Orchestration Administrator Reference

Cloud Manager 2.1

May 9, 2012

Legal Notice

Contents

About This Guide

This *Orchestration Administration Guide* introduces the processes you can use with the Orchestration functionality of NetIQ Cloud Manager 2.1, including the applied use of the Cloud Manager Orchestration Console and various command line tools. The guide provides an introductory overview of Cloud Manager Orchestration components and explains how it administers and manages work on the resources of the data center. The guide is organized as follows:

For reference information about the Orchestration Server or the Orchestration Console, see the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*. For information about the Orchestration Command Line Tools, see *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

Audience

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

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

Feedback

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

Additional Documentation

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

1 Basic Orchestration Concepts

This section contains the followings information:

1.1 Understanding Cloud Manager Orchestration Architecture

This section contains information about the following topics:

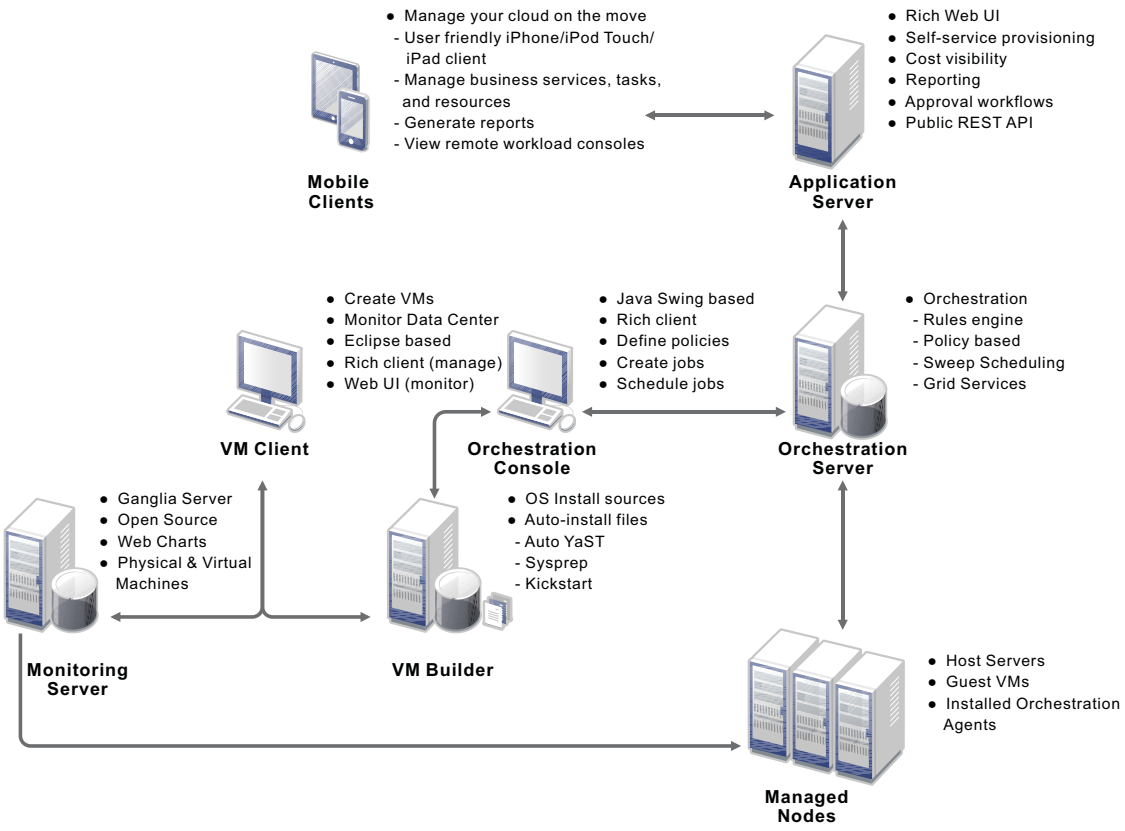
1.1.1 The Orchestration Server

The NetIQ Cloud Manager Orchestration Server is an advanced datacenter management solution designed to manage all network resources. It provides the infrastructure that manages group of ten, one hundred, or thousands of physical or virtual resources.

