

Development Client Reference

Novell® PlateSpin® Orchestrator

2.0.2

August 28, 2009

www.novell.com



Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. See the [Novell International Trade Services Web page \(http://www.novell.com/info/exports/\)](http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2008-2009 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed on the [Novell Legal Patents Web page \(http://www.novell.com/company/legal/patents/\)](http://www.novell.com/company/legal/patents/) and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see [the Novell Documentation Web page \(http://www.novell.com/documentation\)](http://www.novell.com/documentation).

Novell Trademarks

For Novell trademarks, see [the Novell Trademark and Service Mark list \(http://www.novell.com/company/legal/trademarks/tmlist.html\)](http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide	9
1 Layout	11
2 Orchestrate Development Client Menus and Tools	15
2.1 The Operations Menu Bar	15
2.1.1 File	15
2.1.2 Edit	16
2.1.3 View	19
2.1.4 Actions	19
2.1.5 Provision	19
2.1.6 Server	21
2.1.7 Windows	23
2.1.8 Help	23
2.2 The Orchestrate Development Client Toolbar	23
3 The PlateSpin Orchestrate Job Scheduler	25
3.1 Understanding the Job Scheduler View	25
3.1.1 Navigating The Job Schedules Table	26
3.1.2 Creating or Modifying a Job Schedule	28
3.1.3 Understanding Cron Syntax in the Job Scheduler	37
3.2 Walkthrough: Scheduling a System Job	41
3.2.1 Deploying a Sample System Job	41
3.2.2 Creating a New Schedule for the Job	42
3.2.3 Defining the New Schedule	43
3.2.4 Activating the New Schedule	48
3.2.5 Running the New Schedule Immediately	48
4 The Policy Debugger	51
4.1 The Constraints Table View	51
4.1.1 The Match Context Area	52
4.1.2 The Constraint Type List	53
4.1.3 The Verbose Check Box	54
4.1.4 The Capable Resources Summary	54
4.1.5 The Constraints Column of the Constraints Table View	54
4.1.6 The Policy Column of the Constraints Table	55
4.2 The Facts Table View	56
4.2.1 The All Facts Check Box	56
4.3 Policy Debugger Use Cases	57
4.3.1 Use Case 1: Determining Why a Job is in a Waiting State	57
5 The Explorer Tree	59
5.1 The Orchestrate Server Object	59
5.1.1 The Orchestrate Server Info/Configuration Tab	60
5.1.2 The Orchestrate Server Authentication Tab	65
5.1.3 The Orchestrate Server Policies Tab	67

5.1.4	The Orchestrate Server Constraints/Facts Tab	67
5.2	The Server Admin Object.	67
5.3	The Job Object.	68
5.3.1	Job Groups	68
5.3.2	The Job Info/Groups Tab	68
5.3.3	The JDL Editor Tab	78
5.3.4	The Job Library Editor Tab	79
5.3.5	The Job Policies Tab	80
5.3.6	The Job Constraints/Facts Tab	81
5.4	The Resource Object	81
5.4.1	Resource Groups	81
5.4.2	The Resource Info/Groups Tab	82
5.4.3	The Provision Info Tab	102
5.4.4	The Resource Log Tab	103
5.4.5	The Resource Policies Tab	103
5.4.6	The Resource Health Debugger Tab	103
5.4.7	The Resource Constraints/Facts Tab	104
5.5	The VM Host Object.	104
5.5.1	The VM Host Info Tab	104
5.5.2	The VM Host Policies Tab	108
5.5.3	The VM Host Health Debugger Tab	108
5.5.4	The VM Host Constraints/Facts Tab	108
5.5.5	The VM Host Action History Tab	108
5.6	The Repository Object	109
5.6.1	Repository Groups	109
5.6.2	The Repository Info/Groups Tab	109
5.6.3	The Repository Policies Tab	116
5.6.4	The Repository Health Debugger Tab	116
5.6.5	The Repository Constraints/Facts Tab	117
5.6.6	The Repository Action History Tab	117
5.7	The User Object.	117
5.7.1	User Groups	117
5.7.2	The User Info/Groups Tab	118
5.7.3	The User Policies Tab	123
5.7.4	The User Health Debugger Tab	124
5.7.5	The User Constraints/Facts Tab	124
5.7.6	The User Action History Tab	124
5.8	Other Displayed Objects	124
5.8.1	The Policy Object.	125
5.8.2	Computed Fact Objects	125
5.8.3	Event Objects	125
5.8.4	Metrics	125

6 The Health Debugger 127

6.1	The Constraints Table Panel	127
6.1.1	The Match Context Area	128
6.1.2	The Verbose Check Box	129
6.1.3	The Capable Objects Summary	129
6.1.4	The Constraints Column of the Constraints Table View	129
6.2	The Facts Table View	130
6.2.1	The All Facts Check Box	131
6.3	Health Debugger Use Cases	131
6.3.1	Use Case 1: ??	132

A	Grid Object Health Monitoring	133
A.1	Health Facts	133
A.2	Health Events	135
B	Understanding Policy Elements	137
B.1	Constraints	137
B.2	Facts	137
B.3	Computed Facts	137
C	Events	141
C.1	Event Object Visualization and Management in the Development Client	141
C.1.1	Deploying a New Rule-Based Event	142
C.1.2	Deploying a Pre-written Rule-Based Event	142
C.1.3	Undeploying an Event	142
C.1.4	The Event Editor	143
C.2	The Event Debugger	143
C.2.1	The Constraints Table	144
C.2.2	The Facts Table	145
C.3	Understanding the PlateSpin Orchestrate Events System	146
C.3.1	Event Notification	147
C.3.2	Built-in Events	147
C.3.3	Rule-based Events	148
D	Provisioning Actions and History	151
D.1	What are Provisioning Actions?	151
D.2	How Actions Are Displayed in the Development Client	151
D.2.1	Action History in Monitor Views of the Development Client	151
D.2.2	Action History in Admin Views of the Development Client	152
E	Documentation Updates	155
E.1	August 28, 2009	155
E.2	August 7, 2009	155
E.3	July 20, 2009	155
E.4	June 17, 2009 (2.0.2 Release)	156

About This Guide

This PlateSpin Orchestrate Development Client Reference introduces the Development Client of PlateSpin® Orchestrate from Novell®, the product's basic administration environment. The guide provides an introductory overview of the Orchestrate Development Client interface. The guide is organized as follows:

- ♦ Chapter 1, “Layout,” on page 11
- ♦ Chapter 2, “Orchestrate Development Client Menus and Tools,” on page 15
- ♦ Chapter 3, “The PlateSpin Orchestrate Job Scheduler,” on page 25
- ♦ Chapter 4, “The Policy Debugger,” on page 51
- ♦ Chapter 5, “The Explorer Tree,” on page 59
- ♦ Appendix A, “Grid Object Health Monitoring,” on page 133
- ♦ Appendix B, “Understanding Policy Elements,” on page 137
- ♦ Appendix C, “Events,” on page 141
- ♦ Appendix D, “Provisioning Actions and History,” on page 151
- ♦ Appendix E, “Documentation Updates,” on page 155

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.

Documentation Updates

For the most recent version of this *Development Client Reference*, visit the [PlateSpin Orchestrate 2.0 documentation Web site](http://www.novell.com/documentation/pso_orchestrate20/) (http://www.novell.com/documentation/pso_orchestrate20/).

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux* or UNIX, should use forward slashes as required by your software.

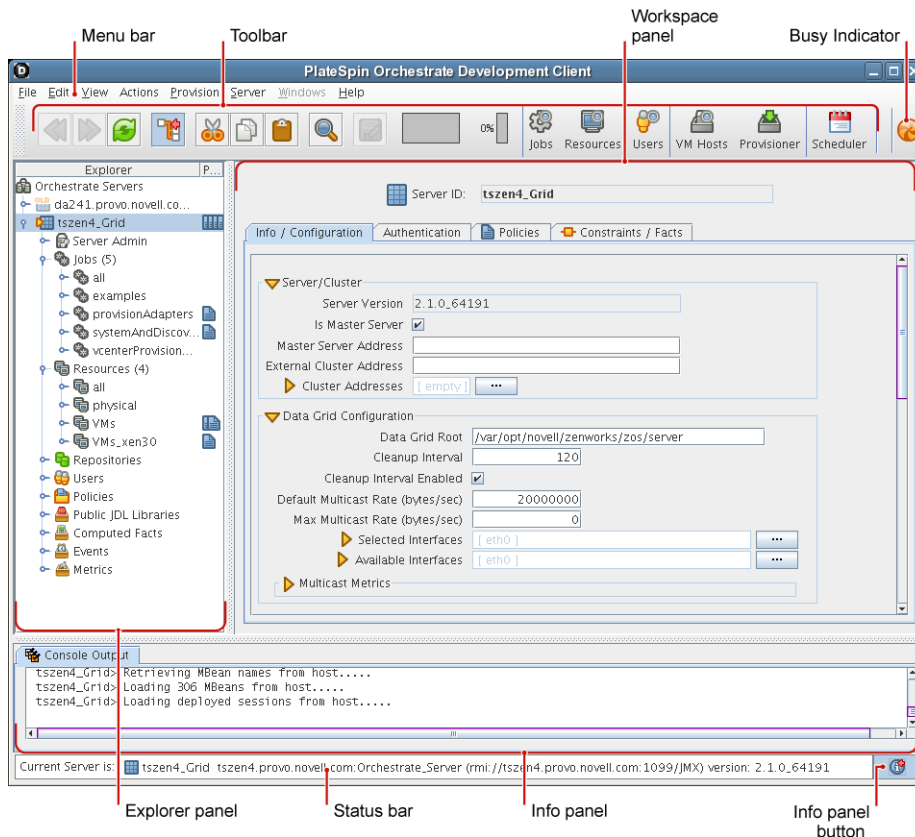
Layout

1

Both the grid administrator and the job developer need to have access to and use the PlateSpin® Orchestrate Development Client. The administrator needs to use the console to perform any management functions, such as creating user accounts and managing Orchestrate 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 PlateSpin Orchestrate Development Client with Parts Identified



The following chart describes the functional areas of the main PlateSpin Orchestrate Development Client display.

Table 1-1 Detailed Description of Console Areas

Area	Description
Menu bar	<p>Provides operations categorized under menus such as <i>File</i>, <i>Edit</i>, <i>View</i>, <i>Grid</i>, <i>Server</i>, <i>Windows</i>, and <i>Help</i>.</p> <ul style="list-style-type: none">◆ The <i>File</i> menu lets you save any changes you've made or exit the console.◆ The <i>Edit</i> menu lets cut, copy, and paste items and choose general and server preferences for console.◆ The <i>View</i> menu lets you manipulate the display of the different components of the console and refresh the Explorer and Workspace panels.◆ The <i>Actions</i> menu lets you launch specific tools that create and delete users or user groups, computing resources, jobs, policies, and computed facts.◆ The <i>Server</i> 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 the server.◆ The <i>Windows</i> 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 (<i><Orchestrator> Log</i> and <i>Console Output</i>) in their own windows by right-clicking the tab and choosing <i>Open in window</i> in the pop-up menu.◆ The <i>Help</i> menu provides access to the About box for the console. It also provides a link to ZENworks Orchestrator documentation on the Web.
Main toolbar	<p>The main toolbar has buttons for executing common tasks. The basic tasks are <i>Go Back</i>, <i>Go Forward</i>, <i>Refresh the view</i>, <i>Hide or Show the Explorer Panel</i>, <i>Cut</i>, <i>Copy</i>, <i>Paste</i>, and <i>Save changes in workspace view</i> and <i>Open the Find Dialog</i>.</p> <p>The toolbar also includes buttons that open monitoring views for <i>Jobs</i>, <i>Resources</i>, and <i>Users</i>.</p> <p>To the far left of the toolbar, a pinwheel icon indicates when the console is busy.</p>
Explorer panel	<p>The Explorer panel displays a hierarchical tree. The tree 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 <i>Computing Resources</i> in the tree, its details appear in the Workspace panel with a list of active computing resources. You can edit the <i>Computing Resource</i> attributes in the workspace panel.</p>
Workspace panel	<p>The Workspace panel displays a detailed view for an item you select in the Explorer panel. For example, if you select a computing resource under <i>physical</i> in the Explorer panel, the Workspace panel view changes to show the details for that resource. You can edit the properties of an Orchestrator object in the views displayed in the Workspace panel.</p>
Info panel	<p>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 <i>Info panel</i> button in the Status bar.</p>

Area	Description
Status bar	The status bar displays general identity information about the Orchestrator Server where you are logged in.

For information about launching the console and using it for the first time, see “[Walkthrough: Launching the PlateSpin Orchestrate Development Client](#)” in the *PlateSpin Orchestrate 2.0 Installation and Configuration Guide*.

For detailed information about the components, icons, and usage of the PlateSpin Orchestrate Development Client, see [Chapter 2, “Orchestrate Development Client Menus and Tools,”](#) on [page 15](#).

Orchestrate Development Client Menus and Tools

2

A number of operations are available from the PlateSpin Orchestrate Development Client from Novell® and can be accessed from its menu bar and toolbar.

- ◆ [Section 2.1, “The Operations Menu Bar,” on page 15](#)
- ◆ [Section 2.2, “The Orchestrate Development Client Toolbar,” on page 23](#)

2.1 The Operations Menu Bar

The Operations Menu Bar in the Orchestrate Development Client provides options that help you to create and administer objects in the Explorer Tree.

- ◆ [Section 2.1.1, “File,” on page 15](#)
- ◆ [Section 2.1.2, “Edit,” on page 16](#)
- ◆ [Section 2.1.3, “View,” on page 19](#)
- ◆ [Section 2.1.4, “Actions,” on page 19](#)
- ◆ [Section 2.1.5, “Provision,” on page 19](#)
- ◆ [Section 2.1.6, “Server,” on page 21](#)
- ◆ [Section 2.1.7, “Windows,” on page 23](#)
- ◆ [Section 2.1.8, “Help,” on page 23](#)

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” on page 15](#)
- ◆ [“Exit” on page 15](#)

Save

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

Exit

The exit operation provides a mouse and keyboard (File > Alt+X) accessible method for users to close all server connections and to exit the Orchestrate Development Client 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” on page 16
- ♦ “Redo” on page 16
- ♦ “Cut” on page 16
- ♦ “Copy” on page 16
- ♦ “Paste” on page 16
- ♦ “Find” on page 17
- ♦ “Find Next” on page 17
- ♦ “Find Previous” on page 17
- ♦ “Enter Find String” on page 17
- ♦ “Load Text” on page 17
- ♦ “Save Text” on page 17
- ♦ “Preferences” on page 17

Undo Addition

The Undo operation provides a mouse-accessible method for users to undo the action they have just performed in the Orchestra Development Client. 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 Orchestra Development Client. 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).

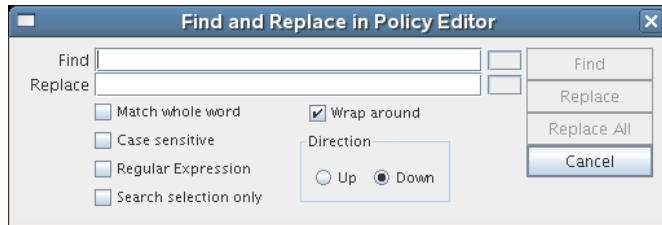
Paste

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.

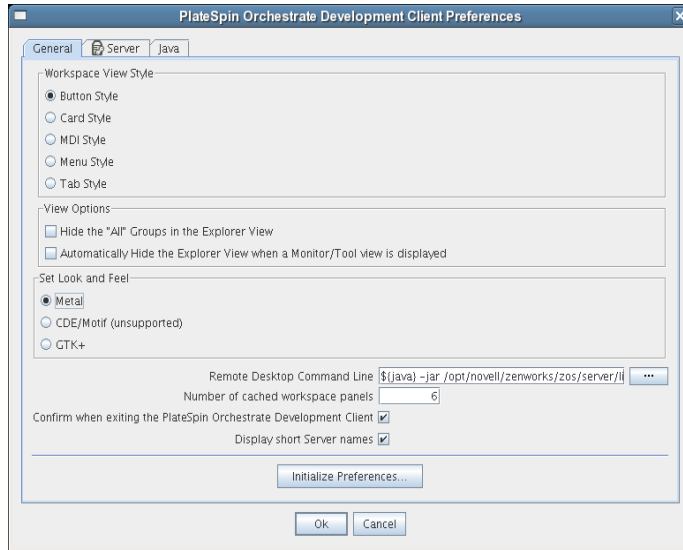
Preferences

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

The dialog box has three tabbed pages.

General Page

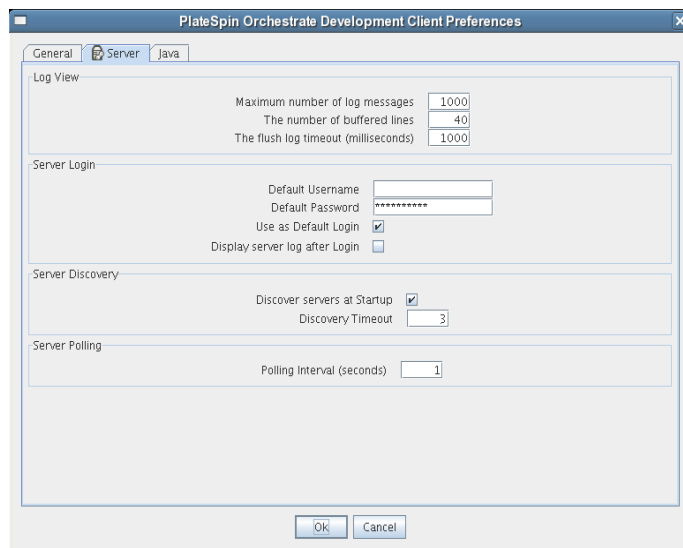
Figure 2-2 General Page of the Orchestrator Development Client Preferences



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

Server Page

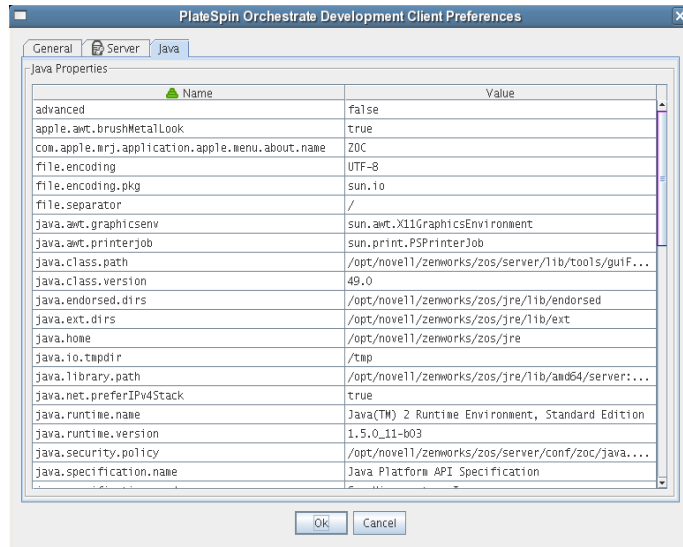
Figure 2-3 Server Page of the Orchestrator Development Client Preferences



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

Java Properties Page

Figure 2-4 The Java Properties Page of the Orchestrate Development Client Preferences



This page lists the Java property names and values that Novell uses to render the Orchestrate Development Client 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 Orchestrate Development Client display of the various PlateSpin Orchestrate 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, “The Orchestrate Development Client 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 remains open after you create each object. Here you can repeatedly create new objects in the dialog, 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 is added to the menu bar only if you have installed Virtual Machine Management. The multiple operations listed in the menu include two of the provisioning actions that you can execute by right-clicking a VM 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*

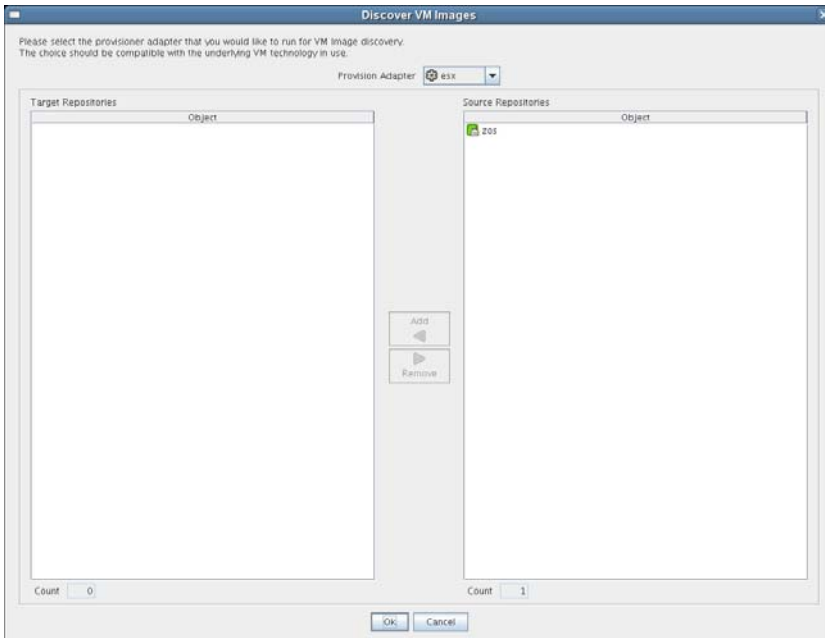


Using this dialog box, you can select a provisioning adapter (`esx`, `hyperv`, `vcenter`, `vmserver`, or `xen30`) that discovers all VM host machines where the PlateSpin Orchestrate Agent is installed and creates objects in the model. The provisioning adapter also discovers the VM Repositories where VM hosts reside.

