

Novell ZENworks® Orchestrator

1.3

June 9, 2008

ADMINISTRATION GUIDE

www.novell.com



Novell®

Legal Notices

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

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

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

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

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

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

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

Novell Trademarks

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

Third-Party Materials

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

Contents

About This Guide	7
1 The Policy Debugger	9
1.1 The Constraints Table View	9
1.1.1 The Match Context Area	10
1.1.2 The Constraint Type List	11
1.1.3 The Verbose Check Box	11
1.1.4 The Capable Resources Summary	12
1.1.5 The Constraints Column of the Constraints Table View	12
1.1.6 The Policy Column of the Constraints Table	13
1.2 The Facts Table View	14
1.2.1 The All Facts Check Box	14
1.3 Policy Debugger Use Cases	15
1.3.1 Use Case 1: Determining Why a Job is in a Waiting State	15
2 Understanding the Orchestrator Job Scheduler	17
2.1 The Job Scheduler View	17
2.1.1 Navigating The Job Schedules Table	18
2.1.2 Creating or Modifying a Job Schedule	20
2.1.3 Understanding Cron Syntax in the Job Scheduler	26
2.2 Walkthrough: Scheduling a System Job	29
2.2.1 Deploying a Sample System Job	29
2.2.2 Creating a New Schedule for the Job	30
2.2.3 Defining the New Schedule	31
2.2.4 Activating the New Schedule	36
2.2.5 Running the New Schedule Immediately	37
3 Security	39
3.1 User and Administrator Password Hashing Methods	39
3.2 User and Agent Password Authentication	39
3.3 Password Protection	40
3.4 TLS Encryption	40
3.5 Security for Administrative Services	41
3.6 Plain Text Visibility of Sensitive Information	41
A The zosadmin Command Line Tool	43
A.1 List of zosadmin Commands	43
A.2 Getting Started with the zosadmin Command	44
A.2.1 Logging In	44
A.2.2 Checking Login Status	45
A.2.3 Logging Out	45
A.3 Details, Usage, and Syntax Examples of zosadmin Commands	46
auditclean	47
auditcount	48
auditreport	49
cancelalljobs	50

create	51
deploy	54
disconnect	55
dump	56
get	57
init	58
invoke	59
list	60
login	61
logout	63
nodes	64
password	65
redploy	66
rotatelogs	67
sessions	68
set	70
start	71
status	73
stop	74
undeploy	75
upgrade	76
users	77
verify	78

About This Guide

This *Administration Guide* introduces Novell® ZENworks® Orchestrator 1.3, including its basic administration environment, which is accessed through the ZENworks Orchestrator Console and various command line tools. The guide provides an introductory overview of the Orchestrator and explains how it administers and manages work on the resources of the data center. The guide is organized as follows:

- ♦ Chapter 1, “The Policy Debugger,” on page 9
- ♦ Chapter 2, “Understanding the Orchestrator Job Scheduler,” on page 17
- ♦ Chapter 3, “Security,” on page 39
- ♦ Appendix A, “The zosadmin Command Line Tool,” on page 43

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 *Administration Guide*, visit the [ZENworks Orchestrator 1.3 Documentation Web site](http://www.novell.com/documentation/zen_orchestrator13/) (http://www.novell.com/documentation/zen_orchestrator13/).

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.

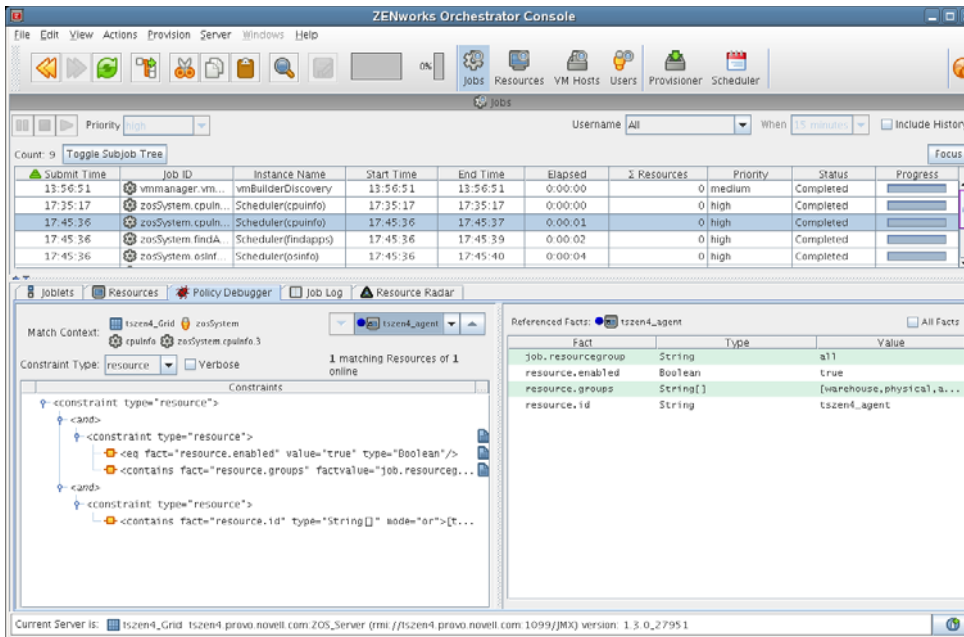
When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux* or UNIX, should use forward slashes as required by your software.

The Policy Debugger

1

The Policy Debugger is a tabbed page available in several views of the ZENworks Orchestrator Console. This tool helps you to determine the reasons for the current state of a job. The following figure shows the Policy Debugger tab opened in the Jobs view of the console.

Figure 1-1 Orchestrator Jobs View with the Policy Debugger Page Open



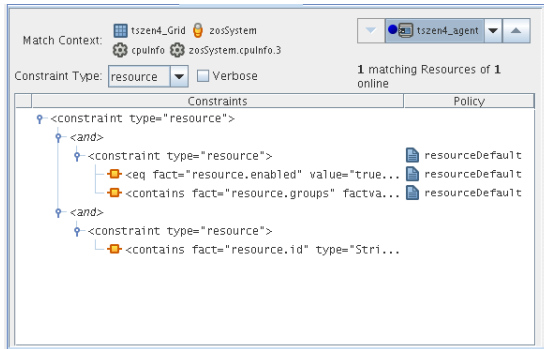
The Policy Debugger tab is also available in the VM Hosts view and in the Provisioner view of the console.

- ◆ [Section 1.1, “The Constraints Table View,” on page 9](#)
- ◆ [Section 1.2, “The Facts Table View,” on page 14](#)
- ◆ [Section 1.3, “Policy Debugger Use Cases,” on page 15](#)

1.1 The Constraints Table View

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

Figure 1-2 The Constraints Table View



The appearance of this view can change, depending on the constraint type you select in the drop down menu. For a description of these types, see [Section 1.1.2, “The Constraint Type List,”](#) on page 11.

The Constraints Table View is composed of several parts:

- ◆ [Section 1.1.1, “The Match Context Area,”](#) on page 10
- ◆ [Section 1.1.2, “The Constraint Type List,”](#) on page 11
- ◆ [Section 1.1.3, “The Verbose Check Box,”](#) on page 11
- ◆ [Section 1.1.4, “The Capable Resources Summary,”](#) on page 12
- ◆ [Section 1.1.5, “The Constraints Column of the Constraints Table View,”](#) on page 12
- ◆ [Section 1.1.6, “The Policy Column of the Constraints Table,”](#) on page 13




1.1.1 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 1-3 The Match Context Area of the Constraints Table View in the Policy Debugger



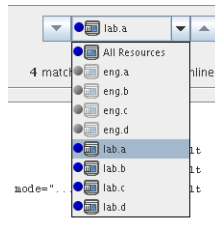
That identifying Facts in the Match Context include:

- ◆ Matrix: The  icon and a text string identifies the machine that matches the grid name given to the Orchestrator 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 1-4 Resource Drop Down List Showing Online and Offline Resources



1.1.2 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

1.1.3 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 1.1.5, “The Constraints Column of the Constraints Table View,”](#) on page 12

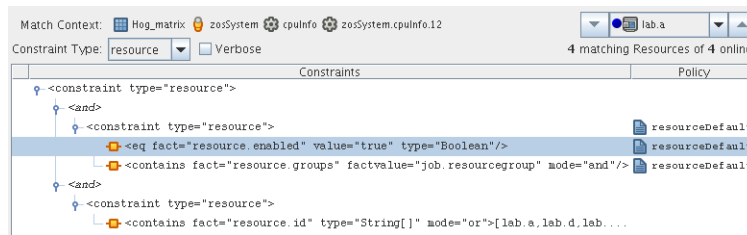
1.1.4 The Capable Resources Summary

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.

