Resources

This button displays a list of resources where the job runs now or where it could run. This list is useful for checking the context of constraints that might have been affected by a choice of user or by manually specifying additional constraints under the *Policy* tab. The list is also useful to verify that a discovery job (that is, one that is triggered by the *Run on Resource Start* option) runs on the preferred set of machines.

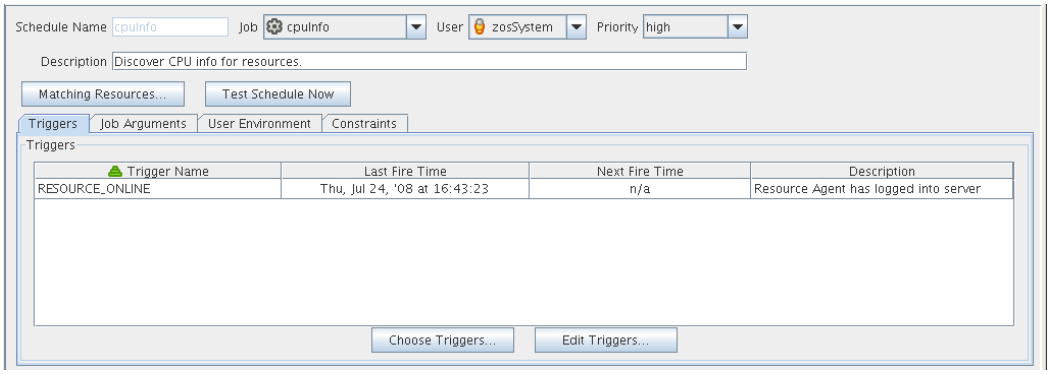
Test Schedule Now

Click this button to test the new or modified schedule you are working with. The test runs the new or modified schedule without permanently saving the current configuration of the schedule or recording statistics. This control differs from the *Run Now* control in the Job Schedules Table, which runs a saved (persisted) schedule, disregarding any unsaved modifications that have been made to it in the Job Schedule Editor.

Triggers

When you click the *Triggers* tab in the Job Scheduler view, the following page opens:

Figure 13-6 *The Schedule Triggers Page in the Job Scheduler*

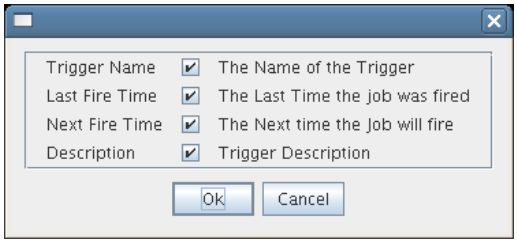


In this view, you can add or define the triggers you want to associate with job schedules. A trigger is the signal to the Job Scheduler to initiate, or "fire" a schedule at a given time or at the occurrence of a given event. Job Scheduler triggers can be classified with regard to two conditions: events and time.

The Triggers table on this page has functionality that lets you decide how you want to display information about the triggers:

- You can drag any column in the table to move it left or right in the table according to your preference.
- You can mouse over any column heading in the table to view tooltip text about the purpose of the data in that column.
- You can right-click any column heading in the table to open the Triggers table column editor dialog box.

Figure 13-7 *Trigger Table Column Editor Dialog Box*



You can select any column heading in this dialog box to display it in the Triggers table. The columns display the attributes of a previously configured Triggers table. As the figure shows, this dialog box also includes text that clarifies the purpose of the data in each column.

You can create as many triggers as you want to meet any scheduling situation you might have. Multiple time triggers can be associated with a schedule and multiple schedules can use the same trigger. The triggers you create are retained by the Job Scheduler for you to choose from when you create a schedule for a job. The currently associated triggers are displayed in the list along with a description.

Choose Triggers

This button opens a dialog box where you can choose both predefined and user defined time triggers to associate with this job schedule.

In this dialog box, you can click *Add* to move a selected trigger to the active, scheduled triggers that are to be associated with this job schedule. You can also click *Remove* to unassociate a trigger.

When a trigger is moved to the scheduled list, it becomes associated to the job schedule and it is displayed in the Job Scheduler view.

Most example jobs in the Orchestration Server are associated with event triggers. The dialog box can also list other job schedule triggers that are based on time.

Event Triggers

An Event trigger is the signal to the Job Scheduler to initiate, or “fire” a job when a given event occurs. An Event can be one of three types:

- **Event objects:** These objects are user defined events that are fired when an event rule is triggered. If an event object is deployed, it automatically shows in the trigger chooser as a possible choice.

- **Built-in events:** These events are system-wide events such as a resource coming online or a changing resource health condition. Built-in events are always available as a trigger choice. The Job Scheduler has eight possible built-in event triggers:
  - AGENT_VERSION_MISMATCH
  - RESOURCE_ONLINE
  - REPOSITORY_HEALTH
  - RESOURCE_HEALTH
  - SERVER_UP
  - USER_HEALTH
  - USER_ONLINE
  - VMHOST_HEALTH

  You can select any combination of these event triggers for a single schedule.

  The first trigger, AGENT_VERSION_MISMATCH, triggers the job when an Orchestration Agent of an incompatible version attempts to connect to this Orchestration Server. It can be used to initiate a configuration management tool for an agent software update or the job could e-mail a message to an administrator to report the incompatible agent. The other seven available built-in event triggers are listed with accompanying descriptions in the dialog box.
- **External events:** These events are fired by an outside process. These are not automatically shown as choices in the trigger chooser, but must be defined by the trigger editor.

An event trigger can be used in conjunction with a time trigger to allow flexibility in scheduling the job application for maximum effectiveness or convenience. Jobs triggered by events require that their job arguments contain a dictionary named context. For example, your event-triggered job should have this jobarg element in its policy:

```
<policy>
   <jobargs>
      <fact name="context" type="Dictionary"
            description="Dictionary containing the context for the event" />
   </jobargs>
</policy>
```

The key/values of the dictionary are dependent on the event type. For event objects, the jobargs.context dictionary contains the matching context of the triggered rule. For built-in events, the jobargs.context dictionary contains the key of the object type corresponding to the built-in event and the object ID that caused the event.

For example, if the USER_ONLINE event triggers because the user named foo logs in, the jobargs.context dictionary contains:

```
{ user : foo }
```

Likewise, if the RESOURCE_ONLINE event is triggered because the resource agent named “vmhost1” comes online, the jobargs.context dictionary contains:

```
{ resource : vmhost1 }
```

For the AGENT_VERSION_MISMATCH event, the jobargs.context dictionary contains more information, as shown in the following table:

Table 13-1 *Dictionary Information*

Key	Type
AgentBuild	Long
AgentIP	String
AgentId	String
AgentMajor	Integer
AgentMinor	Integer
AgentPoint	Integer
JavaMajor	Integer
JavaMinor	Integer
JavaPoint	Integer
JavaVendor	String
JavaVersion	String
OsMajor	Integer
OsMinor	Integer
OsName	String
OsPoint	Integer
OsVendor	String
OsVersion	String
SystemArch	String
UsingJRE	Boolean
resource	String

Time Triggers

A time trigger is the signal to the Job Scheduler to initiate, or "fire" a job when a prescheduled time occurs. A time trigger can be used in conjunction with an event trigger to allow flexibility in scheduling the job application for maximum effectiveness or convenience. No default time triggers are defined in the Job Scheduler. You need to create new time triggers by clicking Edit Triggers.

Edit Triggers

Click *Edit Triggers* to open the Triggers dialog box.

The following controls and information are available in the dialog box:

- **New:** Opens a secondary dialog box where you can create a new time trigger name. When you create the trigger name, the attribute fields in the Triggers dialog box are cleared and you can specify new attributes for the trigger. A new trigger must be given a unique trigger name.

- **Copy:** Lets you modify an existing time trigger by giving it a new name and attributes. This can be helpful if there are only slight differences in the new attributes. A copy of a trigger must be given a unique trigger name.
- **Deploy:** Opens a file chooser where you can choose an existing, stored trigger (that is, a `.trig` file) to deploy.
- **Delete:** Deletes a selected time trigger.

  IMPORTANT: Deleted triggers are not recoverable. If the trigger is used by existing schedules, it is removed from all of those schedules when it is deleted.

- **Trigger Name:** Specifies the unique name of the trigger you are creating or modifying. This name is displayed in the Job Scheduler if you choose to associate this trigger with a schedule. After you create the trigger name, it cannot be modified.
- **Description:** Specifies a description for the time trigger you are creating or modifying. The description is optional and can be as detailed as you want.

  If the number of characters in the description string exceeds the space in the *Description* field, a  button is enabled that opens a string editor when clicked.
- **Save:** Clicking this button saves the defined time trigger and its attributes.
- **Fire Starting In:** Displays multiple fields specifying the time increment and frequency to be used by the trigger to fire the job. If you select this type of time trigger, the *Fire using CRON Expression* button becomes inactive.

  NOTE: You can use the *Fire Starting In* control to create either a "one-shot" time trigger or a "reoccurring" time trigger.

  A one-shot time trigger fires just once after a specified period of time. To specify a one-shot trigger, click *Fire Starting in*, specify the amount of time before firing, then specify 0 as the time to *Repeat Indefinitely*.

  A reoccurring time trigger fires after a specified period and then either fires repeatedly for an indefinite number of times or it fires for a specified number of times. To specify a reoccurring, indefinite trigger, click *Fire Starting in*, specify the amount of time before firing, then select *Repeat Indefinitely*. To specify a reoccurring but finite trigger, click *Fire Starting in*, specify the amount of time before firing, select *Repeat Range*, then specify the number of times you want the trigger to fire.

- **Fire using CRON Expression:** Specifies the cron expression that enables the job to fire automatically at a specified time or date. You need to be familiar with cron to use this field.

  The *Examples* list box of selected cron expressions and their associated descriptions is located just below this button. You can use a listed expression as is, or use it as a template to modify the expression to meet your needs.

  If you select this type of time trigger, the *Fire Starting In* and the *Fire Using Event* buttons become inactive.

  For an example of how a cron expression can be implemented in a trigger, see "Creating and Assigning a Time Trigger for the New Schedule" on page 174. For detailed information about cron syntax, see "Understanding Cron Syntax in the Job Scheduler" on page 167.

- **Fire Using Event:** Specifies a deployed event or an external event that enables the job to fire when a specified event occurs. Deployed events are defined using an XML syntax. You can specify a deployed event from *Events* (that is, listed in the *Events* drop down list) or you can enter the name of an external event.

  If the number of characters in the *Fire Using Event* description string exceeds the space in the field, a ... button is enabled that opens a string editor when clicked.

Job Arguments

This tab displays an area in the lower left corner of the Job Schedule Editor where possible job arguments are listed. If you select an existing schedule in the Job Schedules Table, any optional job arguments (jobargs) for the associated job are displayed in this area.

Figure 13-8 *The Job Arguments Area of the Job Scheduler View*

The jobargs are defined by the deployed job. Some jobs might already have a default value displayed, but others must have values specified in order for the job to be able to run.

IMPORTANT: Job arguments displayed in blue are required. You must supply data in the accompanying fields.

A job argument defines the values that can be passed in when a job is invoked. These values let you statically define and control job behavior. To learn more about each job argument, mouse over each jobarg line to display tool tip text.

The Job Scheduler uses the values you enter into the fields of this area to build a jobargs namespace in the policy for this job.

Each job argument has an accompanying *Lock* check box. When *Lock* is not selected, the accompanying job argument uses the default value specified in the job's policy. When *Lock* is selected, the value specified in the field is locked down and overrides the default value in the policy. A locked value continues to be used even if the policy value is modified.

You can click *Restore Jobargs* to restore job arguments to the values specified in the job policy. This function removes any changes you might have specified in the Job Scheduler and deselects all *Lock* check boxes.

For more information, see "Job Arguments and Parameter Lists in Policies" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

User Environment

This tab displays an area in the lower left corner of the Job Schedule Editor that includes the *Pass User Environment* check box. Select this check box if you want to pass the assigned user's environment variables to the job when it runs. When environment variables are recorded on the user account, selecting the *Pass User Environment* check box makes those environment variables available to the job and joblet.

A user's environment is recorded under the `user.env` fact on his or her account. This fact can be set when a user logs in to the Orchestration Server and is persisted until changed. A user's environment variables can be uploaded with the zos command line tool at login time in one of two variations:

`zos login --user=foo --env`

This command uploads the entire environment to the Job Scheduler. The upload can also be seen on the User object in the Orchestration Console.

`zos login --user=foo --env=PATH`

When the user logs in, he or she can specify one or more environment variables to use at login. The example above would result in just the PATH environment variable being uploaded.

Multiple environment variables can be specified by delimiting with a comma, as in the following example:

```
--env=PATH,LD_PATH,ID
```

NOTE: The user's environment variables can also be passed to the server when the user implements the zos command line tool when running a job (as opposed to logging in). The command passes the environment variable only for that particular job run.

`zos run jobname --env=environment_variable`

Constraints

This tab displays a constraint editor that you can use to create additional constraints for the job being scheduled. Typically, additional resource constraints (such as "start") are useful to delay the start of a job when it is triggered.

Any XML constraints listed in this tab needs to have a top-level XML `<constraints>` element, as in the following example:

```
<constraints>
  <constraint type="resource">
    <contains fact="resource.groups" value="myResourceGroup" />
  </constraint>
</constraints>
```

For more information about working with constraints, see "Scheduling with Constraints" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

13.1.3 Understanding Cron Syntax in the Job Scheduler

The cron triggers you can configure in the Orchestratin Server Job Scheduler use a Quartz crontrigger class for deciding when to invoke job execution. This is based on the standard Quartz format that you can find further described on the OpenSymphony (http://www.opensymphony.com/quartz/wikidocs/CronTriggers%20Tutorial.html) Web site, or the KickJava (http://kickjava.com/src/org/quartz/CronTrigger.java.htm) Web site.