Discover VM Images

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

Figure 2-6 *VM Images Discover Dialog Box*



Using this dialog box, you can select a provisioning adapter (`esx`, `hyperv`, `vcenter`, `vmserver`, or `xen30`) that discovers all VM images and creates objects in the model.

Other Provisioning Operations

The other operations listed in the menu are self-explanatory.

- ◆ Start VM Hosts
- ◆ Shutdown VM Hosts

- ◆ Shutdown VMs
- ◆ Resync VM's State
- ◆ Resync VMs's Host State
- ◆ Reset State 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 22
- ◆ “Display Log” on page 22
- ◆ “Create Custom Log” on page 22

Select Server

The *Select Server* operation lets you select one of the Orchestrate Servers in your grid to log onto. When you select a server, you are required to log on. 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 Orchestrate Development Client starts.

Shutdown Server

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

Figure 2-7 *The Server Shutdown Dialog Box*



Login

The *Login* operation lets you establish a remote connection to another Orchestrate 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.

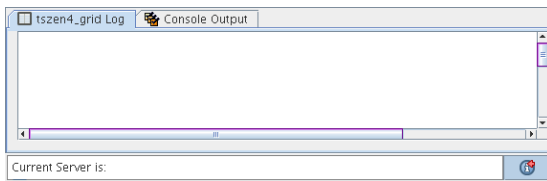
Logout

The *Logout* operation lets you log out of the current, logged on Orchestrate Server without exiting the Orchestrate Development Client. 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 on Orchestrate Server. The display is in the Information window located at the bottom of the Orchestrate Development Client. The server log file is also located by default in the `/var/opt/novell/zenworks/zos/server/logs` directory.

Figure 2-8 Server Log Opened in Information Window of the Orchestrate Development Client



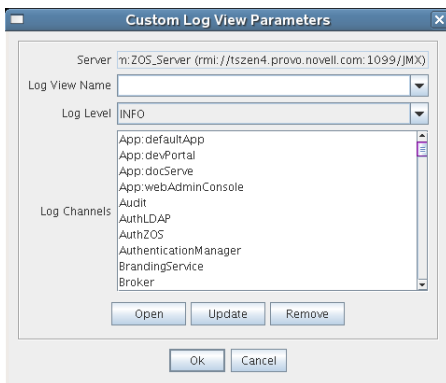
When a log is displayed, you can right-click its tab to further direct the actions of the display. You can pause logging in the window, copy the log to the clipboard, clear its contents, undock the log display as a new window, or remove it from the Information window.

If you right-click on the log display, all of the default editing capabilities of the Orchestrate Development Client are available for your use inside the window. For more information, see [Section 2.1.2, “Edit,” on page 16](#).

Create Custom Log

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

Figure 2-9 The Custom Log View Parameters Dialog Box



By enabling a custom log, you can monitor various components of the Orchestrate 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: Enter the name of the log view. This will be 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 those of greater severity.

Log Channels: A log channel provides log information specific to an PlateSpin Orchestrate 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 Orchestrate Development Client are available for your use inside the window. For more information, see [Section 2.1.2, “Edit,” on page 16](#).

2.1.7 Windows

When you right-click various views and panels in the Orchestrate Development Client, 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 PlateSpin Orchestrate 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 given window is open, its fields and selectable dialogs 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 Orchestrate Development Client.


2.1.8 Help









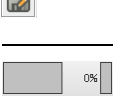

From the *Help* menu, you can access a link to the online PlateSpin Orchestrate documentation (available in `.html` or `.pdf` format) or you can open the About box for the product, where you can view its version number, its license expiration date, and a list of its current management pack capabilities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

2.2 The Orchestrate Development Client Toolbar

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

Table 2-1 *Tool Buttons from the Orchestrate Development Client Toolbar*

Tool Icon	Tool Name	Tool Function
	Back	Go back to the previous workspace view seen.

Tool Icon	Tool Name	Tool Function
	Forward	Go forward to the next workspace view.
	Refresh	Refresh the Explorer and Workspace views.
	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	(Not an active button). This pinwheel shape appears to rotate when the Server is busy performing an operation.

The PlateSpin Orchestrate Job Scheduler

3

You can use the Job Scheduler in PlateSpin® Orchestrate Server from Novell® 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 PlateSpin Orchestrate, 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 PlateSpin Orchestrate 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.

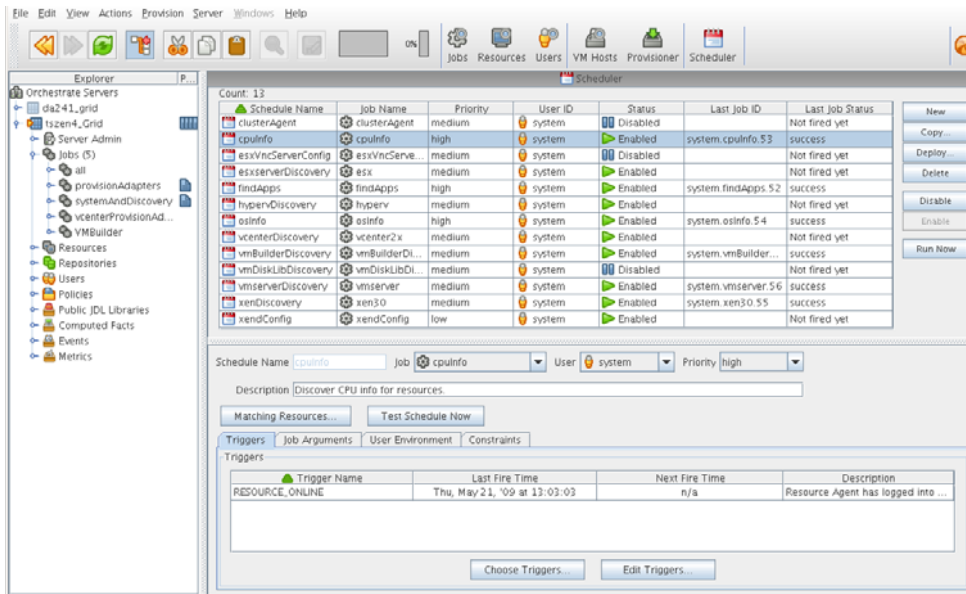
This section includes the following information:

- ♦ [Section 3.1, “Understanding the Job Scheduler View,” on page 25](#)
- ♦ [Section 3.2, “Walkthrough: Scheduling a System Job,” on page 41](#)

3.1 Understanding the Job Scheduler View

Click *Scheduler* on the main toolbar of the PlateSpin Orchestrate Development Client to open the Job Scheduler view.

Figure 3-1 Job Scheduler View of the Orchestrate Development Client



This section includes information to help you understand the functions of the Job Scheduler and how to use it to launch PlateSpin Orchestrate jobs.

- ◆ [Section 3.1.1, “Navigating The Job Schedules Table,”](#) on page 26
- ◆ [Section 3.1.2, “Creating or Modifying a Job Schedule,”](#) on page 28
- ◆ [Section 3.1.3, “Understanding Cron Syntax in the Job Scheduler,”](#) on page 37

3.1.1 Navigating The Job Schedules Table

PlateSpin Orchestrate 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 3-2 The Job Schedules Table in the Job Scheduler View

Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Last Job Status
cpuinfo	cpuinfo	high	zosSystem	Enabled	zosSystem.cpuinfo.2	success
findApps	findApps	high	zosSystem	Enabled	zosSystem.findAp...	success
osinfo	osinfo	high	zosSystem	Enabled	zosSystem.osinfo.3	success
vcenterDiscovery	vcenterDiscov...	medium	zosSystem	Enabled	zosSystem.vcenter...	success
vmBuilderDiscovery	vmBuilderDis...	medium	zosSystem	Enabled		Not fired yet
vmHostVncConfig	vmHostVncCo...	low	zosSystem	Disabled		Not fired yet
vmserverDiscovery	vmserverDisc...	medium	zosSystem	Enabled	zosSystem.vmserv...	success
whdscov	whdscov	medium	zosSystem	Enabled		Not fired yet
xenDiscovery	xenDiscovery	medium	zosSystem	Enabled	zosSystem.xenDis...	success

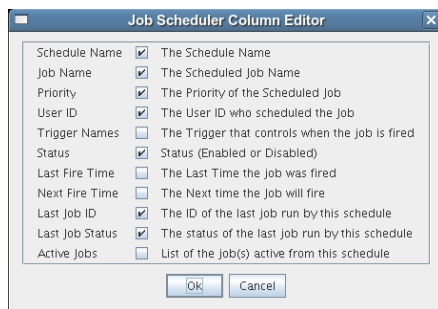
By default, PlateSpin Orchestrate uses schedule names that are similar to the job name so that schedules are easy to match (although this is not required). The schedules list shows all of the existing job schedules that accompany predefined jobs, along with the schedules that you create in the Job Scheduler.

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 tool tip 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 3-3 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 3-2 on page 26](#)) 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 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 PlateSpin Orchestrate, 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 3-4 Some User Object and Job Object Icons 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.

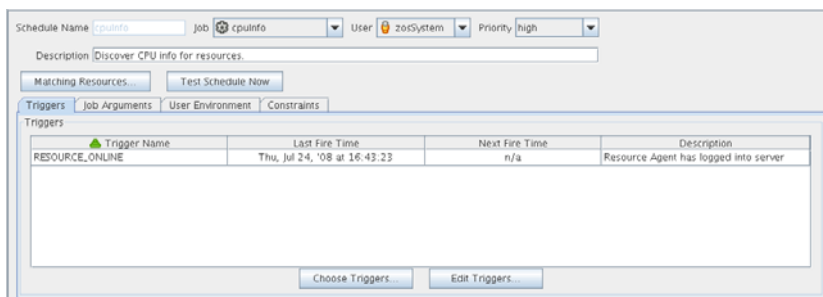
In the *osInfo* schedule, the *system* user has no User icon. That user has been removed from PlateSpin Orchestrate.

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 popup menu for that schedule again.

3.1.2 Creating or Modifying a Job Schedule

The Job Schedule Editor is located immediately below the Job Schedules Table in the Job Scheduler view.

Figure 3-5 The Job Schedule Editor in the Job Scheduler View



There are several times when you can 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 can use the following controls and data when you create or modify a job schedule:

- ◆ “Schedule Name” on page 29
- ◆ “Job” on page 29
- ◆ “User” on page 29
- ◆ “Priority” on page 30
- ◆ “Description” on page 30
- ◆ “Matching Resources” on page 30
- ◆ “Test Schedule Now” on page 30
- ◆ “Triggers” on page 30
- ◆ “Job Arguments” on page 35
- ◆ “User Environment” on page 36
- ◆ “Constraints” on page 37

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, PlateSpin Orchestrate 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.

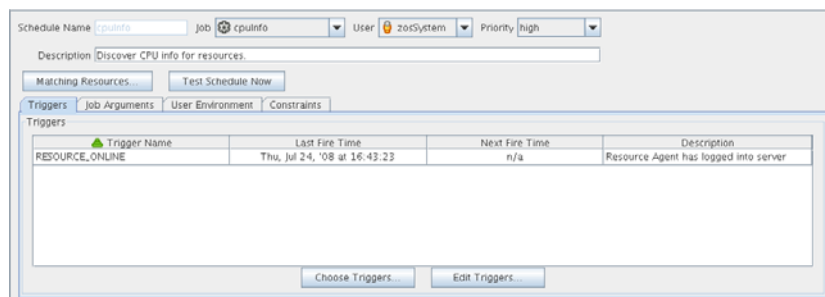
Test Schedule Now

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 3-6 *The Schedule Triggers Page in the Job Scheduler*

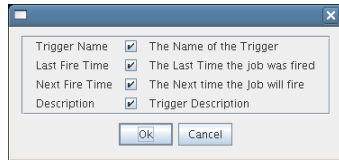


In this view, you can add or define the triggers you want to associate with job schedules. A trigger is the signal to the Job Scheduler to initiate, or “fire” a schedule at a given time or at the occurrence of a given event. Job Scheduler triggers can be classified with regard to two conditions: events 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 tool tip 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 3-7 Trigger Table Column Editor Dialog Box



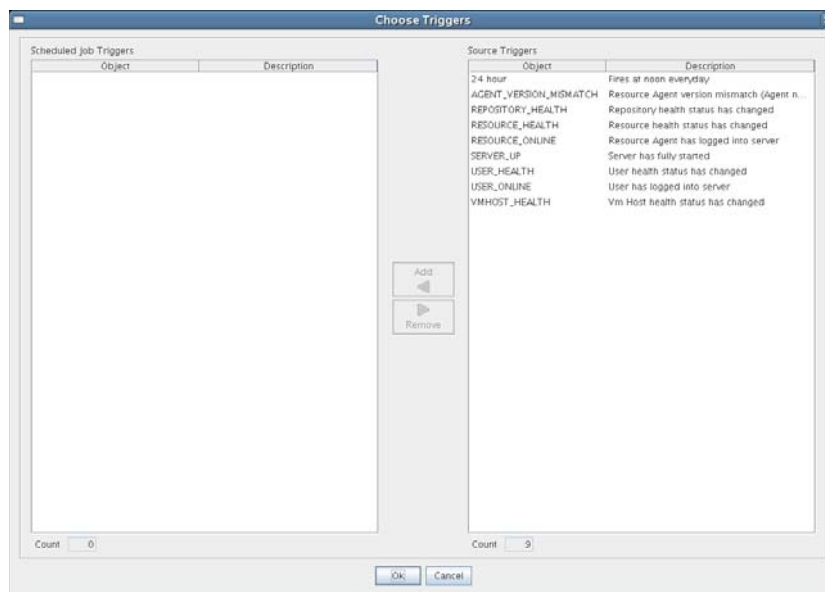
You can select any column heading in this dialog box to display it in the Triggers table. The columns display the attributes of a previously configured Triggers table. As the figure shows, this dialog box also includes text that clarifies the purpose of the data in each column.

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.

Figure 3-8 Choose Triggers Dialog Box



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 PlateSpin Orchestrate are associated with event triggers, which are shown in the previous illustration. 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 when a resource comes online or when a resource health condition change occurs. 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 a PlateSpin Orchestrate Agent of an incompatible version attempts to connect to this Orchestrate Server. It can be used to initiate a configuration management tool for an agent software update or the job could e-mail 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 3-1 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

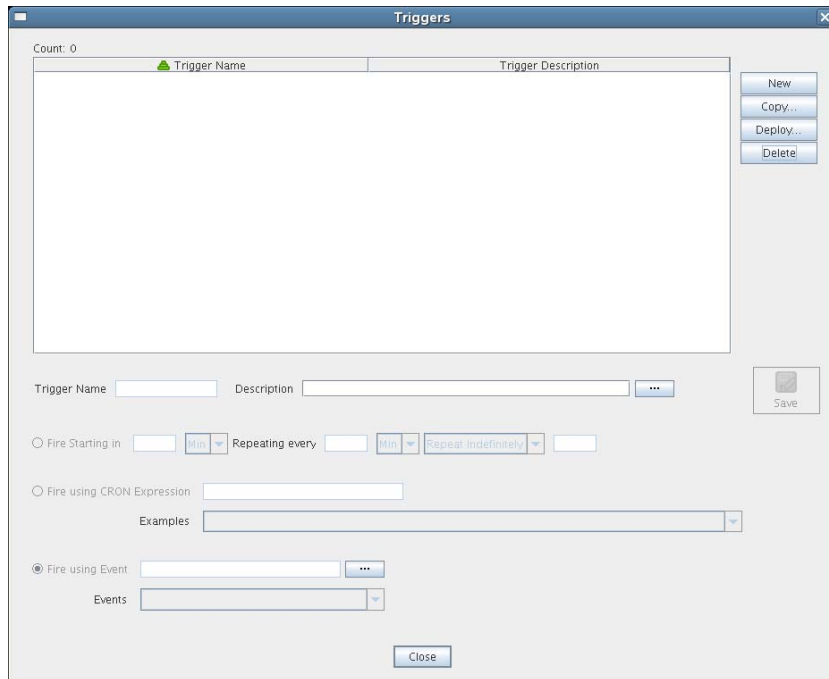
Time Triggers

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.

Figure 3-9 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 icon 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 44](#). For detailed information about cron syntax, see [“Understanding Cron Syntax in the Job Scheduler” on page 37](#).
 - ◆ **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. For more information on creating and firing an event, see [“Creating and Firing an Event” on page 38](#).
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 3-10 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 the *PlateSpin Orchestrate 2.0 Developer Guide and 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 PlateSpin Orchestrate 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 Orchestrate Development Client.

- ◆ `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. For more information about working with constraints, see “[Constraints](#)” in the *PlateSpin Orchestrate 2.0 Developer Guide and Reference*.

3.1.3 Understanding Cron Syntax in the Job Scheduler

The cron triggers you can configure in the PlateSpin Orchestrate 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](#)” on page 37
- ♦ “[Special Characters](#)” on page 38
- ♦ “[Examples of Cron Syntax](#)” on page 39
- ♦ “[Cron Scheduling Precautions](#)” on page 40

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 3-2 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	, - * /

Field Name	Mandatory?	Allowed Values	Allowed special Characters
Day of the Week	Yes	1-7 OR SUN-SAT	, - * ? / L #
Year	No	EMPTY, 1970-2099	, - * /

So 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 2002-2010

Special Characters

Cron syntax incorporates logical operators, special characters that perform operations on the values provided in the cron fields.

Table 3-3 *Special Characters in PlateSpin Orchestrate 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. See the examples below for clarification.
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.

Operator	Purpose	Example
L ("last")	<p>The <code>L</code> character is allowed for the day-of-month and day-of-week fields.</p> <p>Specifies either the last day of the month, or the last xxx day of the month.</p>	<p>The value <code>L</code> in the day-of-month field means "the last day of the month"—day 31 for January, day 28 for February in non-leap years. If you use <code>L</code> 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, <code>6L</code> means "the last Friday of the month."</p> <hr/> <p>TIP: When you use the <code>L</code> option, be careful not to specify lists or ranges of values. Doing so causes confusing results.</p>
W ("weekday")	<p>The <code>W</code> character is allowed for the day-of-month field.</p> <p>Specifies the weekday (Monday-Friday) nearest the given day.</p>	<p>If you specify <code>15W</code> 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 <code>1W</code> 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 <code>W</code> character can only be specified when the day-of-month is a single day, not a range or list of days.</p> <hr/> <p>TIP: You can combine the <code>L</code> and <code>W</code> characters for the day-of-month expression to yield <code>LW</code>, which translates to "last weekday of the month."</p>
pound sign (#)	<p>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.</p>	<p>The value of <code>6#3</code> in the day-of-week field means the third Friday of the month (day 6 = Friday and #3 = the 3rd one in the month).</p> <p>Other Examples: <code>2#1</code> specifies the first Monday of the month and <code>4#5</code> specifies the fifth Wednesday of the month. However, if you specify <code>#5</code> and there are fewer than 5 of the given day-of-week in the month, no firing occurs that month.</p>

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 3-4 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 * * ? 2008	Fire at 10:15 a.m. every day during the year 2008
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 2008-2011	Fire at 10:15 a.m. on every last Friday of every month during the years 2008, 2009, 2010, and 2011
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 out of Daylight Saving Time can cause a skip or a repeat in the schedule firing, depending on whether the clock moves backward or forward.

3.2 Walkthrough: Scheduling a System Job

This section demonstrates how you can use the Orchestrate Development Client to deploy and schedule an existing system job named `auditcleaner.job`. This example job is included in the `examples` directory of your PlateSpin Orchestrate installation.