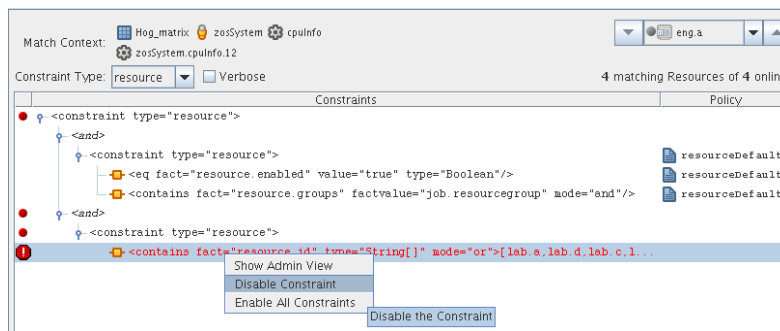
1.1.5 The Constraints Column of the Constraints Table View

The Constraints column of the constraints table view shows the logical hierarchy (that is, a “tree” format) of the constraints that are defined by the Policies associated with the Job. You can identify the status of the listed constraints by the icons that might be displayed in the far left column of the table:

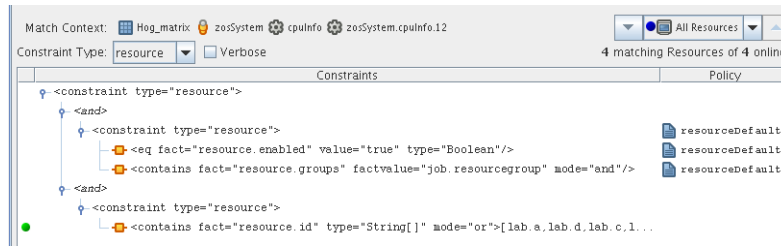
- ◆ no icon: The constraint passes the match. It is a “true” match. The figure below shows that the resource `lab.a` is available to run the job because all of its constraints match. No red icons are displayed next to any listed constraint.



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



- ◆ red octagonal icon: The constraint does not pass the match and is blocking the job. The figure above also shows the blocking constraint (red octagon).
- ◆ green dot icon: A blocking constraint has been disabled so that it behaves like a match. The figure below shows the green dot icon next to that the constraint that was formerly blocked and can now behave as a match.



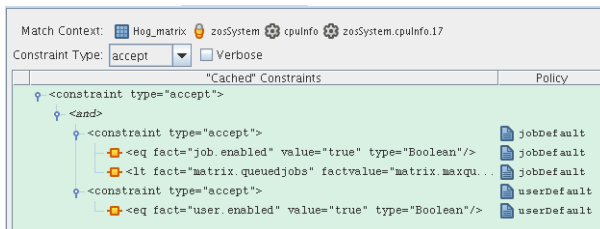
If you right-click a constraint in the table, a popup menu with three options is displayed. This menu lets you change the status of the constraint.

- ◆ **Show Admin View:** Select this option to open the Admin View for the specific resource selected.
- ◆ **Disable Constraint:** Select this option to disable (attach a green dot icon to) a constraint. Disabling a constraint with this function effectively makes it match, a condition that can prove useful if you want to perform a “what if” test without actually changing a policy.
- ◆ **Enable All Constraints:** Select this option if you have disabled one or more constraints during testing and you want to restore them to the enabled state.

Cached Constraints in the Constraints Column

When you change the constraint type in the Constraints Type List, the background of the table changes to green for some types. These are “cached” constraints that are saved with the job after it has completed. Their purpose is to help you debug the policy.

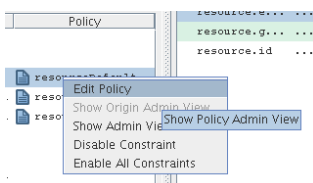
Figure 1-5 Cached Constraints Displayed in the Constraints Table View



1.1.6 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.

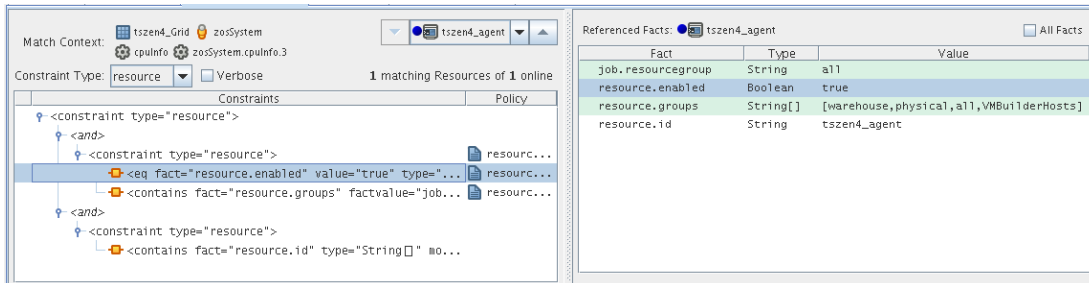
Figure 1-6 The Popup Menu Launched from the Policy Column



1.2 The Facts Table View

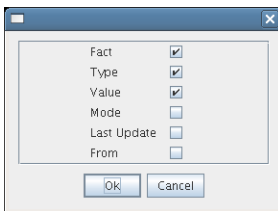
The Facts Table view displays the facts referenced in the Constraint Tree view for a specified Resource. Selecting a fact in the Constraint tree automatically selects that fact in the table.

Figure 1-7 The Constraints Table View and the Accompanying Facts Table View



If you right-click a column head in this table, a menu is launched where you can select the columns that you want to display.

Figure 1-8 Menu Used to Select the Columns Displayed in the Facts Table View of the Policy Debugger



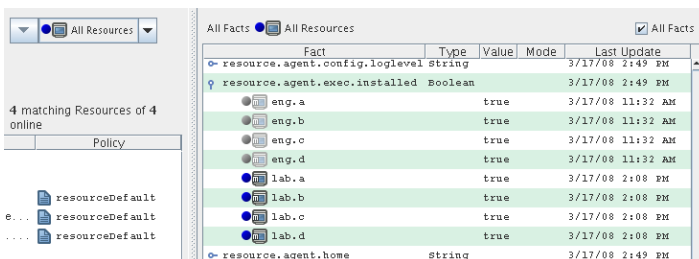
- ♦ [Section 1.2.1, “The All Facts Check Box,” on page 14](#)

1.2.1 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 1.1.1, “The Match Context Area,” on page 10](#)) 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 1-9 All Facts Check Box Selected with All Resources in Match Context



1.3 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:

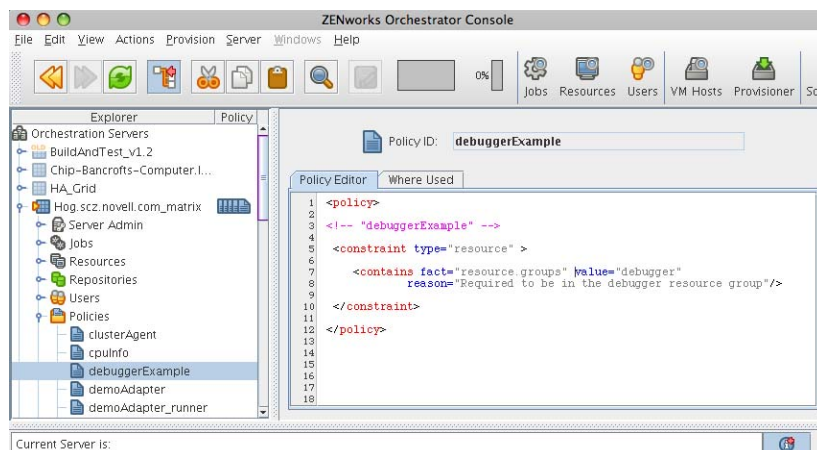
- ♦ [Section 1.3.1, “Use Case 1: Determining Why a Job is in a Waiting State,” on page 15](#)

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

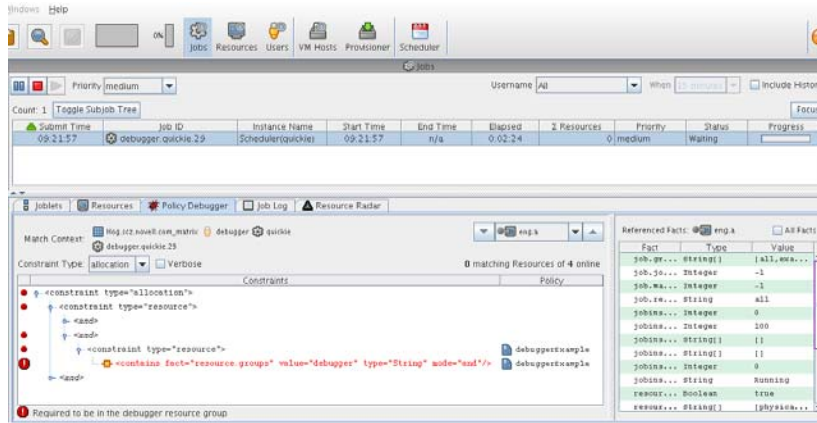
The objective of this use case is to run a job that sits in the waiting state and to use the policy debugger to identify why it is in the state and to make the necessary changes to get the job to run. The `quickie.job` is used along with a simple policy that specifies that the Resources that are to be used must be in a Resource group called `debugger`.

Use the following steps to recreate the use case.

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



- 5 In the console, associate the `debuggerExample` policy to the `quickie` job.
- 6 In the Job Scheduler view of the console, select the `quickie` schedule, then click *Run Now* to run the `quickie` schedule.
- 7 In the Job Monitor view of the console, select the *Policy Debugger* tab and verify that the job is in the waiting state.
- 8 In the Constraints Table view, open the *Constraint Type* drop down list, then select *Allocation*.
- 9 In the Match Context area of the Constraints Table view, open the Resource drop down list, then select any resource to refresh the Constraints Table and Facts Table views.



The Policy Debugger displays a red icon near the constraints that fail to match. The larger, red octagonal icon shows the particular constraint that is “blocking” and preventing the job from running on the resource. This is the constraint that is causing the job to be in a “waiting” state. The Constraints Table also displays the policy name (`debuggerExample`) that is contributing the constraint that is causing problems.