The Orchestration Server can perform a wide range of distributed processing problems including high performance computing, and breaking down work, including VM life cycle management, into jobs that can be processed in parallel through distributed job scheduling.

The following figure shows a high-level perspective of how the Orchestration Server fits into Cloud Manager architecture.

Figure 1-1 *Cloud Manager Orchestration Artchitecture*



The Orchestration Server is the gateway between enterprise applications and resource servers. The server has two primary functions:

- To manage the resource servers
- To manage jobs to run on the computing resource

In the first function, the server manages the computing resources by collecting, maintaining, and updating their status availability, service cost, and other facts. Changes to the computing resources can be made by the administrator.

The second function of the server is to run remote applications—called jobs—on the computing resources. The Orchestration Server uses a policy-based broker and scheduler to decide when and how a job should run on the computing resources. The decisions are based on many controlled factors, including the number of computing resources, their cost, and a variety of other factors as specified by the policy constraints set up by the server administrator. The Orchestration Server runs the job and provides all the job's output responses back to the user. The server provides failover capabilities to allow jobs to continue if computing resources and network conditions degrade.

The core strength of the Orchestration Server is the capability to automatically, rapidly, and securely create and scale heterogeneous virtual environments by using specialized VM provisioning adapter jobs that discover VMs already existing in various hypervisor environments, such as VMware, Microsoft Hyper-V, Citrix XenServer, Xen, and KVM. Once the VMs are discovered, they can be cloned and then provisioned as workloads to suit the business service needs of Cloud Manager users.

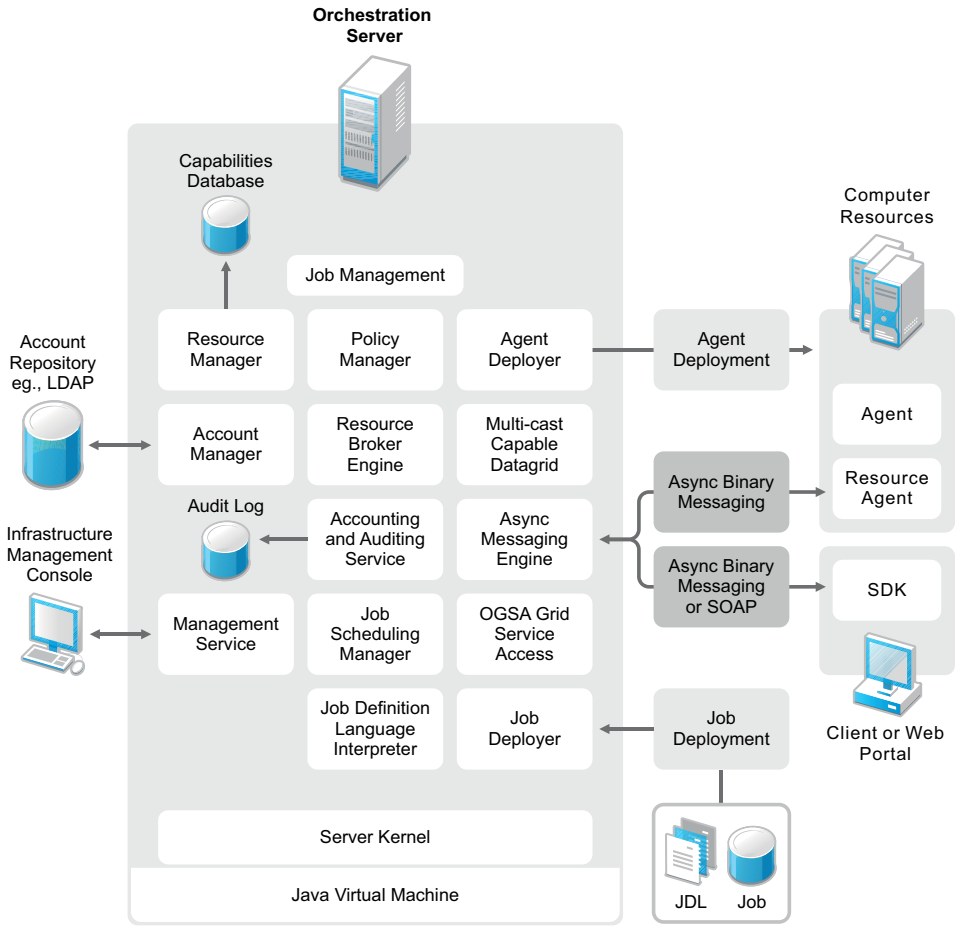
1.1.2 The Orchestration Agent

Agents are installed on all managed resources as part of the product deployment. The agent connects every managed resource to its configured server and advertises to the Orchestration Server that the resource is available for tasks. This persistent and auto-reestablishing connection is important because it provides a message bus for the distribution of work, collection of information about the resource, per-job messaging, health checks, and resource failover control.