This section includes the following information:

- ♦ [Section 3.2.1, “Deploying a Sample System Job,” on page 41](#)
- ♦ [Section 3.2.2, “Creating a New Schedule for the Job,” on page 42](#)
- ♦ [Section 3.2.3, “Defining the New Schedule,” on page 43](#)
- ♦ [Section 3.2.4, “Activating the New Schedule,” on page 48](#)
- ♦ [Section 3.2.5, “Running the New Schedule Immediately,” on page 48](#)

3.2.1 Deploying a Sample System Job

Before a job can run, the PlateSpin Orchestrate 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 PlateSpin Orchestrate Development Client by right-clicking the *Jobs* container in the Explorer panel.
- ♦ Deploy it from the PlateSpin Orchestrate Development Client by selecting the *Actions* menu in the Orchestrate Development Client.
- ♦ Deploy it from the `zosadmin` command line (`zosadmin deploy path_to_job`).
- ♦ Copy the deployable component to the “hot” deployment directory under the Orchestrate 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. PlateSpin Orchestrate 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.

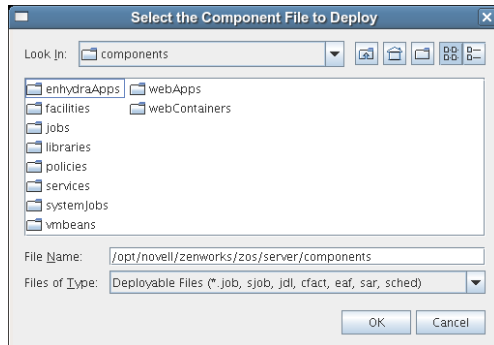
For this walkthrough, you deploy one of several system jobs (`auditCleaner.job`) developed for PlateSpin Orchestrate customers to demonstrate how system jobs are deployed and run. This job package, which is actually a `.jar` archive, includes only a `.policy` component and a `.jdl` component. It does not have a `.sched` component. You can use the Job Scheduler in the Orchestrate Development Client to add the `.sched` component separately.

NOTE: A PlateSpin Orchestrate 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 PlateSpin Orchestrate, 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.

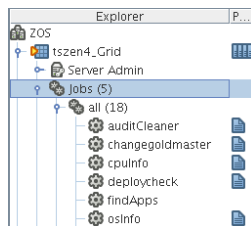
For more information about developing jobs and schedules in a job archive, see “[Job Scheduling](#)” in the *PlateSpin Orchestrate 2.0 Developer Guide and Reference*.

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 PlateSpin Orchestrate Development Client, right-click the *Jobs* container, then click *Deploy Job* to open the Select the Component File to Deploy dialog box.




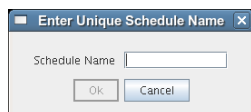
- 2 Open the *Look In* drop-down list, then navigate to the location of the job you want to deploy. Although a job developer can store PlateSpin Orchestrate jobs at any location on the network, the sample jobs shipped with PlateSpin Orchestrate are limited to the directories where the product is installed. 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.




3.2.2 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 Orchestrate Development Client.

- 1 From the toolbar in the Orchestrate Development Client, click the Job Scheduler icon  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.

Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Last Job Status	Last Fire Time	Active Jobs
 cleaner				Paused		Not fired yet	n/a	

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 3.2.3, “Defining the New Schedule,”](#) on page 43 to define this new schedule by adding the specific information you want.

3.2.3 Defining the New Schedule

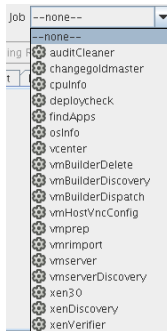
Defining a new job schedule consists of selecting its general properties, its specific properties, and the triggers you want to be associated with it. This section includes the following information:

- ♦ [“Choosing General Properties for a New Schedule”](#) on page 43
- ♦ [“Creating and Assigning a Time Trigger for the New Schedule”](#) on page 44
- ♦ [“\(Optional\) Adding Specific Parameters to the New Schedule”](#) on page 46

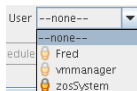
Choosing General Properties for a New Schedule

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.



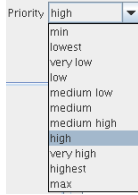
- 2 From the list of available jobs, select *auditCleaner* as the job to which this schedule applies.
- 3 In the Job Schedule Editor, select the *User* drop-down list.




- 4 From the list of available users, select *zosSystem* as the user who runs this job.

The *zosSystem* 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.



- 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.
- Click the *Save* icon  on the toolbar of the Orchestrate Development Client 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.

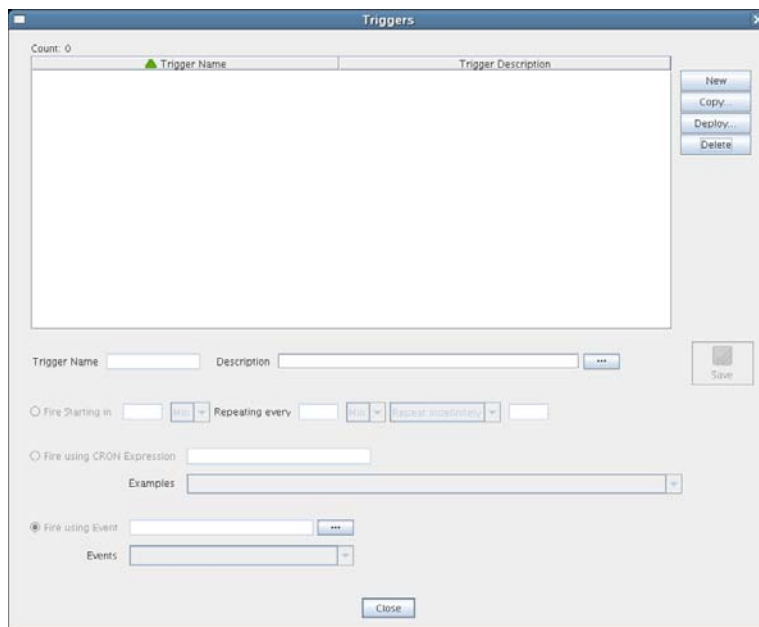
When you have chosen the general properties of the new schedule, you can either continue by [“\(Optional\) Adding Specific Parameters to the New Schedule” on page 46](#) or by proceeding directly to [“Creating and Assigning a Time Trigger for the New Schedule” on page 44](#).

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 use the following steps to define a time trigger.

- 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.

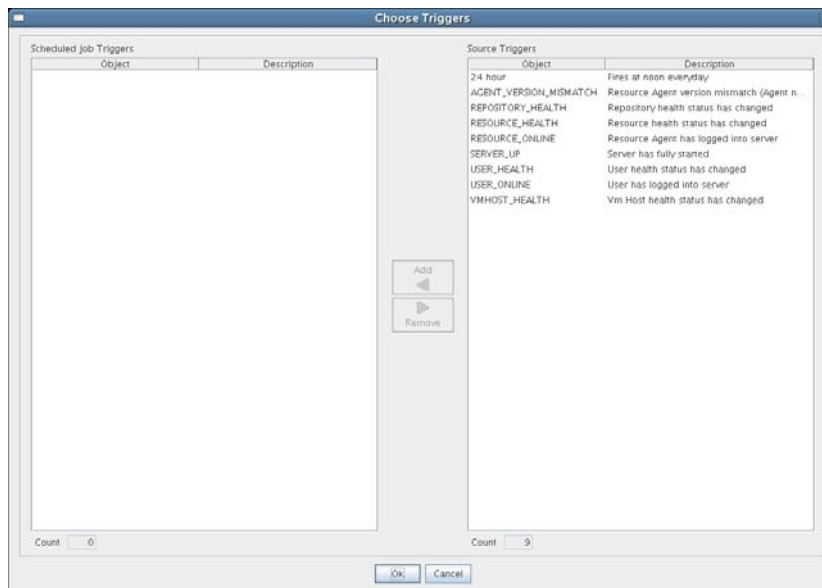
NOTE: For detailed information about cron syntax, see [“Understanding Cron Syntax in the Job Scheduler” on page 37.](#)

- 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.

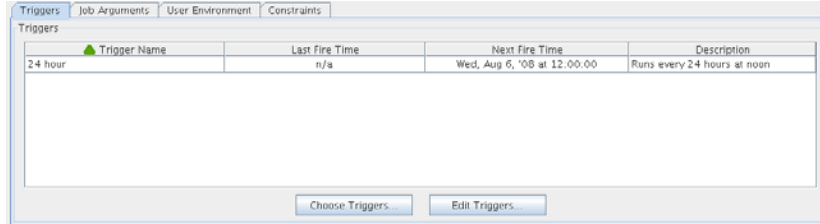
- 7 Click *Save* to save the trigger you just created, then click *Close* to return to the Job Scheduler view.
- 8 From the Job Scheduler view, make sure that the *cleaner* schedule is selected, then click *Choose Triggers* to display the Choose Triggers dialog box.




- 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* icon  to update the Orchestrate Server with the new schedule/trigger association.

(Optional) Adding Specific Parameters to the New Schedule

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:

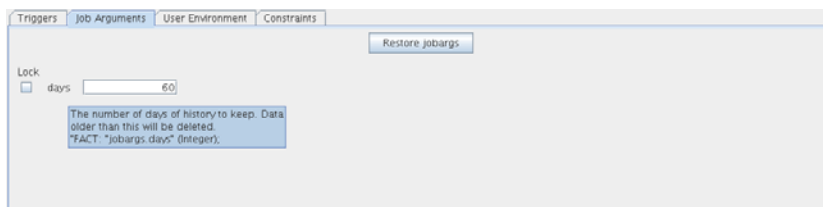
- ◆ “Job Arguments” on page 46
- ◆ “User Environment” on page 47
- ◆ “Constraints” on page 47

Job Arguments

As explained in [Section 3.1.2, “Creating or Modifying a Job Schedule,” on page 28](#), 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 3-11 *The Job Arguments Tab of the Job Schedule Editor*



According to the tool tip 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 PlateSpin Orchestrate 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.

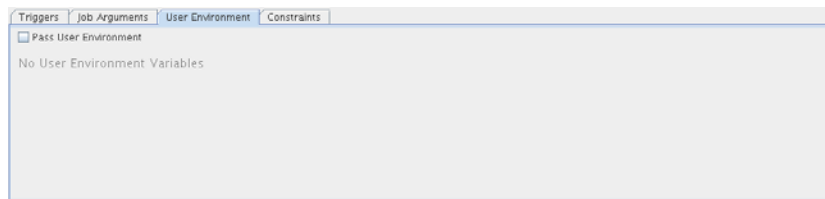
If the default value for a job argument parameter is missing, the job might fail, so you should inspect these parameters carefully.

User Environment

As explained in [Section 3.1.2, “Creating or Modifying a Job Schedule,” on page 28](#), 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 3-12 The User Environment Tab of the Job Scheduler Editor, No User Environment Variables Available

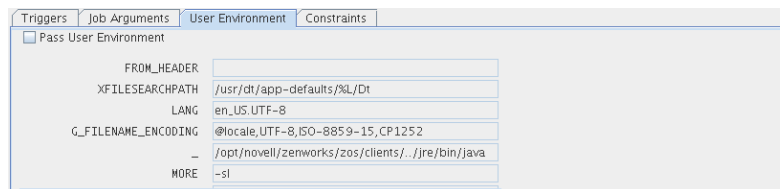


Because there are no environment variables listed, there are none to pass to the schedule, so it is not necessary to select the *Pass User Environment* check box. By default, this check box is not selected, even if environment variables are present for a user specified to run the job.

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 associated a user who had user environment variables with this schedule, you would see a list of those environment variables as they would be passed to the job.

Figure 3-13 The User Environment Tab of the Job Scheduler Editor, User Environment Variables Available

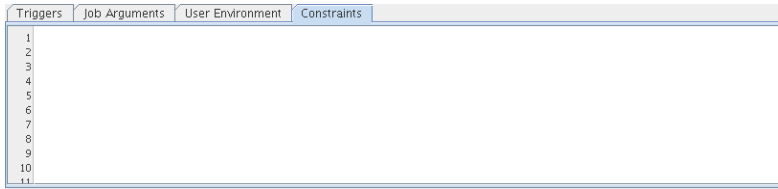


Selecting the *Pass User Environment* check box in this scenario would create these variables as facts used for this job invocation.

Constraints


As explained in [Section 3.1.2, “Creating or Modifying a Job Schedule,” on page 28](#), the *Constraints* tab displays a constraint editor that you can use to create additional constraints for the job being scheduled.

Figure 3-14 The Constraints Tab of the Job Schedule Editor



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 *Save* icon  to update the Orchestrate Server with the new schedule.

For more information about policies, see “[Policy Elements](#)” in the *PlateSpin Orchestrate 2.0 Developer Guide and Reference*.

3.2.4 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.

Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Next Fire Time	Last Fire Time
 cleaner	 auditCleaner	high	 zosSystem	 Disabled		12:00:00	n/a

- 2 Click *Enable* to enable the schedule.

Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Next Fire Time	Last Fire Time
 cleaner	 auditCleaner	high	 zosSystem	 Enabled		12:00:00	n/a

The schedule is now enabled, but has not run yet.

3.2.5 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 icon on the toolbar (*Jobs*) to open the Job Monitor view.

Submit Time	Job ID	Instance Name
16:52:13	zosSyst...	Scheduler(cleaner)


Joblets	Resources	Policy Debugger
Status: 1 Joblet		
Memo:		

The joblet icon shows that the job is running.

- 2 Click the Job Scheduler icon  on the toolbar to open the Job Scheduler view.

Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Last Fire Time	Active Jobs
cleaner	auditCleaner	high	zosSyst...	Enabled	.	12:44:49	zosSystem.audit...

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 icon , 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...

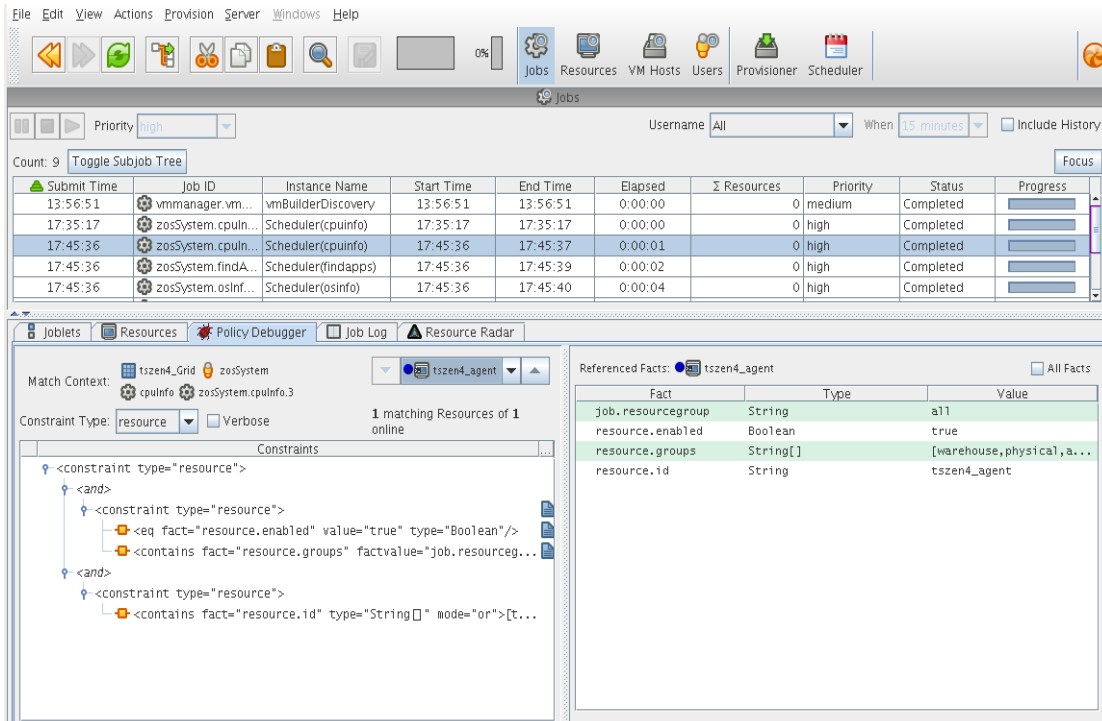
If the job invocation fails, as in this example, a red exclamation icon is also displayed.

The Policy Debugger

4

The Policy Debugger is a tabbed page available in several views of the PlateSpin® Orchestrate Development Client from Novell®. 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 Orchestrate Development Client.

Figure 4-1 PlateSpin Orchestrate Jobs View with the Policy Debugger Page Open



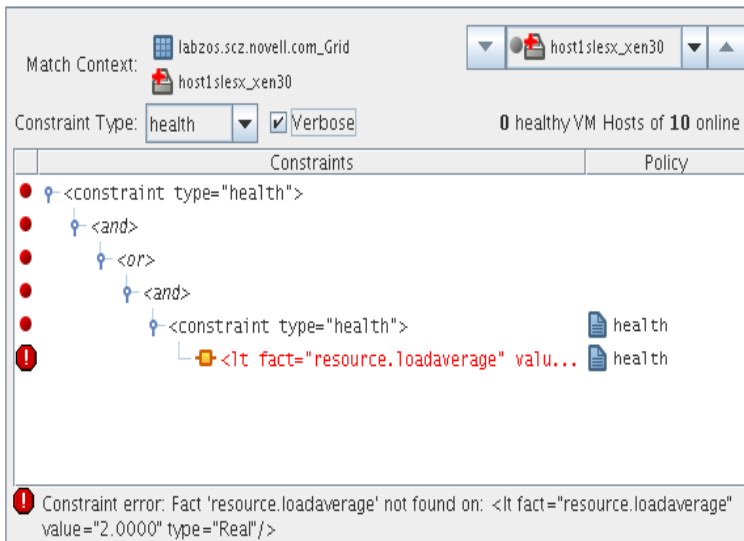
The Policy Debugger tab is also available in the VM Hosts view and in the Provisioner view of the Orchestrate Development Client.

- ◆ [Section 4.1, “The Constraints Table View,” on page 51](#)
- ◆ [Section 4.2, “The Facts Table View,” on page 56](#)
- ◆ [Section 4.3, “Policy Debugger Use Cases,” on page 57](#)

4.1 The Constraints Table View

The left side of the Policy Debugger page is referred to as the Constraints Table view.

Figure 4-2 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 4.1.2, “The Constraint Type List,”](#) on page 53.

The Constraints Table View is composed of several parts:

- ◆ [Section 4.1.1, “The Match Context Area,”](#) on page 52
- ◆ [Section 4.1.2, “The Constraint Type List,”](#) on page 53
- ◆ [Section 4.1.3, “The Verbose Check Box,”](#) on page 54
- ◆ [Section 4.1.4, “The Capable Resources Summary,”](#) on page 54
- ◆ [Section 4.1.5, “The Constraints Column of the Constraints Table View,”](#) on page 54
- ◆ [Section 4.1.6, “The Policy Column of the Constraints Table,”](#) on page 55



4.1.1 The 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 it references Facts, which are also referenced in Policies. The Policies define how, when, and where the job runs.

Figure 4-3 The Match Context Area of the Constraints Table View in the Policy Debugger



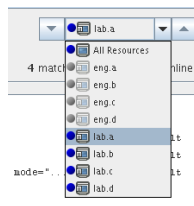
That identifying Facts in the Match Context include:

- ◆ Matrix: The  icon and a text string identifies the machine that matches the grid name given to the Orchestrate 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, and their constraints match so they are capable of servicing the job.

Figure 4-4 Resource Drop Down List Showing Online and Offline Resources



4.1.2 The 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:** accept
- ♦ **start:** start
- ♦ **continue:** continue
- ♦ **provision:** provision
- ♦ **allocation:** allocation
- ♦ **resource:** resource
- ♦ **vmhost:** vmhost
- ♦ **repository:** repository
- ♦ **health:** health

4.1.3 The 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 4.1.5, “The Constraints Column of the Constraints Table View,”](#) on page 54.

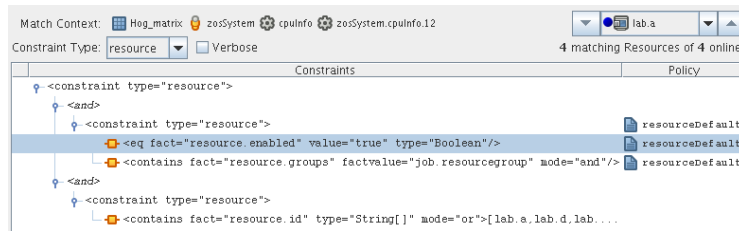
4.1.4 The Capable Resources Summary

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.