There are 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*.

Understanding the Orchestrator Job Scheduler

2

The Job Scheduler in Novell® ZENworks® Orchestrator Server can be used 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 ZENworks Orchestrator, 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 Orchestrator 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 Run On Resource Start event. 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.

Another example would be to run a job on server startup that sends a notification e-mail to an administrator.

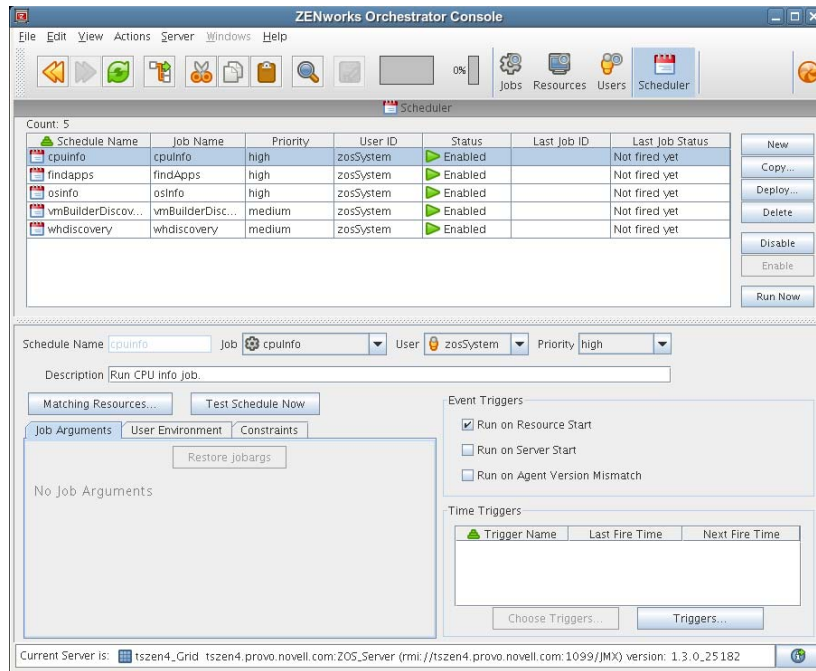
This section of the *ZENworks Orchestrator Administration Guide* includes the following information:

- ♦ [Section 2.1, “The Job Scheduler View,” on page 17](#)
- ♦ [Section 2.2, “Walkthrough: Scheduling a System Job,” on page 29](#)

2.1 The Job Scheduler View

Click *Scheduler* on the main toolbar of the ZENworks Orchestrator Console to open the Job Scheduler view.

Figure 2-1 Job Scheduler View of the ZENworks Orchestrator Console



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

- ◆ [Section 2.1.1, “Navigating The Job Schedules Table,”](#) on page 18
- ◆ [Section 2.1.2, “Creating or Modifying a Job Schedule,”](#) on page 20
- ◆ [Section 2.1.3, “Understanding Cron Syntax in the Job Scheduler,”](#) on page 26

2.1.1 Navigating The Job Schedules Table

ZENworks Orchestrator 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 2-2 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.cpu...	success
findapps	findApps	high	zosSystem	Enabled		Not fired yet
osinfo	osinfo	high	zosSystem	Enabled		Not fired yet
vmBuilderDiscov...	vmBuilderDisc...	medium	zosSystem	Enabled		Not fired yet
whdiscovery	whdiscovery	medium	zosSystem	Enabled		Not fired yet

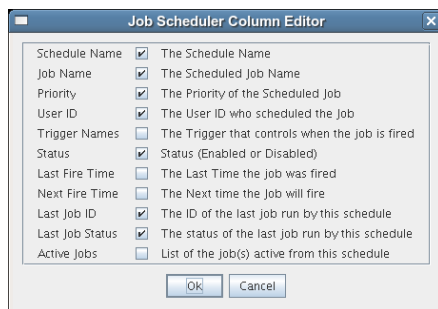
By default, ZENworks Orchestrator uses schedule names that are similar to the job name so that schedules are easy to match (although this is not required). The schedules list shows all of the existing job schedules that accompany predefined jobs, along with the schedules that you create in the Job Scheduler.

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 2-3 Job Scheduler Column Editor Dialog Box



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

In the Job Scheduler view, there are seven function buttons next to the Job Schedules Table 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.

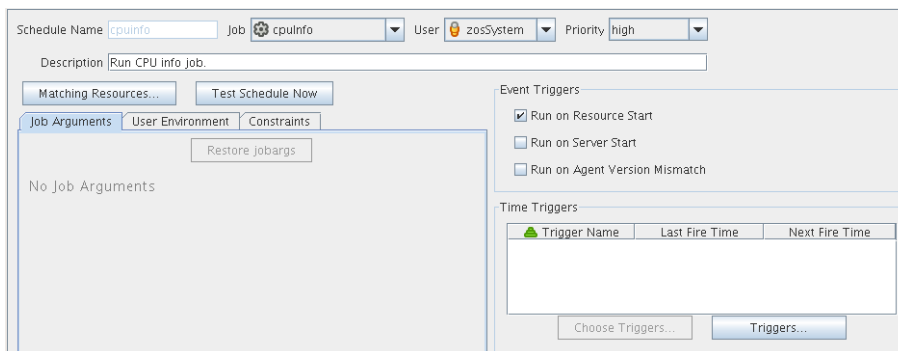
NOTE: Deleting a schedule that was deployed as part of a `.job` or `.sar` displays a confirmation dialog box. Deleting the schedule undeploys all contents of the `.job` or `.sar` that contains the schedule.

- ♦ **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*.

2.1.2 Creating or Modifying a Job Schedule

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

Figure 2-4 The Job Schedule Editor in the Job Scheduler View



There are several times when you can use this part of the Job Scheduler tool:

- ♦ When you create a new schedule by clicking *New*
- ♦ When you modify the attributes of an existing schedule (available when you select a schedule in the table)
- ♦ When you create a copy of an existing schedule by clicking *Copy*

The Job Schedule Editor lets you create or modify a job schedule by specifying its attributes.

You can use the following controls and data when you create or modify a job schedule:

- ♦ “Schedule Name” on page 21
- ♦ “Job” on page 21
- ♦ “User” on page 21
- ♦ “Priority” on page 21
- ♦ “Description” on page 21
- ♦ “Matching Resources” on page 21
- ♦ “Test Schedule Now” on page 22
- ♦ “Job Arguments” on page 22
- ♦ “User Environment” on page 22

- ♦ “Constraints” on page 23
- ♦ “Event Triggers” on page 23
- ♦ “Time Triggers” on page 24
- ♦ “Choose Triggers” on page 25

Schedule Name

When you create a new schedule, the unique name you specify is displayed in this field. If you select a schedule from the Job Schedules Table, the name of the schedule is displayed in this field. The field is not editable, because schedules cannot be renamed after they are created. (You can use a copy if this is required.)

Job

When you create a new schedule, you need to associate a deployed job with it. You can select the job you want to run from this drop-down list.

If you want to use a previously created schedule to run a different job, you can change the job here.

User

When you create a new schedule, you need to associate a user with it. The user represents the user for whom the job will run. The choice of user might affect the permissions, privileges and constraints of the job. You can select the user from this drop-down list.

If you want a different user to run a job on a previously created schedule, you can change the user here.

If you decide to change the user who runs the job, check the *Priority* field to make sure that the priority you want is selected.

Priority

When you create a new schedule and associate a job and a user with it, a list of possible run priorities becomes available in this drop-down list. The list of priorities varies, depending on the user that is specified in the previous field. In this field, you select the priority of the job that is to be run so that if other jobs are to start concurrently or are competing for resources, the Orchestrator 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.

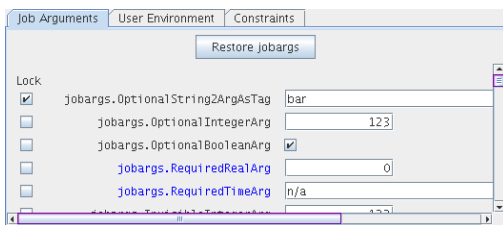
Test Schedule Now

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.

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 2-5 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 *Novell ZENworks Orchestrator 1.3 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 the Orchestrator 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 Orchestrator Console.

- ◆ `zos login --user=foo --env=PATH`

When the user logs in, he or she can specify one or more environment variables to use at login. The example above would result in just the `PATH` environment variable being uploaded.

Multiple environment variables can be specified by delimiting with a comma, as in the following example:

```
--env=PATH,LD_PATH,LD_LIBRARY_PATH
```

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.

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 trigger can be used in conjunction with a time trigger to allow flexibility in scheduling the job application for maximum effectiveness or convenience.