This section includes the following information:

Format

A cron expression is a string comprised of 6 or 7 fields separated by white space. Fields can contain any of the allowed values, along with various combinations of the allowed special characters for that field. The fields are explained in the following table:

Table 13-2 *Fields in a Cron Expression*

Field Name	Mandatory?	Allowed Values	Allowed special Characters
Seconds	Yes	0-59	, - * /
Minutes	Yes	0-59	, - * /
Hours	Yes	0-23	, - * /
Day of the Month	Yes	1-31	, - * ? / L W
Month	Yes	1-12 or JAN-DEC	, - * /
Day of the Week	Yes	1-7 OR SUN-SAT	, - * ? / L #
Year	No	EMPTY, 1970-2099	, - * /

Cron expressions can be as simple as this:

```
* * * * ? *
```

Or cron expressions can be more complex, like this:

```
0 0/5 14,18,3-39,52 ? JAN,MAR,SEP MON-FRI 2002-2011
```

Special Characters

Cron syntax incorporates logical operators, which are special characters that perform operations on the values provided in the cron fields.

Table 13-3 *Special Characters in the Orchestration Server Cron Syntax*

Operator	Purpose	Example
asterisk (*)	Specifies all possible values for a field	An asterisk in the hour time field is equivalent to “every hour.”
question mark (?)	A question mark (?) is allowed in the day-of-month and day-of-week fields. It is used to specify “no specific value,” which is useful when you need to specify something in one of these two fields, but not in the other.	If you want a trigger to fire on a particular day of the month (for example, the 10th), but you don't care what day of the week that is, enter 10 in the day-of-month field, and ? in the day-of-week field.
dash (-)	Specifies a range of values	2-5, which is equivalent to 2,3,4,5
comma (,)	Specifies a list of values	1,3,4,7,8
slash (/)	Used to skip a given number of values	*/3 in the hour time field is equivalent to 0,3,6,9,12,15,18,21. The asterisk (*) specifies “every hour,” but the /3 means only the first, fourth, seventh. You can use a number in front of the slash to set the initial value. For example, 2/3 means 2,5,8,11, and so on.
L (“last”)	The L character is allowed for the day-of-month and day-of-week fields. Specifies either the last day of the month, or the last *xxx* day of the month.	The value L in the day-of-month field means “the last day of the month,” which is day 31 for January, or day 28 for February in non-leap years. If you use L in the day-of-week field by itself, it simply means 7 or SAT. But if you use it in the day-of-week field after another value, it means “the last *xxx* day of the month.” For example, 6L means “the last Friday of the month.” **TIP**: When you use the L option, be careful not to specify lists or ranges of values. Doing so causes confusing results.

Operator	Purpose	Example
W (“weekday”)	The `W` character is allowed for the day-of-month field. Specifies the weekday (Monday-Friday) nearest the given day.	If you specify `15W` as the value for the day-of-month field, the meaning is “the nearest weekday to the 15th of the month.” So if the 15th is a Saturday, the trigger fires on Friday the 14th. If the 15th is a Sunday, the trigger fires on Monday the 16th. If the 15th is a Tuesday, it fires on Tuesday the 15th. However, if you specify `1W` as the value for day-of-month, and the 1st is a Saturday, the trigger fires on Monday the 3rd, because it does not “jump” over the boundary of a month’s days. The `W` character can only be specified when the day-of-month is a single day, not a range or list of days. **TIP**: You can combine the `L` and `W` characters for the day-of-month expression to yield `LW`, which translates to “last weekday of the month.”
pound sign (#)	The pound sign (#) character is allowed for the day-of-week field. This character is used to specify “the nth” *xxx* day of the month.	The value of `6#3` in the day-of-week field means the third Friday of the month (day `6` = Friday and `#3` = the 3rd one in the month). **Other Examples:** `2#1` specifies the first Monday of the month and `4#5` specifies the fifth Wednesday of the month. However, if you specify `#5` and there are fewer than 5 of the given day-of-week in the month, no firing occurs that month.

NOTE: The legal characters and the names of months and days of the week are not case sensitive. `MON` is the same as `mon`.

You can specify days in two fields: month day and weekday. If both are specified in an entry, they are cumulative, meaning that both of the entries are executed.

Examples of Cron Syntax

The following table shows examples of full cron expressions and their respective meanings.

Table 13-4 *Results of Altered Cron Syntax on Execution Times*

Cron Expression Example	Description
`0 0 12 * * ?`	Fire at 12:00 p.m. (noon) every day
`0 15 10 ? * *`	Fire at 10:15 a.m. every day
`0 15 10 * * ?`	Fire at 10:15 a.m. every day
`0 15 10 * * ? *`	Fire at 10:15 a.m. every day
`0 15 10 * * ? 2012`	Fire at 10:15 a.m. every day during the year 2012

Cron Expression Example	Description
`0 * 14 * * ?`	Fire every minute starting at 2:00 p.m. and ending at 2:59.p.m., every day
`0 0/5 14 * * ?`	Fire every five minutes starting at 2:00 p.m. and ending at 2:55 p.m., every day
`0 0/5 14,18 * * ?`	Fire every five minutes starting at 2:00 p.m. and ending at 2:55 p.m., **and** fire every five minutes starting at 6:00 p.m. and ending at 6:55 p.m., every day
`0 0-5 14 * * ?`	Fire every minute starting at 2:00 p.m. and ending at 2:05.p.m., every day
`0 10,44 14 ? 3 WED`	Fire at 2:10 p.m. and at 2:44 p.m. every Wednesday in the month of March
`0 15 10 ? * MON-FRI`	Fire at 10:15 a.m. every Monday, Tuesday, Wednesday, Thursday and Friday
`0 15 10 15 * ?`	Fire at 10:15 a.m. on the 15th day of every month
`0 15 10 15 * ?`	Fire at 10:15 a.m. on the last day of every month
`0 15 10 ? * 6L`	Fire at 10:15 a.m. on the last Friday of every month
`0 15 10 ? * 6L 2011-2014`	Fire at 10:15 a.m. on every last Friday of every month during the years 2011, 20012, 2013, and 2014
`0 15 10 ? * 6#3`	Fire at 10:15 a.m. on the third Friday of every month
`0 0 12 1/5 * ?`	Fire at 12:00 p.m. (noon) every five days every month, starting on the first day of the month
`0 11 11 11 11 ?`	Fire every November 11th at 11:11 a.m.

Cron Scheduling Precautions

You should remember the following items when you use cron scheduling:

- Always check the effect of adding the `?` and `*` characters in the day-of-week and day-of-month fields to make sure the expected behavior fires correctly.
- Support for specifying both a day-of-week and a day-of-month value is not complete. You must currently use the `?` character in one of these fields.
- Be careful when setting fire times to occur between 12:00 a.m. and 1:00 a.m. Changing to or from daylight saving time can cause a skip or a repeat in the schedule firing, depending on whether the clock moves backward or forward.

13.2 Walkthrough: Scheduling a System Job

This section demonstrates how you can use the Orchestration Console to deploy and schedule an existing system job named `auditCleaner.job`. This example job is included in the `examples` directory of your Orchestration Server installation.

This section includes the following information:

13.2.1 Deploying a Sample System Job

Before a job can run, the Orchestration Server administrator must deploy it, which involves moving it from a developed package state to a state where it is ready and available for users. Only the administrator has the necessary rights to deploy a job.

There are four methods you can use to deploy a job:

- Deploy it from the Orchestration Console by right-clicking the *Jobs* container in the Explorer panel.
- Deploy it from the Orchestration Console by selecting the *Actions* menu in the console.
- Deploy it from the `zosadmin` command line (`zosadmin deploy path_to_job`).
- Copy the deployable component to the "hot" deployment directory under the Orchestration Server instance directory. Typically, this directory is located at `/var/opt/novell/zenworks/zos/server/deploy`. Using this method, deployment proceeds within a few seconds. The server monitors this directory.

A runnable job can also be scheduled, which means that the schedule for running the job and the trigger or triggers that initiate or "fire" the schedule (or both) are configured and packaged with the job.

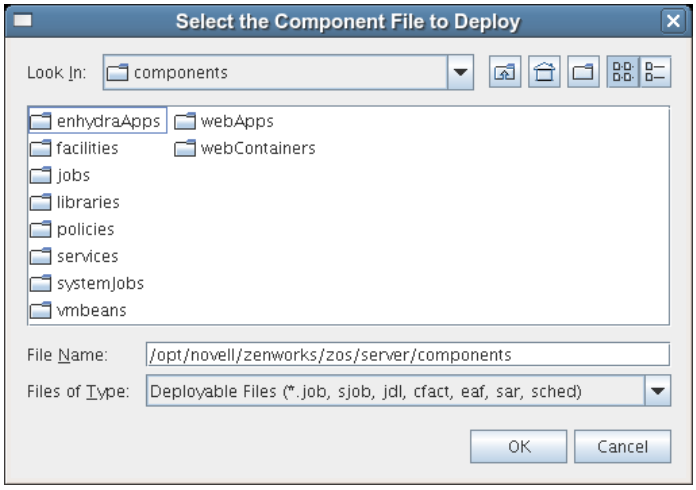
For this walkthrough, you deploy one of several system jobs (`auditCleaner.job`) developed for Cloud Manager Orchestration Server administrators to demonstrate how system jobs are deployed and run. This job package, which is actually a `.jar` archive, includes only a `.policy` component and a `.jdl` component. It does not have a `.sched` component. You can use the Job Scheduler in the Orchestration Console to add the `.sched` component separately.

NOTE: A job developer can create and package jobs that include a `.jdl` file, a `.policy` file, a `.trig` file (trigger), and a `.sched` file (schedule). The presence of the `.sched` file in the job package is also typical of the predeployed discovery jobs installed with the Orchestration Server, which run without intervention when the criteria for firing the schedule are satisfied. Such jobs are visible in the Job Scheduler because they already include the `.sched` components.

For more information about developing jobs and schedules in a job archive, see "Job Scheduling" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

Although this walkthrough demonstrates only the first method listed above for deploying, the other methods are relatively simple, so no further examples are provided to illustrate them.

1 In the Explorer panel of the Orchestration Console, right-click the *Jobs* container, then click *Deploy Job* to open the Select the Component File to Deploy dialog box.

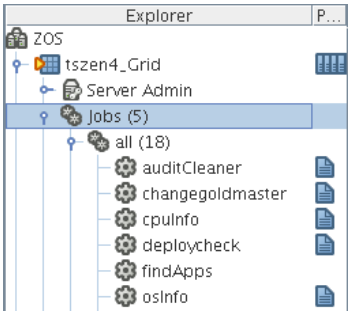


2 Open the *Look In* drop-down list, then navigate to the location of the job you want to deploy.

Although a job developer can store jobs at any location on the network, the sample jobs shipped with the Orchestration Server are limited to the directories where the product is installed. For this walkthrough, navigate to the `/opt/novell/zenworks/zos/server/components/systemJobs` directory.

3 Select *auditCleaner.job*, then click *OK* to deploy the job to the *Jobs* container.

The job appears in the *all* container and in the *examples* container in the tree.



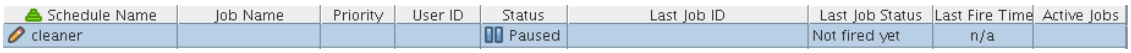
13.2.2 Creating a New Schedule for the Job

When a job has been deployed, you can create a schedule to specify when you want it to run. In this walkthrough, you create a schedule for the `auditCleaner` job by using the Scheduler tool in the Orchestration Console.

1 In the toolbar in the Orchestration Console, click the Job Scheduler button to open the Job Scheduler view.

2 In the Job Scheduler view, click *New* to open the Enter Unique Schedule Name dialog box.

3 Specify a name for the schedule you want to create for this job. For this walkthrough, specify the name `cleaner` in the *Schedule Name* field, then click *OK* to return to the Job Scheduler view.



The new schedule is highlighted in the Job Schedules Table and is flagged with a pencil icon, signifying that the schedule has not been committed yet. Continue with Section 13.2.3, "Defining the New Schedule," on page 173 to define this new schedule by adding the specific information you want.

13.2.3 Defining the New Schedule

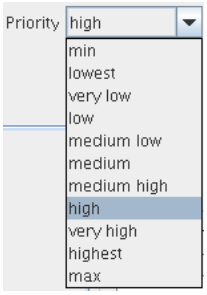
Defining a new job schedule consists of selecting its general properties, its specific properties, and the triggers you want to be associated with it.

Choosing General Properties for a New Schedule

After you have created a new job schedule, its name cannot be changed, but you can add properties to it that help to identify and classify it in a general way. Use the following steps to add these properties:

1. In the Job Schedule Editor panel of the Scheduler view, select the *Job* drop-down list.
2. From the list of available jobs, select *auditCleaner* as the job to which this schedule applies.
3. In the Job Schedule Editor, select the *User* drop-down list.
4. From the list of available users, select *zosSystem* as the user who runs this job.

   The zos user is the built-in user that is always present. It is commonly used for routine jobs like this example.
5. In the Job Schedule Editor, select the *Priority* drop-down list.



6. From the list of available priorities, select *high* as the priority for this job schedule.

   The maximum selectable priority is dependent on an attribute associated with the selected user.
7. Click the *Save* button on the toolbar of the Orchestration Console to save the general properties you have selected for the new schedule.

