AUTHORIZED DOCUMENTATION

Development Client Reference

Novell®
PlateSpin® Orchestrate

2.0

February 23, 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/) 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/) 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).

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (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

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 3, "The PlateSpin Orchestrate Job Scheduler," on page 21
- Chapter 4, "The Policy Debugger," on page 47
- Chapter 5, "The Explorer Panel," on page 55

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/).

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 Orchestrator 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.

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	Provides operations categorized under menus such as *File*, *Edit*, *View*, *Grid*, *Server*, *Windows*, and *Help*. ◆ The *File* menu lets you save any changes you've made or exit the console. ◆ The *Edit* menu lets cut, copy, and paste items and choose general and server preferences for console. ◆ The *View* menu lets you manipulate the display of the different components of the console and refresh the Explorer and Workspace panels. ◆ The *Actions* menu lets you launch specific tools that create and delete users or user groups, computing resources, jobs, policies, and computed facts. ◆ The *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 the server. ◆ The *Windows* menu lets you select console windows to display when you have more than one console window open. You can open the Explorer panel and the two tabs of the Info panel (*<Orchestrator> Log* and *Console Output*) in their own windows by right-clicking the tab and choosing *Open in window* in the pop-up menu. ◆ The *Help* menu provides access to the About box for the console. It also provides a link to ZENworks Orchestrator documentation on the Web.

Area	Description
Main toolbar	The main toolbar has buttons for executing common tasks. The basic tasks are *Go Back*, *Go Forward*, *Refresh the view*, *Hide or Show the Explorer Panel*, *Cut*, *Copy*, *Paste*, and *Save changes in workspace view* and *Open the Find Dialog*. Additional tools are available if you install the Virtual Machine Management Pack. The toolbar also includes buttons that open monitoring views for *Jobs*, *Resources*, and *Users*. And additional monitor for Virtual Machine host groups plugs in to the toolbar if you install Virtual Machine Management. To the far left of the toolbar, a pinwheel icon indicates when the console is busy.
Explorer panel	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 *Computing Resources* in the tree, its details appear in the Workspace panel with a list of active computing resources. You can edit the *Computing Resource* attributes in the workspace panel.
Workspace panel	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 *physical* 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.
Info panel	The Info panel displays a variety of information, such as validation and error messages, log files, and query results. You can display or hide the Info panel by clicking the *Info panel* button in the Status bar.
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 11.

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 11
- Section 2.2, "The Orchestrate Development Client Toolbar," on page 19

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 11
- Section 2.1.2, "Edit," on page 12
- Section 2.1.3, "View," on page 15
- Section 2.1.4, "Actions," on page 15
- Section 2.1.5, "Provision," on page 15
- Section 2.1.6, "Server," on page 17
- Section 2.1.7, "Windows," on page 19
- Section 2.1.8, "Help," on page 19

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 11
- "Exit" on page 11

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 12
- "Redo" on page 12
- "Cut" on page 12
- "Copy" on page 12
- "Paste" on page 12
- "Find" on page 13
- "Find Next" on page 13
- "Find Previous" on page 13
- "Enter Find String" on page 13
- "Load Text" on page 13
- "Save Text" on page 13
- "Preferences" on page 13

Undo Addition

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

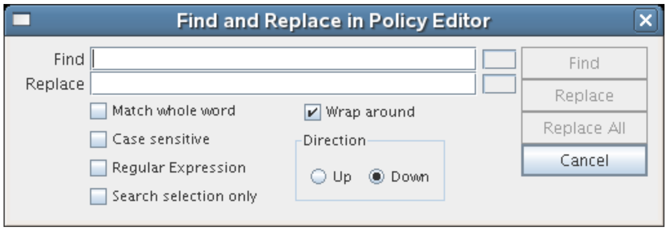
Paste

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.

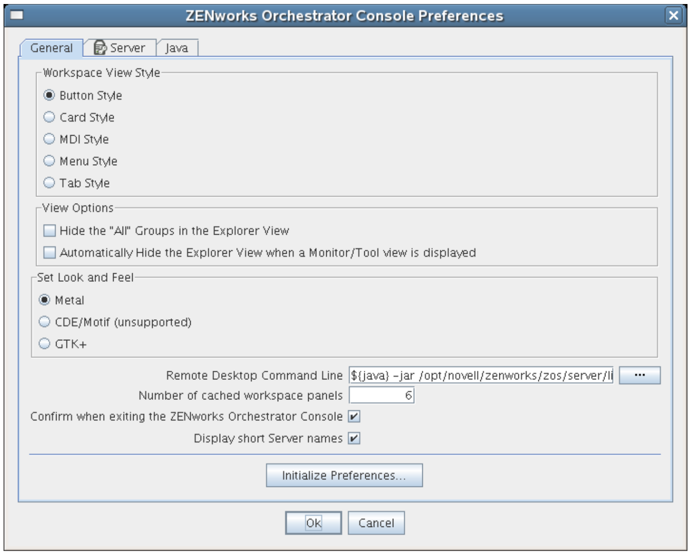
Preferences

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 Orchestrate Development Client Preferences*


ZENworks Orchestrator Console Preferences
General
Server
Java
Initialize Preferences...
Ok
Cancel

Workspace View Style
● Button Style
○ Card Style
○ MDI Style
○ Menu Style
○ Tab Style

View Options
☐ Hide the "All" Groups in the Explorer View
☐ Automatically Hide the Explorer View when a Monitor/Tool view is displayed

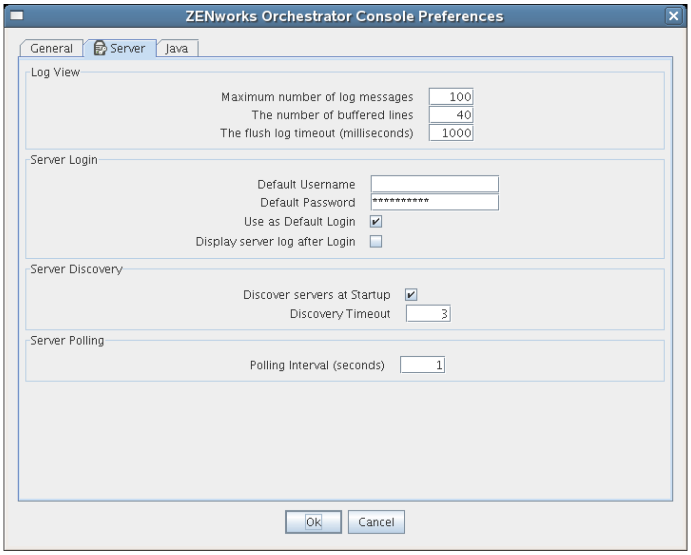
Set Look and Feel
● Metal
○ CDE/Motif (unsupported)
○ GTK+

Remote Desktop Command Line: ${java} -jar /opt/novell/zenworks/zos/server/li
Number of cached workspace panels: 6
Confirm when exiting the ZENworks Orchestrator Console ☑
Display short Server names ☑

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 Orchestrate Development Client Preferences*



Log View
Maximum number of log messages: 100
The number of buffered lines: 40
The flush log timeout (milliseconds): 1000

Server Login
Default Username:
Default Password: **********
Use as Default Login ☑
Display server log after Login ☐

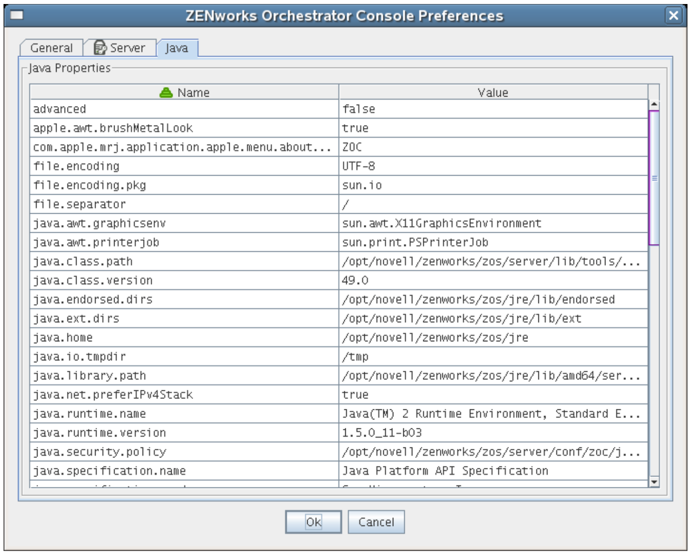
Server Discovery
Discover servers at Startup ☑
Discovery Timeout: 3

Server Polling
Polling Interval (seconds): 1

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*



Name	Value
advanced	false
apple.awt.brushMetalLook	true
com.apple.mrj.application.apple.menu.about...	ZOC
file.encoding	UTF-8
file.encoding.pkg	sun.io
file.separator	/
java.awt.graphicsenv	sun.awt.X11GraphicsEnvironment
java.awt.printerjob	sun.print.PSPrinterJob
java.class.path	/opt/novell/zenworks/zos/server/lib/tools/...
java.class.version	49.0
java.endorsed.dirs	/opt/novell/zenworks/zos/jre/lib/endorsed
java.ext.dirs	/opt/novell/zenworks/zos/jre/lib/ext
java.home	/opt/novell/zenworks/zos/jre
java.io.tmpdir	/tmp
java.library.path	/opt/novell/zenworks/zos/jre/lib/amd64/ser...
java.net.preferIPv4Stack	true
java.runtime.name	Java(TM) 2 Runtime Environment, Standard E...
java.runtime.version	1.5.0_11-b03
java.security.policy	/opt/novell/zenworks/zos/server/conf/zoc/j...
java.specification.name	Java Platform API Specification

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 19.