The Job Scheduler has three possible event triggers: *Run on Resource Start*, *Run on Server Start*, and *Run on Agent Version Mismatch*. The first two triggers are self-explanatory. The third trigger, *Run on Agent Version Mismatch*, triggers the job when an Orchestrator Agent of an incompatible version attempts to connect to this Orchestrator Server. It is used to initiate a configuration management tool for an agent software update. Jobs triggered by *Agent Version Mismatch* must specify a job argument (jobarg) named "resource.id" as in the following example:

```
<jobargs>
  <fact name="resource.id"
    type="String"
    description="The id of the mismatched resource"
    value=""
    visible="true" />
  <fact name="resource.ip"
    type="String"
    description="IP address of the mismatched resource"
    value=""
```

```
        visible="true" />
</jobargs>
```

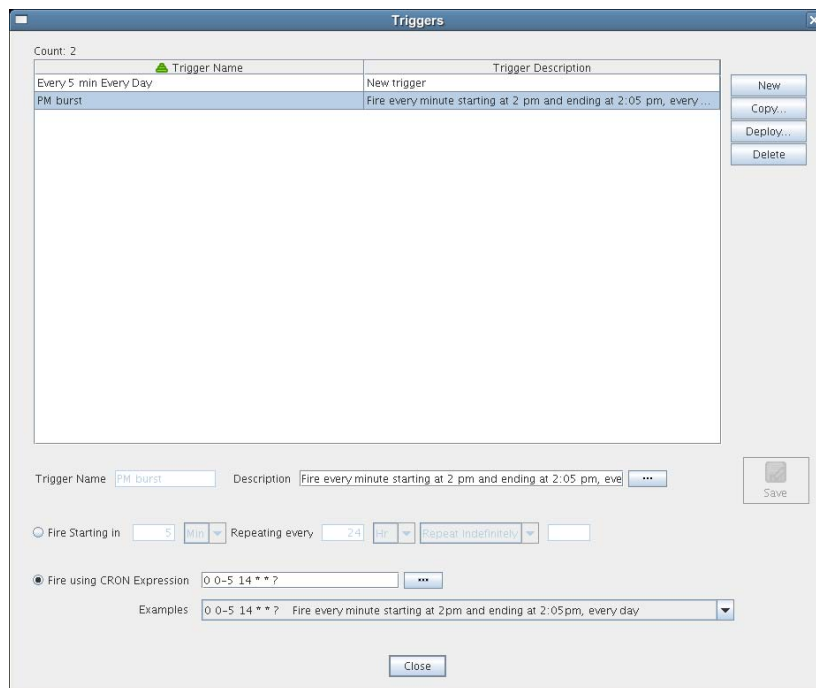
You can select any combination of these event triggers.

Time Triggers

A time trigger is the signal to the Job Scheduler to initiate, or “fire” a schedule at a given time. A time trigger can be used in conjunction with an event trigger to allow flexibility in scheduling the job.

When you click *Triggers* in the Job Scheduler view, the following dialog box opens:

Figure 2-6 The Time Trigger Dialog Box



In this dialog box, you can define the time triggers you want to associate with schedules. You can create as many time 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 time trigger.

The time triggers you create are retained by the Job Scheduler for you to choose from when you create a schedule for a job. The triggers are displayed in the list along with a description.

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.
- ♦ **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.

Click the button to open a dialog box where you can create and edit more lengthy time trigger descriptions.

- ♦ **Save:** 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 this 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.

A 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.

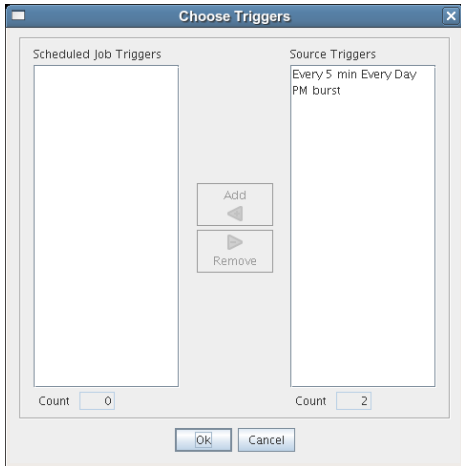
If you select this type of time trigger, the *Fire Starting In* button becomes inactive.

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 34](#). For detailed information about cron syntax, see [“Understanding Cron Syntax in the Job Scheduler” on page 26](#).

Choose Triggers

Opens a dialog box where you can choose the predefined time triggers you want to associate with this job schedule.

Figure 2-7 Choose Triggers Dialog Box



In this dialog box, you can click *Add* to move a selected trigger to the active, scheduled time 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.

2.1.3 Understanding Cron Syntax in the Job Scheduler

The cron triggers you can configure in the Orchestrator Job Scheduler use a standard crontab pattern-matching approach for deciding when to invoke job execution. This is based on the standard crontab format that you can find further described on the the *Elixir Schedule Designer* (<http://www.elixirtech.com/release/Rep7.0.1/Schedule-Designer/ch03s03.html>) Web site, 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.

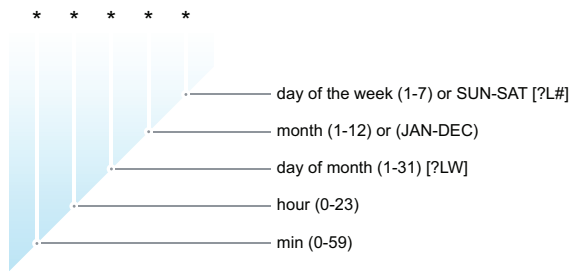
This section includes the following information:

- ♦ “Syntax” on page 26
- ♦ “Standard Operators” on page 27
- ♦ “Specialized Operators” on page 27
- ♦ “Examples of Cron Syntax” on page 28

Syntax

The crontab specification has five fields that are used to specify day, date, and time. The figure below shows how these fields are used.

Figure 2-8 Cron Syntax



Standard Operators

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

Table 2-1 Standard Operators in Orchestrator Cron Syntax

Operator	Purpose	Example
comma (,)	Specifies a list of values	1, 3, 4, 7, 8
dash (-)	Specifies a range of values	2-5, which is equivalent to 2, 3, 4, 5
asterisk (*)	Specifies all possible values for a field	An asterisk in the hour time field is equivalent to “every hour.”
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.

Specialized Operators

In addition to the operators explained in [Table 2-1](#), ZENworks Orchestrator uses other specialized operators in cron syntax:

- ♦ 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.
- ♦ The L character is allowed for the day-of-month and day-of-week fields. This character is short for “last,” but it has different meaning in each of the two fields. For example, 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.

- ♦ The `W` character is allowed for the day-of-month field. This character is used to specify the weekday (Monday-Friday) nearest the given day. As an example, 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.”

- ♦ 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. For example, 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: 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

A specification like the one below runs each day at 6:30 p.m.

```
30 18 * * *
```

Changing the parameter values causes this command to run with a different time schedule, as shown in the syntax examples in the table below.

Table 2-2 Results of Altered Cron Syntax on Execution Times

Minute	Hour	Day of the Month	Month	Day of the Week	Resulting Execution Time
30	0	1	1, 6, 12	*	At 00:30 hours on 1st of Jan, June, and December
0	20	*	10	1-5	At 8.00 p.m. every weekday (Mon.-Fri.) only in October
0	0	1, 10, 15	*	*	At midnight on the 1st, 10th, and 15th of every month
5, 10	0	10	*	1	At 12:05 and 12:10 every Monday and on the 10th of every month

2.2 Walkthrough: Scheduling a System Job

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

This section includes the following information:

- ♦ [Section 2.2.1, “Deploying a Sample System Job,” on page 29](#)
- ♦ [Section 2.2.2, “Creating a New Schedule for the Job,” on page 30](#)
- ♦ [Section 2.2.3, “Defining the New Schedule,” on page 31](#)
- ♦ [Section 2.2.4, “Activating the New Schedule,” on page 36](#)
- ♦ [Section 2.2.5, “Running the New Schedule Immediately,” on page 37](#)

2.2.1 Deploying a Sample System Job

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

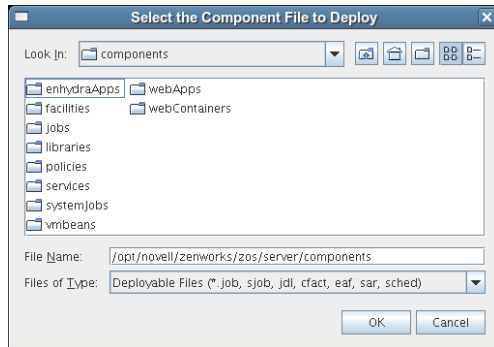
For this walkthrough, you deploy one of several system jobs (`auditCleaner.job`) developed for ZENworks Orchestrator customers to demonstrate how system jobs are deployed and run. This job package, which is actually a `.jar` archive, includes only a `.policy` component and a `.jdl` component. It does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

NOTE: An Orchestrator Job Developer can create and package jobs that include a `.jdl` file, a `.policy` file, 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 Orchestrator, 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.

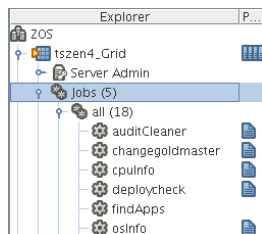
For more information about developing jobs and schedules in a job archive, see “[Job Scheduling](#)” in the *Novell ZENworks Orchestrator 1.3 Developer Guide and Reference*.

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 ZENworks Orchestrator Console, right-click the *Jobs* container, then click *Deploy Job* to open the Select the Component File to Deploy dialog box.




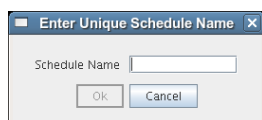
- 2 Open the *Look In* drop-down list, then navigate to the location of the job you want to deploy. Although a job developer can store Orchestrator jobs at any location on the network, the sample jobs shipped with ZENworks Orchestrator are limited to the directories where the product is installed. For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





2.2.2 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 Orchestrator Console.

- 1 From the toolbar in the Orchestrator Console, 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.