   The schedule is now committed, and the attribute columns in the Job Schedules Table are populated with the name of the job that the schedule will run, the user it will run as, the priority at which it will run, and its current status. Because the schedule has not been activated yet, it remains in a *Disabled* state.

When you have chosen the general properties of the new schedule, you can either continue with "Adding Specific Parameters to the New Schedule" on page 175 or proceed directly to "Creating and Assigning a Time Trigger for the New Schedule" on page 174.

Creating and Assigning a Time Trigger for the New Schedule

A job already defined in a schedule can be triggered with two main themes: the occurrence of an event or the arrival of a point in time. In this walkthrough, you define a time trigger for the `cleaner` schedule.

In this example, there are no defined time triggers in the Job Scheduler, so you use the following steps to define a time trigger.

1. In the Job Schedule view, click *Edit Triggers* to display the Triggers dialog box.

   Time triggers are shareable across schedules. After a time trigger is defined, it is added to a list of triggers in this dialog box. You can select a predefined trigger from this list when you create a new schedule, or you can create a new time trigger, as the next steps demonstrate.
2. In the Triggers dialog box, click *New* to clear and activate the fields in the dialog box for the creation of a unique time trigger.
3. In the Enter Unique Trigger Name dialog box, specify `24 hour` as the unique name of this time trigger, then click *OK*.
4. In the *Description* field, specify `Runs every 24 hours at noon` as the description for this time trigger.
5. Click *Fire Using CRON Expression* to activate the fields for defining a cron expression.

6. Click the drop-down list of sample cron expressions, then select the default cron expression, `0 0 12 * * ?`, which is listed first.

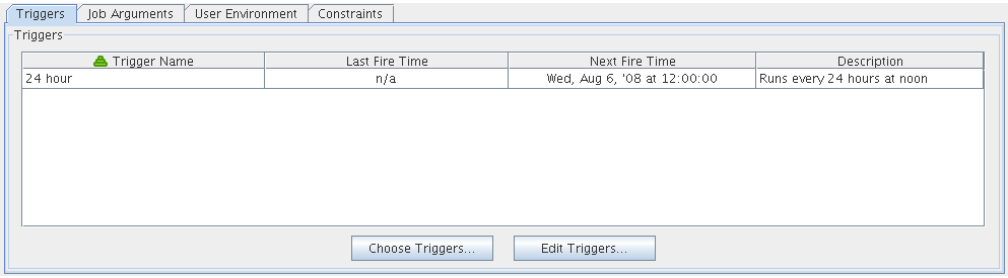
   The sample expressions in the drop-down list show cron strings with accompanying descriptions to remind you how a cron string is constructed. The examples are selectable and editable and can be used in the schedule, just as you have done in this step.

   NOTE: For detailed information about cron syntax, see "Understanding Cron Syntax in the Job Scheduler" on page 167.

   ---
7. Click *Save* to save the trigger you just created, then click *Close* to return to the Job Scheduler view.
8. From the Job Scheduler view, make sure that the *cleaner* schedule is selected, then click *Choose Triggers* to display the Choose Triggers dialog box.
9. In the Choose Triggers dialog box, select *24 hour* (the name of the trigger you created), click *Add* to move the trigger definition to the *Scheduled Job Triggers* list, then click *OK* to return to the Job Scheduler view.

   NOTE: You can select and combine as many time triggers as you want to apply to a given schedule. You can also combine time triggers with event triggers on a given schedule.

   In the *Triggers* list of the Job Scheduler view, the *24 hour* trigger is now associated with the new schedule.



10 Click the *Save* button to update the Orchestration Server with the new schedule/trigger association.

Adding Specific Parameters to the New Schedule

If necessary, you can now add specific parameters to the schedule to edit its job arguments, to choose whether you want to pass the user environment variables to the schedule, or to specify policy constraints to further focus the purpose of this schedule when it fires.

For the purpose of this walkthrough, none of these specific parameters is modified, although a general overview of how to do so is explained.

The following specific parameters can be managed in the Job Scheduler Editor:

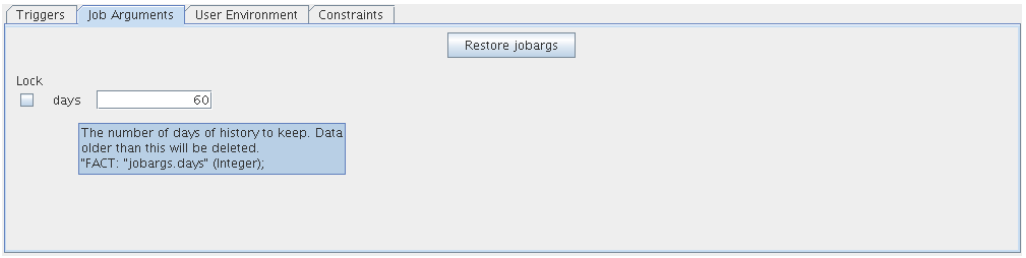
- "Job Arguments" on page 175
- "User Environment" on page 176
- "Constraints" on page 176

Job Arguments

As explained in Section 13.1.2, "Creating or Modifying a Job Schedule," on page 158, a job argument defines the values that can be passed in to the process when a job is invoked. These values let you statically define and control job behavior. The job arguments that appear in the *Job Arguments* tab of the Schedule Editor depend on the job. The job might have no arguments.

By default, the `auditCleaner` job lists only one job argument, *jobargs.days*.

Figure 13-9 *The Job Arguments Tab of the Job Schedule Editor*



According to the tooltip text, this argument is the number of days of job history to keep, so this job cleans up the history of the job in the Orchestration Server audit database after the job reaches the age of 60 days. Data older than 60 days is to be deleted. If you want to, you can change this parameter, or any other parameter in a job argument.

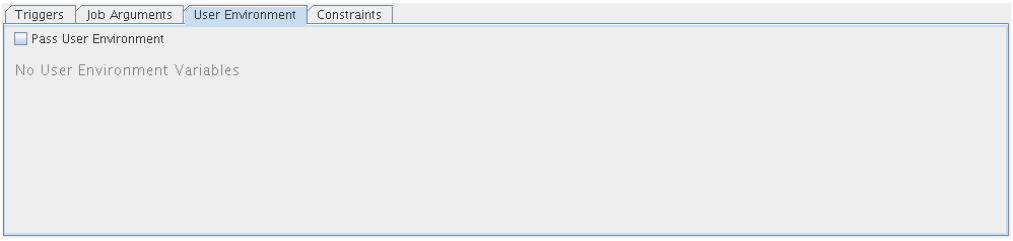
If the default value for a job argument parameter is missing, the job might fail, so you should inspect these parameters carefully.

User Environment

As explained in Section 13.1.2, “Creating or Modifying a Job Schedule,” on page 158, a user’s environment variables are available in the Job Scheduler only if that user utilizes the zos command line tool and elects to pass those environment variables to the server at login time or when he or she runs a job (running the job creates the environment variables as facts in the job). The `zos run` command passes the environment for that particular job run only.

In this walkthrough, the `zosSystem` user shows no user environment variables.

Figure 13-10 *The User Environment Tab of the Job Scheduler Editor, No User Environment Variables Available*

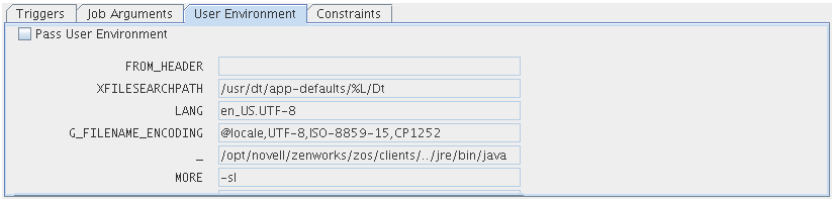


Because there are no environment variables listed, there are none to pass to the schedule, so it is not necessary to select the *Pass User Environment* check box. By default, this check box is not selected, even if environment variables are present for a user specified to run the job.

Sometimes a job is written to work from a user’s environment variables. In this case, if a user has not logged in or has not run the job from the zos command line using the necessary environment option, the schedule must pass those variables to the job when it is invoked.

If you associate a user who has user environment variables with this schedule, you would see a list of those environment variables as they would be passed to the job.

Figure 13-11 *The User Environment Tab of the Job Schedule Editor, User Environment Variables Available*



Selecting the *Pass User Environment* check box in this scenario would create these variables as facts used for this job invocation.

Constraints

As explained in Section 13.1.2, “Creating or Modifying a Job Schedule,” on page 158, the *Constraints* tab displays a constraint editor that you can use to create additional constraints for the job being scheduled.

Any other constraints associated with the context of this job invocation (including but not limited to this job), with the user you’ve selected, with that user’s group, with the jobs group, with the resources that the job uses, or with the resource groups that the job uses, run in spite of the policy that you define here. These additional constraints usually restrict or refine what the job does when this schedule fires.

These constraints are passed to the job only when this schedule is invoked. For example, you could add a start constraint to delay the start of a job, a resource constraint to run on only one of three named machines, or a continue constraint to automatically time out the job if it takes too long to run. Anything you can do with a regular job policy constraint, you can add as a special constraint here for this particular schedule invocation.

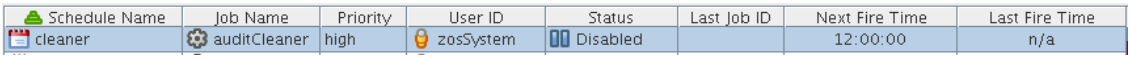
Click the *Save* button to update the Orchestration Server with the new schedule.

For more information about policies, see "Policies" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

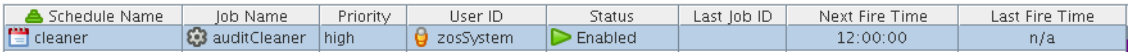
13.2.4 Activating the New Schedule

When the new schedule has been created and its triggers defined, you need to take it from the disabled state to an active state where it is ready to run.

1 In the Job Scheduler view, select the newly created job. The job shows that it is in a *Disabled* state.



2 Click *Enable* to enable the schedule.

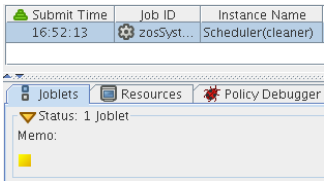


The schedule is now enabled, but has not run yet.

13.2.5 Running the New Schedule Immediately

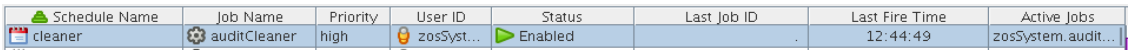
You can trigger the schedule immediately, rather than waiting for the triggers to fire.

1 In the Job Schedules Table of the Job Scheduler view, select *cleaner* (the name of the schedule you want to run), click *Run Now*, then click the job monitor button on the toolbar (*Jobs*) to open the Job Monitor view.



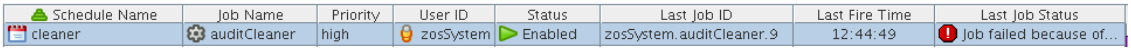
The joblet icon shows that the job is running.

2 Click the Job Scheduler button on the toolbar to open the Job Scheduler view.



The `cleaner` schedule is listed as an active job. This indicates that the schedule has started the job as anticipated.

If you click the *Refresh* button, you can see that the job now has a Job ID.



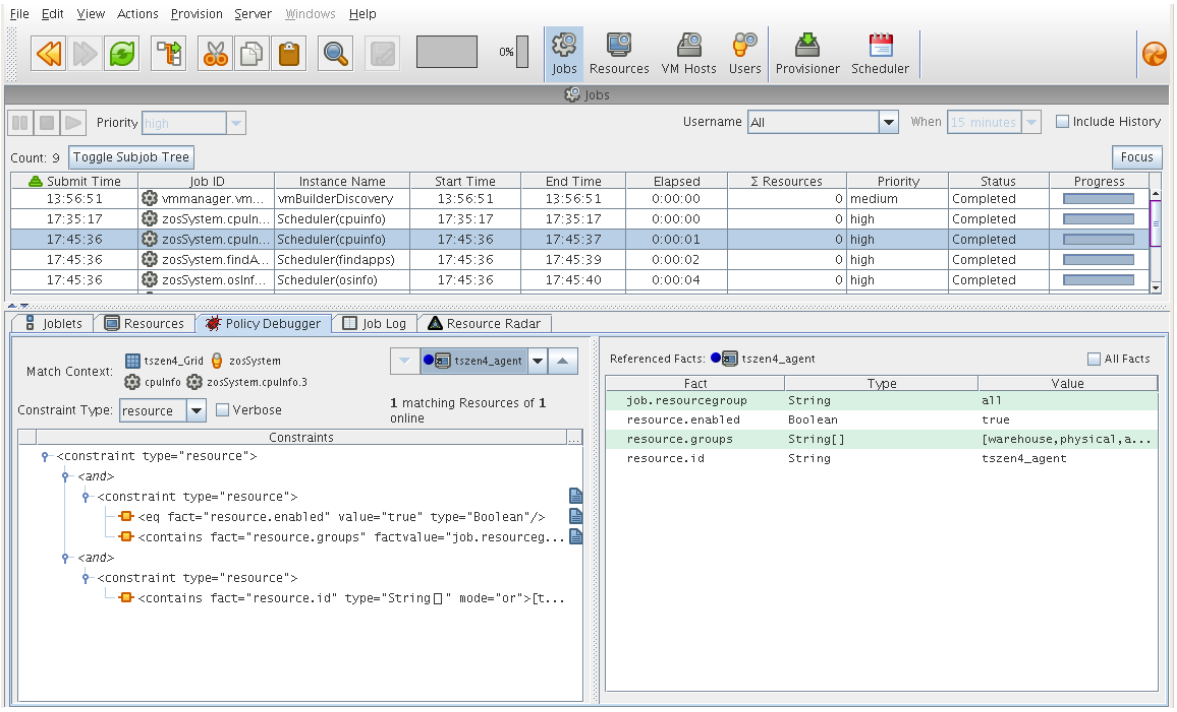
Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Last Fire Time	Last Job Status
cleaner	auditCleaner	high	zosSystem	Enabled	zosSystem.auditCleaner.9	12:44:49	Job failed because of...