After resources are enabled, Cloud Manager Orchestration can discover, access, and store detailed abstracted information—called "facts"—about every resource. Managed resources, referred to as "nodes," are addressable members of the Orchestration Server "grid" (also sometimes called the "matrix"). When integrated into the grid, nodes can be deployed, monitored, and managed by the Orchestration Server, as discussed in Section 1.2, "Understanding Orchestration Functionality," on page 19.

An overview of Cloud Manager Orchestration grid architecture is illustrated in the figure below, much of which is explained in this guide:

Figure 1-2 *Cloud Manager Orchestration Grid Architecture*



For additional information about job architecture, see "Job Architecture" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

1.1.3 The Resource Monitor

Cloud Manager Orchestration enables the monitoring of your system computing resources by using its built-in Resource Monitor.

1.1.4 The Orchestration Console and Command Line Tools

The Orchestration Console and the Orchestration command line tools (sometimes called the "Orchestration Clients") let a computing resource administrator troubleshoot, initiate, change, or shut down server functions for the Orchestration Server and its computing resources. The clients also monitor all managed computing resource job activity and provide facilities to manage application jobs. When you install the Clients on a computing resource, you are installing the following tools:

- zos command line interface
- zosadmin command line interface
- Orchestration Console
- Java SDK (toolkit)

NOTE: The Cloud Manager VM Client is installed separately.

The Orchestration Console is a graphical user interface running on Java. It provides a way for the server administrator to troubleshoot and to initiate, change, or shut down the functioning of the Orchestration Server and its resources. It also functions as a monitor of all Orchestration job activity, and it provides an interface for managing Orchestration Server jobs. For more information about the console, see the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*.

1.1.5 Entity Types and Managers

The following entities are some of key components of the Orchestration Server model:

Resources

All managed resources, which are called nodes, have an agent with a socket connection to the Orchestration Server. All resource use is metered, controlled, and audited by the Orchestration Server. Policies govern the use of resources.

The Orchestration Server allocates resources by reacting as load is increased on a resource. As soon as we go above a threshold that was set in a policy, a new resource is allocated and consequently the load on that resource drops to an acceptable rate.

You can also write and jobs that perform cost accounting to account for the cost of a resource up through the job hierarchy, periodically, about every 20 seconds. For more information, see "Auditing and Accounting Jobs" on page 18.

A collection of jobs, all under the same hierarchy, can cooperate with each other so that when one job offers to give up a resource it is reallocated to another similar priority job. Similarly, when a higher priority job becomes overloaded and is waiting on a resource, the system "steals" a resource from a lower priority job, thus increasing load on the low priority job and allocating it to the higher priority job. This process satisfies the policy, which specifies that a higher priority job must complete at the expense of a low priority job.

Users

An Orchestration user is an individual who authenticates to the Orchestration Server for the purpose of managing (that is, running, monitoring, canceling, pausing, stopping, or starting) a deployed job, or a user who authenticates through the Cloud manager VM Client to manage virtual machines. The Orchestration Server administrator can use the Orchestration Console to identify users who are running jobs and to monitor the jobs that are currently running or that have run during the current server session.

Orchestration Server users must authenticate to access the system. Access and use of system resources are governed by policies. For more information, see "The User Object" in the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*.

Job Definitions

A job definition is described in the embedded enhanced Python script that you create as a job developer. Each job instance runs a job that is defined by the Job Definition Language (JDL). Job definitions might also contain usage policies. For more information, see "Job Class" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

Job Instances

Jobs are instantiated at runtime from job definitions that inherit policies from the entire context of the job (such as users, job definitions, resources, or groups). For more information, see "JobInfo" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

Policies

Policies are XML documents that contain various constraints and static fact assignments that govern how jobs run in the Cloud Manager Orchestration environment.