- 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 2.2.3, “Defining the New Schedule,”](#) on page 31 to define this new schedule by adding the specific information you want.

2.2.3 Defining the New Schedule

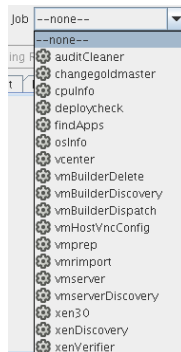
Defining a new job schedule consists of selecting its general properties, its specific properties, and the triggers you want to be associated with it. This section includes the following information:

- [“Choosing General Properties for a New Schedule”](#) on page 31
- [“\(Optional\) Adding Specific Parameters to the New Schedule”](#) on page 32
- [“Creating and Assigning a Time Trigger for the New Schedule”](#) on page 34

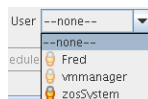
Choosing General Properties for a New Schedule

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:

- In the Job Schedule Editor panel of the Scheduler view, select the *Job* drop-down list.



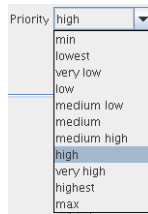
- From the list of available jobs, select *auditCleaner* as the job to which this schedule applies.
- In the Job Schedule Editor, select the *User* drop-down list.




- 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 *medium* as the priority for this job schedule.

7 Click the *Save* icon  on the toolbar of the Orchestrator Console to save the general properties you have selected for the new schedule.

The schedule is now committed, and the attribute columns in the Job Schedules Table are populated with the name of the job that the schedule will run, the user it will run as, the priority at which it will run, and its current status. Because the schedule has not been activated yet, it remains in a *Disabled* state.

When you have chosen the general properties of the new schedule, you can either continue by [“\(Optional\) Adding Specific Parameters to the New Schedule” on page 32](#) or by proceeding directly to [“Creating and Assigning a Time Trigger for the New Schedule” on page 34](#).

(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:

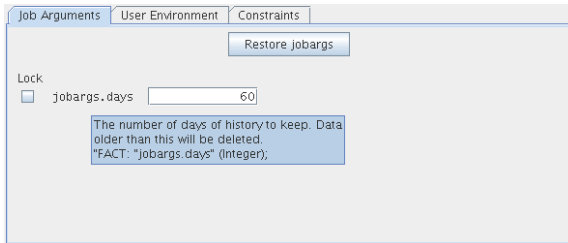
- ♦ [“Job Arguments” on page 32](#)
- ♦ [“User Environment” on page 33](#)
- ♦ [“Constraints” on page 34](#)

Job Arguments

As explained in [Section 2.1.2, “Creating or Modifying a Job Schedule,” on page 20](#), 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 2-9 The Job Arguments Tab of the Job Schedule Editor



According to the tool tip text, this argument is the number of days of job history to keep, so this job cleans up the history of the job in the Orchestrator 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.

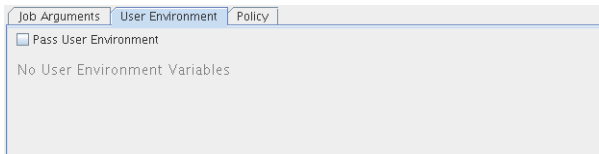
If the default value for a job argument parameter is missing, the job might fail, so you should inspect these parameters carefully.

User Environment

As explained in [Section 2.1.2, “Creating or Modifying a Job Schedule,”](#) on page 20, 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 2-10 The User Environment Tab of the Job Scheduler Editor; No User Environment Variables Available

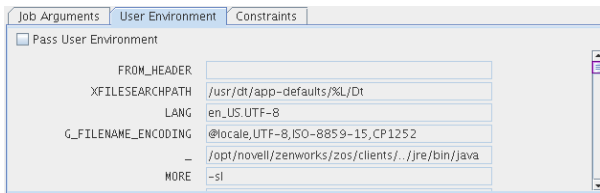


Because there are no environment variables listed, there are none to pass to the schedule, so it is not necessary to select the *Pass User Environment* check box. By default, this check box is not selected, even if environment variables are present for a user specified to run the job.

Sometimes a job is written to work from a user’s environment variables. In this case, if a user has not logged in or has not run the job from the `zos` command line using the necessary environment option, the schedule must pass those variables to the job when it is invoked.

If you 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 2-11 The User Environment Tab of the Job Schedule Editor, User Environment Variables Available

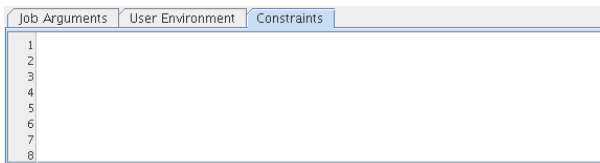


Selecting the *Pass User Environment* check box in this scenario would create these variables as facts used for this job invocation.

Constraints


As explained in [Section 2.1.2, “Creating or Modifying a Job Schedule,”](#) on page 20, the *Constraints* tab displays a constraint editor that you can use to create additional constraints for the job being scheduled.

Figure 2-12 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 Orchestrator Server with the new schedule.

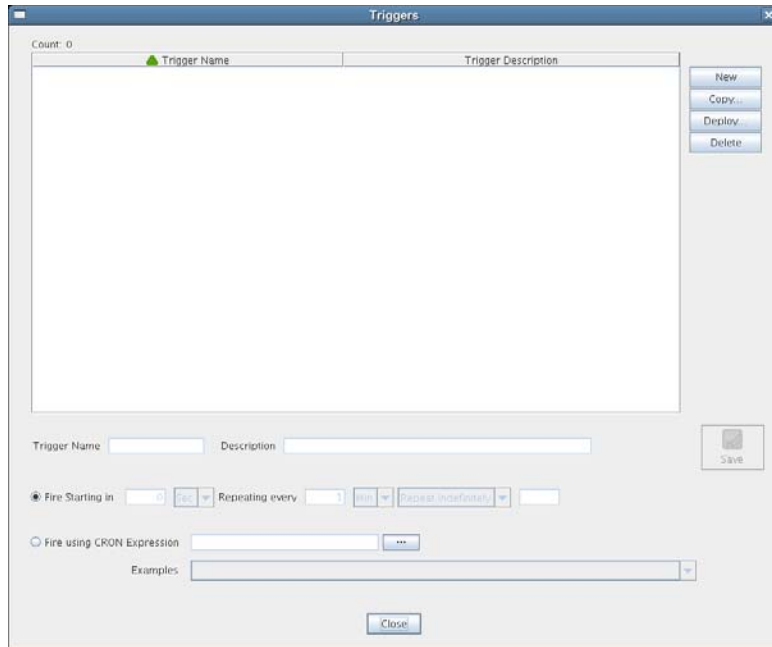
For more information about policies, see “[Developing Policies](#)” in the *Novell ZENworks Orchestrator 1.3 Developer Guide and Reference*.

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 *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 schedule 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 26.](#)

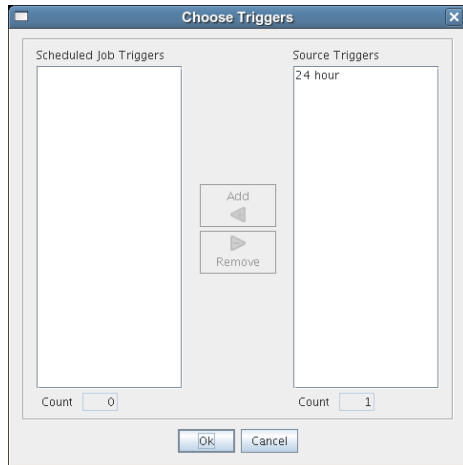
- 2 In the Triggers dialog box, click *New* to clear and activate the fields in the dialog box for the creation of a unique time trigger.
- 3 In the Enter Unique Trigger Name dialog box, specify `24 hour` as the unique name of this time trigger, then click *OK*.
- 4 In the *Description* field, specify `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.

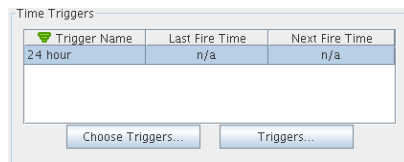
- 7 Click the *Save* icon  to save the trigger you just created, then click *Close* to return to the Job Scheduler view.
- 8 From the Job Scheduler view, make sure that the *cleaner* schedule is selected, then click *Choose Triggers* to display the Choose Triggers dialog box.




- 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 *Time Triggers* list of the Job Scheduler view, the *24 hour* trigger is now associated with the new schedule.



- Click the *Save* icon  to update the Orchestrator Server with the new schedule/trigger association.

2.2.4 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.

- 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	Last Job Status	Last Fire Time	Active Jobs
cleaner	auditCleaner	medium	zosSyst...	Paused		Not fired yet	n/a	

- Click *Enable* to activate the schedule.

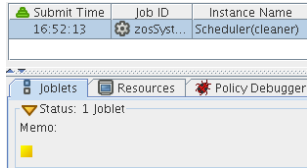
Schedule Name	Job Name	Priority	User ID	Status	Last Job ID	Last Job Status	Last Fire Time	Active Jobs
xenDiscovery	xenDiscovery	medium	zosSyst...	Active	zosSyst...	success	16:17:57	
cpuInfo	cpuInfo	high	zosSyst...	Active	zosSyst...	success	15:02:45	
vmBuilderDisc...	vmBuilderDisc...	medium	zosSyst...	Active	zosSyst...	success	14:43:33	
cleaner	auditCleaner	medium	zosSyst...	Active		Not fired yet	n/a	

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


2.2.5 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 to open the Job Monitor view.