If the job invocation fails, as in this example, a red exclamation icon is also displayed.

14 The Policy Debugger

The Policy Debugger is a tabbed page available in several views of the Cloud Manager Orchestration Console. This tool helps you to determine the reasons for the current state of a job. The following figure shows the *Policy Debugger* tab opened in the Jobs view of the Orchestration Console.

Figure 14-1 *The Jobs View of the Orchestration Console with the Policy Debugger Page Open*



The *Policy Debugger* tab is also available in the VM Hosts view and in the Provisioner view of the Orchestration Console.

14.1 Constraints Table View

The left side of the Policy Debugger page is referred to as the Constraints Table view.

Figure 14-2 *The Constraints Table View*

The appearance of this view can change, depending on the constraint type you select in the drop-down menu. For a description of these types, see Section 14.1.2, “Constraint Type List,” on page 181.

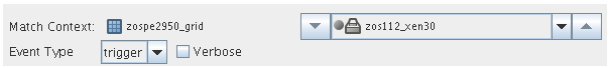
The Constraints Table View is composed of several parts:

- Section 14.1.1, “Match Context Area,” on page 180
- Section 14.1.2, “Constraint Type List,” on page 181
- Section 14.1.3, “Verbose Check Box,” on page 181
- Section 14.1.4, “Capable Resources Summary,” on page 182
- Section 14.1.5, “Constraints Column of the Constraints Table View,” on page 182
- Section 14.1.6, “Policy Column of the Constraints Table,” on page 183

14.1.1 Match Context Area

The policy debugger provides the general identification of a job instance in the *Match Context* area of the Constraints Table view. The Match Context defines everything associated with running a job on a particular resource because it references facts, which are also referenced in policies. The policies define how, when, and where the job runs.

Figure 14-3 *The Match Context Area of the Constraints Table View in the Policy Debugger*



That identifying facts in the Match Context include:

Matrix: The icon and a text string identifies the machine that matches the grid name given to the Orchestration Server where this job is running.

User: The icon and a text string identifies the User object that matches the user who is running the job.

Job: The icon and a text string identifies the deployed job that matches the one that is running on the grid.

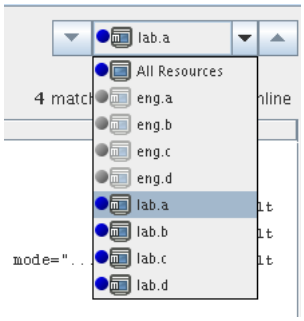
Job Instance: The icon and a fully qualified text string identifies the specific job instance that matches the deployed job running on the grid.

Resource: The Resource drop-down list shows all resources. The list appears in the Match Context if the *resource* constraint type is selected. The resources in the list that are currently offline display with a dimmed icon. If available, a listed resource has a colored dot by its side. The color of the dot (blue or gray) and the resource type it accompanies has significance:

- A blue dot with the *All Resources* label indicates that at least one resource matches the constraints and is capable of servicing the job.
- A gray dot with the *All Resources* label indicates that no resources match the constraints.
- A blue dot with a named, selected resource indicates that its constraints match and it is capable of servicing the job.
- A gray dot by a named, selected resource indicates that its constraints do not match and that it is not capable of servicing the job.

The following figure shows a list of eight resources. Four of those resources (*lab.a*, *lab.b*, *lab.c* and *lab.d*) are online. Their constraints match, so they are capable of servicing the job.

Figure 14-4 *Resource Drop Down List Showing Online and Offline Resources*



14.1.2 Constraint Type List

Select one of the constraint types in the drop down list to specify a policy context. Constraint types in the list are disabled (dimmed) if they do not apply to the job that you are debugging.

- accept
- start
- continue
- provision
- allocation
- resource
- vmhost
- repository
- health

14.1.3 Verbose Check Box

When you select the Verbose check box, the `reason` string specified in the policy is displayed in the *Constraints* tree. For more information, see Section 14.1.5, “Constraints Column of the Constraints Table View,” on page 182.

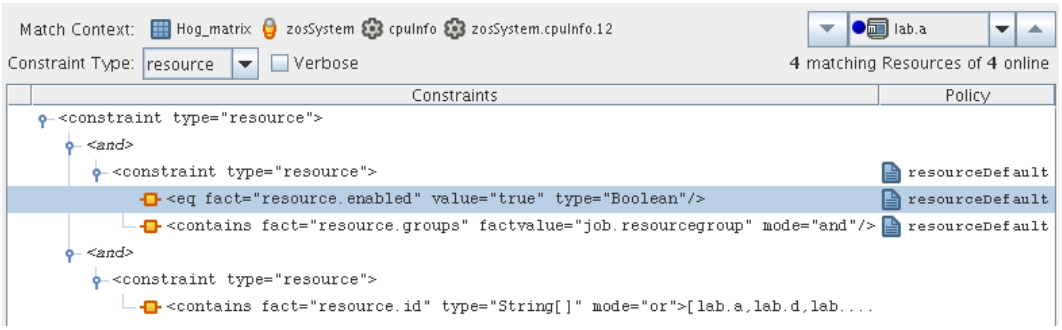
14.1.4 Capable Resources Summary

Directly under the Resource List in the constraint view, a populated string summarizes the resources that are capable of servicing the job. For example, `4 matching Resource of 10 online` indicates that four of the ten total online resources are capable of servicing the job.

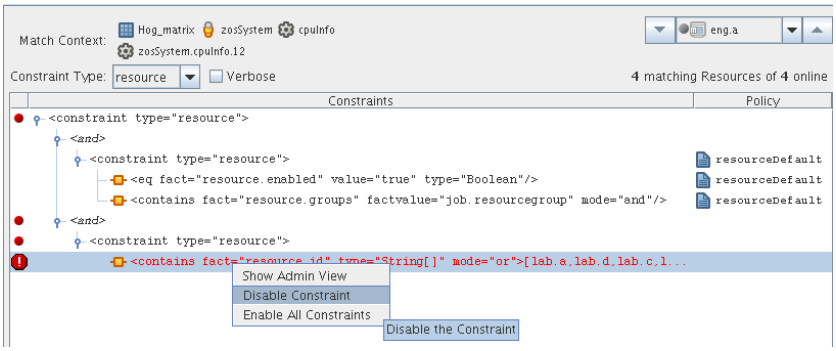
14.1.5 Constraints Column of the Constraints Table View

The Constraints column of the constraints table view shows the logical hierarchy (that is, a "tree" format) of the constraints that are defined by the policies associated with the job. You can identify the status of the listed constraints by the icons that might be displayed in the far left column of the table:

No icon: The constraint passes the match. It is a "true" match. The figure below shows that the resource `lab.a` is available to run the job because all of its constraints match. No red icons are displayed next to any listed constraint.

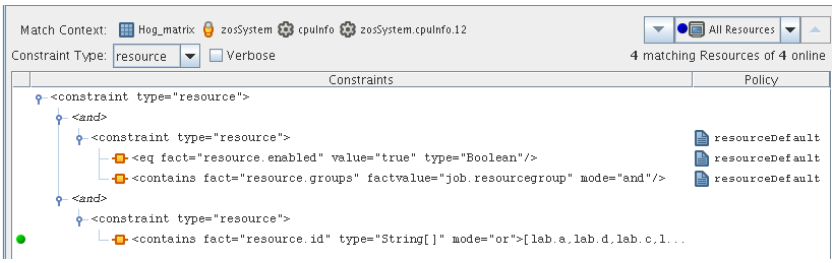


Red dot icon: The constraint does not pass the match. The figure below shows that the resource `eng.a` cannot run the job because its constraints do not match.



Red octagonal icon: The constraint does not pass the match and is blocking the job. The figure above also shows the blocking constraint (red octagon).

Green dot icon: A blocking constraint has been disabled so that it behaves like a match. The figure below shows the green dot icon next to that the constraint that was formerly blocked and can now behave as a match.



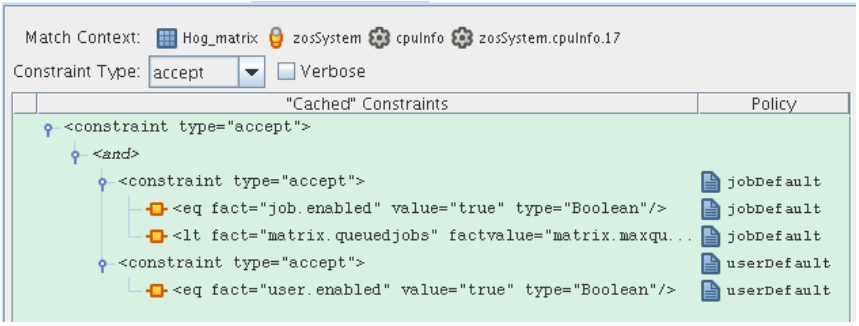
If you right-click a constraint in the table, a popup menu with three options is displayed. This menu lets you change the status of the constraint.

- **Show Admin View:** Select this option to open the Admin View for the specific resource selected.
- **Disable Constraint:** Select this option to disable (attach a green dot icon to) a constraint. Disabling a constraint with this function effectively makes it match, which is a condition that can be useful if you want to perform a "what if" test without actually changing a policy.
- **Enable All Constraints:** Select this option if you have disabled one or more constraints during testing and you want to restore them to the enabled state.

Cached Constraints in the Constraints Column

When you change the constraint type in the Constraints Type List, the background of the table changes to green for some types. These are "cached" constraints that are saved with the job after it is complete. Their purpose is to help you debug the policy.

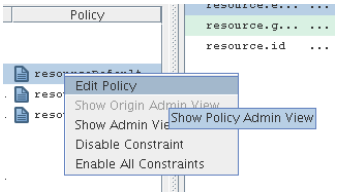
Figure 14-5 *Cached Constraints Displayed in the Constraints Table View*



14.1.6 Policy Column of the Constraints Table

The Policy column of the Constraints Table displays the policy name that contributed the constraint. Right-click a policy name to open a popup menu offering the option to open the policy editor for the specified policy. The menu also includes constraint enabling or disabling options, just as the pop-up menu for constraint column does.

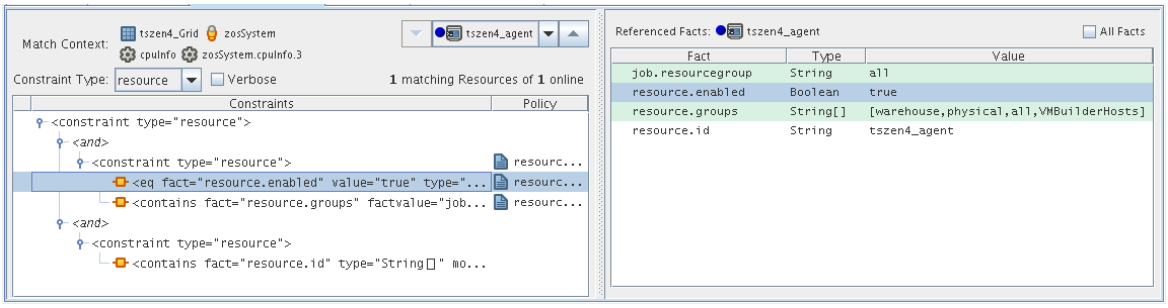
Figure 14-6 *The Pop-up Menu Launched from the Policy Column*



14.2 Facts Table View

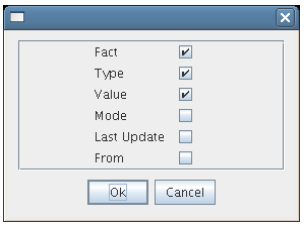
The Facts Table view displays the facts referenced in the Constraint Tree view for a specified Resource. Selecting a fact in the Constraint Tree automatically selects that fact in the table.

Figure 14-7 *The Constraints Table View and the Accompanying Facts Table View*



If you right-click a column head in this table, a menu is launched where you can select the columns that you want to display.

Figure 14-8 *Menu Used to Select the Columns Displayed in the Facts Table View of the Policy Debugger*

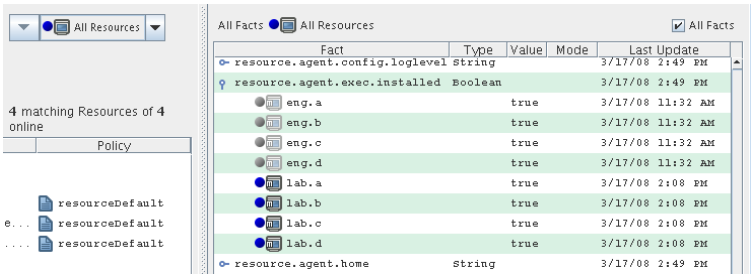


14.2.1 All Facts Check Box

If you select the *All Facts* check box at the top of the Facts Table view, all of the facts (including matrix, user, job, jobargs, jobinstance, and resource facts) associated with the Match Context are listed.

If you select *All Resources* in the Match Context (see Section 14.1.1, “Match Context Area,” on page 180) and you also select the *All Facts* check box, the Facts Table view displays all the facts for all resources for the specified Match Context.