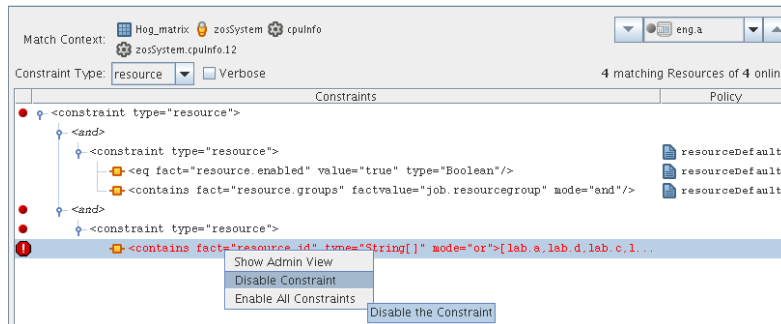
4.1.5 The Constraints Column of the Constraints Table View

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.

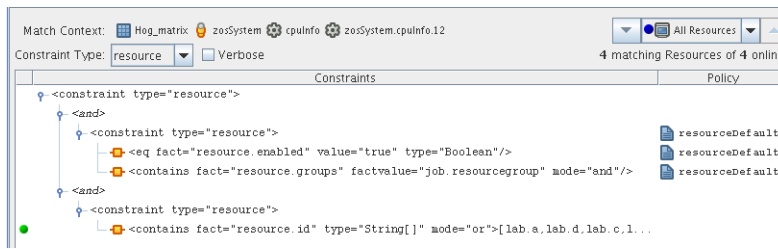


- ◆ red dot icon: The constraint does not pass the match. The figure below shows that the resource `eng.a` cannot run the job because its constraints do not match.



- ◆ 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.



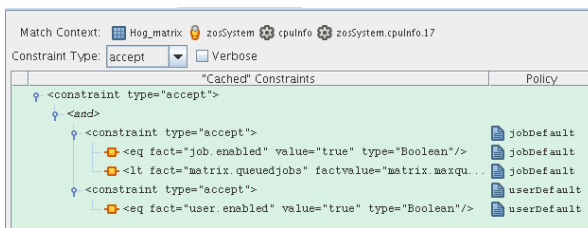
If you right-click a constraint in the table, a popup menu with three options is displayed. This menu lets you change the status of the constraint.

- ◆ **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, a condition that can prove 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 has completed. Their purpose is to help you debug the policy.

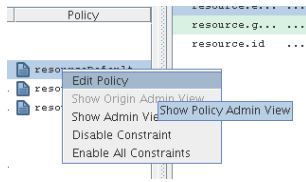
Figure 4-5 Cached Constraints Displayed in the Constraints Table View



4.1.6 The 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 popup menu for constraint column does.

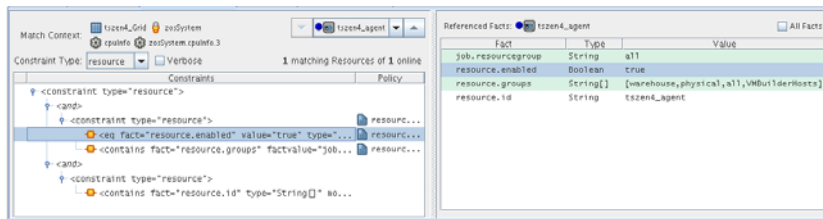
Figure 4-6 The Popup Menu Launched from the Policy Column



4.2 The Facts Table View

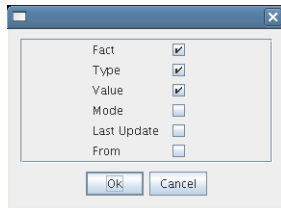
The Facts Table view displays the facts referenced in the Constraint Tree view for a specified Resource. Selecting a fact in the Constraint tree automatically selects that fact in the table.

Figure 4-7 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 4-8 Menu Used to Select the Columns Displayed in the Facts Table View of the Policy Debugger



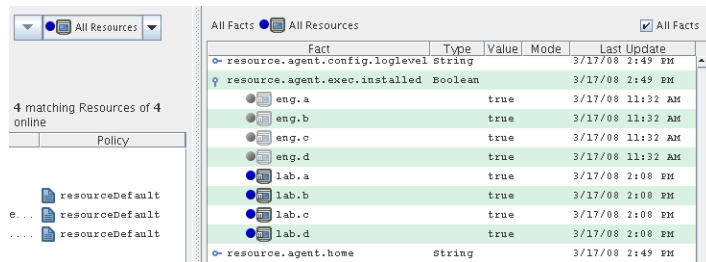
- ◆ [Section 4.2.1, “The All Facts Check Box,” on page 56](#)

4.2.1 The 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 4.1.1, “The Match Context Area,” on page 52](#)) 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 4-9 All Facts Check Box Selected with All Resources in Match Context



4.3 Policy Debugger Use Cases

This section includes several use cases that show how the Policy Debugger works and how to use it. The following use cases are included:

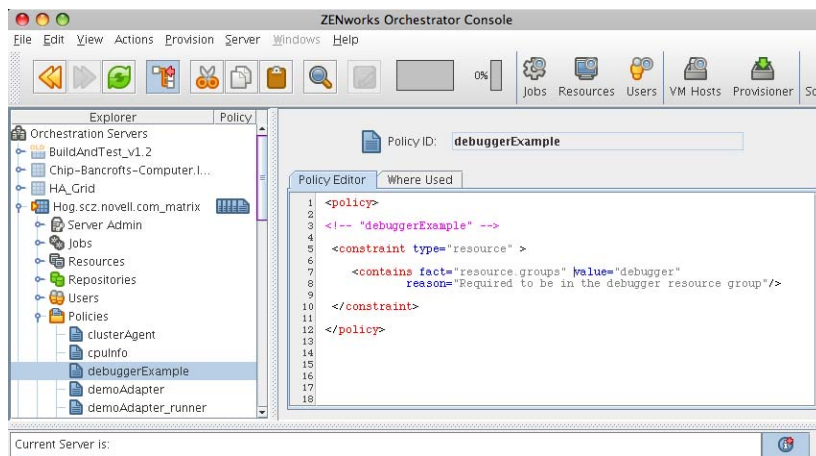
- ♦ Section 4.3.1, “Use Case 1: Determining Why a Job is in a Waiting State,” on page 57

4.3.1 Use Case 1: Determining Why a Job is in a Waiting State

The objective of this use case is to run a job that sits 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`.

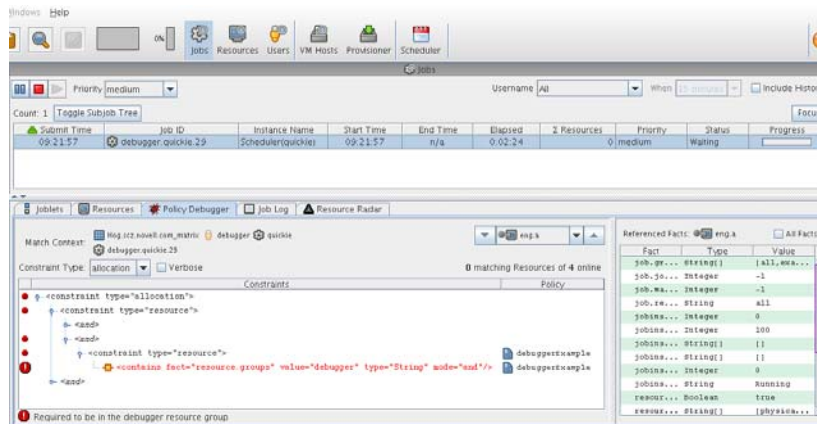
Use the following steps to recreate the use case.

- 1 In the Orchestrator Development Client, create a user named `debugger`.
- 2 Deploy `quickie.job` from the `/examples` directory.
- 3 In the Orchestrator Development Client, create a schedule named `quickie`, specifying the `quickie` job and the `debugger` user.
- 4 In the Orchestrator Development Client, 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 Orchestrator Development Client, associate the `debuggerExample` policy to the `quickie` job.

- 6 In the Job Scheduler view of the Orchestrate Development Client, select the `quickie` schedule, then click *Run Now* to run the `quickie` schedule.
- 7 In the Job Monitor view of the Orchestrate Development Client, 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 a few 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 on the blocking constraint and select *Disable Constraint*.

The Explorer Tree

5

The PlateSpin® Orchestrator Development Client from Novell® lets you visualize the model maintained by the Orchestrator Server used to make Virtual Machine (VM) provisioning decisions. The left pane of the Orchestrator Development Client 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. When you navigate to these objects, you can edit their attributes and view more detail about their configurations. The two objects highest in the tree are [The Orchestrator Server Object \(page 59\)](#) and [The Server Admin Object \(page 67\)](#). Each object in the Explorer Tree is referred to as a “Grid object.” Most Grid objects can also be associated with one or more containers called Groups.

The primary Grid objects include the following:

- ♦ **Jobs:** These objects are deployed to the Orchestrator Server to automate processes, such as on-demand provisioning. Jobs consist of JDL scripts(s) and might have one or more Policies associated with them. For more information about Jobs, see [Section 5.3, “The Job Object,” on page 68](#).
- ♦ **Resources:** These objects represent physical or virtual machines managed by PlateSpin Orchestrator. If a resource is running the PlateSpin Orchestrator Agent, that resource can be scheduled for remote execution of a job. For more information about resources, see [Section 5.4, “The Resource Object,” on page 81](#).
- ♦ **VM Hosts:** These objects represent a VM host technology (for example, Xen*, Hyper-V, and so on) running on a physical resource. VM host objects can be used when making provisioning decisions to a resource. For more information about VM hosts, see [Section 5.5, “The VM Host Object,” on page 104](#).
- ♦ **Repositories:** These objects represent the physical or virtual storage that is accessible for VM images or other use. For more information about repositories, see [Section 5.6, “The Repository Object,” on page 109](#).
- ♦ **Users:** These objects represent the individual accounts that are allowed to connect to the PlateSpin Orchestrator Server. Administrator users are also allowed to connect by using the `zosadmin` command line and the PlateSpin Orchestrator Development Client user interfaces. For more information about Users, see [Section 5.7, “The User Object,” on page 117](#).

For information on creating individual user login accounts for VM operators, see [“Adding User Logins for VM Operators”](#) in the *PlateSpin Orchestrator 2.0 VM Client Guide and Reference*.

Grid objects can be visualized in various ways. One of the most important of these depicts the health of the object. For more information, see [Appendix A, “Grid Object Health Monitoring,” on page 133](#).

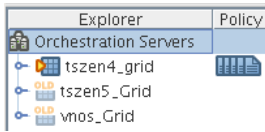
Additional objects, such as Policies, Public JDL Libraries, Computed Facts, Events, and Metrics are also displayed in the Explorer panel. To learn more about these other objects, see [Section 5.8, “Other Displayed Objects,” on page 124](#).

5.1 The Orchestrator Server Object

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

The PlateSpin Orchestrate Development Client is “version aware.” When the Orchestrate Development Client is launched or when server discovery is manually run, the client recognizes both current PlateSpin Orchestrate installations and old installations of discovered servers and displays their icons accordingly. This visual cue helps you to recognize when older Orchestrate Servers need to be upgraded.

Figure 5-1 Current and “Old” Server Objects



The tool tip for a Orchestrate 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 Orchestrate 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 the policy icon(s). From there, you can select a policy to open in the Policy Editor. For more information about policies, see [Section 5.8.1, “The Policy Object,”](#) on page 125.

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 5.1.1, “The Orchestrate Server Info/Configuration Tab,”](#) on page 60
- ◆ [Section 5.1.2, “The Orchestrate Server Authentication Tab,”](#) on page 65
- ◆ [Section 5.1.3, “The Orchestrate Server Policies Tab,”](#) on page 67
- ◆ [Section 5.1.4, “The Orchestrate Server Constraints/Facts Tab,”](#) on page 67

5.1.1 The Orchestrate Server Info/Configuration 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 server.

- ◆ [“Server/Cluster”](#) on page 60
- ◆ [“Data Grid Configuration”](#) on page 61
- ◆ [“Security/TLS Configuration”](#) on page 62
- ◆ [“Agent/User Session Configuration”](#) on page 63
- ◆ [“Audit Database Configuration”](#) on page 64
- ◆ [“job.limits”](#) on page 64

Server/Cluster

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

Is Master Server: This check box in non High Availability cluster configurations. It is unchecked if the server is not the Master Server in a High Availability cluster configuration.

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

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

Cluster Addresses: This list shows the hostname(s) or IP Address(es) associated with a Orchestrate Server when it configured 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 PlateSpin Orchestrate in a High Availability environment, see the [*PlateSpin Orchestrate 2.0 High Availability Configuration Guide*](#).

Data Grid Configuration

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: This field sets the location of the PlateSpin Orchestrate datagrid in the file system. For example, you might change this location to use a different file system mount point (recommended when there is a lot of datagrid I/O).

Cleanup Interval: This is the interval at which the Orchestrate Server scans through 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 to disable the checking.

Default Multicast Rate: This field 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: This field sets the maximum data rate in bytes per second that a client can specify for a multicast file transfer.

Selected Interfaces: This field names 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: This field list lists the network interfaces that are available on the local machine for multicasting.

NOTE: The property is “read-only” and is provided for your information.

Multicast Metrics: 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 due to 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 due to 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 select to clear the total metrics in order to begin monitoring multicast statistics from a new point in time.

Security/TLS Configuration

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

TLS On Agent: 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 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 due to 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`

This level specifies that the server defaults to unencrypted communication, but that the agent can optionally enable encryption.

This is the default setting for the Orchestrate 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 Orchestrate 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 are 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 due to legal or policy restrictions, or where the performance benefits of disabling encryption outweigh security concerns.

- ◆ `Allow TLS on the 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 Orchestrate 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 user can optionally disable encryption.

- ◆ `Make TLS mandatory on the clients`

The Orchestrate 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 are protected from tampering or interception.

TLS Address: This is 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 host name, then the Orchestrate Server listens on all available network interfaces. The default is `*:8101`, which causes the Orchestrate Server to listen for encrypted sessions on all available interfaces on the system.

Agent/User Session Configuration

When nodes (agents) and users log on to the Orchestrate Server, they establish a session context 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:** 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, 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 keep alive “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:** This is the idle timeout (in seconds) for user sessions. If a user’s session encounters no message traffic or requests in either direction for time, then 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 keep alive “ping” in order to detect hung/stalled network connections. This allows the agent to recover from hung connections and to transparently reconnect an 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 long-lived connection.

Audit Database Configuration

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:** 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:** When selected, this indicates that the driver is successfully connected.
- ◆ **Connect (button):** Click to connect using 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.

job.limits

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

- ◆ **max.active.jobs:** Set a (global default) limit on the number of active jobs.

The Orchestrate 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`, below.

- ◆ **max.queued.jobs:** Set a (global default) limit on the number of queued jobs.

This value is similar to `max.active.jobs` (see above) but it is used in the `accept` constraint and limits the number of jobs sitting in a queue waiting to be started. Therefore, the maximum jobs that can be present on an Orchestrate Server is `max.active.jobs + max.queued.jobs`. New jobs are not be accepted by the server if, when added, they would exceed this total.

- ♦ **job.finishing.timeout:** Set a (global default) limit on the timeout for job completion.

This value represents the number of seconds that the Orchestrate Server allows a job to execute it's `job_cancelled_event()` (if defined) before forcibly aborting the job. This prevents jobs from potentially hanging during cancellation.

5.1.2 The Orchestrate Server Authentication Tab

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

- ♦ “Resources” on page 65
- ♦ “Users” on page 65

Resources

The resources in a PlateSpin Orchestrate grid are actually PlateSpin Orchestrate Agents that authenticate or “register” with the PlateSpin Orchestrate Server.

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

Users

Only authenticated users can log into the PlateSpin Orchestrate 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 PlateSpin Orchestrate Server to automatically register users when they first connect to the Orchestrate Server.

Enable LDAP

Select this check box if you want the Orchestrate Server to authenticate users externally using an LDAP server. Additional LDAP-related configuration fields are displayed when you select check box.

Administrators

The Administrators list specifies the group names whose membership includes PlateSpin Orchestrate 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.

Active Directory Service Settings

If you select Active Directory Service in the Server Type drop down list, the following settings are available:


Directory Name: Enter the name of the Active Directory Service server.

Servers: This property is 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 are in use, however, the host name 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 section are for more selective ADS authentication.

- ◆ **SSL:** Selecting this option (assuming that the accompanying *Start TLS* check box is not also selected and also assuming that the ADS server’s SSL certificate has been installed on the PlateSpin Orchestrate Server JVM) securely connects to the ADS server using SSL encryption.

The older style LDAP protocol (`ldaps://`) is used for the connection.

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

Generic Settings

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

Base Domain Name: Specifies 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: Specifies the attribute on a user’s entry that identifies his or her login account name. For ADS servers, this attribute is `sAMAccountName`.

User Filter: Specifies the name of the filter to be used in the lookup for the user’s LDAP distinguished name.

User Prefix: Specifies the prefix used to define the LDAP subtree within the BaseDN tree that contains user accounts. If you leave this property blank, the Orchestrate Server uses the BaseDN.

For ADS, this prefix is `cn=Users`.

Group Attribute: Specifies the attribute of a group entry describing the login name of that group.

Group Filter: Specifies 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}`.

Not used for Active Directory authentication.

Group Prefix: Specifies the prefix used to define the LDAP subtree within the BaseDN tree that contains group accounts.

Not used for Active Directory authentication.

Group DNA Attribute: Specifies 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: Specifies the attribute of a group entry where subgroups can be queried.

5.1.3 The Orchestrate Server Policies Tab

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 icon.

5.1.4 The Orchestrate Server Constraints/Facts Tab

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. In essence, by building, deploying, and running jobs on the PlateSpin Orchestrate Server, you can individually change the functionality of any and all system resources by managing an object’s facts and constraints. The Orchestrate 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 edited (that is, using the “pencil” icon) in order to view their value(s) but changes cannot be made.

5.2 The Server Admin Object

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

5.3 The Job Object

A job is deployed to the Orchestrate 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(s) 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 PlateSpin Orchestrate Server itself and schedules work to run on one or more managed resources that are running the PlateSpin Orchestrate Agent. The logic that is dispatched and run on the managed resources is called a joblet. A job may or may 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 PlateSpin Orchestrate Server dispatches joblets to resources in the grid where the work is done.

The Job object also contains Facts with attributes that are used for job and joblet control. Policies associated with the job also control the job. The Orchestrate Development Client 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 Orchestrate Development Client:

- ♦ [Section 5.3.1, “Job Groups,” on page 68](#)
- ♦ [Section 5.3.2, “The Job Info/Groups Tab,” on page 68](#)
- ♦ [Section 5.3.3, “The JDL Editor Tab,” on page 78](#)
- ♦ [Section 5.3.4, “The Job Library Editor Tab,” on page 79](#)
- ♦ [Section 5.3.5, “The Job Policies Tab,” on page 80](#)
- ♦ [Section 5.3.6, “The Job Constraints/Facts Tab,” on page 81](#)



5.3.1 Job Groups

Any group object displayed in the Explorer panel represents a collection of similar object types. Groups can also be created automatically, as in the case when a provisioning adapter (PA) discovers a local repository on a VM host. For example, the `xen30` PA, 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 Development Client, 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*.

5.3.2 The Job Info/Groups Tab

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.

- ♦ [“Info” on page 69](#)
- ♦ [“Groups” on page 78](#)

NOTE: Whenever you make changes to any Grid object, that object’s icon is overlaid with the write icon , signifying that the object has been altered. If you want to save the changes you have made, you need to click the Save icon  on the Development Client toolbar.

Info

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

- ♦ “Show Inherited Fact Values Check Box” on page 69
- ♦ “Job Control Settings” on page 69
- ♦ “Joblet Control Settings” on page 72
- ♦ “Automatic Resource Provisioning Settings” on page 74
- ♦ “Resource Preemption Settings” on page 75
- ♦ “Job Counts” on page 75
- ♦ “Job History” on page 76

Show Inherited Fact Values Check Box

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).

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 (that is, its value is “true”), the job is enabled (that is, it 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 (that is, its value is “true”), the job can be viewed in the Development Client, by using command line queries, or in the Orchestrate Server Portal. Deselecting this check box does not keep the job from running.