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 Job Status	Last Fire Time	Active Jobs
 cleaner	auditCleaner	medium	zosSystem	 Active		Not fired yet	16:52:13	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	Last Job ID	Last Job Status	Last Fire T...	Active Jobs
 cleaner	auditCleaner	medium	zosSystem	zosSystem.auditCleaner.51	 Job was cancelled	16:52:13	

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

This section explains various security issues related to Novell® ZENworks® Orchestrator:

- ♦ [Section 3.1, “User and Administrator Password Hashing Methods,” on page 39](#)
- ♦ [Section 3.2, “User and Agent Password Authentication,” on page 39](#)
- ♦ [Section 3.3, “Password Protection,” on page 40](#)
- ♦ [Section 3.4, “TLS Encryption,” on page 40](#)
- ♦ [Section 3.5, “Security for Administrative Services,” on page 41](#)
- ♦ [Section 3.6, “Plain Text Visibility of Sensitive Information,” on page 41](#)

3.1 User and Administrator Password Hashing Methods

All passwords used in ZENworks Orchestrator are hashed using Secure Hash Algorithm-1 (SHA-1). User passwords are hashed with the user name as a “salt” to increase the cost of dictionary-based and brute force password-cracking strategies. The “salt” is extra data, hashed with the user’s password and subsequently concatenated with the hashed password. The use of a salt prevents a password cracker from simply comparing intercepted hashed passwords against a large pre-hashed dictionary.

The “salt” is not secret, but rather written in plain text. ZENworks Orchestrator includes it in the hashed portion of the password and prepends it (again in plain text) to the hashed result, so that the Orchestrator Server can prepend it when testing the password. By using a salt, ZENworks Orchestrator requires an individual hash for each password check during a brute force attempt at cracking. This makes it far more expensive, computationally, to execute a security breach.

WARNING: The `zosadmin` command line and the ZENworks Orchestrator Console do not use SSL encryption, nor do they support TLS/SSL, so they should only be used over a secure network.

All agent and client connections support TLS encryption. This includes the `zos` command line and the ZENworks Orchestrator Agent.

3.2 User and Agent Password Authentication

The ZENworks Orchestrator Server stores all user and agent passwords in its data store as double-hashed strings. The single-hashed password sent by the user is hashed again with a random salt chosen by the server and compared to the stored double-hashed password stored by the server for authentication. This procedure prevents a stolen server data store (including passwords) that could be used to try to crack the passwords of user accounts on the server, or to use those passwords on other servers.

This also means that the “user hashed” (that is, the single-hashed) password is used as the actual password. Because the password is sent over an SSL connection, it is secure, but even so, the single-hashed password is best not stored on disk. The single hashing for user passwords is done so that users who use the same password for multiple applications and Web sites do not have their accounts compromised if their Orchestrator password is intercepted while they are browsing network traffic.

This method is also useful for agents, since agents need to store their authentication credentials to disk in order to start up automatically without user intervention. It allows administrators to use user-friendly passwords without compromising the actual password string by storing it to disk.

3.3 Password Protection

You should take measures to protect the passwords of both the Orchestrator Server and Orchestrator Agents by ensuring that only the user account of the Orchestrator Server (currently `root` or `Administrator`, by default) has access to the `/store` and `/tls` directories on the server, so that general users are prevented from obtaining the password.

Currently, ZENworks Orchestrator restricts file access on the server, but we recommend that you further restrict shell accounts on server machines for general users as a precaution.

For users, none of the Novell-provided client utilities stores the user-entered password to disk in either plain text or hashed form. However, temporary once-per-session credentials are stored to the disk in the users `$HOME/.novell/zos/client` directory. Theft of this session credential could allow someone else to take over that user session, but not to steal the user's password. Users can protect their logged-in session by making sure the permissions either on their home directory or on the `~/ .novell/zos/client` directory are set to forbid both read and write access by other users.

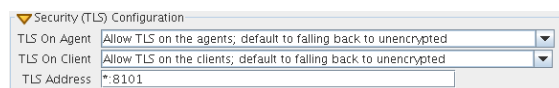
Orchestrator Agents use the same authentication protocol and password hashing as users (agent passwords are stored to disk in hashed form, not plain text) with the exception that agent passwords are not salted, allowing agents to be renamed by the server. Because agent passwords are not salted, we recommend that you generate and use random non-mnemonic strings for agent passwords.

3.4 TLS Encryption

ZENworks Orchestrator 1.3 uses Transport Layer Security (TLS) to provide encryption for both user and agent connections. By default, both the Orchestrator Agent and the Orchestrator Clients use TLS to initiate their connections to the Orchestrator Server, and then the server specifies whether to “fall back” to plain text or continue the session fully encrypted.

Although you can manually configure the agent and clients to either always require TLS encryption or to fully disable TLS encryption, we recommended that you leave the agents and clients in their default configuration, and then use the Orchestrator Console on the server to specify the default behavior. This is the purpose of the TLS Options section on the main server tab of the Orchestrator Console.

Figure 3-1 *TLS Options in the Orchestrator Console*



Here, there are 4 levels that you can set separately for both agent connections and user/client connections:

- ◆ **Forbid TLS for (agents/clients):** This option is to fully disable and prohibit TLS encryption altogether. This is the least secure option and is therefore usually not the desirable choice, but it could be required in countries that restrict encryption or in low security environments where performance is more critical than security.

- ◆ **Allow TLS on the (agents/clients); default to falling back to unencrypted:** This option (the factory default for both agents and clients) is to allow TLS encryption if the agent or client explicitly requests it, but to default to falling back to plain text after authentication.

NOTE: Authentication always occurs over SSL, regardless of settings.

- ◆ **Allow TLS on the (agents/clients); default to TLS encrypted if not configured encrypted:** This option is similar to the second option. Agents/clients may specify whether or not to use TLS, but if they use the default of “server specified,” the server defaults to using TLS.
- ◆ **Make TLS mandatory on the (agents/clients):** This option is the most secure, locked down option. It requires TLS at all times, and fails connections if the agent or the client tries to specify plain text.

In addition to these settings for TLS configuration, there are files that need to be protected on both the server and on the client/agent. For more information, search for the *TLS Certificate Installation On ZENworks Orchestrator* article at the *Novell Cool Solutions Community* (<http://www.novell.com/communities/cool solutions/>).

3.5 Security for Administrative Services

The ZENworks Orchestrator Console and the `zosadmin` command line tool are clients to the MBean and RMI servers. ZENworks Orchestrator does not provide encryption for these administrative services, so you should be careful to use them only in a secure environment.

The `zosadmin` tool stores the administrator’s user name and password in single-hashed form on disk to prevent the plain text user password from being stolen. Even so, this credential is still vulnerable and could be stolen to gain unauthorized access, so it must be protected.

The Orchestrator Console secrets are stored in `$HOME/.novell/zoc` and the `zosadmin` secrets are stored in `$HOME/.novell/zos/admin`. You need to protect both of these directories with permissions that restrict both Read and Write access by other users.

3.6 Plain Text Visibility of Sensitive Information

The following table outlines where sensitive information might be visible as plain text:

Table 3-1 *Locations Where Sensitive Information Might Be Stored As Plain Text*

Information	Storage Location	Visibility Issue
VM configuration data	MS SQLite database	The VM configuration data can be viewed by anyone.
Orchestrator Server credentials	VM Warehouse configuration file	<p>The VM Warehouse stores this information as plain text in a configuration file that is readable as root.</p> <p>By default, the repository is located in: <code>/etc/opt/novell/zenworks/vmwarehouse/vmwarehouse.conf</code></p> <p>If necessary, you can determine the configuration file’s location using the <code>vmr cn -a</code> command.</p>

Information	Storage Location	Visibility Issue
Audit Database configuration	ZOS properties store	Contains plain text information including user/ password for allowing ZENworks Orchestrator to log into the Audit Database (i.e. mysql) for logging. You should use a non-privileged database account for logging.

The zosadmin Command Line Tool

A

The zosadmin command line tool is used by Novell® ZENworks® Orchestrator Server administrators to log in to the server, add or remove server components such as jobs and policies, to report on the status of nodes, users, and the audit database, and to perform other administrative functions.

This section includes information about the following:

- ♦ [Section A.1, “List of zosadmin Commands,” on page 43](#)
- ♦ [Section A.2, “Getting Started with the zosadmin Command,” on page 44](#)
- ♦ [Section A.3, “Details, Usage, and Syntax Examples of zosadmin Commands,” on page 46](#)

A.1 List of zosadmin Commands

The following table includes a list of zosadmin commands in order of common usage, and a description for each command.