Figure 14-9 *All Facts Check Box Selected with All Resources in Match Context*

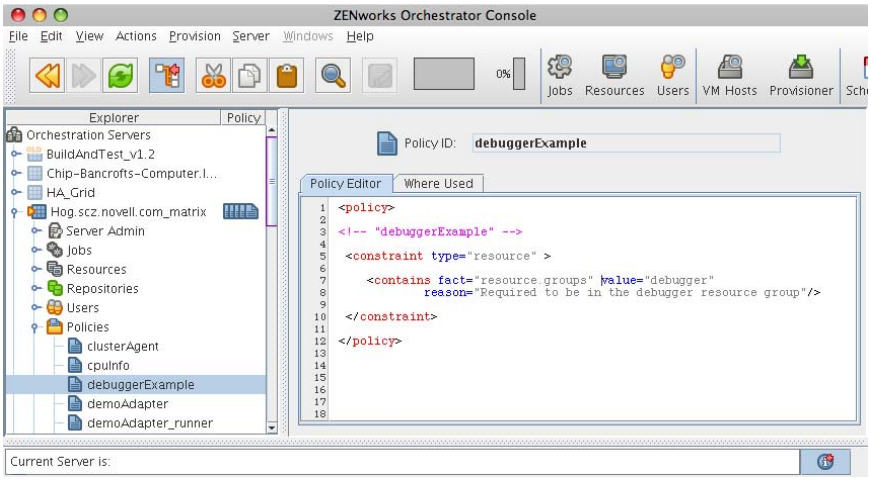


14.3 Policy Debugger Use Cases

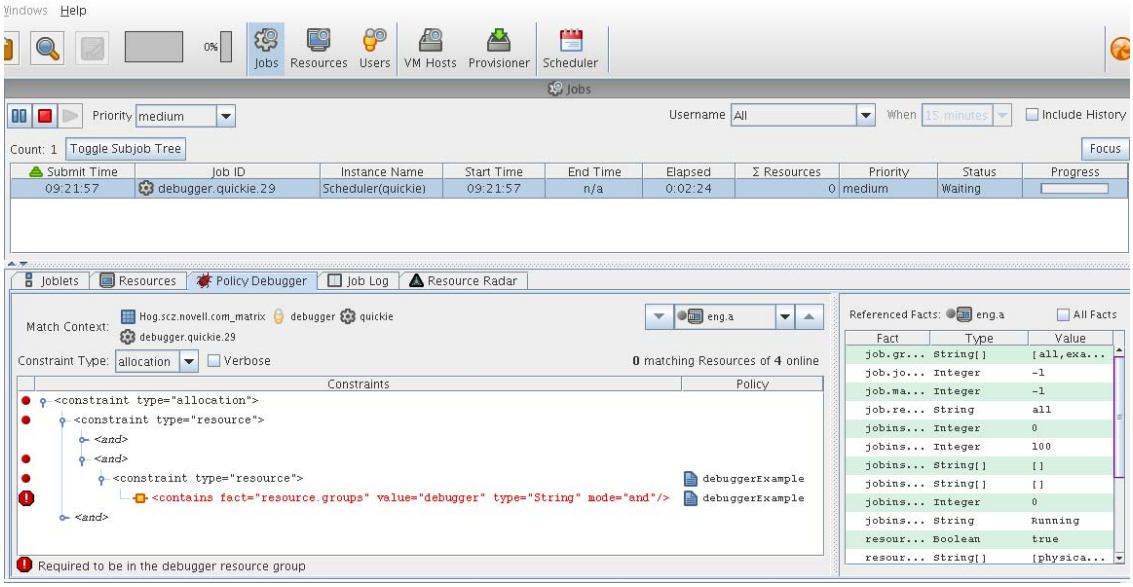
The objective of this use case is to run a job that remmians in the waiting state and to use the policy debugger to identify why it is in the state and to make the necessary changes to get the job to run. The `quickie.job` is used along with a simple policy that specifies that the resources that are to be used must be in a Resource group called `debugger`.

Use the following steps to re-create the use case.

1. In the Orchestration Console, create a user named `debugger`.
2. Deploy `quickie.job` from the `/examples` directory.
3. In the Orchestration Console, create a schedule named `quickie`, specifying the `quickie` job and the `debugger` user.
4. In the Orchestration Console, create a policy and name it `debuggerExample`. The policy needs to specify that the resource used belongs to the group called `debugger`.



5. In the Orchestration Console, associate the `debuggerExample` policy to the `quickie` job.
6. In the Job Scheduler view of the Orchestration Console, select the `quickie` schedule, then click *Run Now* to run the `quickie` schedule.
7. In the Job Monitor view of the Orchestration Console, select the *Policy Debugger* tab and verify that the job is in the waiting state.
8. In the Constraints Table view, open the *Constraint Type* drop-down list, then select *Allocation*.
9. In the Match Context area of the Constraints Table view, open the *Resource* drop-down list, then select any resource to refresh the Constraints Table and Facts Table views.



The Policy Debugger displays a red icon near the constraints that fail to match. The larger, red octagonal icon shows the particular constraint that is "blocking" and preventing the job from running on the resource. This is the constraint that is causing the job to be in a waiting state. The Constraints Table also displays the policy name (`debuggerExample`) that is contributing the constraint that is causing problems.

There are several ways to get the job to run:

- Create a Resource group called `debugger`, then place a resource in that group to satisfy the constraint specified in the policy.
- Disassociate the policy (`debuggerExample`) from the job (`quickie`).
- In the Constraints Table, right-click the blocking constraint and select *Disable Constraint*.

A Grid Object Health Monitoring

This section includes the following information:

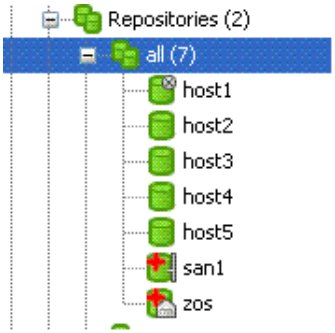
A.1 Health Facts

The Resource grid object, the VM host grid object, the User grid object, and the Repository grid object each have an attribute or Fact that denotes the health of the object.

- `resource.health`
- `vmhost.health`
- `user.health`
- `repository.health`

Empirically, object health is a simple Boolean value, with `True` indicating that the object is healthy. This value can be controlled in a number of ways. An unhealthy object is displayed in the Cloud Manager Orchestration Console with a red “plus” symbol to signal the object’s condition.

Figure A-1 *Tree View of Repository Grid Objects in the “all” Group, Some Objects Unhealthy*



You can define what constitutes the health or non-health of the grid object by setting this health fact. The health fact can be set or cleared in several ways:

A.1.1 Explicitly Set or Cleared by the Administrator

The administrator can use tools in the Orchestration Console to explicitly set or clear the fact.

1. Select any grid object in the Orchestration Console, then click the *Info/Groups* tab in the Workspace view. This is the "info" attribute editor. The attributes on this page let you edit facts.

   The object information panel of the page has a *Healthy* check box that you can select or deselect to set the health of the object.

2. On the Constraint/Fact fact page of a grid object, right click the `xxx.health` fact name, then click *Edit/View Fact* to open the Edit Fact dialog box.

   You can set the health of the object by selecting or deselecting the health check box. Changing the value in the Orchestration Console in this way has an immediate effect unless the value is overridden by an attached policy (this follows the normal rules of policy inheritance).

A.1.2 Set by Using a Discovery Job

A discovery job is a job that is periodically scheduled to run on resources and to explicitly set the health fact, much like it sets other discovered facts). In this case, the discovery job performs a setFact (`xxx.health`) from JDL code.

A.1.3 Set by Using a Policy

This method has little practical use except for locking the value immediately to override the setting (that is, the typical policy behavior) on the grid object:

```
<policy>
  <fact name="resource.health" value="true" type="boolean" />
</policy>
```

A.1.4 Set by Using a Computed Fact

Set by using a computed fact. This method can be used to monitor the health according to a computed value. One applied scenario for this method might be a computed fact that performs a statistical analysis of historical load data, perhaps provided by the Metrics facility.

A.1.5 Set Automatically by Using a Health Constraint

Set automatically by using a health constraint. This is the most practical use and is best illustrated with examples.

Example 1: Define resources as "unhealthy" if their 10-minute load average is greater than 5.

```
<policy>
   <constraint type="health">
       <lt fact="resource.metrics.loadaverage.history.10_min" value="5.0" />
   </constraint>
</policy>
```

You could attach this policy directly to the Resource grid object or to a Resource group. Attaching it to a Resource group is more practical.

Example 2: Define a user as unhealthy if he or she has no money in an account.

```
<policy>
   <constraint type="health">
       <ge fact="user.account.balance" value="0" />
   </constraint>
</policy>
```

You could attach this policy directly to the User grid object, or to a User group. Attaching it to a User group is more practical.

You can aggregate (that is, group together with "and" or "or") health constraints by using normal rules of policy aggregation.

By default, the Orchestration Server runs health constraints every 30 seconds. To alter this interval, you must contact NetIQ Support.

A.2 Health Events

Each time the value of a health fact changes, an event is generated. This event can be subscribed to by long-running Jobs (see "Receiving Event Notifications in a Running Job" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*) or the event can be used to trigger Jobs in the Job Scheduler (see Chapter 13, "The Orchestration Server Job Scheduler," on page 155). The event names are different for each object type. They are listed in the following table.

Table A-1 *Event Names for Grid Objects*

Object	Event Name
User	`USER_HEALTH`
Resource	`RESOURCE_HEALTH`
Repository	`REPOSITORY_HEALTH`
VM host	`VMHOST_HEALTH`

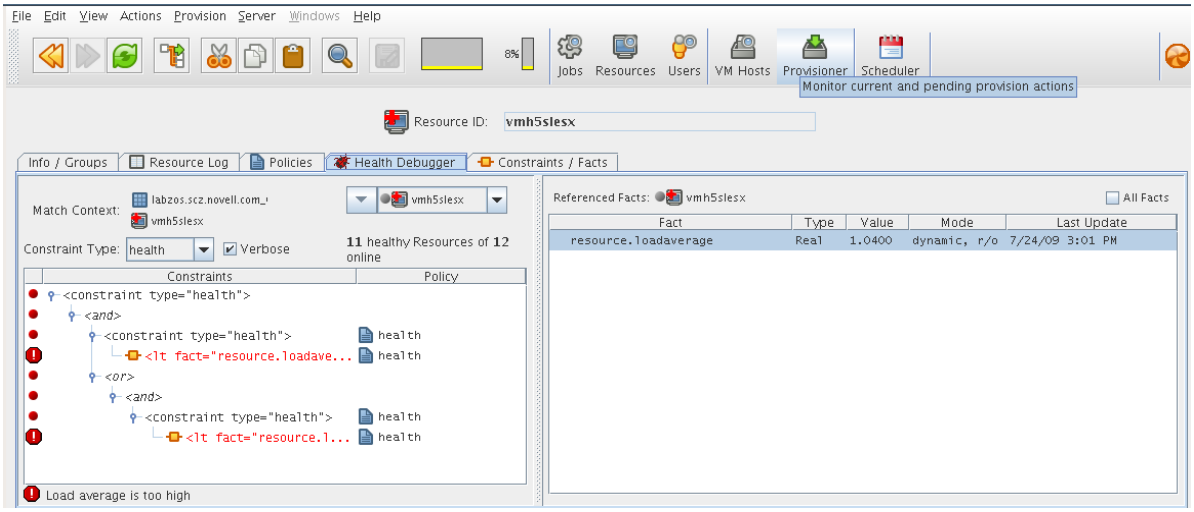
A.3 Health Debugger

Several objects modeled by the Orchestration Console have a health fact that can be used to visually show the health of the object or to trigger a job (see Chapter 13, "The Orchestration Server Job Scheduler," on page 155) upon a state change. This fact can optionally be manually set, or more

usually automatically set through the periodic execution of the health constraint placed on that object. When such a health constraint is active, you might need to debug to discover the reasons for the changed state. The Health Debugger is a useful debugging tool.

The Health Debugger is a tabbed page (sometimes called an “admin view”) available in several views of the Orchestration Console and functions much like the more generic “Policy Debugger.” This tool helps you to determine the reasons for the current Health of a Grid object. The following figure shows the Health Debugger tab opened in the Resource object view of the Orchestration Console.

Figure A-2 *Orchestration Console Jobs View with the Health Debugger Page Open*



The Health Debugger tab is also available in the VM Host object view, the Repository object view and in the User object view of the Orchestration Console.

- Section A.3.1, “Constraints Table Panel,” on page 190
- Section A.3.2, “Facts Table View,” on page 193

A.3.1 Constraints Table Panel

The left side of the Health Debugger page is referred to as the Constraints Table view.

Figure A-3 *The Constraints Table View*

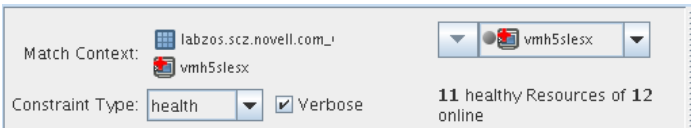
The appearance of this view can change, depending on the Constraint Type you select in the Constraint drop down list. In effect, if you change from the health selection, you will be debugging other constraints. For a description of these constraint types, see Section 14.1.2, "Constraint Type List," on page 181. Different objects selected for the view change the *Match Context* in which the constraint is executed, which is useful for comparing how the constraint evaluates other objects.