In the Fact Editor, this 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 (that is, its value is “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: This drop-down list lets you choose the job type that applies to this job. This setting is optional and is leveraged by the server 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: Enter 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 (that is, its value is “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" />
```

Queue Type: This drop-down list 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 quality 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 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.queue.time.average` fact.

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

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

Job Queued Timeout: Enter 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.queued.timeout`:

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

Resource Match Cache TTL: This value 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 zero (<0) to disable caching.

In the Fact Editor, this fact is listed as `job.cache.resource.matches.ttl`:

```
<fact name="job.cache.resource.matches.ttl" value="30" type="Integer" />
```

Preemptible: This check box is not selected by default. When it is selected (that is, its value is “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 (that is, its value is “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: This value 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: This value 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.max.failures`:

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

Max Node Failures: This value specifies the number 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: This value specifies the absolute maximum number of resources that you want the job to use at one time. PlateSpin Orchestrate 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: This value specifies the absolute maximum number of joblets that you want the job to have running at one time. PlateSpin Orchestrate 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: This value specifies the absolute 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: This field displays ranking specification used to select suitable resources. 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. In this dialog box 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 (that is, its value is `"true"`), you specify that the Grid objects that this job modifies are persisted at the end of 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: This value specifies the amount of time (in seconds) you want the Orchestrate Server to wait until cancelling 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: This value specifies the number of joblet retries (of any type) to be attempted before the Orchestrate 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.

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

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

Retry Limit (Forced): This value specifies the number of forced joblet retries (that is, requested by the joblet to run on another resource) to be allowed before the Orchestrate 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.forced`:

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

Retry Limit (Unforced): This value specifies the number of unforced joblet retries to be allowed before the Orchestrate 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): This value specifies the number of joblet retries caused by unexpected resource disconnect to be allowed before the Orchestrate 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): This value specifies the number of joblet retries caused by server-initiated joblet timeout to be allowed before the Orchestrate 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.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 (that is, its value is “true”), you specify that you want the system to immediately retry a joblet rather than waiting until all others 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: This value 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 (that is, its value is “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: From the drop-down list, you can 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 Orchestrate 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 Orchestrate Server user who matches the PlateSpin Orchestrate 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 Orchestrate Agent (also known as the “Node User”). It does not run as the OS user whose username matches the PlateSpin Orchestrate 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: This value 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: This value 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: This value specifies the maximum number of provision failures resources 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: This field displays ranking specification used to select suitable resources to automatically provision. 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.provision.rankby">
  <array type="String">
    </array>
</fact>
```

You can edit this array by clicking the  button to open the Attribute element values dialog box. In this dialog box you can add or remove fact specifications to the array of element choices.

Host Selection Strategy: This drop-down list lets you choose the type of strategy you want to use in finding a host for any automatically provisioned resource. The choices include:

- ♦ **queue:** The *queue* option 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 completes.
- ♦ **immediate:** The *immediate* option 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: This field displays ranking specification used to select suitable jobs to automatically preempt on a resource. 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.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. In this dialog box you can add or remove fact specifications to the array of element choices.

Job Counts

Total Instances: This field displays the total number of job instances of this type that exist in the PlateSpin Orchestrate system.

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

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

Active Instances: This field displays the total number of job instances of this type that are in a queued state in the PlateSpin Orchestrate system.

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

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

Queued Instances: This field displays the total number of job instances of this type that are active in the PlateSpin Orchestrate 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: This drop-down list 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: This drop-down list 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) This field displays the total number of job instances (including those denied by “accept” constraints) of this job that have ever been initiated on this PlateSpin Orchestrate 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) This field 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) This field 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) This field 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: This field 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: This field 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: This field 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: This field 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: This field 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: This field 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: This field 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 really a relative measure and approximates to a second of real processing time of a 2Ghz Pentium* class Intel* processor.

Average Grid Time: This field 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: This field 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.queueetime.total`:

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

Average Queue Time: This field 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.queue.time.average`:

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

Average Sample Size: This field 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.sample.size`:

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

NOTE: Similar to a moving average, a trailing average is the mean average measured over the last x datapoints.

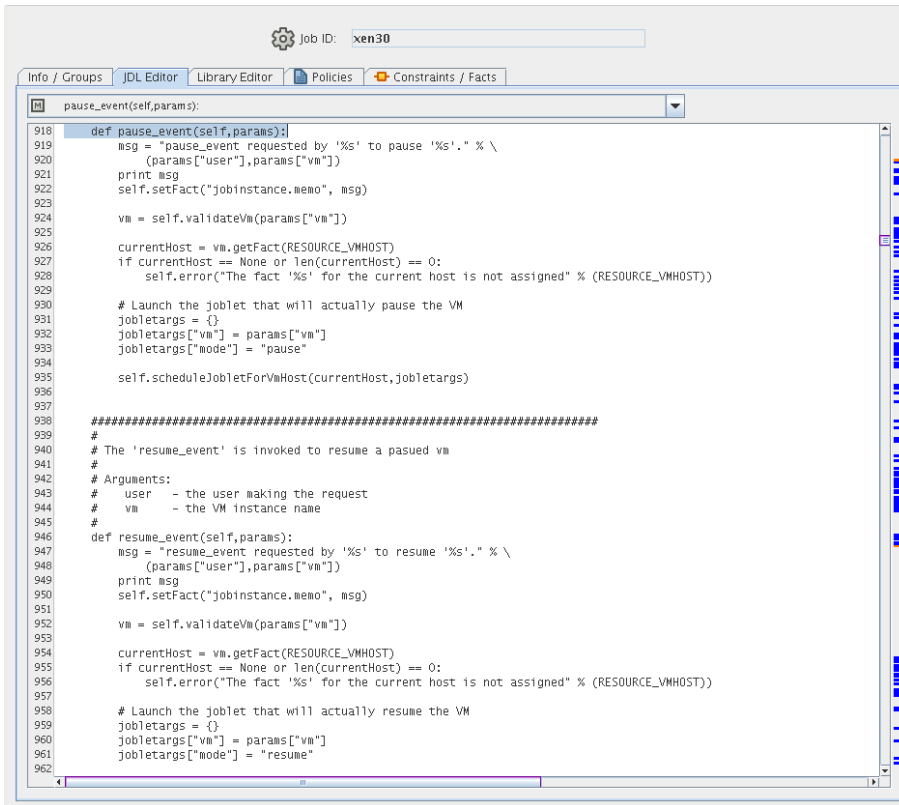
Groups

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.

5.3.3 The JDL Editor Tab

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-based script and 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 5-2 The JDL Editor

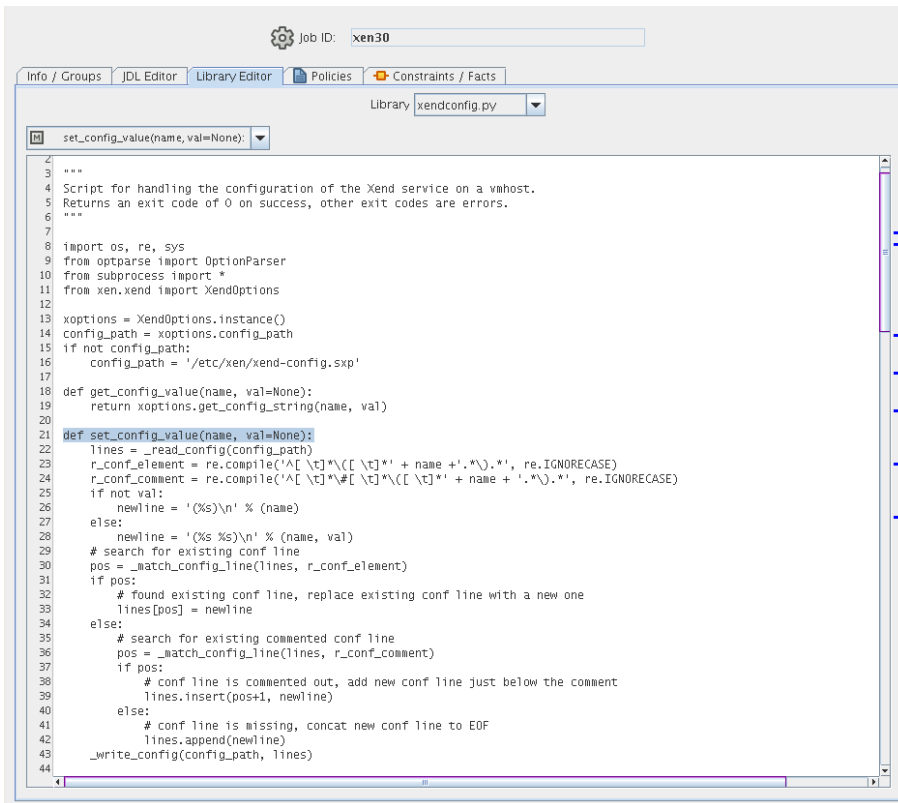


A drop-down list at the top of the editor includes the Java classes and their methods that are bookmarked in the code. Select any of these to go to the location in the code where they are invoked. Clickable, colored blocks on the editor scroll bar perform a similar bookmarking function.

5.3.4 The Job Library Editor Tab

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 Orchestrate Server for handling job functions.

Figure 5-3 The Job Library Editor



There are two drop down lists located at the top of the Library Editor view. The first labeled “Library” lists the different libraries for the job, and the second lists 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.

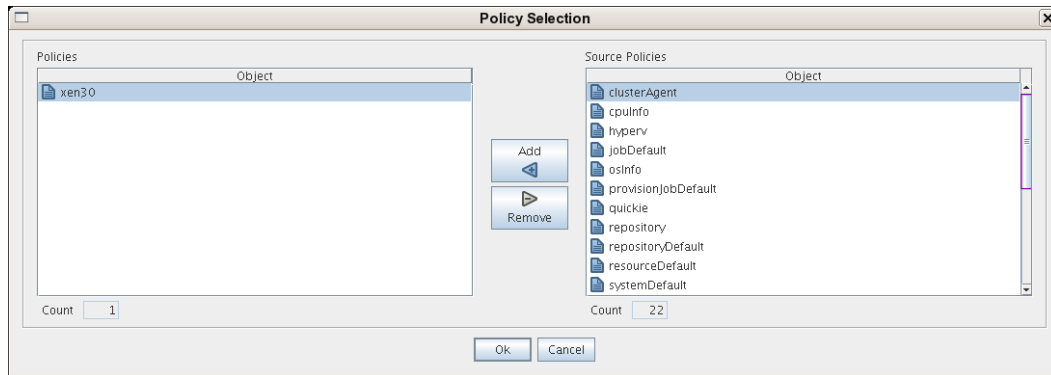
5.3.5 The Job Policies Tab

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 using the Policy Grid object. for more information see [Section 5.8.1, “The Policy Object,”](#) on page 125.

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 5-4 The Policy Selection Dialog Box



5.3.6 The Job 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 changing the policy constraints and fact values for a job, you can change the behavior of the job and how the PlateSpin Orchestrate Server allocates available system resources to it. The Orchestrate 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 (that is, using the edit “pencil” icon) but changes cannot be made.

5.4 The Resource Object

A Resource object in the Explorer Panel represents a physical or virtual machine that is managed by PlateSpin Orchestrate. If a resource is running the PlateSpin Orchestrate 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 view and the accompanying Admin view of the Orchestrate Development Client:

- ◆ [Section 5.4.1, “Resource Groups,” on page 81](#)
- ◆ [Section 5.4.2, “The Resource Info/Groups Tab,” on page 82](#)
- ◆ [Section 5.4.3, “The Provision Info Tab,” on page 102](#)
- ◆ [Section 5.4.4, “The Resource Log Tab,” on page 103](#)
- ◆ [Section 5.4.5, “The Resource Policies Tab,” on page 103](#)
- ◆ [Section 5.4.6, “The Resource Health Debugger Tab,” on page 103](#)
- ◆ [Section 5.4.7, “The Resource Constraints/Facts Tab,” on page 104](#)



5.4.1 Resource Groups

Any group object displayed in the Explorer panel represents a collection of similar object types. Groups can also be created automatically, as in the case when a provisioning adapter (PA) discovers a local repository on a VM host. For example, the `xen30` PA, 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 Development Client, 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*.

5.4.2 The Resource Info/Groups Tab

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, that object's icon is overlaid with the write icon , signifying that the object has been altered. If you want to save the changes you have made, you need to click the Save icon  on the Development Client toolbar.

- ◆ [“Info” on page 82](#)
- ◆ [“Groups” on page 102](#)

Info

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

- ◆ [“Show Inherited Fact Values Check Box” on page 82](#)
- ◆ [“Resource Information” on page 82](#)
- ◆ [“VM Host Info” on page 86](#)
- ◆ [“Virtual Machine Configuration” on page 87](#)
- ◆ [“Provisioning Information” on page 91](#)
- ◆ [“OS Information” on page 96](#)
- ◆ [“CPU Information” on page 98](#)
- ◆ [“Memory Information” on page 98](#)
- ◆ [“Disk/Network Information” on page 99](#)
- ◆ [“Agent Information” on page 99](#)
- ◆ [“Agent Configuration” on page 100](#)
- ◆ [“Installed Components” on page 102](#)

Show Inherited Fact Values Check Box

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).

Resource Information

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.

Resource Type: This drop-down list 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 a 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 (that is, its value is “true”), the resource is enabled (that is, it is 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 (that is, its value is “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 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 133](#).

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

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

Shutting Down: (Read Only) When this check box is selected (that is, its value is “true”), the node is attempting to shut down, pause, or suspend and does not want to 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 Orchestrate Agent. Often, the resource ID and the hostname are 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 Orchestrate 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 PlateSpin Orchestrate Agent on this node to use for authentication to the PlateSpin Orchestrate 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 Orchestrate 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: This drop-down list 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) This value specifies 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) This value specifies the percentage of CPU utilization currently used by the resource.

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

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

Joblet Slots: This value specifies 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: This value specifies 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) This value specifies 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) This field displays 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="7/23/09 5:02 PM" type="Date" />
```

Total Joblets Started: (Read Only) This field displays 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) This field displays 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) This field displays the total number of joblets that have been cancelled 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) This field displays 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) This field displays 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) This field displays 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) This field displays 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) This field displays 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 (that is, its value is “true”), you specify that this resource is a provisionable type. On a VM resource and a VM template resource are currently provisionable.

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


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

VM Host Info

VM Host Containers: This field displays 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. In this dialog box you can add or remove VM host containers to the array of element choices.

VM Host Repositories: This field displays 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. In this dialog box 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: This check box is not selected by default. When it is selected (that is, its value is “true”), the VM is currently under construction (that is it is 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: This value specifies the vendor name of the hypervisor that has provided 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: This value specifies 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: This drop-down list 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="vmh1s1esx" type="String" />
```

Storage Location in Repository: This value specifies the file system location (either absolute or relative to `repository.location`) where the VM files are located.

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

```
<fact name="resource.vm.basepath" value="/var/lib/xen/images/win2kbuild" type="String" />
```

Virtual Disks Layout: This visual map specifies the virtual disks that make up this VM.

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

```
<fact name="resource.vm.vdisks">
  <list>
    <listelement>
      <dictionary>
        <dictelement key="moveableprefix">
```

```

    <string>file:</string>
  </dictelement>
  <dictelement key="vdev">
    <string>hda</string>
  </dictelement>
  <dictelement key="mode">
    <string>w</string>
  </dictelement>
  <dictelement key="moveable">
    <boolean>true</boolean>
  </dictelement>
  <dictelement key="size">
    <integer>4096</integer>
  </dictelement>
  <dictelement key="location">
    <string>file:/var/lib/xen/images/win2kbuild/disk0</string>
  </dictelement>
  <dictelement key="repository">
    <string>vmh1slesx</string>
  </dictelement>
</dictionary>
</listelement>
</list>
</fact>

```


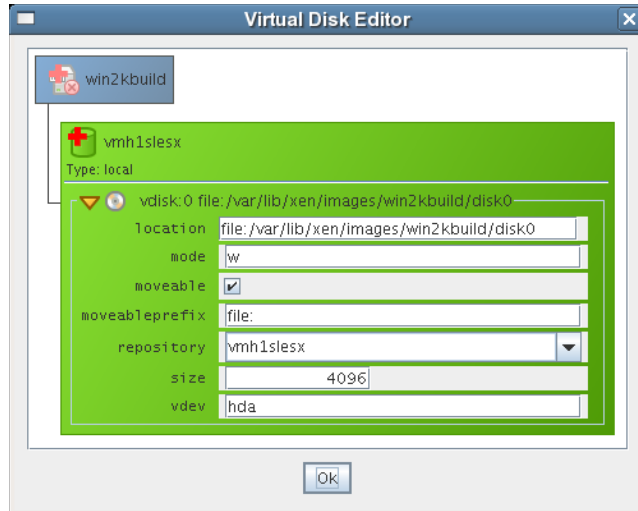
You can edit these dictionary keys by clicking the  button to open the Virtual Disk Editor dialog box and then expanding the map to open the details of the dictionary.

Figure 5-5 *The Virtual Disk Editor*



The editable dictionary keys (String values) include the following:

- ◆ **repository:** The optional repository housing disk.
- ◆ **location:** The URL referencing disk image.
- ◆ **size:** The optional size (measured in Megabytes) of the virtual disk.
- ◆ **moveable:** The true or false value showing whether the disk can be moved by PlateSpin Orchestrate.

VM Files: This list includes the files (described by file types and file paths) that make up this VM.

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>
```

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`.

You can edit this dictionary by clicking the button to open an array editor. In this dialog box you can add or remove file types and paths to the array of element choices.

Required Virtual Memory: This value specifies 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" />
```

Host CPU Architecture: This value specifies 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 (that is, its value is “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: This value specifies 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 would get 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: This value specifies 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: This value specifies 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" />
```

Prevent Disk Relocation: This check box is not selected by default. When it is selected (that is, its value is “true”), PlateSpin Orchestrate cannot move VM disks and thus consideration of other potential VM hosts.

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

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

Max VMs Per Host: This value specifies the number of instances of this VM image that are allowed on a VM host, configurable based on the known capacity of the VM host/hypervisor.


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

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

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. In this dialog box you can add or remove ranking specifications to 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 using the PlateSpin Orchestrate 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 the dictionary by clicking the  button to open an attribute editor where you can add or remove dialog box and then expanding the map to open the details of the dictionary.

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: This drop-down list 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="xen30" type="String" />
```

VM Warehouse Version: This value specifies the warehouse version number for this VM. This fact is no longer used.

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

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

VM Warehouse GUID: This value specifies the warehouse ID for this VM. This fact is no longer used.

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

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

Provisioned Instances: This value specifies 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: This value specifies 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: This list includes 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. In this dialog box you can add or remove instance IDs to the array of choices.

Max Provisioned Instances: (For VM templates only) This value specifies 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: This value specifies the maximum amount of time (measured in seconds) allowed for this VM to shut down and for the Orchestrate Agent to disconnect from the Orchestrate 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: This value specifies the maximum amount of time (measured in seconds) allowed for a VM instance to be idle before relaxing reservation policy or shutting down. Behavior depends on mode, and can be overridden by the provision 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: This value specifies 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: This value specifies 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: This drop-down list includes the names of VM hosts that you can choose from 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 (that is, its value is “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: This value specifies 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: This value specifies 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: This value specifies 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_xen30" type="String" />
```

Current Status: (Read Only) This value specifies 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) This value specifies 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) This value specifies 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="8/24/09 4:36 PM" type="Date" />
```

Start Time: (Read Only) This value specifies 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) This value specifies 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) This value specifies 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) This value specifies 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.xen30.74239" type="String" />
```

Automatic Provision: (Read Only) This check box is not selected by default. When it is checked (that is, its value is “true”), the resource was cloned or provisioned automatically and so 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 (that is, its value is “true”), you specify that the provisioned resource’s state needs to be resynchronized 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: 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:** This value specifies the host name of a new VM. Enter “*” to indicate that the VM ID is to be used rather than the host name 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" />
```

Windows Sysprep Config: Although the fields in this section can be defined with a Fact Editor, the entire section of facts is non-functional and is not currently supported.

Autoprep Network Adapter 0: This section of the Info/Groups page displays the configuration information for a network adapter (0) you can prepare for autoprep work. A new VM resource or VM template resource does not have these facts defined. When you click *Define*, a Fact Editor is enabled where you can define the facts for the adapter.

The section includes the following settings/facts:

- ♦ **MAC Address:** The value you specify in this field is the name for each NIC that represents the MAC address of the interface. Specify an asterisk (*) to generate a new MAC address. If the value is not set, the current configuration is used.

In the Fact Editor, this fact is listed as

```
resource.provisioner.autoprep.adapters[0].MACAddress:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].MACAddress"
value="dadf" type="String" />
```

- ♦ **Use DHCP:** When this check box is selected (that is, its value is “true”), the new VM retrieves its network settings from a DHCP server and any adapter settings are ignored. If the check box is not selected (that is, its value is “false”), any required adapter settings must be defined.