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.

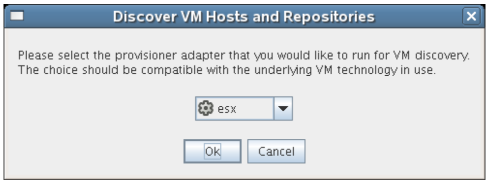
2.1.5 Provision

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*



Discover VM Hosts and Repositories

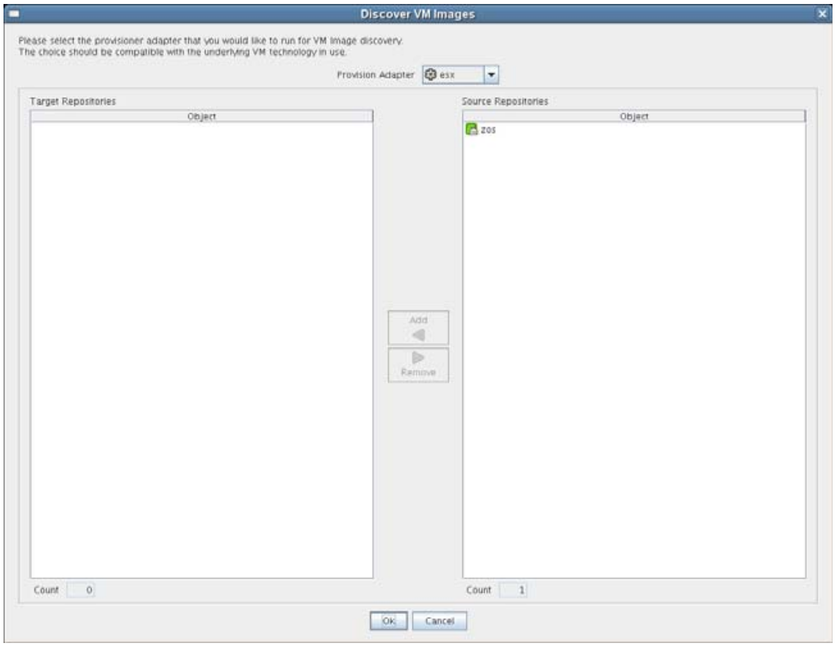
Please select the provisioner adapter that you would like to run for VM discovery.
The choice should be compatible with the underlying VM technology in use.

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*



Discover VM Images

Please select the provisioner adapter that you would like to run for VM image discovery.
The choice should be compatible with the underlying VM technology in use.

Provision Adapter: esx

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 17
- "Discover Servers" on page 17
- "Shutdown Server" on page 17
- "Login" on page 17
- "Logout" on page 18
- "Display Log" on page 18
- "Create Custom Log" on page 18

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.

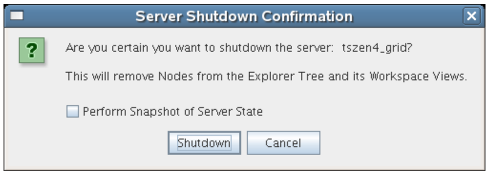
Discover Servers

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*



Are you certain you want to shutdown the server: tszen4_grid?

This will remove Nodes from the Explorer Tree and its Workspace Views.

☐ Perform Snapshot of Server State

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.

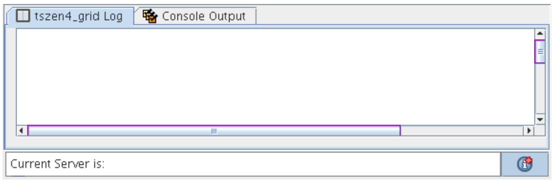
Logout

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*



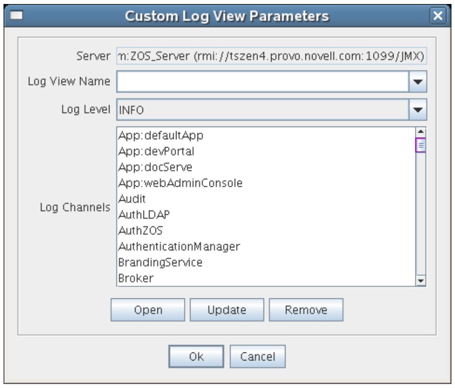
When a log is displayed, you can right-click its tab to further direct the actions of the display. You can pause logging in the window, copy the log to the clipboard, clear its contents, undock the log display as a new window, or remove it from the Information window.

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 12.

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*



Server n:ZOS_Server (rmi://tszen4.provo.novell.com:1099/JMX)
Log View Name
Log Level INFO
Log Channels
App:defaultApp
App:devPortal
App:docServe
App:webAdminConsole
Audit
AuthLDAP
AuthZOS
AuthenticationManager
BrandingService
Broker

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 12.

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 Al*l, 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)
0%	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.

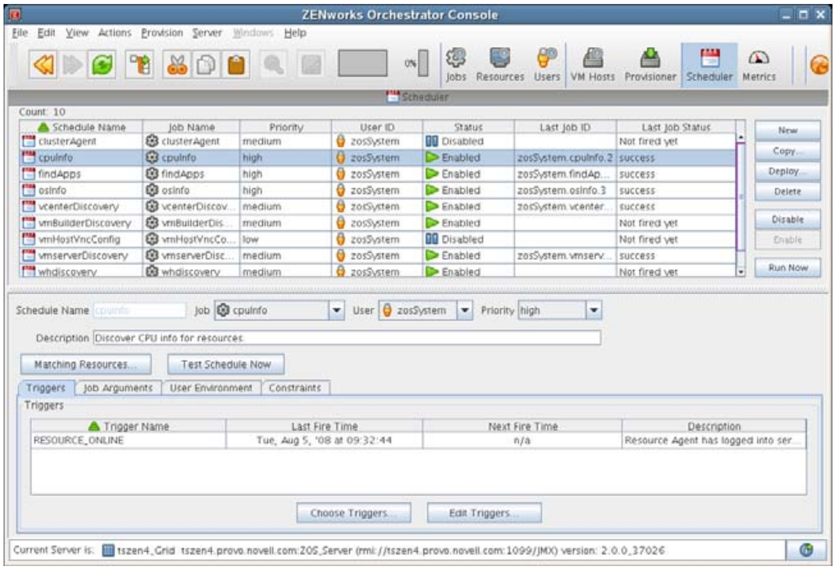
This section includes the following information:

- Section 3.1, "Understanding the Job Scheduler View," on page 21
- Section 3.2, "Walkthrough: Scheduling a System Job," on page 36

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*



Count: 10

Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Last Job Status
clusterAgent	clusterAgent	medium	zosSystem	Disabled		Not fired yet
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
whdiscovery	whdiscovery	medium	zosSystem	Enabled		Not fired yet

Schedule Name: cpuInfo
Job: cpuInfo
User: zosSystem
Priority: high
Description: Discover CPU info for resources

Triggers

Trigger Name	Last Fire Time	Next Fire Time	Description
RESOURCE_ONLINE	Tue, Aug 5, '08 at 09:32:44	n/a	Resource Agent has logged into ser...

Current Server is: tszen4_Grid tszen4.provo.novell.com:ZOS_Server (rmi://tszen4.provo.novell.com:1099/JMX) version: 2.0.0_37026

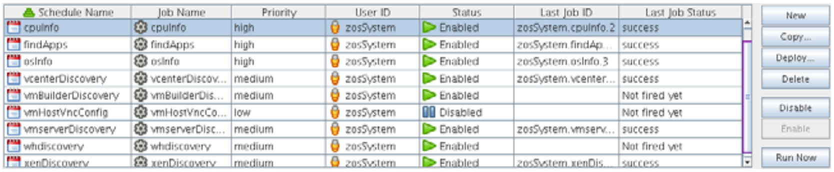
This section includes information to help you understand the functions of the Job Scheduler and how to use it to launch PlateSpin Orchestrate jobs.