The Constraints Table view is composed of several parts:

Match Context Area

The Health Debugger provides the general identification of a Grid object in the *Match Context* area of the Constraints Table view. The Match Context defines every object that is available to the constraint you are viewing.

Figure A-4 *The Match Context Area of the Constraints Table View in the Health Debugger*



In this example, the identifying Facts in the Match Context include:

Matrix: The icon and a text string identifies the machine that matches the grid name given to the Orchestration Server where this object exists.

Object icon: The object icon and a text string identifies the object (VM host, Repository, or that matches the user who is running the job. If the object icon has a red cross overlaid, it is unhealthy.

Object list: The object drop down list shows all named objects of the type selected in the Explorer Tree. The objects in the list that are currently offline display with a dimmed icon. If available, a listed object has a colored dot by its side. The color of the dot (blue or gray) and the object type it accompanies has significance:

- A blue dot with the *All <Object Type>* label indicates that at least one object in the list matches the constraints and is healthy.
- A gray dot with the *All <Object Type>* label indicates that no objects of this type match the constraints.
- A blue dot with a named, selected object indicates that its constraints match and it is healthy.
- A gray dot by a named, selected object indicates that its constraints do not match and that it is not healthy.

Verbose Check Box

When you select the Verbose check box, the `reason` string specified in the policy is displayed in the *Constraints* tree. For more information, see Section 14.1.5, "Constraints Column of the Constraints Table View," on page 182.

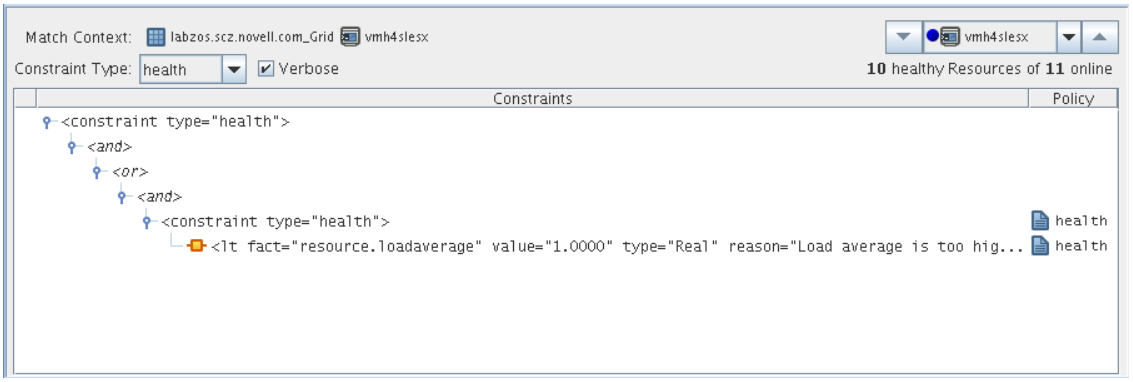
Capable Objects Summary

Directly under the Object list in the constraint view, a populated string summarizes the resources that are capable of servicing the job. For example, `11 healthy Resources of 12 online` indicates that 11 of the 12 total online Resources are healthy.

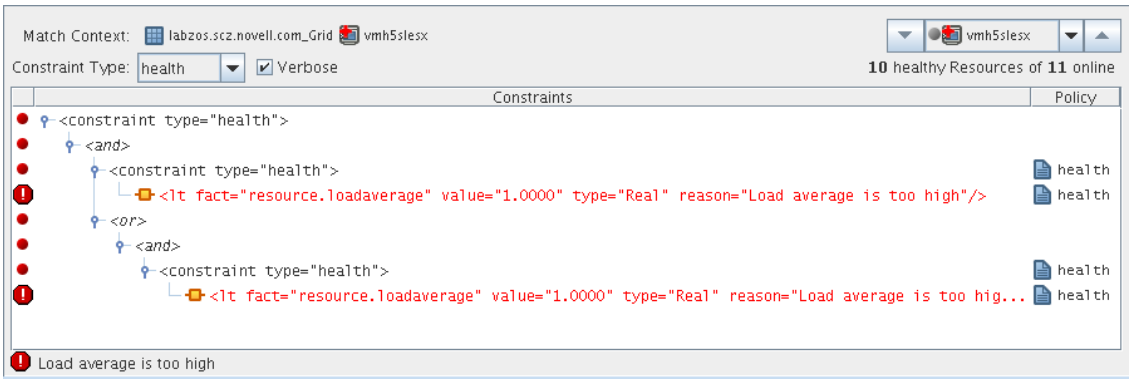
Constraints Column of the Constraints Table View

The Constraints column of the constraints table view shows the logical hierarchy (that is, a "tree" format) of the constraints that are defined by the Policies associated with the Job. You can identify the status of the listed constraints by the icons that might be displayed in the far left column of the table:

No icon: no icon: The constraint passes the match. It is a "true" match. No red icons are displayed next to any listed constraint.

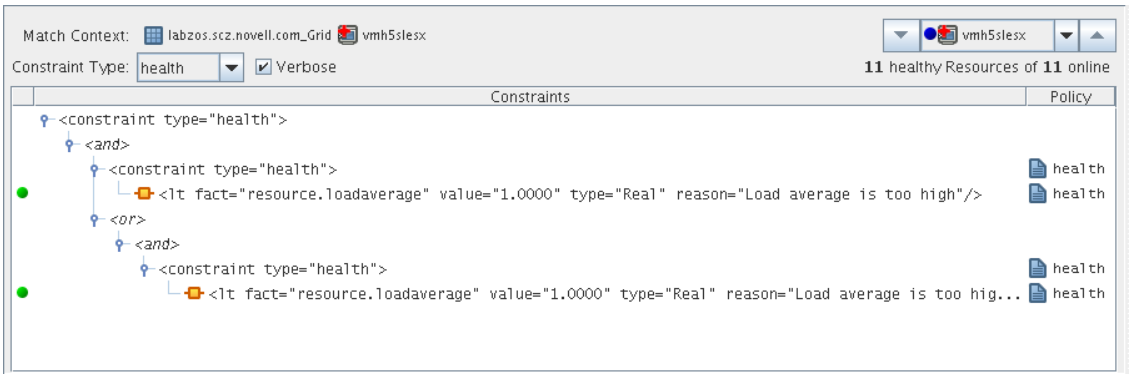


Red dot icon: The constraint does not pass the match. The figure below shows that the resource `vmh5slesx` cannot run the job because its health constraints do not match.



Red octagonal icon: For convenience, only one of the constraints is identified as the blocking constraint. This is the constraint that the system has determined is responsible for the constraint as a whole to fail (note that individual constraint lines can fail without causing the entire constraint to fail).

Green dot icon: green dot icon: A failing constraint has been disabled so that it behaves like a match (pass). The figure below shows the green dot icon next to that the constraint that was formerly failing and can now be forced to behave as a match.



If you right-click a constraint in the table, a popup menu with three options is displayed. This menu lets you change the status of the constraint. Disabling a constraint is useful if you want to temporarily relax a condition without editing or redeploying the entire policy and potentially affecting other objects that share the policy. A disabled constraint can be re-enabled later.

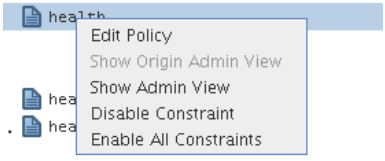
The constraint status change options include the following:

- **Show Admin View:** Select this option to open the Admin view for the specific object selected.
- **Disable Constraint:** Select this option to disable (attach a green dot icon to) a constraint. Disabling a constraint with this function effectively makes it match, a condition that can prove useful if you want to perform a "what if" test without actually changing a policy. Green
- **Enable All Constraints:** Select this option if you have disabled one or more constraints during testing and you want to restore them to the enabled state.

NOTE: Health constraints are always re-evaluated in the debugger. The last system execution (cached constraint) is not available for health constraints.

The Policy column of the constraints table displays the policy name that contributed the constraint. Right-click a policy name to open a popup menu offering the option to open the policy editor for the specified policy. The menu also includes constraint enabling or disabling options, just as the pop-up menu for constraint column does.

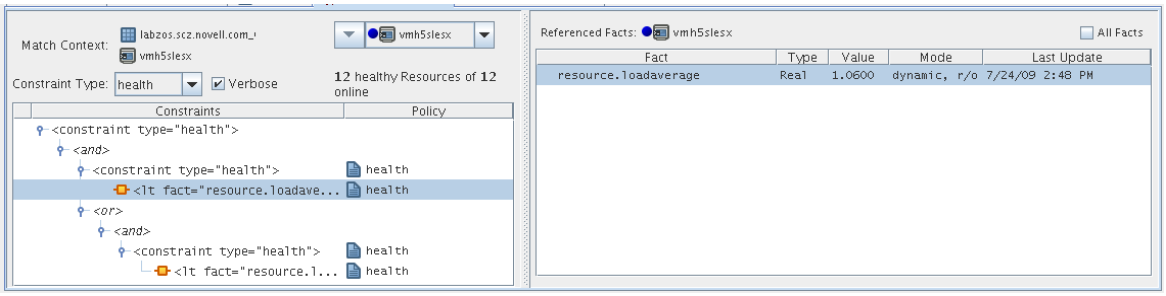
Figure A-5 *The Pop-up Menu Launched from the Policy Column*



A.3.2 Facts Table View

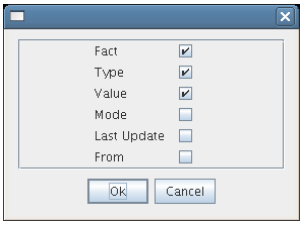
The Facts Table view displays the facts referenced in the Constraint Tree view for a specified object. Selecting a fact in the Constraint tree automatically selects that fact in the table.

Figure A-6 *The Constraints Table View and the Accompanying Facts Table View*



If you right-click a column head in this table, a menu is launched where you can select the columns that you want to display.

Figure A-7 *Menu Used to Select the Columns Displayed in the Facts Table View of the Policy Debugger*

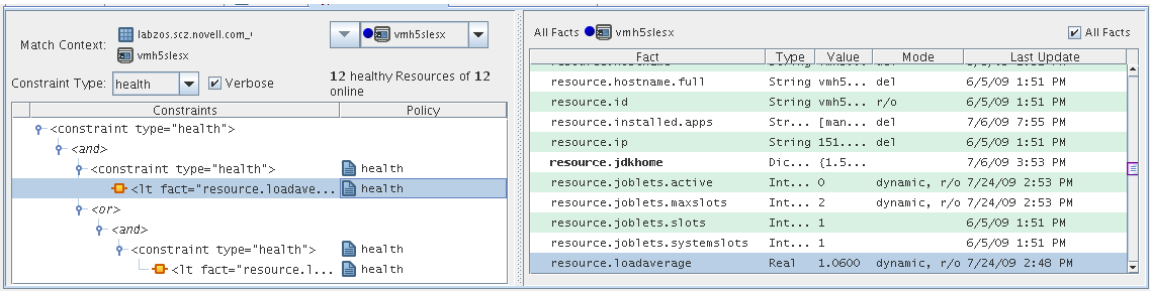


All Facts Check Box

If you select the *All Facts* check box at the top of the Facts Table view, all of the facts (including matrix, and <Object type> facts) associated with the Match Context are listed.

If you select *All <Object Type>* in the Match Context (see Section 14.1.1, "Match Context Area," on page 180) and you also select the *All Facts* check box, the Facts Table view displays all the <object type> facts for the specified Match Context of the selected object.

Figure A-8 *All Facts Check Box Selected with All VM Hosts in Match Context*



B Events

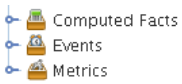
This section contains the following information:

- Section B.1, “Event Object Visualization and Management in the Orchestration Console,” on page 195
- Section B.2, “Event Debugger,” on page 197
- Section B.3, “Understanding the Orchestration Server Events System,” on page 200

B.1 Event Object Visualization and Management in the Orchestration Console

The Events folder is displayed in the Explorer tree between the Computed Facts and Metrics folders.

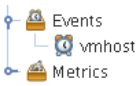
Figure B-1 *The Events Folder in the Orchestration Console Explorer View*



Although the Cloud Manager Orchestration Server includes several built-in Events (see Section B.3.2, “Built-in Events,” on page 201), these Events are not displayed in the Explorer view. Only custom Events (defined in XML by the administrator and then deployed on the server) are displayed in the tree.

When an Event object is deployed, its icon is displayed in the tree in the Events folder.

Figure B-2 *An Event Object in the Events Folder*



The icon in the Explorer tree might be overlaid with a write symbol  to indicate that its XML content has changed and needs to be saved. For more information about changing the XML content, see Section B.1.4, “Event Editor,” on page 196.

This section includes the following information about how the Event object is managed in the Orchestration Console:

- Section B.1.1, “Deploying a New Rule-Based Event,” on page 196
- Section B.1.2, “Deploying a Pre-written Rule-Based Event,” on page 196
- Section B.1.3, “Undeploying an Event,” on page 196
- Section B.1.4, “Event Editor,” on page 196

B.1.1 Deploying a New Rule-Based Event

Use the following steps to create a new event.

1. In the Explorer view, right-click the Events folder, then select *New Event* to open the Create a New Event dialog box.
2. Specify the name for the new Event, then click *OK* to create the new Event object.

   The Orchestration Server then deploys the new Event object on the server, where it can be managed. The Orchestration Console opens the Event Editor, where you can edit the XML definition of this Event. For more information, see Section B.1.4, "Event Editor," on page 196 and Section B.3.3, "Rule-based Events," on page 202.

B.1.2 Deploying a Pre-written Rule-Based Event

Use the following steps to deploy a pre-written Event (an XML `.event` file).