In the Fact Editor, this fact is listed as

```
resource.provisioner.autoprep.adapters[0].UseDHCP:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].UseDHCP"
value="true" type="Boolean" />
```

- ♦ **IP Address:** This field displays the IP address for the adapter.

In the Fact Editor, this fact is listed as

```
resource.provisioner.autoprep.adapters[0].IPAddress:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].IPAddress"
value="fddfasdf" type="String" />
```

- ♦ **Subnet Mask:** This field displays the subnet mask for the adapter.

In the Fact Editor, this fact is listed as


```
resource.provisioner.autoprep.adapters[0].subnetMask:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].subnetMask"
value="fadsafdasdf" type="String" />
```

- ♦ **Gateway IP Address:** This field displays a list of the internet gateway IP addresses available to the interface.

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

```
<fact name="resource.provisioner.autoprep.adapters[0].Gateways">
  <array>
    <string>afdasads</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.

- ♦ **DNS from DHCP:** (Optional. SUSE VM only) When this check box is selected (that is, its value is “true”), the SUSE VM is configured to retrieve its DNS server settings from DHCP.

In the Fact Editor, this fact is listed as


```
resource.provisioner.autoprep.adapters[0].DNSFromDHCP:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].DNSFromDHCP"
value="true" type="Boolean" />
```

- ♦ **DNS Server IP Addresses:** This field displays a list of DNS servers for name lookup.

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

```
<fact name="resource.provisioner.autoprep.adapters[0].DNSServers">
  <array>
    <string>adfadsasdf</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 server IP address or change its order in the array of element choices.

- ◆ **DNS Domain:** (Windows only) This field displays the name of the adapter's domain.

In the Fact Editor, this fact is listed as

```
resource.provisioner.autoprep.adapters[0].DNSDomain:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].DNSDomain"
value="afds" type="String" />
```

- ◆ **Primary WINS Server:** This field displays the name of the adapter's primary WINS server.

In the Fact Editor, this fact is listed as

```
resource.provisioner.autoprep.adapters[0].primaryWINS:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].primaryWINS"
value="dfasfasddasf" type="String" />
```

- ◆ **Secondary WINS Server:** This field displays the name of the adapter's secondary WINS server.

In the Fact Editor, this fact is listed as


```
resource.provisioner.autoprep.adapters[0].secondaryWINS:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].secondaryWINS"
value="fdasadsfasd" type="String" />
```

- ◆ **DNS Suffixes:** This field displays a list of the suffixes associated with this adapter that are appended to the name for lookup.

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

```
<fact name="resource.provisioner.autoprep.adapters[0].DNSSuffixes">
  <array>
    <string>afjdkd</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.

- ◆ **NetBIOS:** This drop-down list box includes the NetBIOS options for this VM. Options include *EnableNetBIOSviaDhcp*, *EnableNetBIOS*, and *DisableNetBIOS*.

In the Fact Editor, this fact is listed as

```
resource.provisioner.autoprep.adapters[0].netBIOS:
```

```
<fact name="resource.provisioner.autoprep.adapters[0].netBIOS"
value="EnableNetBIOS" type="String" />
```

Autoprep Network Adapter 1: This section of the Info/Groups page displays the configuration information for a network adapter (1) you can prepare for autoprep work. Definitions of its facts are similar to **Autoprep Network Adapter (0)**, except for its identification in the as Adapter [1] rather than Adapter [0].

OS Information

OS Name: (Read Only) This field displays 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="Linux" type="String" />
```

OS Version: (Read Only) This field displays 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" value="2.6.16.46-0.12-smp" type="String" />
```

OS Version String: (Read Only) This field displays the full identification string of the operating system as supplied by the vendor of the operating system. The information is not available until the `osinfo` system job runs. For a VM resource, this fact remains undefined.

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

```
<fact name="resource.os.version.string" value="Linux version 2.6.16.46-0.12-smp (geeko@buildhost) (gcc version 4.1.2 20070115 (prerelease) (SUSE Linux)) #1 SMP Thu May 17 14:00:09 UTC 2007" type="String" />
```

OS Architecture: This field displays the name of 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: This value indicates 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) This field displays the unique string identifier for each OS release.

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

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

OS Vendor Version: (Read Only) This field displays the vendor-defined version for the operating system (for example, *10* for SUSE Linux Enterprise Server 10).

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="SUSE Linux Enterprise Server 10 (i586)" type="String" />
```

OS File Path Separator: (Read Only) This field displays the character used by the resource operating system for file path separation.

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

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

CPU Information

Number of CPUs: (Read only) This field displays the number 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: (Read only) This field displays 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) This field displays 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 Number: (Read only) This field displays 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: This field displays 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

Virtual Memory: (Read only) This field displays 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) This field displays 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: (Read only) This field displays 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) This field displays 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: (Read only) This field displays 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) This field displays 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

Agent Version: (Read only) This field displays the PlateSpin Orchestrate 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) This field displays the name of the home directory of the PlateSpin Orchestrate 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) This field displays the version of the Java JVM currently in use by the PlateSpin Orchestrate 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) This field displays the version of the Java JVM runtime currently in use by the PlateSpin Orchestrate 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) This field displays the name of the vendor of the Java JVM currently in use by the PlateSpin Orchestrate 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) This field displays the path to the home directory of the Java JVM currently in use by the PlateSpin Orchestrate 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) This field displays the amount of memory (measured in Mb) available to the PlateSpin Orchestrate 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 (that is, its value is “true”), it indicates that the PlateSpin Orchestrate Agent installed on this resource is able to use enhanced exec features (as opposed to unsupported agent installs, for example, 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 (that is, its value is “true”), you specify that the agent is “clustered” on this VM resource; that is, 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

Gmond Port: This field specifies the port that the agent uses for gmond. Port 8649 is the default port. A setting of zero (90) or less means that the values is not read.

In the Fact Editor, this fact is listed as `resource.agent.config.gmond.port`:

```
<fact name="resource.agent.config.gmond.port" value="8649" type="Integer" />
```

Datagrid Cache TTL: This field specifies 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: This field specifies the amount of time (measured in minutes) that the Orchestrate 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: This field specifies 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 (that is, its value is “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 (that is, the `job.joblet.runtime` 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 (that is, its value is “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 (that is, its value is “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 not to 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: This drop-down list 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 (that is, its value is “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: This field displays 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. In this dialog box you can add or remove the application name or change its order in the array of element choices.

Groups

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.

5.4.3 The Provision Info Tab

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:** This page displays the log that includes historical details about the history of the provisioning of the VM.
- ♦ **Host Assignment Log:** This page displays the log of VM host assignment constraint errors of the last migration or provision action.
- ♦ **Autoprep Data:** You can use this page to help you debug the policy (and its resulting constraints and facts) associated with this VM.
- ♦ **Policy Debugger:** You can use this page to help you debug the policy (and its resulting constraints and facts) associated with this VM. For more information, see [Chapter 4, “The Policy Debugger,” on page 51](#).
- ♦ **Action History:** This page displays the log that includes historical details about the history of the provisioning of the VM. For more information, see [Appendix D, “Provisioning Actions and History,” on page 151](#).

5.4.4 The Resource Log Tab

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.

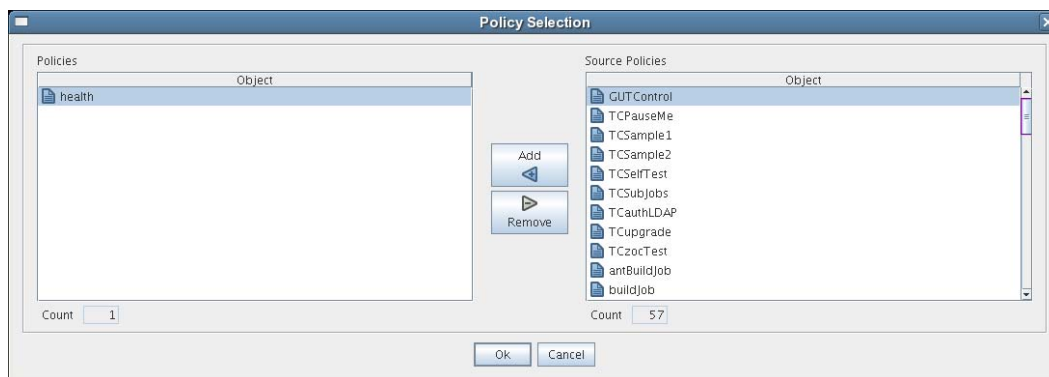
5.4.5 The Resource Policies Tab

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 using the Policy Grid object. For more information see [Section 5.8.1, “The Policy Object,” on page 125](#).

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 5-6 The Policy Selection Dialog Box



5.4.6 The Resource Health Debugger Tab

The Health Debugger is a common Admin view in the Development Client for most Grid objects. For information about this tool, see [Chapter 6, “The Health Debugger,” on page 127](#).

5.4.7 The Resource 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 changing the policy constraints and fact values for a job, you can change the behavior of the job and how the PlateSpin Orchestrate Server allocates available system resources to it. The Orchestrate 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 (that is, using the edit “pencil” icon) but changes cannot be made.

5.5 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.



This section includes the following information:

- ♦ [Section 5.5.1, “The VM Host Info Tab,” on page 104](#)
- ♦ [Section 5.5.2, “The VM Host Policies Tab,” on page 108](#)
- ♦ [Section 5.5.3, “The VM Host Health Debugger Tab,” on page 108](#)
- ♦ [Section 5.5.4, “The VM Host Constraints/Facts Tab,” on page 108](#)
- ♦ [Section 5.5.5, “The VM Host Action History Tab,” on page 108](#)

5.5.1 The VM Host Info Tab

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 repository.

- ♦ [“Show Inherited Fact Values Check Box” on page 104](#)
- ♦ [“VM Host Information” on page 104](#)
- ♦ [“Provisioning Adapter Config” on page 107](#)
- ♦ [“Guest VM Monitor Information” on page 107](#)

NOTE: Whenever you make changes to any Grid object, that object’s icon is overlaid with the write icon , signifying that the object has been altered. If you want to save the changes you have made, you need to click the Save icon  on the Development Client toolbar.

Show Inherited Fact Values Check Box

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).

VM Host Information

The VM Host Information panel on the Info page includes the following fields:

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

Physical Resource: (Read Only) this field displays 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" />
```

Enabled: This check box is selected by default. When it is selected (that is, its value is “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 (that is, its value is “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 (that is, its value is “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 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 133](#)

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

```
<fact name="vmhost.health" value="false" type="Boolean" />
```

Shutting Down: When this check box is selected (that is, its value is “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: 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 (that is, its value is “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 (that is, its value is “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: From the drop-down list, you can select 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: This value specifies 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 Virtual Memory: This field displays 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: This field that displays the list of repositories (that is, 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. In this dialog box 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. In this dialog box you can add, remove, or edit the resource groups (element values) in an array of choices.

Managing Job: In this field, you can specify 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 (that is, its value is “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" />
```

Provisioning Adapter Config

Adapter Job Name: From the drop-down list, you can select 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="esx" type="String" />
```

Username: In this field, specify 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: In this field, specify 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" />
```

Guest VM Monitor Information

Current VM Count: (Read Only) This field displays 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 Virtual Memory: This field displays 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: This field displays 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 and then by adding or removing the names in the dictionary.

Running VM Instances: (Read Only) This field displays a list of active VM instances.

Load Index (Slots): (Read Only) This field displays 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) This field displays 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" />
```

5.5.2 The VM Host 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.

5.5.3 The VM Host Health Debugger Tab

The Health Debugger is a common Admin view in the Development Client for most Grid objects. For information about this tool, see [Chapter 6, “The Health Debugger,” on page 127](#).

5.5.4 The VM Host 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 PlateSpin Orchestrate Server, you can individually change the functionality of any and all system resources by managing an object’s facts and constraints. The Orchestrate 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 (that is, using the edit “pencil” icon) but changes cannot be made.

5.5.5 The VM Host 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 VM provisioning actions performed on this Grid object.

The Orchestrate 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 icon in the toolbar to fetch and display fresh data.

For more details about the information listed on the Action History page, see [Section D.2.2, “Action History in Admin Views of the Development Client,” on page 152](#).

5.6 The Repository Object

Repositories are storage areas for VMimage 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"](#) on page 110.

The repository groups and their constituent repository objects are displayed in the Explorer panel and the accompanying Repository Admin View of the Development Client.

- ♦ [Section 5.6.1, "Repository Groups,"](#) on page 109
- ♦ [Section 5.6.2, "The Repository Info/Groups Tab,"](#) on page 109
- ♦ [Section 5.6.3, "The Repository Policies Tab,"](#) on page 116
- ♦ [Section 5.6.4, "The Repository Health Debugger Tab,"](#) on page 116
- ♦ [Section 5.6.5, "The Repository Constraints/Facts Tab,"](#) on page 117
- ♦ [Section 5.6.6, "The Repository Action History Tab,"](#) on page 117



5.6.1 Repository Groups

Any group object displayed in the Explorer panel represents a collection of similar object types. Groups can also be created automatically, as in the case when a provisioning adapter (PA) discovers a local repository on a VM host. For example, the Xen30 PA, 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 Development Client, 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*.

5.6.2 The 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.

- ♦ ["Info"](#) on page 110
- ♦ ["Best Practices for Entering Repository File Paths"](#) on page 115
- ♦ ["Groups"](#) on page 116

NOTE: Whenever you make changes to any Grid object, that object's icon is overlaid with the write icon , signifying that the object has been altered. If you want to save the changes you have made, you need to click the Save icon  on the Development Client toolbar.

Info

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

- ♦ [“Show Inherited Fact Values Check Box” on page 110](#)
- ♦ [“Repository Information” on page 110](#)
- ♦ [“SAN Adapter Configuration” on page 114](#)

Show Inherited Fact Values Check Box

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).

Repository Information

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: Enter information in this box that describes 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 (that is, its value is “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 (that is, its value is “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 (this follows the normal rules of policy inheritance). For more information, see [Appendix A, “Grid Object Health Monitoring,” on page 133](#)

In the Fact Editor, this 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 5-1 Repository Types and Descriptions

Repository Type	Description	Development Client 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)	A remote storage location that can be configured as a shared store for VM images (which can be moved) or a block-based disk used by a single VM. This repository can be configured as either iSCSI or Fibre Channel. using iSCSI Qualified Name, NPIV, or EMC. The modeling of SAN storage requires special setup, particularly when configured as shared storage where LUN masks are created for each of the hosts with visibility to the SAN.	
datagrid	The shared datagrid repository (named <code>zos</code>) is located on the Orchestrate Server and is accessible to all host servers in the datagrid. By default, each host server has access to the <code>zos</code> datagrid repository.	
virtual	Represents an externally managed virtual disk (for example, VMware Virtual Center).	

SAN ID: (SAN repositories only) This field displays 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: Enter the repository's logical root location in this field. 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 115](#).

Table 5-2 Repository Types and Root Location Examples

Repository Type	Root Location Examples
local	<ul style="list-style-type: none"> ◆ / (root) ◆ c:/vm

Repository Type	Root Location Examples
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. <ul style="list-style-type: none"> ◆ /u ◆ /mnt/myshreddisk
SAN (Storage Area Network)	not required
datagrid	grid:///vms
virtual	<ul style="list-style-type: none"> ◆ / (root) ◆ c:/vm

VM Config Search Path: Enter 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>
```

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 115](#).

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 5-3 Repository Types and VM Config Search Path Examples

Repository Type	VM Config Search Path Examples
local	<code>/etc/xen/vm</code>
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. <ul style="list-style-type: none"> ◆ <code>my_vms</code> ◆ <code>saved_vms</code> ◆ null (no path entry)
SAN (Storage Area Network)	not required
datagrid	<code>grid:///vms</code>
virtual	

Preferred Storage Path: Enter the relative path (from `repository.location`) where you want PlateSpin Orchestrate 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 115](#).

Table 5-4 *Repository Types and Preferred Storage Path Examples*

Repository Type	Preferred Storage Path Examples
local	<code>var/lib/xen/images</code> for Xen VMs
NAS (Network Attached Storage)	<code>my_vms</code>
SAN (Storage Area Network)	not required
datagrid	<code>grid:///vms</code>
virtual	

Capacity (Mb): Enter the maximum amount (in megabytes) of storage space available to VMs. 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): This is a non-editable field that displays the amount (in megabytes) 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): This is a non-editable field that displays the amount (in megabytes) 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" />
```

Efficiency: Enter an efficiency coefficient that PlateSpin Orchestrate 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 megabytes) to determine an efficiency score. A score of zero (0) means no cost (very efficient).


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

```
<fact name="repository.efficiency" value="1.0000" type="Real" />
```

Stored VMs: This field displays a list of 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>
```

You can edit this array by clicking the  button to open the Attribute element values dialog box. In this dialog box you can add, remove, or edit the VM host IDs (element values) in an array of VM host ID choices.

Compatible VM Hosts: This field displays a list of 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. In this dialog box you can add, remove, or edit the VM host IDs (element values) in an array of VM host ID choices.

Accessed By Provision Adapters: This field displays a list of 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. In this dialog box you can add or remove provisioning adapters to 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) Enter the name of the vendor of the SAN. This should be adapter specific, for example: *iqn, npiv, emc*. An empty field (that is, no value in the string) indicates bind/unbind is a noop (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" />
```

Best Practices for Entering Repository File Paths

Because of some limitations in the current (2.0.2) release of PlateSpin Orchestrate, we suggest that you use the following guidelines in scenarios where you need VM repositories.

- ♦ [“Creating a Repository to Use with New VMs” on page 115](#)
- ♦ [“Creating a Repository to Use with Existing VMs” on page 115](#)
- ♦ [“Creating a Repository for Existing VMs with Shared Root Locations and Separate Configuration Directories” on page 116](#)

Creating a Repository to Use with New VMs

If you are creating a repository for new VMs that you will eventually provision, use the following steps:

- 1 In the *Root Location* field, enter the location for the new repository.

Example: /vms_new

- 2 In the *Preferred Storage Path* field, enter 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

When you already have VMs in your grid, so a store for the VM configuration and disk image files already exists,

- 1 In the *Root Location* field, enter the shared location for this repository.

Example: /vms_new

- 2 In the *VM Config Search Path* field, enter 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, enter 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

When you want to create a repository for existing VMs that have a shared root path but separate configuration file directories, (for example, `/vms_new/old_config1` and `/vms_new/old_config2`)

- 1 In the *Root Location* field, enter the shared location for this repository.

Example: `/vms_new`

- 2 In the *VM Config Search Path* field, enter 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, enter `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, enter the path to your existing image file store (relative to the Root Location path). This path will also become 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`.

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 and then clicking *Add* or *Remove* to move it to or from the Source Repository Groups list.

5.6.3 The 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* and clicking the Save icon.

5.6.4 The Repository Health Debugger Tab

The Health Debugger is a common Admin view in the Development Client for most Grid objects. For information about this tool, see [Chapter 6, “The Health Debugger,” on page 127](#).

5.6.5 The 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. In essence, by building, deploying, and running jobs on the PlateSpin Orchestrate Server, you can individually change the functionality of any and all system resources by managing an object's facts and constraints. The Orchestrate 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 (that is, using the edit “pencil” icon) but changes cannot be made.

5.6.6 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 Orchestrate 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 icon in the toolbar to fetch and display fresh data.

For more details about the information listed on the Action History page, see [Section D.2.2, “Action History in Admin Views of the Development Client,” on page 152](#).

5.7 The User Object

A User object represents an individual account that is allowed to connect to the PlateSpin Orchestrate Server. Administrator users are also allowed to connect by using the `zosadmin` command line and the PlateSpin Orchestrate Development Client user interfaces.

You can use the Orchestrate Development Client user interface to manually create a User object, or you can create them 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.

- ♦ [Section 5.7.1, “User Groups,” on page 117](#)
- ♦ [Section 5.7.2, “The User Info/Groups Tab,” on page 118](#)
- ♦ [Section 5.7.3, “The User Policies Tab,” on page 123](#)
- ♦ [Section 5.7.4, “The User Health Debugger Tab,” on page 124](#)
- ♦ [Section 5.7.5, “The User Constraints/Facts Tab,” on page 124](#)
- ♦ [Section 5.7.6, “The User Action History Tab,” on page 124](#)

5.7.1 User Groups



Any group object displayed in the Explorer panel represents a collection of similar object types. Groups can also be created automatically, as in the case when a provisioning adapter (PA) discovers a local repository on a VM host. For example, the `xen30` PA, 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 Development Client, 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*.

5.7.2 The 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.

- ♦ “Info” on page 118
- ♦ “Groups” on page 123

NOTE: Whenever you make changes to any Grid object, that object’s icon is overlaid with the write icon , signifying that the object has been altered. If you want to save the changes you have made, you need to click the Save icon  on the Development Client toolbar.

Info

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

- ♦ “Show Inherited Fact Values Check Box” on page 118
- ♦ “User Information” on page 118
- ♦ “Personal Information” on page 119
- ♦ “Job Information” on page 120
- ♦ “Accounting Information” on page 121
- ♦ “Job Control” on page 121
- ♦ “Quota Information” on page 123

Show Inherited Fact Values Check Box

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).

User Information

The User 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.

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 (that is, “enabled”).

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 (that is, its value is “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 (that is, its value is “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 (this follows the normal rules of policy inheritance). For more information, see [Appendix A, “Grid Object Health Monitoring,” on page 133](#)


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

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

External Groups: This field displays 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. In this dialog box you can add, remove, or edit the name and value for every user environment you want to use.

Personal Information

First Name: This value specifies 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: This value specifies 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: This value specifies 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" />
```

Email: This value specifies 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: This value specifies 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: This value specifies 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: This value specifies 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: This value specifies 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: This field displays a list of default user environment variable names and values that the Orchestrate 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: This value specifies the total number of jobs that this user has historically initiated on this Orchestrate server.