- Section 3.1.1, "Navigating The Job Schedules Table," on page 22
- Section 3.1.2, "Creating or Modifying a Job Schedule," on page 24
- Section 3.1.3, "Understanding Cron Syntax in the Job Scheduler," on page 33

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
whdiscovery	whdiscovery	medium	zosSystem	Enabled		Not fired yet
xenDiscovery	xenDiscovery	medium	zosSystem	Enabled	zosSystem.xenDis...	success

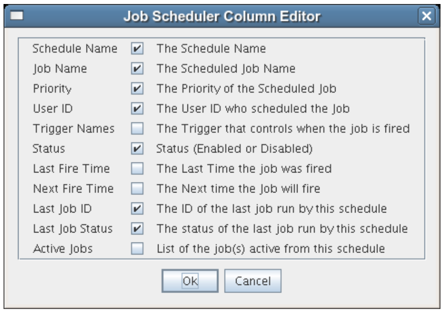
By default, PlateSpin Orchestrate uses schedule names that are similar to the job name so that schedules are easy to match (although this is not required). The schedules list shows all of the existing job schedules that accompany predefined jobs, along with the schedules that you create in the Job Scheduler.

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*



Schedule Name	☑	The Schedule Name
Job Name	☑	The Scheduled Job Name
Priority	☑	The Priority of the Scheduled Job
User ID	☑	The User ID who scheduled the Job
Trigger Names	☐	The Trigger that controls when the job is fired
Status	☑	Status (Enabled or Disabled)
Last Fire Time	☐	The Last Time the job was fired
Next Fire Time	☐	The Next time the Job will fire
Last Job ID	☑	The ID of the last job run by this schedule
Last Job Status	☑	The status of the last job run by this schedule
Active Jobs	☐	List of the job(s) active from this schedule

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 22) 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.

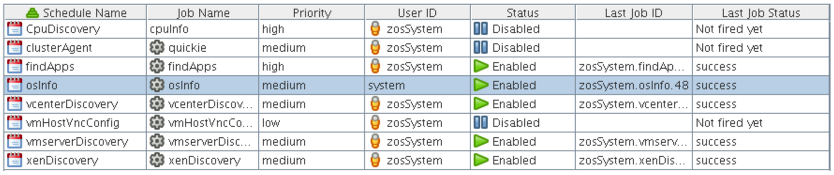
  NOTE: Deleting a schedule that was deployed as part of a `.job` or `.sar` displays a confirmation dialog box. Deleting the schedule undeploys all contents of the `.job` or `.sar` that contains the schedule.
- **Disable:** Disables the selected schedule in the Job Schedules Table. The jobs associated 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.

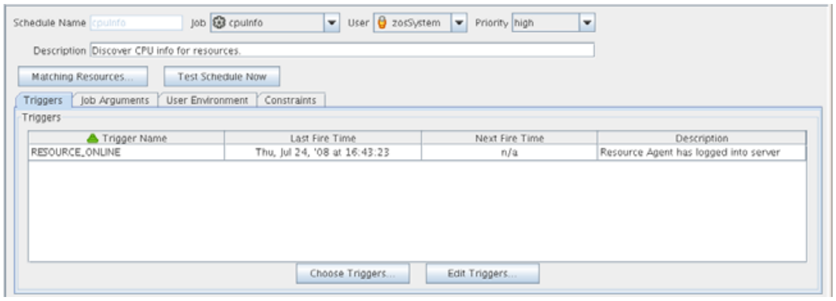
In the *osinfo* schedule, the *system* user has no User icon. That user has been removed from PlateSpin Orchestrate.

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*



Schedule Name cpuInfo
Job cpuInfo
User zosSystem
Priority high
Description Discover CPU info for resources.

Trigger Name	Last Fire Time	Next Fire Time	Description
RESOURCE_ONLINE	Thu, Jul 24, '08 at 16:43:23	n/a	Resource Agent has logged into server

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 25
- "Job" on page 25
- "User" on page 25
- "Priority" on page 25
- "Description" on page 26
- "Matching Resources" on page 26
- "Test Schedule Now" on page 26
- "Triggers" on page 26
- "Job Arguments" on page 31
- "User Environment" on page 32
- "Constraints" on page 33

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.

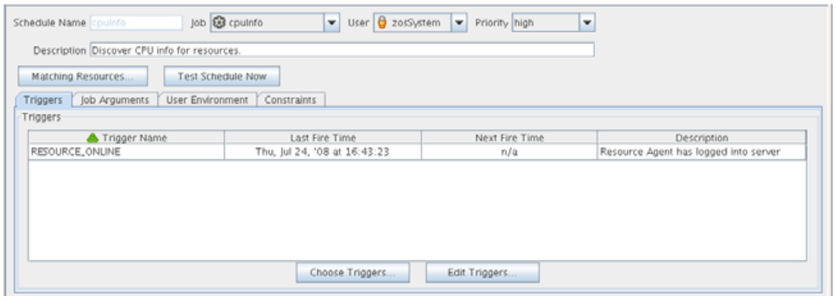
Test Schedule Now

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*



Schedule Name cpuInfo
Job cpuInfo
User zosSystem
Priority high
Description Discover CPU info for resources.

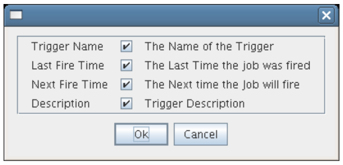
Trigger Name	Last Fire Time	Next Fire Time	Description
RESOURCE_ONLINE	Thu, Jul 24, '08 at 16:43:23	n/a	Resource Agent has logged into server

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*



Trigger Name	☑	The Name of the Trigger
Last Fire Time	☑	The Last Time the job was fired
Next Fire Time	☑	The Next time the Job will fire
Description	☑	Trigger Description

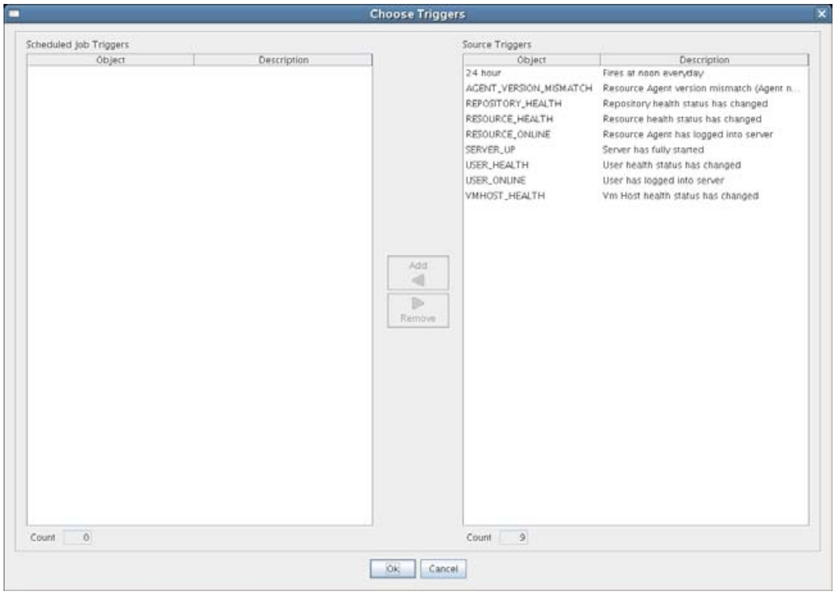
> You can select any column heading in this dialog box to display it in the Triggers table. The columns display the attributes of a previously configured Triggers table. As the figure shows, this dialog box also includes text that clarifies the purpose of the data in each column.

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*



Choose Triggers

Scheduled Job Triggers

Object	Description

Count 0

Source Triggers

Object	Description
24 hour	Fires at noon everyday
AGENT_VERSION_MISMATCH	Resource Agent version mismatch (Agent n...
REPOSITORY_HEALTH	Repository health status has changed
RESOURCE_HEALTH	Resource health status has changed
RESOURCE_ONLINE	Resource Agent has logged into server
SERVER_UP	Server has fully started
USER_HEALTH	User health status has changed
USER_ONLINE	User has logged into server
VMHOST_HEALTH	Vm Host health status has changed

Count 9

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 more details see). 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 *Enter Table Title Here*

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

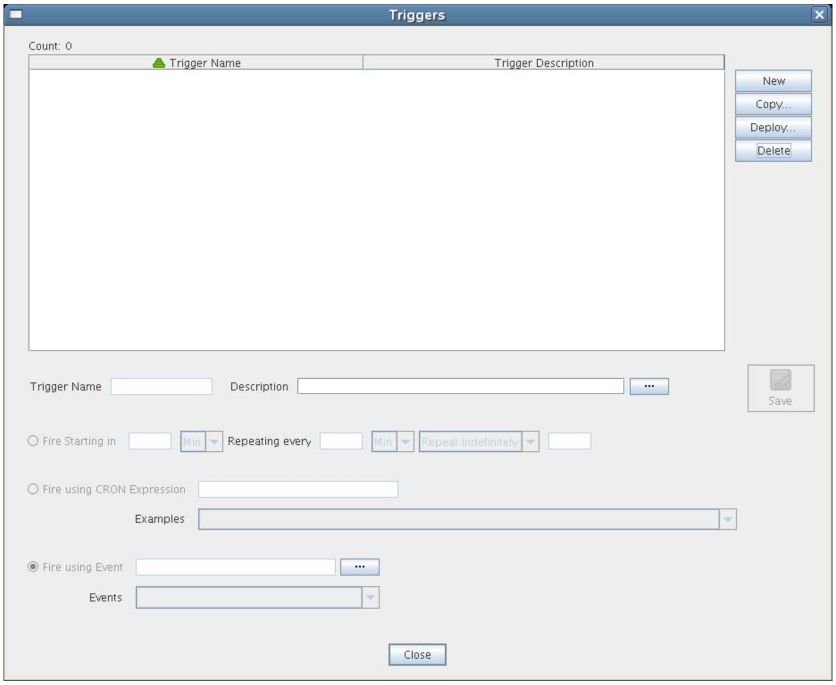
Time Triggers

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 40. For detailed information about cron syntax, see “Understanding Cron Syntax in the Job Scheduler” on page 33.