1. Right-click the Events container, then select *Deploy Event* to open the Select the Component File to Deploy dialog box.
2. In the dialog box, navigate to the file system location of the Event file you previously created, or to an example `.event` file from `/opt/novell/zenworks/zos/server/examples/events`, then click *OK* to deploy the pre-written Event.

   When you deploy the rule-based Event, the Orchestration Console opens the Event Editor, where you can edit the XML definition of this Event. For more information, see Section B.1.4, "Event Editor," on page 196 and Section B.3.3, "Rule-based Events," on page 202.

B.1.3 Undeploying an Event

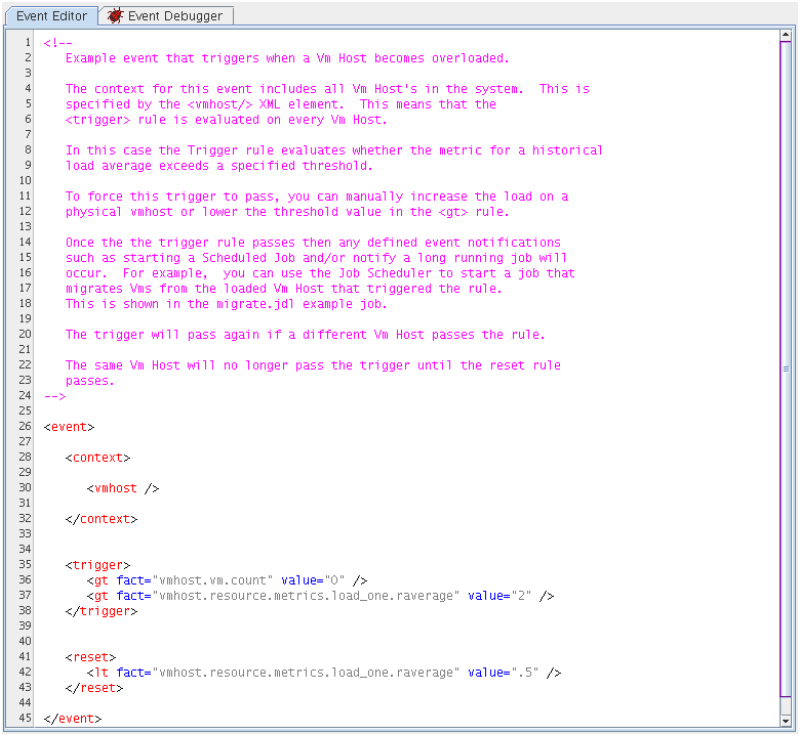
When an Event has been deployed, it can be undeployed. Undeploying deletes the Event object within the server, but it does not delete the source `.event` file, which still exists and can be redeployed.

To undeploy an Event, right-click the Event object in the Explorer tree, then select *Undeploy*. You can also simply select the object and press Delete.

B.1.4 Event Editor

The Event Editor opens when you select a deployed Event in the Explorer tree of the Orchestration Console.

Figure B-3 *The Event Editor*



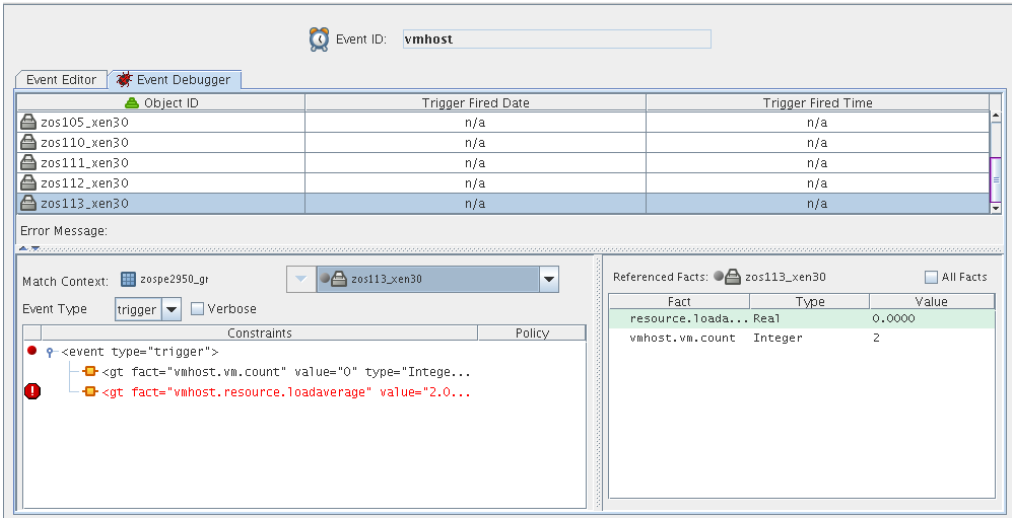
Inside this editor, you can make changes to the XML content of the Event. Example Events contain comments that explain how you can use them and the behavior you can expect to see as a result of deploying them. For the changes you make to be effective, you need to click the Save tool.

For more information about the allowed XML syntax within an Event, see Section B.3.3, “Rule-based Events,” on page 202.

B.2 Event Debugger

The Event Debugger is a tabbed page available from the Explorer view of an Event object when you select an event and then click the Event Debugger tab. This tool helps you to determine the reasons for the current state (`triggered` or `reset`) of an Event. The following figure shows the Event Debugger view:

Figure B-4 *The Event Debugger*



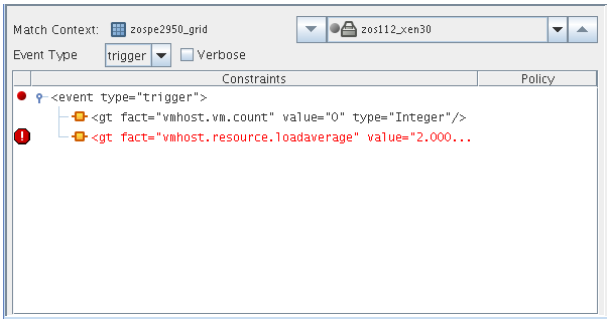
The information in this section describes the various parts of the debugger.

- Section B.2.1, “Constraints Table,” on page 198
- Section B.2.2, “The Facts Table,” on page 199

B.2.1 Constraints Table

The left side of the Event Debugger panel includes the Constraints Table. The rules that define the `<trigger>` and `<reset>` of an Event are defined using the same XML constraint syntax used in policies.

Figure B-5 *The Constraints Table Area of the Events Debugger*



The Constraints Table has several parts:

- “Match Context” on page 199
- “Event Type List” on page 199
- “Verbose Check Box” on page 199
- “Constraints List (Tree)” on page 199

Match Context

Depending on the Event, the debugger identifies the Grid objects (`Job`, `Jobinstance`, `Resource`, `Repository`, `User`, or `VMHost`) that define the context of the trigger or reset rules specified in the Event XML.

Figure B-6 *The Match Context Area of the Constraints Table View in the Event Debugger*

Event Type List

Select one of the Event types in the drop-down list to debug how either the `<trigger>` or `<reset>` rules are being applied. Constraint types in the list are disabled (dimmed) if they do not apply to the Event that you are debugging.

- **trigger:** The rules defining the conditions (through a constraint) in which an Event is generated.
- **reset:** The rules defining the conditions (through a constraint) in which an Event is reset (that is, able to be triggered again).

Verbose Check Box

When you select the Verbose check box, additional constraint information is displayed.

Constraints List (Tree)

The Constraints tree, whichh is a column in the constraints table, lists the constraints that constitute a particular rule in a hierarchical view.

Each constraint is flagged with an icon to signify whether it "passes" or not. A constraint flagged with an exclamation point indicates a constraint causing the rule to not "pass."

Right-click a constraint to display a menu where you can perform one of the following actions:

Show Admin View: Selects the currently evaluated Grid object in the Explorer Tree and displays its Info/Groups administration information view.

Disable Constraint: Passes the constraint, regardless of how it evaluates.

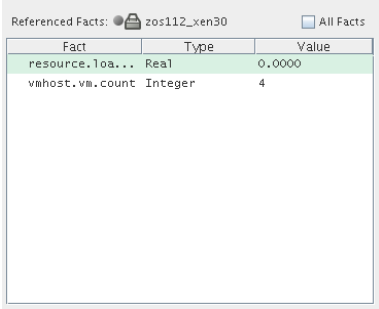
Enable All Constraints: Re-enables any disabled constraints.

NOTE: The right-click menu is available only when you select specific constraints.

B.2.2 The Facts Table

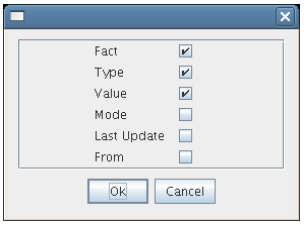
The Facts Table view displays the facts referenced in the Constraint Tree view for a specified Event. Selecting a rule containing a particular fact in the Constraint tree automatically selects that fact and its current value in the table.

Figure B-7 *The Facts Table View*



If you right-click a column head in this table, a menu is launched where you can select the columns that you want to display.

Figure B-8 *Menu Used to Select the Columns Displayed in the Facts Table View of the Policy Debugger*



All Facts Check Box: If you select the *All Facts* check box at the top of the Facts Table view, all facts for the selected Grid object, as well as those for the Server itself (`matrix.* facts`) are displayed.

If you right-click a fact, you have the option of adding a new fact, deleting the selected fact, or viewing/editing the selected fact, if the fact is editable or can be deleted.

B.3 Understanding the Orchestration Server Events System

The the Orchestration Server Event System integrates with the Job Scheduler. Event notifications can start jobs and can also invoke Event handler methods in long-running jobs. In turn, a job can react to the Event by starting other server actions, by modifying object attributes, or by executing another external process.

For example, an Event notification can occur when a VM Host has exceeded its configured load limits. This Event can start a job that migrates VMs off of the loaded VM Host or VM Hosts.

The Orchestration Server supports two Event types:

- Built-in Events, such as change of status of the health of a resource's or a change in the online status of the resource.
- Rule-based Events that are triggered when the attributes of an object satisfy the rules (constraint conditions) defining the Event.

This section includes the following information:

- Section B.3.1, “Event Notification,” on page 201
- Section B.3.2, “Built-in Events,” on page 201
- Section B.3.3, “Rule-based Events,” on page 202

B.3.1 Event Notification

An Event notifies two other Cloud Manager Orchestration services, the Job Scheduler and the Job Broker. The Job Scheduler starts jobs that are awaiting an Event to trigger them. The Job Broker invokes a callback on any long-running job that has registered for notification of an Event.

See Chapter 13, "The Orchestration Server Job Scheduler," on page 155 for more information about setting up a Job Schedule. Also see "Job Architecture" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference* for more information about how jobs can register for event notification.

B.3.2 Built-in Events

Built-in events occur when a managed object comes online/offline or has a health status change.

The Orchestration Server uses the following built-in Events to keep managed objects synchronized.

Table B-1 *The Orchestration Server Built-in Events*

Event Name	Description
AGENT_VERSION_MISMATCH	Resource Agent version mismatch (agent needs upgrade)
REPOSITORY_HEALTH	Repository health status has changed
RESOURCE_HEALTH	Resource health status has changed
RESOURCE OFFLINE	Resource Agent has logged out of the server
RESOURCE_ONLINE	Resource Agent has logged in to the server
SERVER_UP	Server has fully started
USER_HEALTH	User health status has changed
USER_ONLINE	User has logged in to the server
VMHOST_ADDED	VM Host has been added
VMHOST_HEALTH	VM Host health status has changed
VMHOST_NOT_AVAILABLE	No VM Host is available
VMHOST_REMOVED	VM Host has been removed

For example, when a resource comes online (that is, the agent connects to the server), the `RESOURCE_ONLINE` Event is fired and both scheduled jobs with a trigger for that Event and long-running jobs with Event handlers are notified.

The `RESOURCE_ONLINE` built-in Event is used by the embedded discovery jobs, such as for discovering operating system and CPU information (`osInfo` and `cpuInfo` jobs). Both `osInfo` and `cpuInfo` job archives (`.job`) include a schedule file (`.sched`) specifying a trigger (`.trig`) that allows these jobs be started when notification of the `RESOURCE_ONLINE` Event occurs.

B.3.3 Rule-based Events

Rule-based Events are defined in an XML document. They are deployed to the Orchestration Server and managed through the Orchestration Console. Rules can be a simple object attribute (fact) equivalency check or they can use AND,OR, IF, ELSE logic, among other things, in an Event ruleset.

The rules follow the same syntax as the constraints that are defined in XML policy files for all Grid Objects, such as Jobs, VM Hosts, etc.

The Orchestration Server Event Service evaluates the rules; if the rules pass, an Event notification occurs.

The XML Schema document specification can be found in `<install dir>/doc/xsds/event_1_0_0.xsd`.

The Event XML specification is composed of three sections.

- `<context>`
- `<trigger>`
- `<reset>`

NOTE: Both the `<context>` and `<trigger>` sections are required.

<context> section

The `<context>` section defines the context in which the Event rules are evaluated. With Events, you specify what objects are in the Event rule context in this section. The available objects are `Job`, `Jobinstance`, `Resource`, `Repository`, `User`, and `VMHost`. From these objects, you can specify one object set to iterate over and optionally a single instance of the object.

<trigger> section

The `<trigger>` section defines the rules for when an Event notification occurs. The `<trigger>` format is the same syntax as `<constraints>` used in policies.

<reset> section

The optional `<reset>` section defines the rules for when an Event is reset. If the `<reset>` rule is not used, an Event is reset based on a timeout. The `<reset>` format is also the same syntax as in `<constraints>` used in policies.