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

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

Active Jobs: This value specifies the number 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: This value specifies the number 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: This value specifies 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: This value specifies the number of currently active sessions (that is, connections) that the user has established with the Orchestrate Server.

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

```
<fact name="user.sessions" value="1" type="Integer" />
```

Accounting Information

Total Spending: This field displays 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: This field displays 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: This value specifies 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: This drop-down list 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: This field displays 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: This field displays 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: This value specifies 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: This field displays 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: This value specifies 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: This value specifies 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 willing 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: This value specifies 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: This value specifies 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: This field displays 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. In this dialog box 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: This field specifies 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: This field specifies 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: This field specifies 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: This field specifies 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" />
```

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.

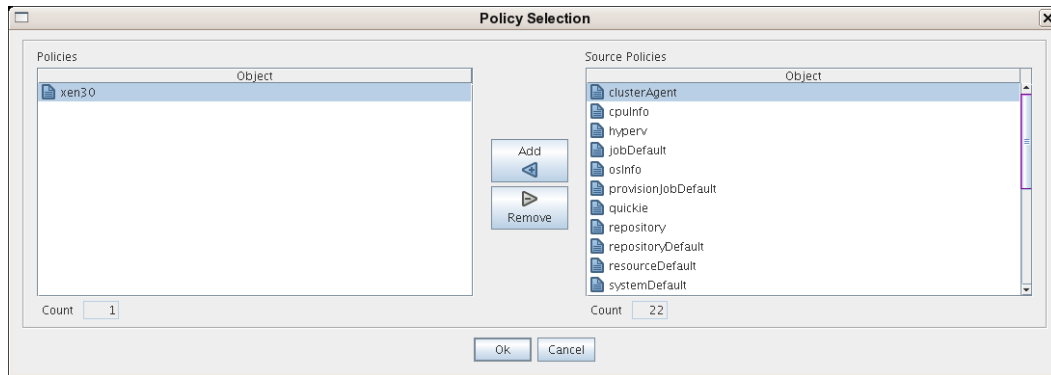
5.7.3 The User Policies Tab

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 using the Policy Grid object. For more information, see [Section 5.8.1, “The Policy Object,” on page 125](#).

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 5-7 The Policy Selection Dialog Box



5.7.4 The User Health Debugger Tab

The Health Debugger is a common Admin view in the Development Client for most Grid objects. For information about this tool, see [Chapter 6, “The Health Debugger,” on page 127](#).

5.7.5 The 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. In essence, by building, deploying, and running jobs on the PlateSpin Orchestrate Server, you can individually change the functionality of any and all system resources by managing an object’s facts and constraints. The Orchestrate 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 (that is, using the edit “pencil” icon) but changes cannot be made.

5.7.6 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 Orchestrate 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 icon in the toolbar to fetch and display fresh data.

For more details about the information listed on the Action History page, see [Section D.2.2, “Action History in Admin Views of the Development Client,” on page 152](#).

5.8 Other Displayed Objects

The Explorer also includes Policies, Public JDL Libraries, Computed Facts, Events, and Metrics.

- ♦ [Section 5.8.1, “The Policy Object,” on page 125](#)
- ♦ [Section 5.8.2, “Computed Fact Objects,” on page 125](#)

- ◆ [Section 5.8.3, “Event Objects,” on page 125](#)
- ◆ [Section 5.8.4, “Metrics,” on page 125](#)

5.8.1 The Policy Object

XML is used to define PlateSpin Orchestrate 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 a policy in the Explorer tree view, making your changes, and then clicking the Save icon. The *Where Used* tab of the Policy Editor lists the jobs where the selected policy associated.

Policy Constraints

A policy can define a collection of constraints which 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` located on the PlateSpin Orchestrate Server as an example policy file with constraints.

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 PlateSpin Orchestrate Server at `/opt/novell/zenworks/zos/server/examples/allTypes.policy`.

5.8.2 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 PlateSpin Orchestrate Server at `/opt/novell/zenworks/zos/server/examples/activejobs.cfact` and `/opt/novell/zenworks/zos/server/examples/repostiory.cfact`.

5.8.3 Event Objects

An Event object in the Explorer tree represents a user-described set of rules which 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 *PlateSpin Orchestrate 2.0 Developer Guide and Reference*, [“Event Triggers” on page 32](#) of this guide, and [Section A.2, “Health Events,” on page 135](#) of this guide.

5.8.4 Metrics

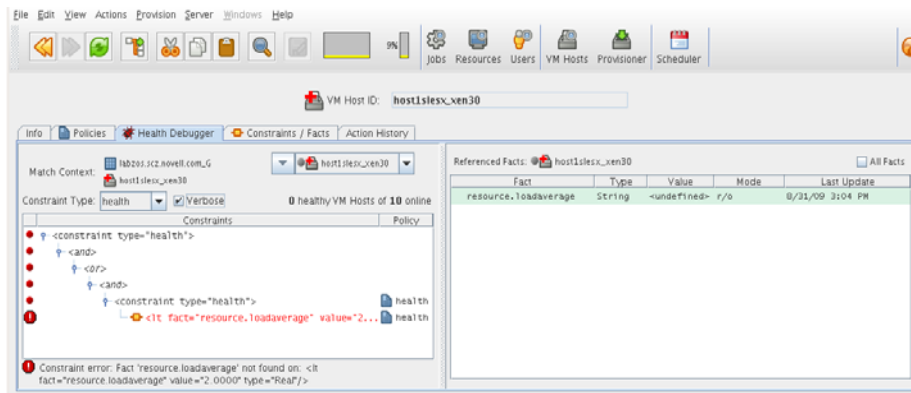
Although the Metrics object is visible in the Explorer panel, configuration for this object is not available in this release.

The Health Debugger

Several objects modeled by PlateSpin® Orchestrate have a health fact that can be used to visually show the health of the object or to trigger a job (see [Chapter 3, “The PlateSpin Orchestrate Job Scheduler,”](#) on page 25) 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 PlateSpin Orchestrate Development Client 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 Orchestrate Development Client.

Figure 6-1 PlateSpin Orchestrate Jobs View with the Health Debugger Page Open



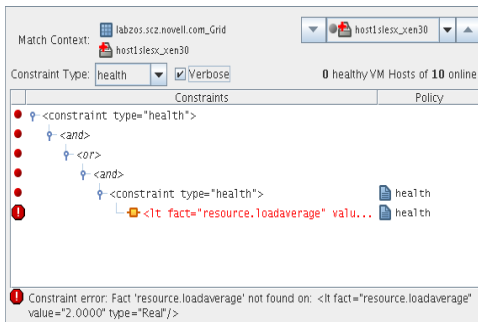
The Health Debugger tab is also available in the VM Host object view, the Repository object view and in the User object view of the Orchestrate Development Client.

- ◆ [Section 6.1, “The Constraints Table Panel,”](#) on page 127
- ◆ [Section 6.2, “The Facts Table View,”](#) on page 130
- ◆ [Section 6.3, “Health Debugger Use Cases,”](#) on page 131

6.1 The Constraints Table Panel

The left side of the Health Debugger page is referred to as the Constraints Table view.

Figure 6-2 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 4.1.2, “The Constraint Type List,” on page 53](#). 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:

- ◆ [Section 6.1.1, “The Match Context Area,” on page 128](#)
- ◆ [Section 6.1.2, “The Verbose Check Box,” on page 129](#)
- ◆ [Section 6.1.3, “The Capable Objects Summary,” on page 129](#)
- ◆ [Section 6.1.4, “The Constraints Column of the Constraints Table View,” on page 129](#)

6.1.1 The 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.

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 Orchestrate 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.

6.1.2 The 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 4.1.5, “The Constraints Column of the Constraints Table View,”](#) on page 54.

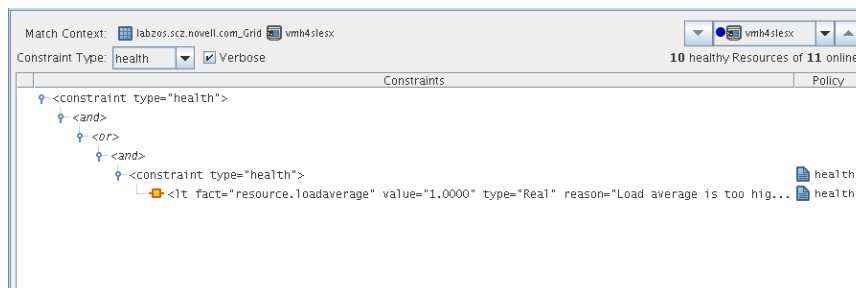
6.1.3 The Capable Objects Summary

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.

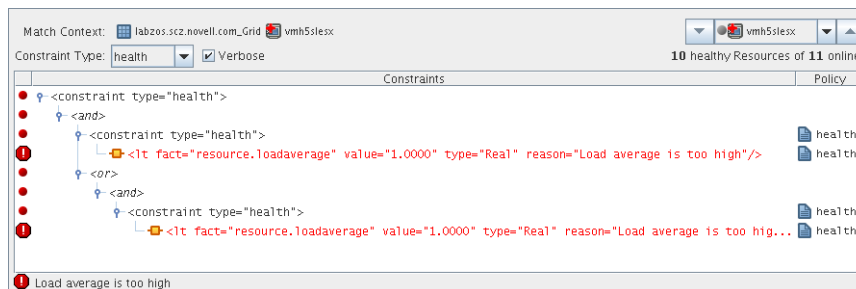
6.1.4 The Constraints Column of the Constraints Table View

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:

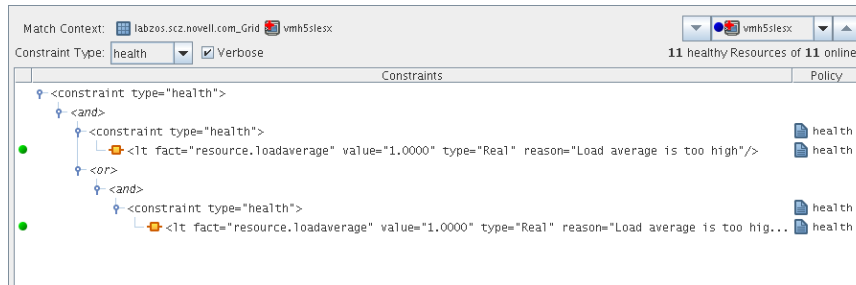
- ◆ no icon: The constraint passes the match. It is a “true” match. No red icons are displayed next to any listed constraint.



- ◆ red dot icon: The constraint does not pass the match. The figure below shows that the resource `vmh5s1esx` cannot run the job because its health constraints do not match.



- ◆ red octagonal icon: For convenience, just one of the constraints is identified as the blocking constraint. This is the constraint that the system has determined as responsible for the constraint as a whole to fail (note that individual constraint lines can fail without causing the the entire constraint to fail).
- ◆ 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.



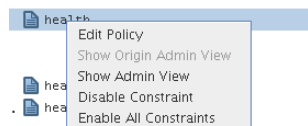
If you right-click a constraint in the table, a popup menu with three options is displayed. This menu lets you change the status of the constraint. Disabling a constraint is useful if you want to temporarily relax a condition without editing or redeploying the whole policy and potentially affecting other objects that share the policy. A disabled constraint can be re-enabled later.

- ◆ **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.
- ◆ **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 popup menu for constraint column does.

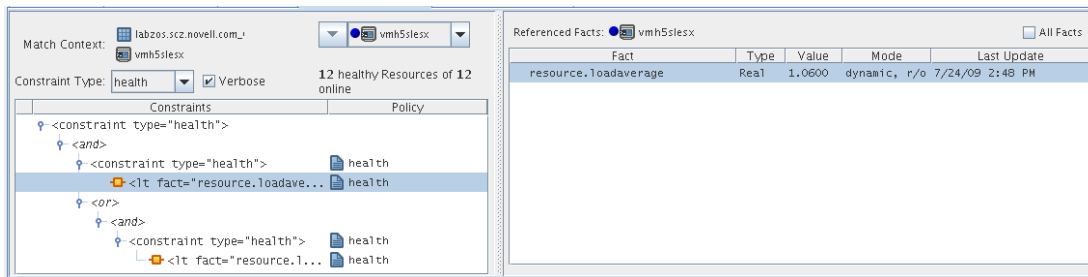
Figure 6-3 The Popup Menu Launched from the Policy Column



6.2 The Facts Table View

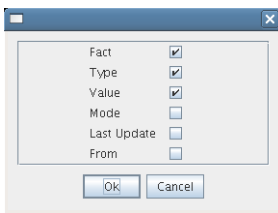
The Facts Table view displays the facts referenced in the Constraint Tree view for a specified object. Selecting a fact in the Constraint tree automatically selects that fact in the table.

Figure 6-4 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 6-5 Menu Used to Select the Columns Displayed in the Facts Table View of the Policy Debugger



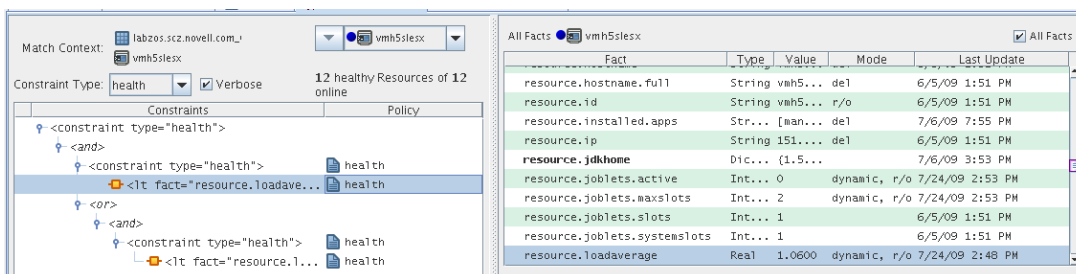
- ◆ Section 6.2.1, “The All Facts Check Box,” on page 131

6.2.1 The 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 4.1.1, “The Match Context Area,” on page 52) 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 6-6 All Facts Check Box Selected with All VM Hosts in Match Context



6.3 Health Debugger Use Cases

This section includes several use cases that show how the Health Debugger works and how to use it. The following use cases are included:

- ◆ Section 6.3.1, “Use Case 1: ??,” on page 132

6.3.1 Use Case 1: ??

The objective of this use case is to

Use the following steps to recreate the use case.

Grid Object Health Monitoring

A

This section includes the following information:

- ♦ [Section A.1, “Health Facts,” on page 133](#)
- ♦ [Section A.2, “Health Events,” on page 135](#)

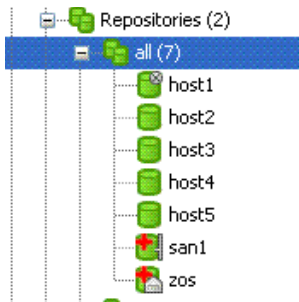
A.1 Health Facts

Starting with PlateSpin® Orchestrator 2.0, the Resource grid object, the VM host grid object, the User grid object and the Repository grid object each has 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 PlateSpin Orchestrator Development Client with a red cross to signal the object’s condition.

Figure A-1 Tree View of Repository Grid Objects in the “all” Group, Some Objects Unhealthy

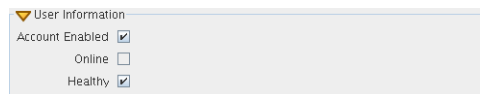


You can define what constitutes the health or non-health of the grid object by setting this health fact. The health fact can be set or cleared in several ways:

- ♦ Explicitly set or cleared by the administrator using tools in the PlateSpin Orchestrator Development Client.
 - ♦ Select any grid object in the Development Client, 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.

Figure A-2 The “Healthy” Check box on the Info/Groups Page



- ◆ On the Constraint/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.

Figure A-3 The Edit Fact Dialog Box Displayed from the Constraint/Fact Page



You can set the health of the object by selecting or deselecting the health check box. Changing the value in the Development Client in this way has an immediate effect unless the value is overridden by an attached policy (this follows the normal rules of policy inheritance).

- ◆ Set by using a discovery job (a job 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.
- ◆ 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>
```

- ◆ 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.
- ◆ 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 (more practical).

Example 2: Define a user as unhealthy if he or she has no money in their 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 (more practical).

You can aggregate (that is, group together with “and” or “or”) health constraints by using normal rules of policy aggregation.

By default, PlateSpin Control runs health constraints every 30 seconds. To alter this interval, you must contact Novell 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 *PlateSpin Orchestrate 2.0 Developer Guide and Reference*) or the event can be used to trigger Jobs in the Job Scheduler (see “[The PlateSpin Orchestrate Job Scheduler](#)” in the *PlateSpin Orchestrate 2.0 Development Client Reference*). 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
VmHost	VMHOST_HEALTH

Understanding Policy Elements

B

XML is the representation for PlateSpin® Orchestrate policy elements. 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.

- ♦ [Section B.1, “Constraints,” on page 137](#)
- ♦ [Section B.2, “Facts,” on page 137](#)
- ♦ [Section B.3, “Computed Facts,” on page 137](#)

B.1 Constraints

The constraint element defines the selection of grid objects such as resources. The required type attribute defines the selection type. Supported types are:

- ♦ Resource
- ♦ Provision
- ♦ Allocation
- ♦ Accept
- ♦ Start
- ♦ Continue
- ♦ vmhost
- ♦ Repository

Constraints can also be constructed in JDL and in the Java Client SDK. A JDL constructed constraint can be used for grid search and for scheduling. A Java Client SDK constructed constraint can only be used for grid object search. For additional information, see “[Working with Facts and Constraints](#)” in the *PlateSpin Orchestrate 2.0 Developer Guide and Reference*.

B.2 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.

See the examples in the directory, `/allTypes.policy`. This example policy has an XML representation for all the fact types.

Facts can also be created and modified in JDL and in the Java Client SDK.

B.3 Computed Facts

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.

To create a new computed fact, you subclass the `ComputedFact` class with the `.cfact` extension. An implementation uses the `ComputedFactContext` to get the evaluation context. For more information, see the job structure from the following examples in the *PlateSpin Orchestrate 2.0 Developer Guide and Reference*:

- ◆ “`ComputedFact`”
- ◆ “`ComputedFactContext`”

After the new computed fact is created, you deploy it using the same procedures required for jobs (using either the `zosadmin` command line tool or the PlateSpin Orchestrate Development Client).

The following example shows a computed fact that returns the number of active job instances for a specific job for the current job instance. This fact can be used in an accept or start constraint to limit how many jobs a user can run in the system. The constraint is added to the job policy in which to have the limit. In this example, the start constraint uses this fact to limit the number of active jobs for a user to one:

```
"""
    <constraint type="start" >
      <lt fact="cfact.activejobs"
        value="1"
        reason="You are only allowed to have 1 job running at a time" />
    </constraint>
```

Change `JOB_TO_CHECK` to define which job is to be limited.

```
"""
JOB_TO_CHECK="quickie"

class activejobs(ComputedFact):

    def compute(self):

        j = self.getContext()
        if j == None:
            # This means computed Fact is executed in a non running
            # job context. e.g., the ZOC fact browser
            print "no job instance"
            return 0
        else:
            # Computed fact is executing in a job context
            user = j.getFact("user.id")
            activejobs = self.getMatrix().getActiveJobs()
            count = 0
            for j in activejobs:
                jobname = j.getFact("job.id")

                # Don't include queued in count !
                state = j.getFact("jobinstance.state.string")
                if jobname == JOB_TO_CHECK \
                    and j.getFact("user.id") == user \
                    and (state == "Running" or state == "Starting"):
                    count+=1

            jobid = j.getFact("jobinstance.id")
            print "jobid=%s count=%d" % (jobid,count)
            return count
```

For another computed fact example, see `activejobs.cfact` (located in the `examples/activejobs.cfact` directory).