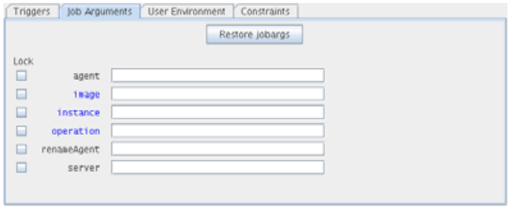
- **Fire Using Event:** Specifies a deployed event or an external event that enables the job to fire when a specified event occurs. Deployed events are defined using an XML syntax. You can specify a deployed event from *Events* (that is, listed in the *Events* drop down list) or you can enter the name of an external event. For more information on creating and firing an event, see

  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 33
- "Special Characters" on page 34
- "Examples of Cron Syntax" on page 35
- "Cron Scheduling Precautions" on page 36

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	, - * /
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.
L (“last”)	The `L` character is allowed for the day-of-month and day-of-week fields. Specifies either the last day of the month, or the last *xxx* day of the month.	The value `L` in the day-of-month field means “the last day of the month”—day 31 for January, day 28 for February in non-leap years. If you use `L` in the day-of-week field by itself, it simply means `7` or `SAT`. But if you use it in the day-of-week field after another value, it means “the last *xxx* day of the month.” For example, `6L` means “the last Friday of the month.” **TIP:** When you use the `L` option, be careful not to specify lists or ranges of values. Doing so causes confusing results.

Operator	Purpose	Example
W (“weekday”)	The `W` character is allowed for the day-of-month field. Specifies the weekday (Monday-Friday) nearest the given day.	If you specify `15W` as the value for the day-of-month field, the meaning is “the nearest weekday to the 15th of the month.” So if the 15th is a Saturday, the trigger fires on Friday the 14th. If the 15th is a Sunday, the trigger fires on Monday the 16th. If the 15th is a Tuesday, it fires on Tuesday the 15th. However, if you specify `1W` as the value for day-of-month, and the 1st is a Saturday, the trigger fires on Monday the 3rd, because it does not “jump” over the boundary of a month’s days. The `W` character can only be specified when the day-of-month is a single day, not a range or list of days. **TIP:** You can combine the `L` and `W` characters for the day-of-month expression to yield `LW`, which translates to “last weekday of the month.”
pound sign (#)	The pound sign (`#`) character is allowed for the day-of-week field. This character is used to specify “the nth” *xxx* day of the month.	The value of `6#3` in the day-of-week field means the third Friday of the month (day `6` = Friday and `#3` = the 3rd one in the month). **Other Examples:** `2#1` specifies the first Monday of the month and `4#5` specifies the fifth Wednesday of the month. However, if you specify `#5` and there are fewer than 5 of the given day-of-week in the month, no firing occurs that month.

NOTE: The legal characters and the names of months and days of the week are not case sensitive. `MON` is the same as `mon`.

You can specify days in two fields: month day and weekday. If both are specified in an entry, they are cumulative, meaning that both of the entries are executed.

Examples of Cron Syntax

The following table shows examples of full cron expressions and their respective meanings.

Table 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

Cron Expression Example	Description
`0 * 14 * * ?`	Fire every minute starting at 2:00 p.m. and ending at 2:59.p.m., every day
`0 0/5 14 * * ?`	Fire every five minutes starting at 2:00 p.m. and ending at 2:55 p.m., every day
`0 0/5 14,18 * * ?`	Fire every five minutes starting at 2:00 p.m. and ending at 2:55 p.m., **and** fire every five minutes starting at 6:00 p.m. and ending at 6:55 p.m., every day
`0 0-5 14 * * ?`	Fire every minute starting at 2:00 p.m. and ending at 2:05.p.m., every day
`0 10,44 14 ? 3 WED`	Fire at 2:10 p.m. and at 2:44 p.m. every Wednesday in the month of March
`0 15 10 ? * MON-FRI`	Fire at 10:15 a.m. every Monday, Tuesday, Wednesday, Thursday and Friday
`0 15 10 15 * ?`	Fire at 10:15 a.m. on the 15th day of every month
`0 15 10 15 * ?`	Fire at 10:15 a.m. on the last day of every month
`0 15 10 ? * 6L`	Fire at 10:15 a.m. on the last Friday of every month
`0 15 10 ? * 6L 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 37

- Section 3.2.2, "Creating a New Schedule for the Job," on page 38
- Section 3.2.3, "Defining the New Schedule," on page 39
- Section 3.2.4, "Activating the New Schedule," on page 44
- Section 3.2.5, "Running the New Schedule Immediately," on page 44

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.

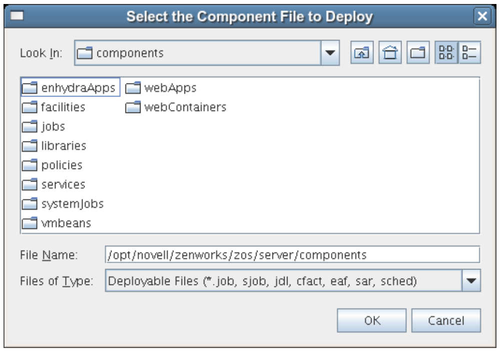
For this walkthrough, you deploy one of several system jobs (`auditCleaner.job`) developed for PlateSpin Orchestrate customers to demonstrate how system jobs are deployed and run. This job package, which is actually a `.jar` archive, includes only a `.policy` component and a `.jdl` component. It does 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.



File Name: /opt/novell/zenworks/zos/server/components

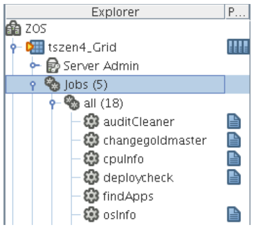
Files of Type: Deployable Files (*.job, sjob, jdl, cfact, eaf, sar, sched)

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.

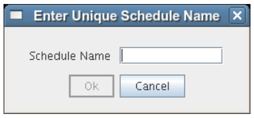


3.2.2 Creating a New Schedule for the Job

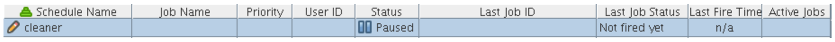
When a job has been deployed, you can create a schedule to specify when you want it to run. In this walkthrough, you create a schedule for the `auditCleaner` job by using the Scheduler tool in the Orchestrate Development Client.

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 39 to define this new schedule by adding the specific information you want.

3.2.3 Defining the New Schedule

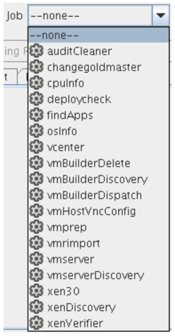
Defining a new job schedule consists of selecting its general properties, its specific properties, and the triggers you want to be associated with it. This section includes the following information:

- "Choosing General Properties for a New Schedule" on page 39
- "Creating and Assigning a Time Trigger for the New Schedule" on page 40
- "(Optional) Adding Specific Parameters to the New Schedule" on page 42

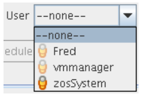
Choosing General Properties for a New Schedule

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.

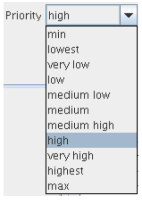


2. From the list of available jobs, select *auditCleaner* as the job to which this schedule applies.
3. In the Job Schedule Editor, select the *User* drop-down list.



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.



6 From the list of available priorities, select *high* as the priority for this job schedule.

The maximum selectable priority is dependent on an attribute associated with the selected user.