Policies are used to enforce quotas, job queuing, resource restrictions, permissions, and other job parameters. Policies can be associated with any Orchestration Server object. For more information, see Section 1.2.4, "Policy-Based Management," on page 24 and "Policies" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

Facts

Facts represent the state of any object in the Orchestration Server grid. They can be discovered through a job or they can be explicitly set.

Facts control the behavior a job (or joblet) when it's executing. Facts also detect and return information about that job in various UIs and server functions. For example, a job description that is set through its policy and has a specified value might do absolutely nothing except return immediately after network latency.

The XML fact element defines a fact to be stored in the Grid object's fact namespace. The name, type and value of the fact are specified as attributes. For list or array fact types, the element tag defines list or array members. For dictionary fact types, the dict tag defines dictionary members.

Facts can also be created and modified in JDL and in the Java Client SDK. For more information, see "Using Facts in Job Scripts" in the "Job Development Concepts" section of the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

There are three basic types of facts:

- **Static:** Facts that require you to set a value. For example, in a policy, you might set a value to be False. Static facts can be modified through policies.
- **Dynamic:** Facts produced by the Orchestration Server system itself. Policies cannot override dynamic facts. They are read only and their value is determined by the Orchestration Server itself.
- **Computed:** Facts derived from a value, like that generated from the cell of a spreadsheet. Computed facts have some kind of logic behind them which derive their values. For more information, see "Computed Facts" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

See the example, `/opt/novell/zenworks/zos/server/examples/allTypes.policy`. This example policy has an XML representation for all the fact types. For a comprehensive list of facts and fact junctions used in Cloud Manager Orchestration, see "Grid Object Facts and Fact Junctions" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

Constraints

The constraint element of a policy can define the selection and allocation of Grid objects (such as resources) in a job. The required type attribute of a constraint defines the selection of the resource type.

For example, in order for the Orchestration Server to choose resources for a job, it uses a "resource" constraint type. A resource constraint consists of Boolean logic that executes against facts in the system. Based upon this evaluation, the Orchestration Server considers only resources that match the criteria that have been defined in constraints.

For more information about the types of constraints and how they are applied in jobs, see "The Role of Policy Constraints in Job Operation" in the "Job Development Concepts" section of the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

Groups

Resources, users, job definitions and virtual machines (VM) are managed in groups with group policies that are inherited by members of the group.

VM: Hosts, Images, and Instances

A virtual machine host is a resource that is able to run guest operating systems. Attributes (facts) associated with the VM host control its limitations and functionality within the Orchestration Server. A VM image is a resource image that can be cloned and/or provisioned. A VM instance represents a running copy of a VM image.

Templates

Templates are images that are meant to be cloned (copied) prior to provisioning the new copy.

1.1.6 Jobs

The Orchestration Server manages all nodes by administering jobs (and the functional control of jobs at the resource level by using joblets), which control the properties (facts) associated with every resource. In other words, jobs are units of functionality that dispatch data center tasks to resources on the network such as management, migration, monitoring, load balancing, etc.

The Orchestration Server provides a unique job development, debugging, and deployment environment that expands with the demands of growing data centers.

As a job developer, your task is to develop jobs to perform a wide array of work that can be deployed and managed by the Orchestration Server.