For more information about policies, see “**Policy Elements**” in the *PlateSpin Orchestrate 2.0 Developer Guide and Reference*.

Events

C

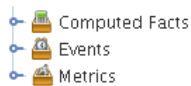
This section contains the following information:

- ◆ [Section C.1, “Event Object Visualization and Management in the Development Client,” on page 141](#)
- ◆ [Section C.2, “The Event Debugger,” on page 143](#)
- ◆ [Section C.3, “Understanding the PlateSpin Orchestrate Events System,” on page 146](#)

C.1 Event Object Visualization and Management in the Development Client

The Events folder is displayed in the Explorer tree between the Computed Facts and Metrics folders.

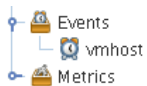
Figure C-1 *The Events Folder in the PlateSpin Orchestrate Explorer View*




Although the PlateSpin Orchestrate system includes several built-in Events (see [Section C.3.2, “Built-in Events,” on page 147](#)), these Events are not displayed in the Explorer view. Only custom Events, defined by an administrator 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 C-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 C.1.4, “The Event Editor,” on page 143](#).

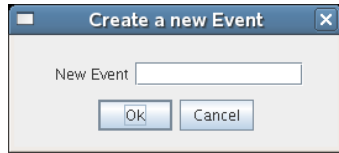
This section includes the following information about how the Event object is managed in the Development Client:

- ◆ [Section C.1.1, “Deploying a New Rule-Based Event,” on page 142](#)
- ◆ [Section C.1.2, “Deploying a Pre-written Rule-Based Event,” on page 142](#)
- ◆ [Section C.1.3, “Undeploying an Event,” on page 142](#)
- ◆ [Section C.1.4, “The Event Editor,” on page 143](#)

C.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.



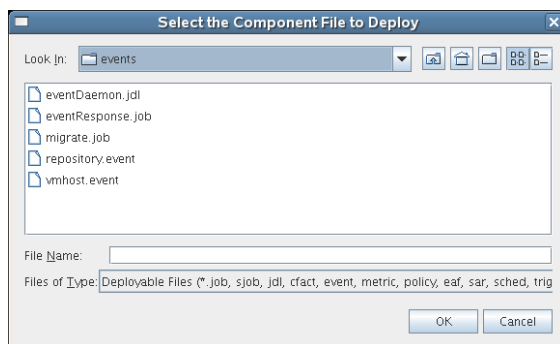
- 2 Enter the name for the new Event, then click *OK* to create the new Event object.

PlateSpin Orchestrate then deploys the new Event object on the server, where it can be managed. The Development Client opens the Event Editor, where you can edit the XML definition of this Event. For more information, see [Section C.1.4, “The Event Editor,” on page 143](#) and [Section C.3.3, “Rule-based Events,” on page 148](#).

C.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 Development Client opens the Event Editor, where you can edit the XML definition of this Event. For more information, see [Section C.1.4, “The Event Editor,” on page 143](#) and [Section C.3.3, “Rule-based Events,” on page 148](#).

C.1.3 Undeploying an Event

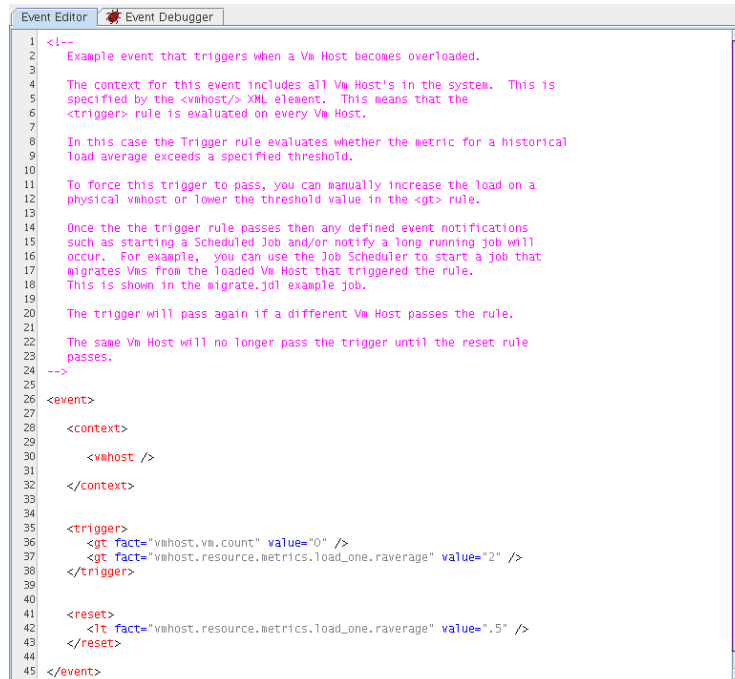
When an Event has been deployed, it can be undeployed. Undeploying deletes the Event object within PlateSpin Orchestrate, but it does not delete the source `.event` file, which still exists and can be redeployed.

To undeploy an Event, right-click on the Event object in the Explorer tree, then select *Undeploy*. You can also simply select the object and press Delete.


C.1.4 The Event Editor

The Event Editor opens when you select a deployed Event in the Explorer tree of the Development Client.

Figure C-3 *The Event Editor*



```
1 <!--  
2 Example event that triggers when a Vm Host becomes overloaded.  
3  
4 The context for this event includes all Vm Host's in the system. This is  
5 specified by the <vmhost/> XML element. This means that the  
6 <trigger> rule is evaluated on every Vm Host.  
7  
8 In this case the Trigger rule evaluates whether the metric for a historical  
9 load average exceeds a specified threshold.  
10  
11 To force this trigger to pass, you can manually increase the load on a  
12 physical vmhost or lower the threshold value in the <gt> rule.  
13  
14 Once the the trigger rule passes then any defined event notifications  
15 such as starting a Scheduled Job and/or notify a long running job will  
16 occur. For example, you can use the Job Scheduler to start a job that  
17 migrates Vms from the loaded Vm Host that triggered the rule.  
18 This is shown in the migrate_vm.example.job.  
19  
20 The trigger will pass again if a different Vm Host passes the rule.  
21  
22 The same Vm Host will no longer pass the trigger until the reset rule  
23 passes.  
24 -->  
25  
26 <event>  
27  
28 <context>  
29  
30 <vmhost />  
31  
32 </context>  
33  
34 <trigger>  
35 <gt fact="vmhost.vm.count" value="0" />  
36 <gt fact="vmhost.resource.metrics.load_one.raverage" value="2" />  
37 </trigger>  
38  
39  
40 <reset>  
41 <lt fact="vmhost.resource.metrics.load_one.raverage" value=".5" />  
42 </reset>  
43  
44  
45 </event>
```

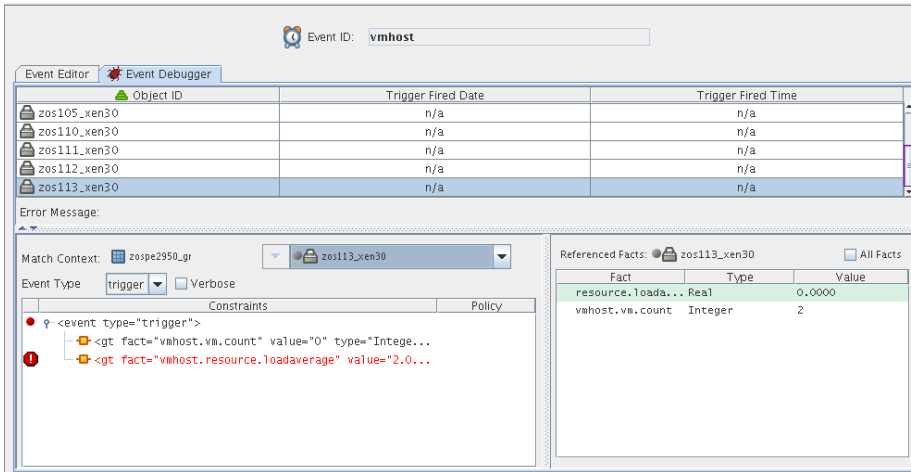
Inside this editor, you can make changes to the XML content of the Event. Example Events contain comments that explain how you can use them and the behavior you can expect to see as a result of deploying them. For the changes you make to be effective, you need to click the Save  tool.

For more information about the allowed XML syntax within an Event, see [Section C.3.3, “Rule-based Events,”](#) on page 148.

C.2 The 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 C-4 *The Event Debugger*



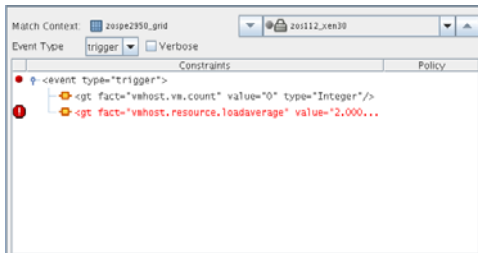
The information in this section describes the various parts of the debugger.

- ◆ [Section C.2.1, “The Constraints Table,” on page 144](#)
- ◆ [Section C.2.2, “The Facts Table,” on page 145](#)

C.2.1 The 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 C-5 *The Constraints Table Area of the Events Debugger*



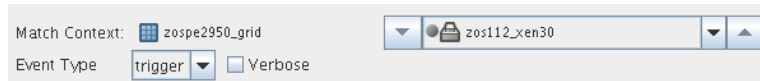
The Constraints Table has several parts:

- ◆ [“Match Context” on page 144](#)
- ◆ [“Event Type List” on page 145](#)
- ◆ [“Verbose Check Box” on page 145](#)
- ◆ [“Constraints List \(Tree\)” on page 145](#)

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 C-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> rule(s) 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, 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 on a constraint to expose 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.

C.2.2 The Facts Table

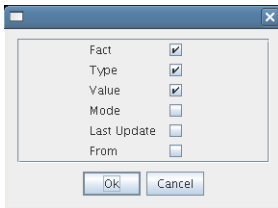
The Facts Table view displays the facts referenced in the Constraint Tree view for a specified Event. Selecting a rule containing a particular fact in the Constraint tree automatically selects that fact (and its current value) in the table.

Figure C-7 *The Facts Table View*

Fact	Type	Value
resource.10a...	Real	0.0000
vmhost.vm.count	Integer	4

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 C-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 (if the fact is editable or can be deleted) the selected fact.

C.3 Understanding the PlateSpin Orchestrate Events System

The PlateSpin Orchestrate 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 PlateSpin Orchestrate 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.

PlateSpin Orchestrate 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 C.3.1, “Event Notification,” on page 147](#)

- ♦ [Section C.3.2, “Built-in Events,” on page 147](#)
- ♦ [Section C.3.3, “Rule-based Events,” on page 148](#)

C.3.1 Event Notification

An Event notifies two other PlateSpin Orchestrate 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 “[The PlateSpin Orchestrate Job Scheduler](#)” in the *PlateSpin Orchestrate 2.0 Development Client Reference* for more information about setting up a Job Schedule. Also see “[Job Architecture](#)” in the *PlateSpin Orchestrate 2.0 Developer Guide and Reference* for more information about how jobs can register for event notification.

C.3.2 Built-in Events

Built-in events occur when a managed object comes online/offline or has a health status change.

PlateSpin Orchestrate uses the following built-in Events to keep managed objects synchronized.

Table C-1 *PlateSpin Orchestrate 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 off from server
RESOURCE_ONLINE	Resource Agent has logged in to server
SERVER_UP	Server has fully started
USER_HEALTH	User health status has changed
USER_ONLINE	User has logged in to 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 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.

C.3.3 Rule-based Events

Rule-based Events are defined in an XML document and deployed to the PlateSpin Orchestrate Server and managed using the PlateSpin Orchestrate Development Client. 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 PlateSpin Orchestrate 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>` is 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: Defines the context for the Event's rule evaluation.

Line 4: The context specifies for 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: Defines the Trigger rule to determine if this Event is to fire notifications or not. If the trigger rule does not pass, then 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 if the average is greater than 2.

Lines 12-14: 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.

So 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.

Provisioning Actions and History

D

The following information is included in this section:

- ◆ [Section D.1, “What are Provisioning Actions?,” on page 151](#)
- ◆ [Section D.2, “How Actions Are Displayed in the Development Client,” on page 151](#)

D.1 What are Provisioning Actions?

Beginning with the 2.0.2 release, the provisioning operations you perform in PlateSpin® Orchestrate are recorded as “actions.” For example, in the PlateSpin Orchestrate Development Client main menu:

- ◆ A VM Host Discovery action is initiated if you select *Provision > Discover VM Hosts and Repositories* and then you select a provisioning adapter in the Discover VM Hosts and Repositories dialog box.
- ◆ A VM Discovery action is initiated if you select *Provision > Discover VM Images* and then you select a provisioning adapter in the Discover VM Images dialog box. An action is specified for each Repository you specify.
- ◆ A Migrate action is initiated when you perform the migration of a Virtual Machine (VM).

D.2 How Actions Are Displayed in the Development Client

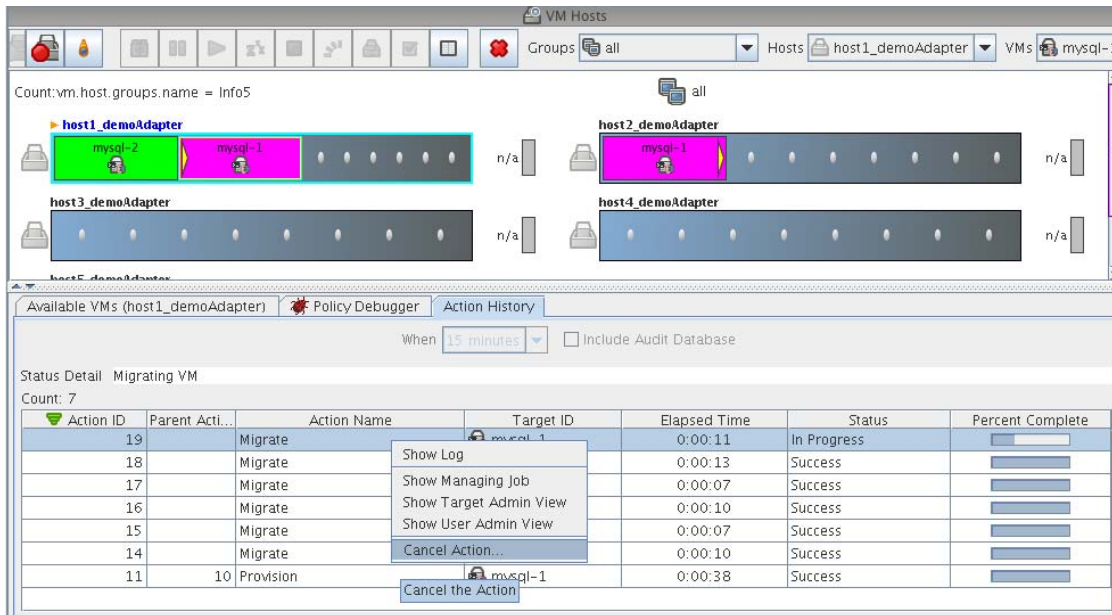
Depending on the Grid object you select, an Action History tab is displayed in several views of the Orchestrate Development Client.

- ◆ [Section D.2.1, “Action History in Monitor Views of the Development Client,” on page 151](#)
- ◆ [Section D.2.2, “Action History in Admin Views of the Development Client,” on page 152](#)

D.2.1 Action History in Monitor Views of the Development Client

You can see the Action History tab in the VM Hosts monitor view if you select a migrating VM:

Figure D-1 VM Hosts Actions



Two action-specific menu selections are available if you right-click an action in the action history table:

- ◆ *Show Log* opens the provision log for the VM
- ◆ *Cancel Action* cancels the selected active action

The action history table is updated at the same time the polling view is updated.

NOTE: The format of action history table is similar in the Provisioner monitor view and in the Users monitor view.

If the *Include Audit Database* check box is selected in this view, the action status is not polled. Click the refresh icon to fetch and display fresh data.

NOTE: The Orchestrate Server must be connected to an audit database for the *Include Audit Database* check box to be available. This behavior is the same in the Job monitor view.

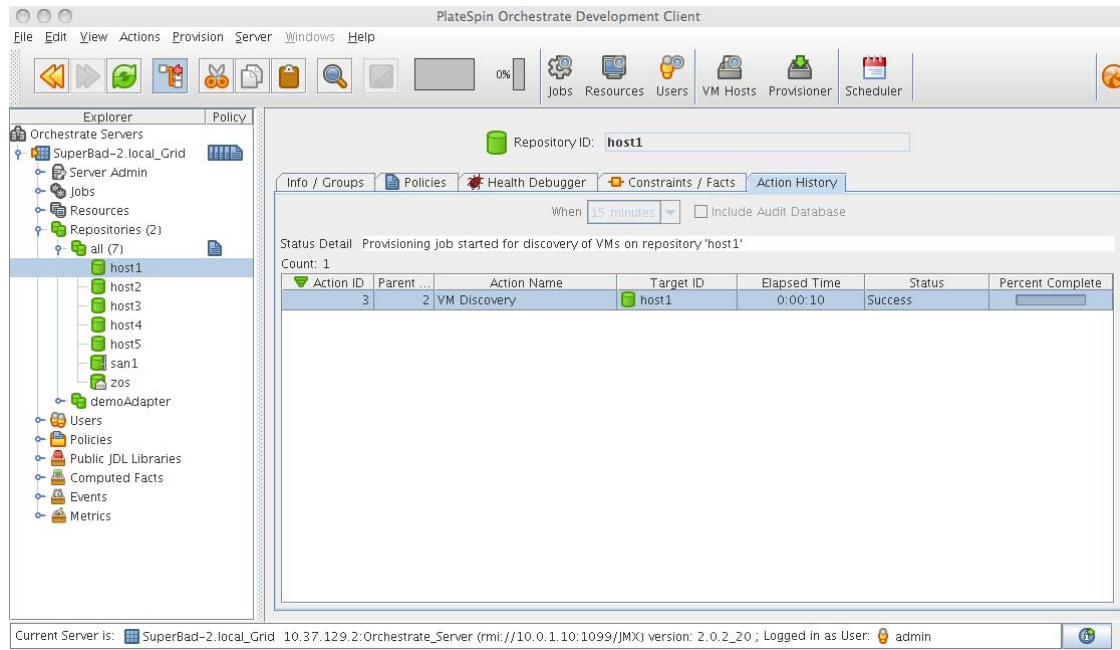
D.2.2 Action History in Admin Views of the Development Client

Action history is displayed in other Grid object admin views of the Development Client:

- ◆ User object
- ◆ Resource object
- ◆ Repository object

The following illustration shows an example of action history in the Repository admin view:

Figure D-2 Repository Action History

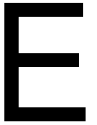


The table below defines some of the column names and the values that can populate those columns in the action history table:

Table D-1 Action History Table Columns

Column Name	Purpose
Action ID	A unique integer value used to identify the action.
Parent Action ID	If a value is displayed, the action is a child of this parent, identified with a unique integer value.
Action Name	Specific to the action being invoked.
Target ID	The identifying string of the object where the action is to occur.
Status	Displays the status of this action (specific to the action being invoked). Possible values in this column include: <ul style="list-style-type: none"> ◆ Started ◆ In Progress ◆ Success ◆ Failed ◆ Canceled

Documentation Updates



This section contains information about documentation content changes that were made in this *PlateSpin Orchestrate Development Client Reference* after the initial release of PlateSpin® Orchestrate 2.0. The changes are listed according to the date they were published.

The documentation for this product is provided on the Web in two formats: HTML and PDF. The HTML and PDF documentation are both kept up-to-date with the changes listed in this section.

If you need to know whether a copy of the PDF documentation that you are using is the most recent, the PDF document includes a publication date on the title page.

The documentation was updated on the following dates:

E.1 August 28, 2009

Updates were made to the following sections:

Location	Update
Section 5.4, "The Resource Object," on page 81	New section.
Section 5.5, "The VM Host Object," on page 104	New section.

E.2 August 7, 2009

Updates were made to the following sections:

Location	Update
Section 5.3, "The Job Object," on page 68	New section.
Section 5.7, "The User Object," on page 117	New section.
Chapter 6, "The Health Debugger," on page 127	New section.

E.3 July 20, 2009

Updates were made to the following sections:

Location	Update
Section 5.6, "The Repository Object," on page 109	New section.

E.4 June 17, 2009 (2.0.2 Release)

Updates were made to the following sections:

Location	Update
Appendix C, "Events," on page 141	New section.
Appendix D, "Provisioning Actions and History," on page 151	New section.