7 Click the *Save* 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.

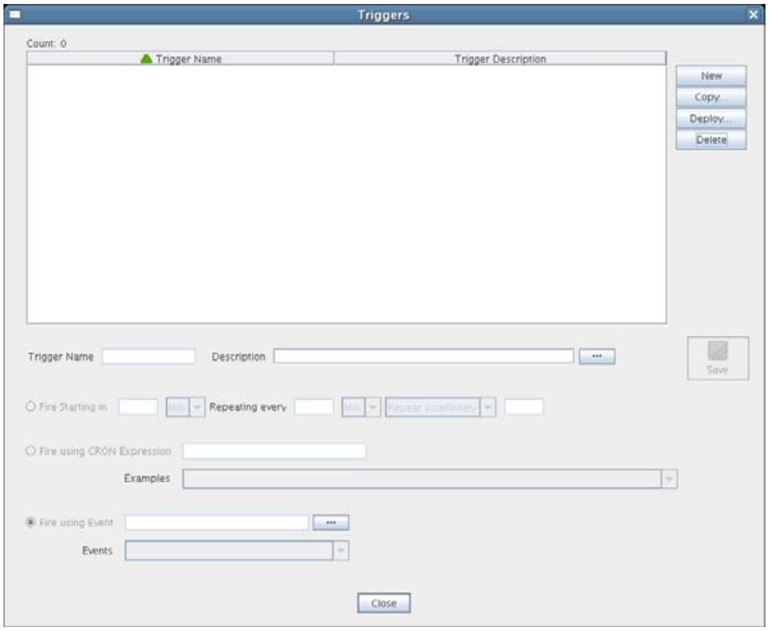
When you have chosen the general properties of the new schedule, you can either continue by "(Optional) Adding Specific Parameters to the New Schedule" on page 42 or by proceeding directly to "Creating and Assigning a Time Trigger for the New Schedule" on page 40.

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.

1 In the Job Schedule view, click *Edit Triggers* to display the Triggers dialog box.

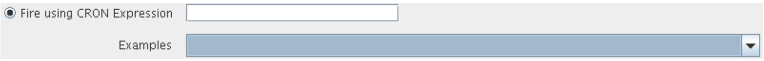


Time triggers are shareable across schedules. After a time trigger is defined, it is added to a list of triggers in this dialog box. You can select a predefined trigger from this list when you create a new schedule, or you can create a new time trigger, as the next steps demonstrate.

NOTE: For detailed information about cron syntax, see "Understanding Cron Syntax in the Job Scheduler" on page 33.

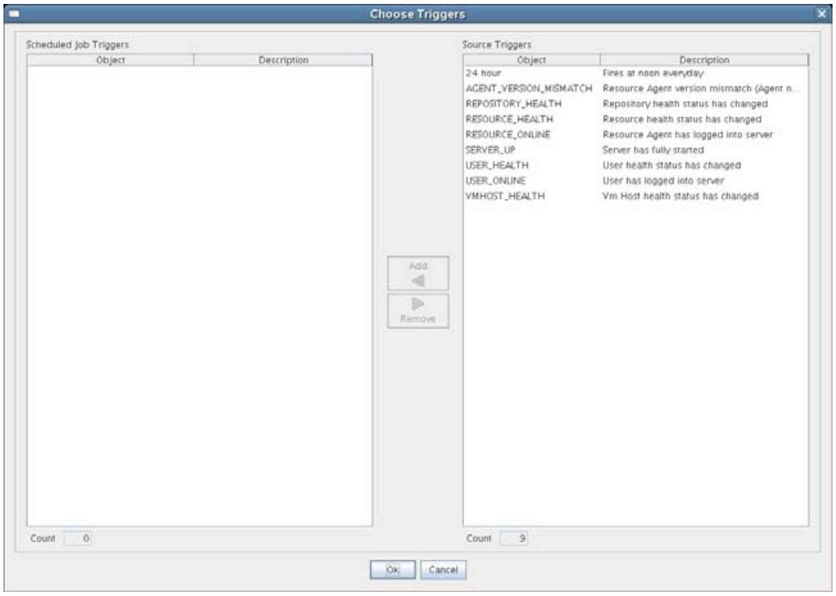
---

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.

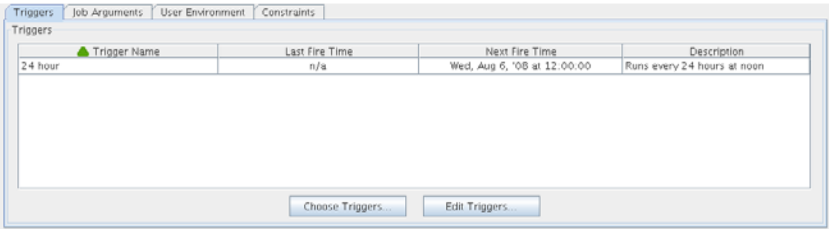


Object	Description
24 hour	Fires at noon everyday
AGENT_VERSION_MISMATCH	Resource Agent version mismatch (Agent n...
REPOSITORY_HEALTH	Repository health status has changed
RESOURCE_HEALTH	Resource health status has changed
RESOURCE_ONLINE	Resource Agent has logged into server
SERVER_UP	Server has fully started
USER_HEALTH	User health status has changed
USER_ONLINE	User has logged into server
VMHOST_HEALTH	Vm Host health status has changed

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.



Trigger Name	Last Fire Time	Next Fire Time	Description
24 hour	n/a	Wed, Aug 6, '08 at 12:00:00	Runs every 24 hours at noon

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:

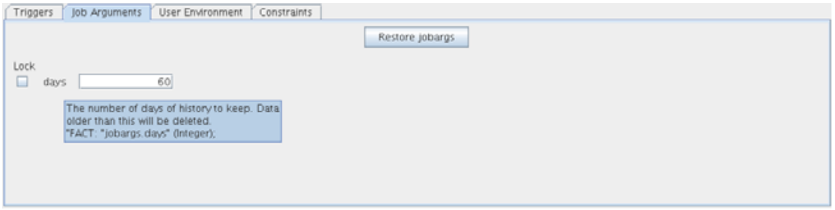
- "Job Arguments" on page 42
- "User Environment" on page 43
- "Constraints" on page 43

Job Arguments

As explained in Section 3.1.2, "Creating or Modifying a Job Schedule," on page 24, 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*



☐ days 60

The number of days of history to keep. Data older than this will be deleted.
"FACT: "jobargs.days" (Integer);

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.

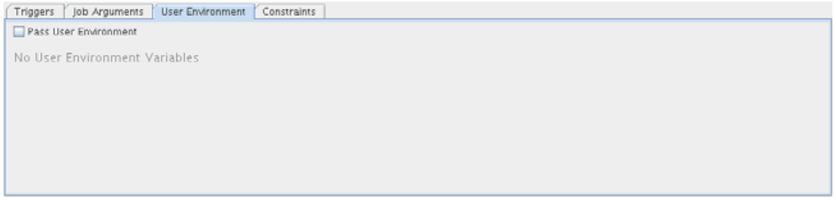
If the default value for a job argument parameter is missing, the job might fail, so you should inspect these parameters carefully.

User Environment

As explained in Section 3.1.2, "Creating or Modifying a Job Schedule," on page 24, 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*



☐ Pass User Environment

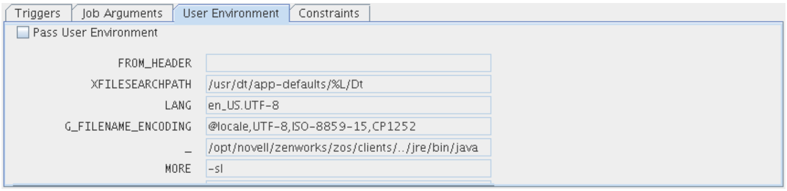
No User Environment Variables

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 Schedule Editor, User Environment Variables Available*



☐ Pass User Environment

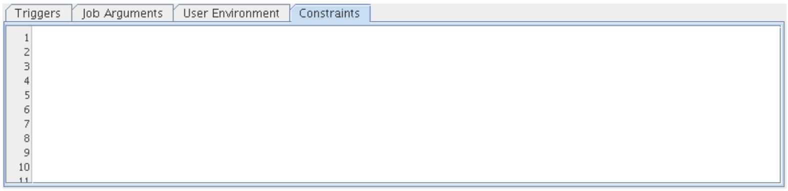
FROM_HEADER	
XFILESEARCHPATH	/usr/dt/app-defaults/%L/Dt
LANG	en_US.UTF-8
G_FILENAME_ENCODING	@locale,UTF-8,ISO-8859-15,CP1252
_	/opt/novell/zenworks/zos/clients/../jre/bin/java
MORE	-sl

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 24, 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.

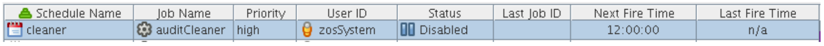
Click *Save* icon to update the Orchestrate Server with the new schedule.

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

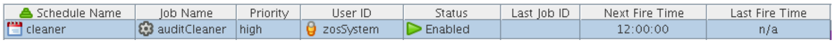
When the new schedule has been created and its triggers defined, you need to take it from the disabled state to an active state where it is ready to run.