Table A-1 Available Zosadmin Commands and Their Descriptions

Command	Description
auditclean	Clean the audit database by removing old data
auditcount	Count the number of jobs in the audit database
auditreport	Generate an audit report
cancelalljobs	Cancel all running jobs
create	Create a new server instance
deploy	Deploy a new component onto a server
disconnect	Disconnect and/or revoke user or node sessions
dump	Dump contents of the namespace (advanced diagnostics)
get	Retrieve an attribute
help	Displays help for any of the commands in this list
init	Restore a server configuration to initial state
invoke	Invoke an mbean method (advanced diagnostics)
list	List the running servers
login	Log in to the server
logout	Log out of the server
nodes	Retrieve the list of active or inactive nodes
password	Change admin password

Command	Description
redeploy	Redeploy a component on a server
rotatelogs	Back up and rotate the log files
sessions	Display session information
set	Set an attribute
start	Start a local server
status	Collect the status of a server
stop	Shut down the server
undeploy	Undeploy a component from the server
upgrade	Upgrade old server snapshot to current version
users	Retrieve the list of active or inactive users
verify	Verify a component for syntax

A.2 Getting Started with the zosadmin Command

All zosadmin commands begin with zosadmin on the command line. The general format for a zosadmin command is zosadmin followed by the command name, followed by command line parameters, if needed:

```
zosadmin [standard_options] command [command_options_and_arguments]
```

Before you use the zosadmin CLI, make sure that your path is correctly pointing to the Orchestrator /bin directory.

This section includes the following information:

- ◆ [Section A.2.1, “Logging In,” on page 44](#)
- ◆ [Section A.2.2, “Checking Login Status,” on page 45](#)
- ◆ [Section A.2.3, “Logging Out,” on page 45](#)

A.2.1 Logging In

Login is required to operate on a running server. The commands start, list, init, and create do not require a login. Use the following syntax to log in:

```
>zosadmin login -user=username Orchestrator_Server_name
Please enter current password for 'username': ****
Logged into Orchestrator_grid_name> on Orchestrator_Server_name
```

The login should now be complete.

Login information is stored in the /home directory, so further zosadmin commands use the saved login information. To operate on a different Orchestrator Server, run zosadmin login to log in to the new Orchestrator Server.

A.2.2 Checking Login Status

Enter the following command and parameter to retrieve the status of the current login:

```
>zoadmin login -c
```

```
Currently logged into testgrid on server 'tszen5'
```

A.2.3 Logging Out

Enter the following command to log out of the Orchestrator Server:

```
>zoadmin logout
```

```
Logged out from testgrid
```

A.3 Details, Usage, and Syntax Examples of zosadmin Commands

This section includes a detailed list of the `zosadmin` commands you can use. It also includes examples for using these commands and shows the syntax of typical commands.

NOTE: Items shown in brackets [] are optional. Items shown in *italics* are contextual examples.

The section is organized according to the command names, which include the following:

- ♦ “auditclean” on page 47
- ♦ “auditcount” on page 48
- ♦ “auditreport” on page 49
- ♦ “cancelalljobs” on page 50
- ♦ “create” on page 51
- ♦ “deploy” on page 54
- ♦ “disconnect” on page 55
- ♦ “dump” on page 56
- ♦ “get” on page 57
- ♦ “init” on page 58
- ♦ “invoke” on page 59
- ♦ “list” on page 60
- ♦ “login” on page 61
- ♦ “logout” on page 63
- ♦ “nodes” on page 64
- ♦ “password” on page 65
- ♦ “redeploy” on page 66
- ♦ “rotatelogs” on page 67
- ♦ “sessions” on page 68
- ♦ “set” on page 70
- ♦ “start” on page 71
- ♦ “status” on page 73
- ♦ “stop” on page 74
- ♦ “undeploy” on page 75
- ♦ “upgrade” on page 76
- ♦ “users” on page 77
- ♦ “verify” on page 78

auditclean

This command cleans the audit database by removing old data.

Syntax

```
zosadmin auditclean      --dayskept= [--matrix=]
```

Options

-d, --dayskept=

Specify the number of days of history kept in the database (0 removes all).

-M=, --matrix=

Specify a different grid (the default grid is the server's grid, * means all).

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

auditcount

This command retrieves the count of the number of jobs in the audit database.

Syntax

```
zosadmin auditcount [--root] [matrix=]
```

Options

-R, --Root

Display the count for root jobs only (default counts all jobs).

-M=, --matrix=

Specify a different grid (default grid is the server's, * means all).

-v, --verbose

Turns on verbose output during this operation.

-V, --debug

Turns on debug output during this operation.

-h, --help

Displays a help message for this operation.

auditreport

This command displays an audit report generated from the audit database.

Syntax

```
zosadmin auditreport [--username=] [--from=] [--to=] [--limit=] [--matrix=] [--childjobs]
```

Options

-m, --username=

Specify the username of the user who ran the job (default is all users).

-F=, --from=

Select jobs submitted on or after this date. For example, `Fri, 12 Oct 2007` or `Fri, 12 Oct 2007 13:30:00`. The default day is the current day.

-T=, --to=

Select jobs submitted on or before this date. For example, `Fri, 12 Oct 2007` or `Fri, 12 Oct 2007 13:30:00`. The default day is the current time.

-L=, --limit=

Limit the number of jobs reported (the default is 500).

-M=, --matrix=

Specify a different grid (the default grid is the server's grid, * means all).

-I=, --childjobs

Include child jobs (the default is root jobs only).

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

cancelalljobs

This command cancels all running jobs.

Syntax

```
zosadmin cancelalljobs
```

Options

-v, --verbose

Turns on verbose output during this operation.

-V, --debug

Turns on debug output during this operation.

-h, --help

Displays a help message for this operation.

create

This command creates a new server instance.

Syntax

```
zosadmin create [--newdir=] [--upgrade=] [--profile=] [--id=] [--grid=] [--user=] [--passwd=] [--rmiport=] [--rmihost=] [--httpport=] [--adminport=] [--agentport=] [--auditurl=] [--audituser=] [--auditpasswd=] [--cert=] [--key=] [--<hostname>=]
```

Arguments

hostname

Specify the hostname to use for this server. The default is the system hostname.

Options

-N, --newdir=

The location where the new Orchestrator Server should be installed.

-G, --upgrade

Switch the upgrade behavior for snapshot handling.

-e, --profile=

Create a new instance based on named profile (the default is *server*).

-i, --id=

Select by kernel ID.

-g, --grid=

Select by grid name.

-u, --user=

Username used in accessing secure remote sites.

-p, --password=

Password used in accessing secure remote sites.

-P, rmiport=

Select by RMI port.

-H, --rmihost=

Select or specify the RMI host (can be different from *server host*).

-W=, --httpport=

Specify the http port used for the User Portal (the default is 80).

-M, --adminport=

Specify the http port used for Administrative Information Portal (default 8001).

-A, --agentport=

Specify the communication port for agent connections (default 8100).

-X, --auditurl=

Specify the connection URL to a PostgreSQL database (for example, jdbc:postgresql://localhost/matrix_db).

-Y, --audituser=

Specify the audit database user.

-Z, --auditpasswd=

Specify the audit database password.

-k, --cert=

Specify the .pem file containing the new server's TLS certificate.

-K, --key=

Specify the .pem file containing the new server's TLS key.

-v, --verbose

Turns on verbose output during this operation.

-V, --debug

Turns on debug output during this operation.

-h, --help

Displays a help message for this operation.

Examples

Example 1

To create a new server instance in the install directory (default), use the following command:

```
zosadmin create
```

Example 2

To create a new server instance using host name *myhost.mydomain.com*, use the following command:

```
zosadmin create myhost.mydomain.com
```

Example 3

To create a new server instance in the */zos/myserver/* directory, use the following command:

```
zosadmin create --newdir=/zos/myserver
```

Example 4

To create a new upgraded server instance in an existing directory, use the following command:

```
zosadmin create --newdir=/zos/myserver --upgrade
```

IMPORTANT: This command removes previous serve instance data from the directory!

deploy

This command deploys a new component onto a server.

Syntax

```
zosadmin deploy [--port=] file|dir
```

Arguments

file|dir

Specify the deployable file or a directory containing deployable files.

Options

-P=, --port=

The deployment port of the target server (the default is 8001, set with the `-M` option in the `create` command).

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To deploy the job file `quickie.job`, use the following command:

```
zosadmin deploy quickie.job
```

Example 2

To deploy a job file in the `/test/myjob.job` directory, use the following command:

```
zosadmin deploy /test/myjob.job
```

disconnect

This command allows forceful disconnect or revocation of user or node sessions.

Syntax

```
zosadmin disconnect [--id=] [--node=] [--user=] [--all] [--allUsers]
[--allNodes] [--revoke]
```

Options

-i, --id=*value*

Disconnect an active session by its session ID.

-n, --node=*value*

Disconnect a node by name.

-u, --user=*value*

Disconnect all of a user's sessions by name.

-a, --all

Disconnect all user and node sessions.

-U, --allUsers

Disconnect all user sessions.

-N, --allNodes

Disconnect all node sessions.

-r, --revoke

Revoke the session or sessions in addition to disconnecting.

-V, --debug

Turn on debug output during this operation.

-v, --verbose

Turn on verbose output during this operation.

-h, --help

Display a help message for this operation.

dump

This command dumps the contents of namespace.

Syntax

```
zosadmin dump [--dir=] --jndi=
```

Options

-d=, --dir=

Specify the working directory for a Orchestrator Server installation.

-j=, --jndi=

Specify the JNDI path to be displayed.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To dump the configuration for the naming facility, use the following command:

```
zosadmin dump --jndi=/facility/naming/config
```


get

This command retrieves an attribute.

Syntax

```
zosadmin get --mbean= --attr=
```

Options

-m=, --mbean=

Specifies the name of the mbean to view.

-a=, --attr=

Specifies the attribute name of the mbean to view.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To get the node autoregistration setting, use the following command:

```
zosadmin get --mbean=local:facility=nodeManager --attr=Autoregister
```

init

This command restores a server configuration to its initial state.