The `resetInterval` attribute is set on the `<event>` XML element. If `"resetInterval"` and `<reset>` are not used, the default timeout for resetting is 10 minutes.

The following example, taken from the `"vmhost.event"` in `<install dir>/examples/events`), defines that a notification occurs when a VM host becomes overloaded.

```
1<event>
2
3   <context>
4      <vmhost />
5   </context>
6
7   <trigger>
8      <gt fact="vmhost.vm.count" value="0" />
9      <gt fact="vmhost.resource.loadaverage" value="2" />
10   </trigger>
11
12   <reset>
13      <lt fact="vmhost.resource.loadaverage" value=".5" />
14   </reset>
15
16</event>
```

Lines 3-5: This section defines the context for the Event's rule evaluation.

Line 4: The context specifies all VM host objects, so the Event Service iterates over all VM hosts. On each VM host, the `<trigger>` rule will be evaluated, so in this case, the Event context is composed of one or more VM hosts.

Lines 7-12: This section defines the Trigger rule to determine if this Event is to fire notifications or not. If the trigger rule does not pass, no Event notifications occur.

Line 8: Consider only VM hosts that have at least one VM instance running.

Line 9: Check the running average of the VM host's load average if it exceeds a threshold value. In this case, run the check if the average is greater than 2.

Lines 12-14: This section defines the Reset rule to determine if a previously triggered VM host can be reset and triggered again.

Line 13: Only reset if the running average of the VM host's load average drops below a threshold.

When a VM host passes the trigger rule, the VM host does not pass the trigger rule again until the reset rule (load average drops below threshold) passes.

See the `repository.event` example (`<install dir>/examples/events/repository.event`) for an Event with a rule that evaluates the `freespace` fact on all repository objects.

C The Metrics Facility

The the Orchestration Server Metrics Facility collects, aggregates, and allows simple fact-based retrieval of metric values by jobs and computed facts (via JDL), policy constraints, and Event triggers on a per-resource basis. This provides aggregated metrics generated by gmond without the need for the gmetad Ganglia service. Note that gmetad can still be used in parallel for aggregating gmond reported metrics for visualization purposes.

C.1 Metrics Facility Functionality

The Metrics Facility provides the following functionality:

- Collection of gmond provided metrics using the Orchestration Agent.
- Retrieval of instantaneous metric values via `resource.metrics.<METRIC_NAME>` fact space, where `<METRIC_NAME>` is the name of the metric.
- Deployable Round Robin Database (RRD) (data aggregation) definition using XML `.metric` files, which allows flexible definition of aggregation periods. For example, using `resource.metrics.<METRIC_NAME>.10_minute.average` as a 10-minute aggregation period separate from `resource.metrics.<METRIC_NAME>.1_hour.average`.
- Retrieval of an array of aggregated metric values using `resource.metrics.<METRIC_NAME>.xxx.values`.
- Zero-configuration for core Ganglia metrics. The Orchestration Agent automatically discovers if gmond is running on a resource, and the Orchestration Server collects and, if a `.metric` file is configured, aggregates those metrics.
- Persistence of collected RRD data across server restart and high availability fail-over conditions.

C.2 Ganglia Metrics

Not all Ganglia metrics are suitable for aggregation, such as those of “String” type. By default, only the 24 metrics listed in the following table are supported (either of type “Real” or type “Integer”):

Table C-1 *Supported Ganglia Metrics*

Metric	Type
bytes_in	Real
bytes_out	Real
cpu_aidle	Real
cpu_idle	Real
cpu_nice	Real
cpu_system	Real
cpu_user	Real
cpu_wio	Real
disk_free	Integer
disk_total	Integer
load_fifteen	Real
load_five	Real
load_one	Real
mem_buffers	Integer
mem_cached	Integer
mem_free	Integer
mem_shared	Integer
mem_total	Integer
pkts_in	Real
pkts_out	Real
proc_run	Integer
proc_total	Integer
swap_free	Integer
swap_total	Integer

C.3 How Does the Metrics Facility Impact Orchestration Server Performance?

The Metrics Facility balances flexibility and minimizing the impact on overall Orchestration Server performance. There is no continuous parsing of XML on the server. Instead, parsing of gmond-generated XML is performed by each managed resource.

This section includes the following information:

- Section C.3.1, “I/O Contention,” on page 207
- Section C.3.2, “Too Many Open Files,” on page 207

C.3.1 I/O Contention

By default, RRD-based data aggregation is file-based. Because of the frequency of updates and queries of RRD files, this poses a significant performance issue. The Metrics Facility minimizes I/O contention by using in-memory caching and batched write operations to avoid I/O contention and the resulting performance degradation.

C.3.2 Too Many Open Files

The the Orchestration Server Monitoring Server uses a "one RRD per resource" approach, where a RRD contains the AVERAGE, MIN, and MAX RRAs for multiple metrics (DS). In contrast, the Metrics Facility takes an "inside out" approach, which results in "one RRD per metric." For 1,000 agents reporting 24 Ganglia metrics, this reduces the number of files dramatically (from 24,000 in the "one RRD per metric, per resource" case, and 1,000 in the "one RRD per resource" case, to 24 in the "one RRD per metric" case). This approach avoids a "too many open files" condition.

NOTE: A "too many open files" condition occurs when the default maximum file descriptors available to a process launched from the Linux shell is exceeded.

C.4 RRD Definition Using Deployable .metric Files

Definition of the "aggregation" functions performed by the Metrics Facility's internal RRD data structures are customizable using deployable XML `.metric` definition files. This accommodates a flexible configuration of the following:

- The Ganglia-reported metrics to include in aggregated data structures
- The data aggregation periods that are of specific interest

The deployable definition files, one per metric to be aggregated, consist of the following:

- The name of the "instantaneous" metric to be aggregated, for example, `load_one`
- An optional description of the metric to be aggregated.
- A "heartbeat" value that governs the updates to the contained RRAs.
- One or more named Periods (corresponding to the RRAs to be created) with an optional description.
- For each Period, the number of data points to aggregate (steps), the number of aggregated data points to archive (rows), and the "xff' (x-files factor) allowed for the metric.

  NOTE: The xff determines how many of the samples can be NaN for the consolidated sample to be considered NaN. Usually, this is set to 0.5, or 50%.).

When an RRD is defined through deployment of its definition file, three RRAs are created for each Period: AVERAGE, MAX, and MIN. A new DS (datasource) is added to the RRD for each resource reporting the metric to be aggregated. This requires the RRD file to be re-created each time a new resource begins reporting a given metric and the previously aggregated values copied from the old RRD to the new one. This approach enhances performance and flexibility, but the RRD file is not of fixed size: Over time, the RRD grows or shrinks as new resources are added to the system or are deleted from the Orchestration Server model.

NOTE: The RRD is actually re-created with a new DS added for each new resource and the "old" RRA's data copied into it.

Deleting a Resource Grid object removes its DS from the RRD file (actually, from all RRDs with metrics reported by that resource).

One optimization you can implement for storing the smallest Period (consisting of a single step) is to create only a single RRA (vs. three), because the average of a single datapoint is equal to the maximum and minimum of a single datapoint.

NOTE: The RRD files created by the Java rrd4j library are not binary compatible with RRD files generated by the rrdtool used by gmetad. They are however portable across operating system architectures (e.g., 32-bit bigendian vs. 64-bit little-endian) which is not possible with traditional RRD files created using rrdtool.

C.4.1 XML Format for Deployable .metric Definitions

An example of the format of the deployable RRD definition is shown below.

```
<metric name="load_one" heartbeat="120"
        description="Ganglia oneminute load average">


    <period name="1_minute" steps="1" rows="60" xff=".5"
        description="1 hour worth of 1 minute (raw) data"/>


    <period name="5_minute" steps="5" rows="12" xff=".5"
        description="1 hour worth of 5 minute aggregations"/>


    <period name="10_minute" steps="10" rows="72" xff=".5"
        description="12 hours worth of 10 minute aggregations"/>


    <period name="1_hour" steps="60" rows="24" xff=".5"
        description="1 days worth of 1 hour aggregations"/>


</metric>
```

This example creates an RRD for the `load_one` metric, with four aggregation periods (RRAs) called `1_minute`, `5_minute`, `10_minute`, and `1_hour`. The default sample (RRD update) time is one minute, so the `10_minute` aggregation period has 10 steps. 72 "rows" of aggregated datapoints are retained in the RRA before the oldest is dropped off, representing 12 hours (12 * 60 / 10) worth of data.

NOTE: The `1_minute` period is not a true aggregation because the default sample (RRD update) time is also one minute. In this case, the "raw" datapoints are stored for historical reference.

C.5 Query of Aggregated Metric Values

Aggregated metric values can be queried similar to the instantaneous values either from JDL, from within a policy/event constraint, or from an array of multiple metrics values. This section includes the following information:

- Section C.5.1, “Example of a JDL Query for Aggregated Metric Values,” on page 209
- Section C.5.2, “Example of a Policy Constraint or Event Constraint Using Aggregated Metric Values,” on page 209
- Section C.5.3, “Example of Using Non-aggregated (“Raw”) Historical Metric Values,” on page 209

C.5.1 Example of a JDL Query for Aggregated Metric Values

```
r = getMatrix().getResource(“local_vmhost”)

print "resource.id: %s" % (r.getFact("resource.id"))

print "load_one.10_minute.average: %s" %
( r.getFact("resource.metrics.load_one.10_minute.average") )
```

C.5.2 Example of a Policy Constraint or Event Constraint Using Aggregated Metric Values

```
   <event>
      <context>
         <vmhost />
      </context>
      <trigger>
         <gt fact="vmhost.vm.count" value="0" />
        <gt fact="vmhost.resource.metrics.load_one.10_minute.average" value="2" />
      </trigger>
      <reset>
        <lt fact="vmhost.resource.metrics.load_one.10_minute.average" value=".5" /
>
      </reset>
   </event>
```

The consolidation functions (AVERAGE,MAX, and MIN) are supported for each defined aggregation period. In RRD terminology, this means that for each metric, there are three RRAs defined for each “period” element in the `.metric` deployable definition.

C.5.3 Example of Using Non-aggregated (“Raw”) Historical Metric Values

You can query for an array of “raw” values that constitute the aggregated datapoints for a given RRA within the RRD data structure by appending `.values` to the factname representing the metric (and period) of interest. For example, to print all the MIN values collected for the `10_minute` aggregation period, the JDL is

```
r = getMatrix().getResource(“local_vmhost”)

print "resource.id: %s" % (r.getFact("resource.id"))
print "load_one.10_minute.min.values: %s" %

( r.getFact("resource.metrics.load_one.10_minute.min.values") )
```

Similarly, to print the array of AVERAGE and MAX values collected, the JDL is

```
print "load_one.10_minute.average.values: %s" %
( r.getFact("resource.metrics.load_one.10_minute.average.values") )

print "load_one.10_minute.max.values: %s" %
( r.getFact("resource.metrics.load_one.10_minute.max.values") )
```

C.6 MetricsManager MBean API

The MetricsManager facility exposes a small number of methods for disabling and enabling a RRD file creation and update, and for query of both instantaneous and aggregated metric values. The facility also allows a UI to query for information needed for populating a pull-down list that could include, for example, valid metric names for a specified resource or valid aggregation periods for such a metric. The API also currently provides a way to fetch a "running average" of the raw datapoints for a metric (a cached value that does not require a fetch operation from an RRD) and server-side generation of a simple graph of metric data (requires a fetch from an RRD file).

This section includes the following information:

- Section C.6.1, "MBean Methods Exposed by the MetricsManager Facility," on page 210
- Section C.6.2, "The MetricsDeployer Facility," on page 210

C.6.1 MBean Methods Exposed by the MetricsManager Facility

The following table lists the MBean methods that are exposed by the MetricsManager Facility.

Table C-2 *MBean Methods Exposed by the MetricsManager Facility*

MBean Method	Description
`setMetricsEnabled()`	A parameter of false disables the creation or update of RRD files.
`getObjectNames()`	Returns a list of current deployed metric definitions
`update()`	Updates RRDs with instantaneous values collected by the Metrics Facility.
`getFact()`	Given the fully qualified fact name, fetches the value of a specified resource's fact.
`getMetricNames()`	Fetches the metric names relevant for a specified resource .
`getPeriodNames()`	Fetches the names of periods (RRAs) defined for a specified metric.
`getRunningAverage()`	Fetches the running average of a metric for a specified resource.
`getRrdGraphic()`	Used by the Orchestration Console to fetch a server-generated ImageIconInfo object.

C.6.2 The MetricsDeployer Facility

The MetricsDeployer deployment facility parses the `.metric` definition files and creates the associated grid objects that are used to maintain the metadata related to the RRD used for aggregation.

C.7 Using the Metrics Facility in the Orchestration Console

The Orchestration Console supports the Metrics Facility in the following ways:

- **Metrics object:** A Metrics object is deployed in the Explorer tree. Use the right-click menu to display the “deploy” and “undeploy” actions. Pre-defined `.metric` files are located in the `/opt/novell/zenworks/zos/server/components/metrics` folder, or you can create a new `.metric` file and paste in the XML as shown in Section C.4.1, “XML Format for Deployable .metric Definitions,” on page 208.

  Metrics objects are listed by their deployment name, which may or may not be the same as the name of the actual metric. This potentially allows multiple, separately deployable, RRD definitions for a single “instantaneous” metric, with different aggregation periods defined.
- **Metrics Editor:** Selecting the Metrics object opens a free-form XML editor in the admin view of the client. The editor is similar to the Event” editor for viewing RRD definition XML.