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.



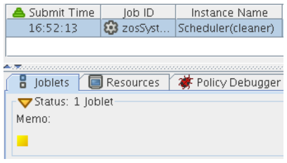
Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Next Fire Time	Last Fire Time
cleaner	auditCleaner	high	zosSystem	Enabled		12:00:00	n/a

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.



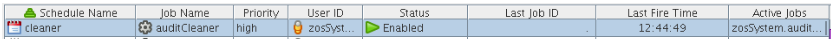
Submit Time	Job ID	Instance Name
16:52:13	zosSyst...	Scheduler(cleaner)

Status: 1 Joblet

Memo:

The joblet icon shows that the job is running.

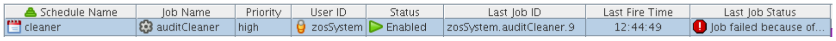
2 Click the Job Scheduler icon on the toolbar to open the Job Scheduler view.



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...

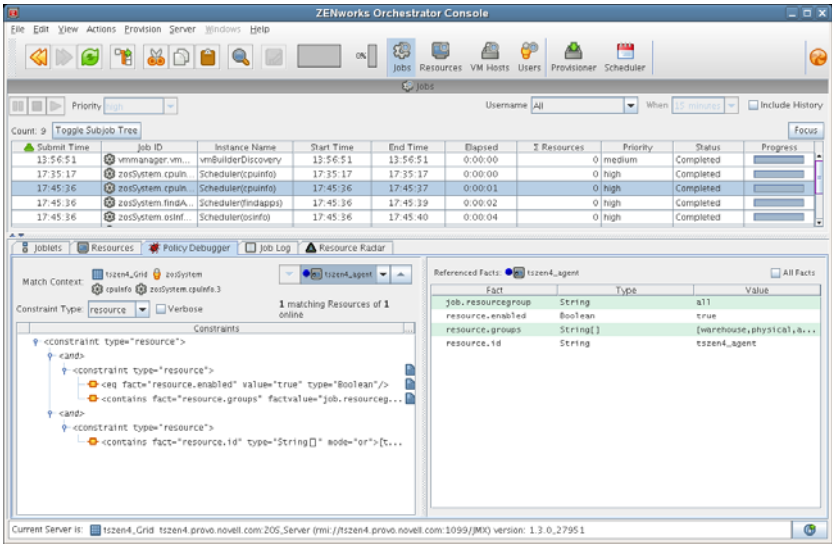
If the job invocation fails, as in this example, a red exclamation icon is also displayed.

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*



Priority: high

Username: All

When: 15 minutes

☐ Include History

Count: 9

Submit Time	Job ID	Instance Name	Start Time	End Time	Elapsed	Σ Resources	Priority	Status	Progress
13:56:51	vmmanager.vm...	vmBuilderDiscovery	13:56:51	13:56:51	0:00:00	0	medium	Completed	
17:35:17	zosSystem.cpuIn...	Scheduler(cpuinfo)	17:35:17	17:35:17	0:00:00	0	high	Completed	
17:45:36	zosSystem.cpuIn...	Scheduler(cpuinfo)	17:45:36	17:45:37	0:00:01	0	high	Completed	
17:45:36	zosSystem.findA...	Scheduler(findapps)	17:45:36	17:45:39	0:00:02	0	high	Completed	
17:45:36	zosSystem.osInf...	Scheduler(osinfo)	17:45:36	17:45:40	0:00:04	0	high	Completed	

Match Context: tszen4_Grid zosSystem cpuinfo zosSystem.cpuinfo.3

Constraint Type: resource ☐ Verbose

tszen4_agent

1 matching Resources of 1 online

Constraints

```
<constraint type="resource">
  <and>
    <constraint type="resource">
      <eq fact="resource.enabled" value="true" type="Boolean"/>
      <contains fact="resource.groups" factvalue="job.resourceg...
  <and>
    <constraint type="resource">
      <contains fact="resource.id" type="String[]" mode="or">[t...
```

Referenced Facts: tszen4_agent ☐ All Facts

Fact	Type	Value
job.resourcegroup	String	all
resource.enabled	Boolean	true
resource.groups	String[]	[warehouse,physical,a...
resource.id	String	tszen4_agent

Current Server is: tszen4_Grid tszen4.provo.novell.com:ZOS_Server (rmi://tszen4.provo.novell.com:1099/JMX) version: 1.3.0_27951

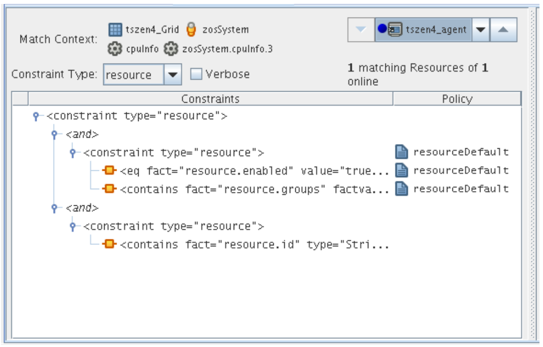
The Policy Debugger tab is also available in the VM Hosts view and in the Provisioner view of the Orchestrate Development Client.

- Section 4.1, "The Constraints Table View," on page 47
- Section 4.2, "The Facts Table View," on page 52
- Section 4.3, "Policy Debugger Use Cases," on page 53

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*



Match Context: tszen4_Grid zosSystem cpuInfo zosSystem.cpuInfo.3

tszen4_agent

Constraint Type: resource ☐ Verbose

1 matching Resources of **1** online

Constraints	Policy
`<constraint type="resource">`	
`<and>`	
`<constraint type="resource">`	resourceDefault
`<eq fact="resource.enabled" value="true...`	resourceDefault
`<contains fact="resource.groups" factva...`	resourceDefault
`<and>`	
`<constraint type="resource">`	
`<contains fact="resource.id" type="Stri...`	

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 49.

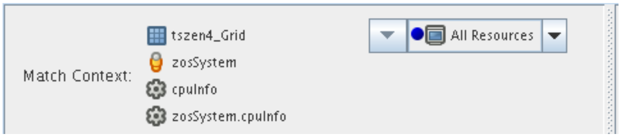
The Constraints Table View is composed of several parts:

- Section 4.1.1, "The Match Context Area," on page 48
- Section 4.1.2, "The Constraint Type List," on page 49
- Section 4.1.3, "The Verbose Check Box," on page 49
- Section 4.1.4, "The Capable Resources Summary," on page 50
- Section 4.1.5, "The Constraints Column of the Constraints Table View," on page 50
- Section 4.1.6, "The Policy Column of the Constraints Table," on page 51

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*



Match Context: tszen4_Grid zosSystem cpuInfo zosSystem.cpuInfo

All Resources

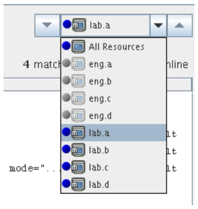
That identifying Facts in the Match Context include:

- 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

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 50.

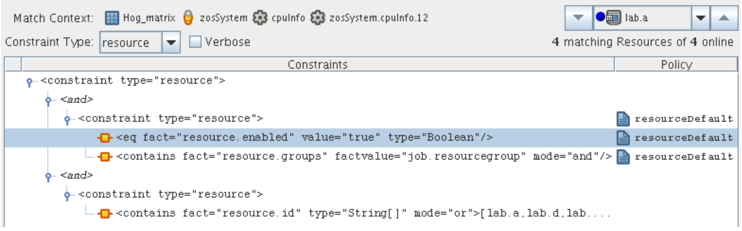
4.1.4 The Capable Resources Summary

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.

  Match Context: Hog_matrix zosSystem cpuInfo zosSystem.cpuInfo.12

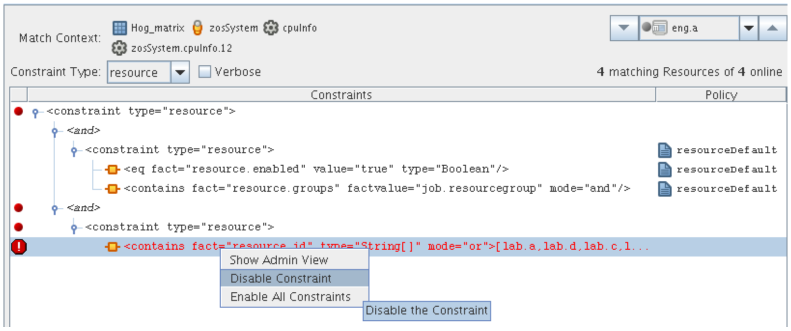
  lab.a

  Constraint Type: resource ☐ Verbose

  4 matching Resources of 4 online

  ```
  Constraints                                                                          Policy
  <constraint type="resource">
    <and>
      <constraint type="resource">                                                      resourceDefault
        <eq fact="resource.enabled" value="true" type="Boolean"/>                       resourceDefault
        <contains fact="resource.groups" factvalue="job.resourcegroup" mode="and"/>     resourceDefault
    <and>
      <constraint type="resource">
        <contains fact="resource.id" type="String[]" mode="or">[lab.a,lab.d,lab....
  ```

- 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.