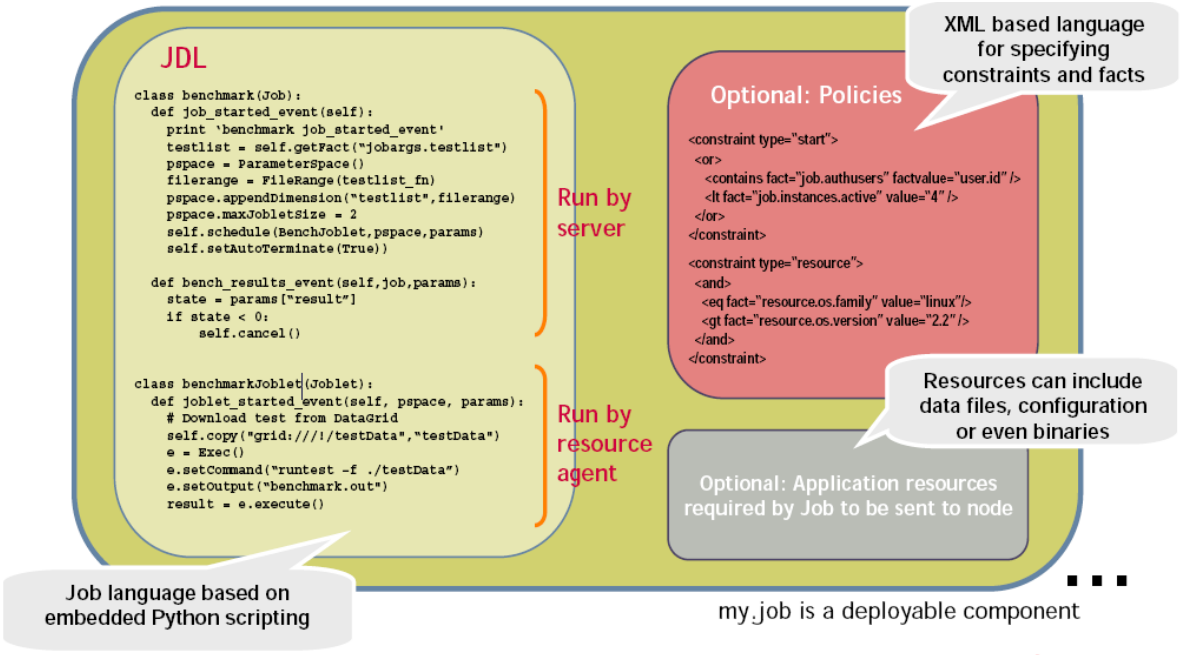
Jobs, which run on the Orchestration Server, can provide functions within the Orchestration environment that might last from seconds to months. Job and joblet code exist in the same script file and are identified by the `.jdl` extension. The `.jdl` script contains only one job definition and zero or more joblet definitions. A `.jdl` script can have only one Job subclass. As for naming conventions, the Job subclass name does not have to match the `.jdl` filename; however, the `.jdl` filename is the defined job name, so the `.jdl` filename must match the `.job` filename that contains the `.jdl` script. For example, the job files (`demoIterator.jdl` and `demoIterator.policy`) included in the `demoIterator` example job are packaged into the archive file named `demoIterator.job`, so in this case, the name of the job is `demoIterator`.

A job file also might have policies associated with it to define and control the job's behavior and to define certain constraints to restrict its execution. A `.jdl` script that is accompanied by a policy file is typically packaged in a job archive file (`.job`). Because a `.job` file is physically equivalent to a Java archive file (`.jar`), you can use the JDK JAR tool to create the job archive.

Multiple job archives can be delivered as a management pack in a service archive file (SAR) identified with the .sar extension. Typically, a group of related files are delivered this way. For example, the Xen30 management pack is a SAR.

As shown in the following illustration, jobs include all of the code, policy, and data elements necessary to execute specific, predetermined tasks administered either through the Cloud Manager Orchestration Console, or from the zos command line tool.

Figure 1-3 *Components of a Job (`my.job`,)*



Because each job has specific, predefined elements, jobs can be scripted and delivered to any agent, which ultimately can lead to automating almost any datacenter task. Jobs provide the following functionality:

- "Controlling Process Flow" on page 17
- "Parallel Processing" on page 17
- "Managing the Cluster Life Cycle" on page 17
- "Discovery Jobs" on page 17
- "System Jobs" on page 17
- "Provisioning Jobs" on page 18
- "Auditing and Accounting Jobs" on page 18

For more information, see "Job Development Concepts" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference* and the following JDL job class definitions in the same guide:

- "Job"
- "JobInfo"

Controlling Process Flow

Jobs can written to control all operations and processes of managed resources. Through jobs, the Orchestration Server manages resources to perform work. Automated jobs (written in JDL), are broken down into joblets, which are distributed among multiple resources.

Parallel Processing

By managing many small joblets, the Orchestration Server can enhance system performance and maximize resource use.

Managing the Cluster Life Cycle

Jobs can detect demand and monitor health of system resources, then modify clusters automatically to maximize system performance and provide failover services.