Syntax

```
zosadmin init
```

Options

-d, --dir=

The working directory for a Orchestrator Server installation.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

invoke

This command invokes an mbean method.

Syntax

```
zosadmin invoke --mbean= --method=
```

Options

-m=, --mbean=

Specify the name of the mbean to view.

-i=, --method=

Specify the method name of the mbean to invoke.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

list

This command lists all of the running servers.

Syntax

```
zosadmin list [--grid=] [--id=] [--server=] [--rmiport] [--rmipath] [-  
-rmihost=] [--rmiurl=] [--timeout=]
```

Options

-g, --grid=

Select by grid name.

-i, --id=

Select by kernel ID.

-s=, --server=

Select by host name.

-P=, --rmiport=

Select by RMI port.

-J=, --rmipath=

Select or specify the RMI bind path.

-H=, --rmihost=

Select or specify the RMI host (can be different from *server host*).

-r=, --rmiurl=

Select by full RMI URL.

-t=, --timeout=

Sets the query timeout (in seconds).

-V, --debug

Turn on debug output during this operation.

-v, --verbose

Turn on verbose output during this operation.

-h, --help

Display a help message for this operation.

login

This command logs into the Orchestrator Server.

Syntax

```
zosadmin login [--grid=] [--id=] [--rmiport=] [--rmipath=] [--  
rmihost=] [--rmiurl=] [--timeout=] [--user=] [--passwd=] [--check=]  
[serverhost]
```

Arguments

server host

Enter the server host name that you are logging into. The command polls for a server if none is supplied.

Options

-g, --grid

Select by grid name.

-i, --id=

Select by kernel ID.

-P, --rmiport=

Select by RMI port.

-J, --rmipath=

Select or specify the RMI bind path.

-H, --rmihost=

Select/specify RMI host (can be different from *server host*).

-r, --rmiurl=

Select by full RMI URL.

-t, --timeout

Sets the query timeout (in seconds).

-u, --user

Username used in accessing secure remote sites.

-p, --passwd

Password used in accessing secure remote sites.

-c, --check

Check and report existing login. Ignores other options.

-V, --debug

Turns on debug output during the login operation.

-v, --verbose

Turns on verbose output during the login operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To login to server *Eng*, use the following command:

```
zosadmin login Eng
```

Example 2

To check the current login, use the following command:

```
zosadmin login --check
```

logout

This command logs out of the Orchestrator Server.

Syntax

```
zosadmin logout
```

Options

-v, --verbose=

Specifies the terminal width for formatting. The user must enter a value with this option.

-V, --debug

Turns on debug output during the logout operation.

-h, --help

Displays a help message for this operation.

nodes

This command retrieves a list of all, online, offline, or mismatched nodes.

Syntax

```
zosadmin nodes [--offline] [--mismatch] [--all]
```

Options

-o, --offline

Retrieve a list of offline users and nodes.

-m, --mismatch

Retrieve a list of nodes requiring update because of a version mismatch.

-a, --all

Retrieve a list of all users and nodes.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

password

This command lets the administrator or developer change his or her password on the Orchestrator Server.

Syntax

```
zosadmin password [--passwd=] [--newpasswd=]
```

Options

-p=, --passwd=

Password used in accessing secure remote sites.

-l=, --newpasswd=

New password to use in accessing secure remote sites.

-v, --verbose

Turns on verbose output during this operation.

-d, --debug

Turns on debug output during this operation.

-h, --help

Displays a help message for this operation.

redeploy

This command redeploys a component onto a server.

Syntax

```
zosadmin redeploy [--session=] file|dir
```

Arguments

file|dir

Specify the deployable file or a directory containing deployable files.

Options

-i, --session=

Specifies the deployment session ID.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To redeploy the service file `myservice.sar`, use the following command:

```
zosadmin redeploy myservice.sar
```

Example 2

To redeploy the job file `quickie.job`, use the following command:

```
zosadmin redeploy quickie.job
```

Example 3

To redeploy the job files in directory `/test/myjob.job`, use the following command:

```
zosadmin redeploy /test/myjob.job
```

rotatelog

This command backs up and rotates the log files.

Syntax

```
zosadmin rotatelog
```

Options

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

sessions

This command displays session information.

Syntax

```
zosadmin sessions [id=] [--node=] [--user=] [--all] [--allNodes] [--allUsers] [--subSessions] [--inactive] [--full] [--connaddr] [--noDns]
```

Options

-i, --id <value>

Display this session for this session ID.

-n, --node <value>

Display sessions for the specified node.

-u, --user <value>

Display sessions for the specified user.

-a, --all

Display sessions for all clients.

-N, --allNodes

Display sessions for all nodes.

-U, --allUsers

Display sessions for all users.

-s, --subSessions

Include subsession entries.

-I, --inactive

Include inactive entries.

-f, --full

Include full session information.

-c, --connaddr

Include local and remote addresses.

-d, --noDns

Do not look up host names in DNS.

-V, --debug

Turn on debug output during this operation.

-v, --verbose

Turn on verbose output during this operation.

-h, --help

Display a help message for this operation.

set

This command sets an attribute on an MBean.

Syntax

```
zosadmin set --mbean= --attr= --value= --type=
```

Options

-m, --mbean=

Specify the name of the MBean to view.

-a, --attri=

Specify the attribute name of the MBean to view.

-o=, --value=

Specify the attribute value of the MBean to set in string form.

-t=, --type=

Specify the attribute value type to convert string form into (String/Boolean/Integer).

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To change the nodeManager autoregistration setting to false, use the following command:

```
zosadmin set --mbean=local:facility=nodeManager --attr=Autoregister --  
value=false --type=Boolean
```

start

This command starts a local server.

Syntax

```
zosadmin start [--dir=] [--jvmargs=] [--javaargs=] [--timeout=] [--upgrade] [--snapshot=]
```

Options

-d=, --dir=

Specify the working directory for a Orchestrator Server installation.

-b=, --jvmargs=

Pass the value as an argument to an invoked JVM process.

-B=, --javaargs=

Pass the value as an argument to an invoked Java program.

-t=, --timeout=

Specify the query timeout (in seconds).

-G=, --upgrade

Switch upgrade behavior for snapshot handling.

-S=, --snapshot=

Upgrade a server using the snapshot directory of an existing server.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To start a server instance using the install directory (default), use the following command:

```
zosadmin start
```

Example 2

To start a server instance using directory `/zos/server/`, use the following command:

```
zosadmin start --dir=/zos/server
```

Example 3

To start a server instance and upgrade from an existing server snapshot, use the following command:

```
zosadmin start --upgrade --snapshot=/oldzos/server/snapshot
```

Example 4

To start --upgrade --snapshot=/oldzos/server/snapshot, use the following command:

```
zosadmin start --jvmargs=-Xmx4g
```


status

This command displays the status of a server (including information on managed components)

Syntax

```
zosadmin status [--mbeans] [--sessions] [--facilities]
```

Options

-m, --mbeans

Retrieve the list of all MBeans.

-s, --sessions

Display deployment sessions ID along with component name.

-f, --facilities

Retrieve information on all facilities.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

stop

This command shuts down the Orchestrator Server.

Syntax

```
zosadmin stop [--force] [--snap]
```

Options

-f, --force

Do not prompt for server shutdown confirmation.

-s, --snap

Create a snapshot of server state for use in a later upgrade.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To stop a server instance without prompting for confirmation, use the following command:

```
zosadmin stop --force
```

Example 2

To stop a server instance and create a snapshot, use the following command:

```
zosadmin stop --snap
```

undeploy

This command undeploys a component from the server.

Syntax

```
zosadmin undeploy [--session=] component|dir
```

Arguments

component|dir

Specify the deployed file or a directory containing the deployed files.

Options

-i, --session=

Specify the deployment session ID.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To undeploy the service `myservice.sar`, use the following command:

```
zosadmin undeploy myservice.sar
```

Example 2

To undeploy the job file `quickie.job` from the server, use the following command:

```
zosadmin undeploy quickie.job
```

Example 3

To undeploy the job files in directory `/test/myjob.job`, use the following command:

```
zosadmin undeploy quickie.job
```

upgrade

This command upgrades an older server snapshot to a current version.

Syntax

```
zosadmin upgrade snapshotdir
```

Arguments

snapshotdir

Specify the directory containing a server configuration snapshot.

Options

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

NOTE: The `zosadmin upgrade` command does not actually upgrade a server instance. Use the `zosadmin create` and `zosadmin start` commands for manually upgrading an instance using a transformed configuration snapshot.

Examples

Example 1

To upgrade the server snapshot in the current directory, use the following command:

```
zosadmin upgrade ./snapshot
```

users

This command retrieves a list of all active or inactive users.

Syntax

```
zosadmin users    [--offline] [--all]
```

Options

-o, --offline

Retrieve a list of offline users/nodes.

-a, --all

Retrieve a list of all users/nodes.

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

verify

This command verifies a component for syntax.

Syntax

```
zosadmin verify file
```

Arguments

file

Specify the deployable file to verify.

Options

-v, --verbose

Turn on verbose output during this operation.

-V, --debug

Turn on debug output during this operation.

-h, --help

Display a help message for this operation.

Examples

Example 1

To verify the policy file `myjob.policy`, use the following command:

```
zosadmin verify myjob.policy
```

Example 2

To verify the schedule file `'mysched.sched'` and dump reconstituted XML, use the following command:

```
zosadmin verify mysched.sched --debug
```