  Match Context: Hog_matrix zosSystem cpuInfo zosSystem.cpuInfo.12

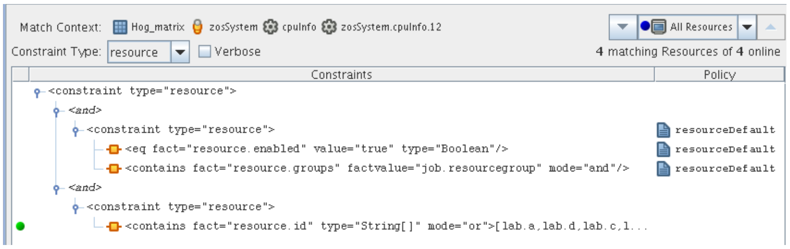
  eng.a

  Constraint Type: resource ☐ Verbose

  4 matching Resources of 4 online

  ```
     Constraints                                                                          Policy
  ●  <constraint type="resource">
       <and>
         <constraint type="resource">                                                      resourceDefault
           <eq fact="resource.enabled" value="true" type="Boolean"/>                       resourceDefault
           <contains fact="resource.groups" factvalue="job.resourcegroup" mode="and"/>     resourceDefault
  ●    <and>
  ●      <constraint type="resource">
  !          <contains fact="resource.id" type="String[]" mode="or">[lab.a,lab.d,lab.c,l...
  ```

- 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.



Match Context: Hog_matrix zosSystem cpuInfo zosSystem.cpuInfo.12

Constraint Type: resource ☐ Verbose

4 matching Resources of 4 online

```
                          Constraints                                            Policy
<constraint type="resource">
  <and>
    <constraint type="resource">                                                 resourceDefault
      <eq fact="resource.enabled" value="true" type="Boolean"/>                  resourceDefault
      <contains fact="resource.groups" factvalue="job.resourcegroup" mode="and"/> resourceDefault
  <and>
    <constraint type="resource">
●     <contains fact="resource.id" type="String[]" mode="or">[lab.a,lab.d,lab.c,l...
```

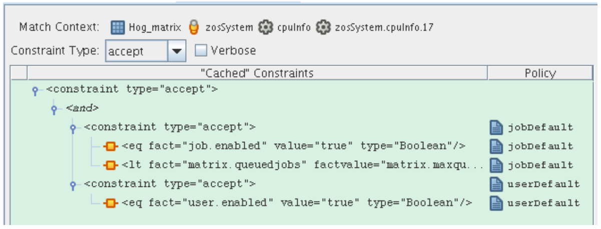
If you right-click a constraint in the table, a popup menu with three options is displayed. This menu lets you change the status of the constraint.

- **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*



Match Context: Hog_matrix zosSystem cpuInfo zosSystem.cpuInfo.17

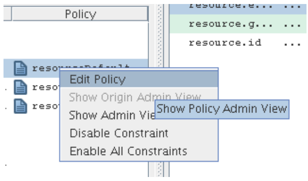
Constraint Type: accept ☐ Verbose

```
                  "Cached" Constraints                            Policy
<constraint type="accept">
  <and>
    <constraint type="accept">                                    jobDefault
      <eq fact="job.enabled" value="true" type="Boolean"/>        jobDefault
      <lt fact="matrix.queuedjobs" factvalue="matrix.maxqu...     jobDefault
    <constraint type="accept">                                    userDefault
      <eq fact="user.enabled" value="true" type="Boolean"/>       userDefault
```

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.

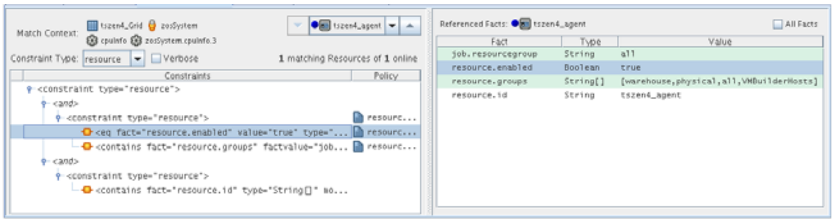
Figure 4-6 *The Popup Menu Launched from the Policy Column*



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*



Constraint Type: resource

☐ Verbose

1 matching Resources of 1 online

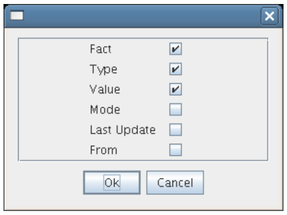
Constraints	Policy
<constraint type="resource">	
<and>	
<constraint type="resource">	resourc...
<eq fact="resource.enabled" value="true" type="...	resourc...
<contains fact="resource.groups" factvalue="job...	resourc...
<and>	
<constraint type="resource">	
<contains fact="resource.id" type="String[]" mo...	

Referenced Facts: tszen4_agent ☐ All Facts

Fact	Type	Value
job.resourcegroup	String	all
resource.enabled	Boolean	true
resource.groups	String[]	[warehouse,physical,all,VMBuilderHosts]
resource.id	String	tszen4_agent

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*



Fact ☑
Type ☑
Value ☑
Mode ☐
Last Update ☐
From ☐

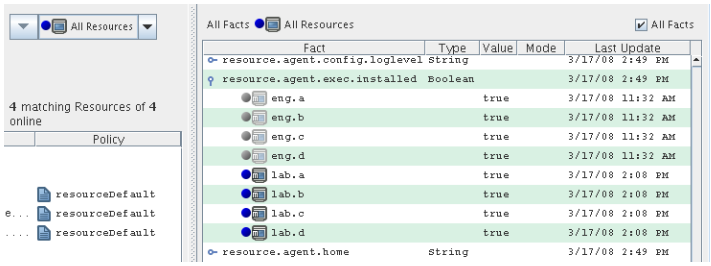
- Section 4.2.1, "The All Facts Check Box," on page 52

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 48) 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 matching Resources of 4 online

Policy
resourceDefault
resourceDefault
resourceDefault

All Facts All Resources ☑ All Facts

Fact	Type	Value	Mode	Last Update
resource.agent.config.loglevel	String			3/17/08 2:49 PM
resource.agent.exec.installed	Boolean			3/17/08 2:49 PM
eng.a		true		3/17/08 11:32 AM
eng.b		true		3/17/08 11:32 AM
eng.c		true		3/17/08 11:32 AM
eng.d		true		3/17/08 11:32 AM
lab.a		true		3/17/08 2:08 PM
lab.b		true		3/17/08 2:08 PM
lab.c		true		3/17/08 2:08 PM
lab.d		true		3/17/08 2:08 PM
resource.agent.home	String			3/17/08 2:49 PM

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:

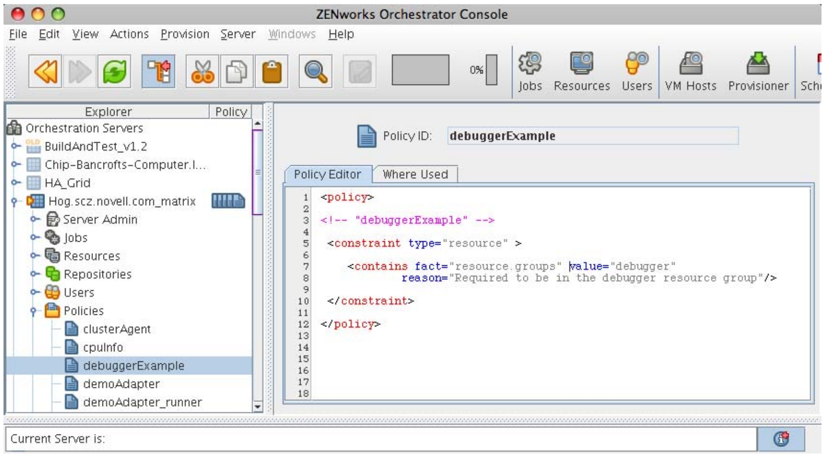
- Section 4.3.1, "Use Case 1: Determining Why a Job is in a Waiting State," on page 53

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.

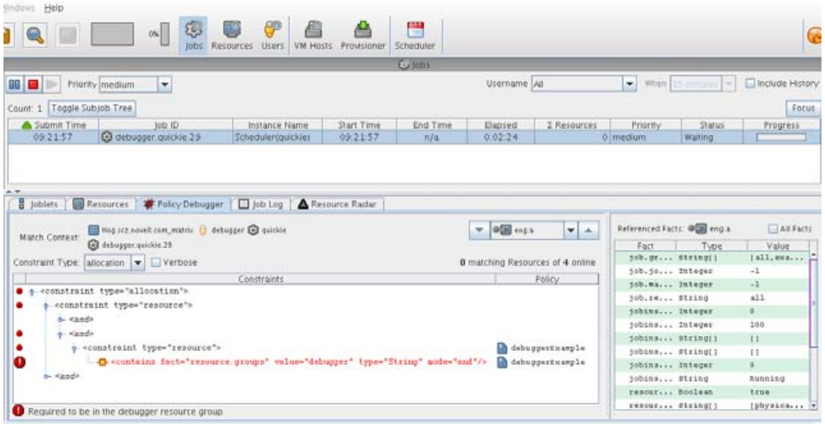
1. In the Orchestrate Development Client, create a user named `debugger`.
2. Deploy `quickie.job` from the `/examples` directory.
3. In the Orchestrate Development Client, create a schedule named `quickie`, specifying the `quickie` job and the `debugger` user.
4. In the Orchestrate Development Client, create a policy and name it `debuggerExample`. The policy needs to specify that the resource used belongs to the group called `debugger`.



Policy ID: debuggerExample

```
<policy>

<!-- "debuggerExample" -->

 <constraint type="resource" >

    <contains fact="resource.groups" value="debugger"
            reason="Required to be in the debugger resource group"/>

 </constraint>

</policy>
```

5. In the Orchestrate 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.



Priority: medium
Username: All
Count: 1

Submit Time	Job ID	Instance Name	Start Time	End Time	Elapsed	Σ Resources	Priority	Status	Progress
09:21:57	debugger.quickie.29	Scheduler(quickie)	09:21:57	n/a	0:02:24	0	medium	Waiting	

Match Context: Hog.icz.novell.com_matrix debugger quickie debugger.quickie.29

Constraint Type: allocation ☐ Verbose

0 matching Resources of 4 online

Constraints	Policy
`<constraint type="allocation">`	
`<constraint type="resource">`	
`<and>`	
`<and>`	
`<constraint type="resource">`	debuggerExample
`<contains fact="resource.groups" value="debugger" type="String" mode="and"/>`	debuggerExample
`<and>`	

Required to be in the debugger resource group

Referenced Facts: eng.a

Fact	Type	Value
job.gr...	String[]	[all,exa...
job.jo...	Integer	-1
job.ma...	Integer	-1
job.re...	String	all
jobins...	Integer	0
jobins...	Integer	100
jobins...	String[]	[]
jobins...	String[]	[]
jobins...	Integer	0
jobins...	String	Running
resour...	Boolean	true
resour...	String[]	[physica...

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 Panel

5

The PlateSpin® Orchestrate™ Development Client from Novell® lets you visualize the model maintained by the Orchestrate Server that you can use to make work allocation and Virtual Machine (VM) provisioning decisions. The left pane of the Orchestrate Development Client displays a hierarchical tree known as the Explorer Tree or the Explorer Window. The tree lets you navigate to different objects to see their details. When you navigate to these objects, you can edit their attributes and learn more detail about their configurations. The following sections provide more information about these objects.

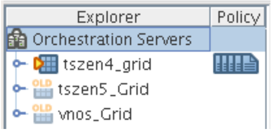
- Section 5.1, "Orchestrate Server Object," on page 55
- Section 5.2, "Server Admin Object," on page 63
- Section 5.3, "Grid Objects," on page 63
- Section 5.4, "Other Displayed Objects," on page 64

5.1 Orchestrate Server Object

The object highest in the Explorer Tree is the Orchestrate Server Object, sometimes called the "grid server" object because it represents the PlateSpin Orchestrate 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 polices, 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 5.4.1, "Policy Objects," on page 64.

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, "Info/Configuration Tab," on page 56
- Section 5.1.2, "Authentication Tab," on page 60

- Section 5.1.3, “Policies Tab,” on page 63
- Section 5.1.4, “Constraints/Facts Tab,” on page 63

5.1.1 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 56
- “Data Grid Configuration” on page 56
- “Security/TLS Configuration” on page 58
- “Agent/User Session Configuration” on page 59
- “Audit Database Configuration” on page 59
- “job.limits” on page 60

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 hostnames(s) or IP Address(es) associated with a Orchestrate Server when it configured in a High Availability configuration.

The <ellipsis> 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 <ellipsis> 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 umber 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 reauthenticate 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 reauthenticate 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 reauthenticate.
- **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 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 60
- "Users" on page 61

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 ellipsis 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 ellipsis 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 Policies Tab

The Polices 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 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 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 Grid Objects

The PlateSpin Orchestrate Development lets you visualize the model maintained by the PlateSpin Orchestrate Server that you can use to make work allocation and Virtual Machine (VM) provisioning decisions. Each object in the modeled world is referred to as a Grid object. The primary grid objects include:

- Users
- Resources (physical or virtual)
- Storage Repositories
- Jobs

Each of the Grid objects listed above can also be accumulated into respective containers called groups.

- Section 5.3.1, "Users," on page 64
- Section 5.3.2, "Resources," on page 64
- Section 5.3.3, "Storage Repositories," on page 64
- Section 5.3.4, "Jobs," on page 64

These Grid objects can be visualized in various ways, one of the most important of these depicts the health of the object and thus its readiness to work. For more information, see Appendix A, "Grid Object Health Monitoring," on page 67.

5.3.1 Users

Users are allowed to connect to the PlateSpin Orchestrate Server and run jobs. Administrator users are also allowed to connect by using the `zosadmin` command line and the PlateSpin Orchestrate Development Client user interfaces.

5.3.2 Resources

Resources are physical or virtual machines running a PlateSpin Orchestrate Agent, which can be scheduled for remote execution using joblets, or monitored for VM provisioning decisions.

5.3.3 Storage Repositories

Repositories are physical or virtual storage accessible for VM images or other use.

5.3.4 Jobs

Jobs are deployed to the Orchestration Server to automate processes, such as on-demand provisioning. For more information, see

5.4 Other Displayed Objects

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

- Section 5.4.1, "Policy Objects," on page 64
- Section 5.4.2, "Computed Fact Objects," on page 65
- Section 5.4.3, "Event Objects," on page 65
- Section 5.4.4, "Metrics," on page 65

5.4.1 Policy Objects

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.

Constraints

A policy may define a collection of constraints which are applied appropriately based on context. For example, a resource constraint may limit the selection of a resource to a subset based on resource group membership, or any number of other fact-based evaluations.

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.

5.4.2 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. To see a computed fact example, open `activejobs.cfact` in the `examples/activejobs.cfact directory`.

5.4.3 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 28 of this guide, and Section A.2, “Health Events,” on page 69.

5.4.4 Metrics

The Metrics Object is not visible in the Explore unless you start the PlateSpin Orchestrate Development Client in the “Advanced” mode. For more information, see...

Grid Object Health Monitoring

A

This section includes the following information:

- Section A.1, “Health Facts,” on page 67
- Section A.2, “Health Events,” on page 69

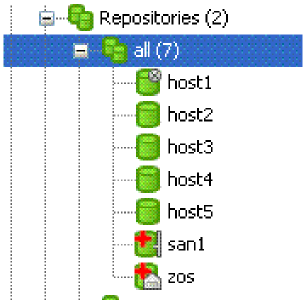
A.1 Health Facts

Starting with PlateSpin® Orchestratel 2.0, the Resource grid object, the Vmhost 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 Orchestrate 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*

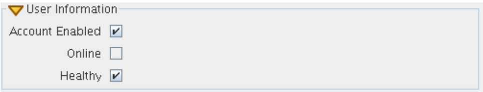


You can define what constitutes the health or non-health of the grid object by setting this health fact. The health fact can be set or cleared in several ways:

- Explicitly set or cleared by the administrator using tools in the PlateSpin Orchestrate 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* checkbox 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*



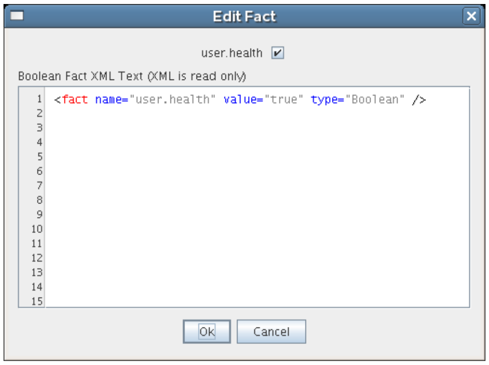
Account Enabled ☑

Online ☐

Healthy ☑

- On the Constraint/Fact fact page of a grid object, right click the `xxx.health` fact name, then click *Edit/View Fact* to open the Edit Fact dialog box.

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



user.health ☑

Boolean Fact XML Text (XML is read only)

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

You can set the health of the object by selecting or deselecting the health checkbox. Changing the value in the Development Client in this way has an immediate effect unless the value is overriden 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 71
- Section B.2, “Facts,” on page 71
- Section B.3, “Computed Facts,” on page 71

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*.