Discovery Jobs

Some jobs provide inspection of resources to more effectively management assets. These jobs enable all agents to periodically report basic resource facts and performance metrics. In essence, these metrics are stored as facts consisting of a key word and typed-value pairs like the following example:

```
resource.loadaverage=4.563, type=float
```

Jobs can poll resources and automatically trigger other jobs if resource performance values reach certain levels.

The system job scheduler is used to run resource discovery jobs to augment resource facts as demands change on resources. This can be done on a routine, scheduled basis or whenever new resources are provisioned, new software is installed, bandwidth changes occur, OS patches are deployed, or other events occur that might impact the system.

Consequently, resource facts form a capabilities database for the entire system. Jobs can be written that apply constraints to facts in policies, thus providing very granular control of all resources as required. All active resources are searchable and records are retained for all off-line resources.

The following `osInfo.job` example shows how a job sets operating system facts for specific resources:

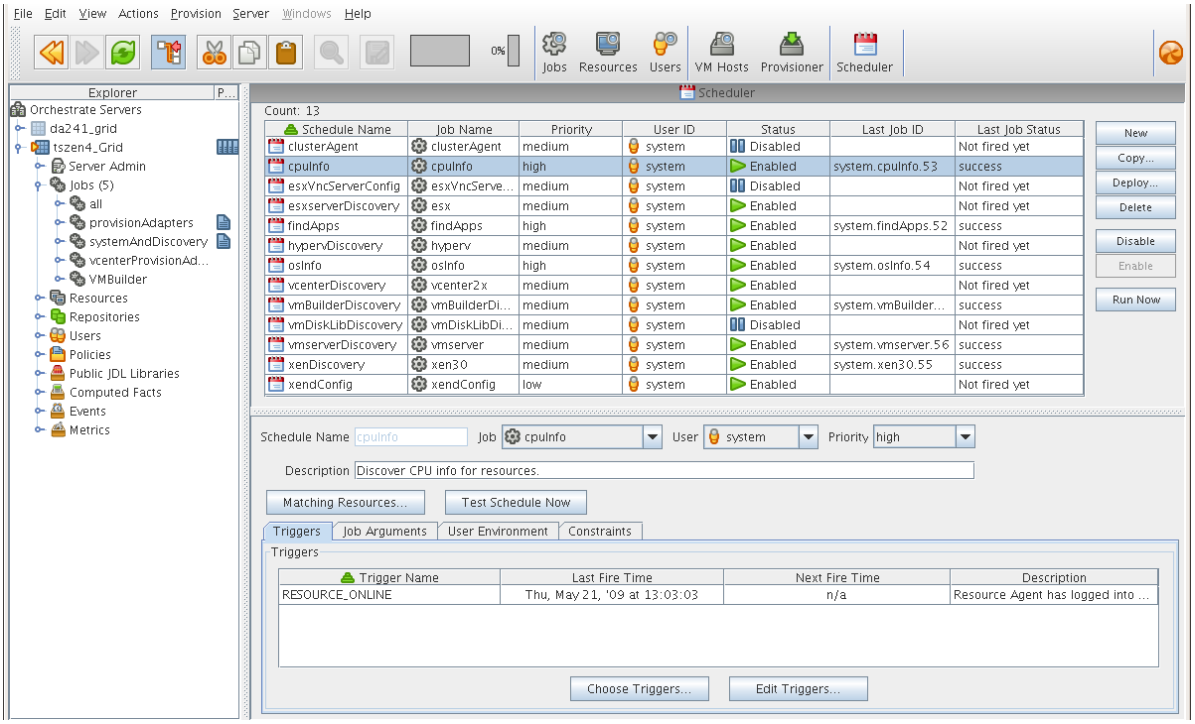
```
resource.cpu.mhz (integer) e.g., "800" (in Mhz)
    resource.cpy.vendor (string) e.g. "GenuineIntel"
    resource.cpu.model (string) e.g. "Pentium III"
    resource.cpu.family (string) e.g. "i686"
```

`osInfo.job` is packaged as a single cross-platform job and includes the Python-based JDL and a policy to set the timeout. It is run each time a new resource appears and once every 24 hours to ensure validity of the resources. For a more detailed review of this example, see "osInfo.job" in "Job Development Concepts" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

System Jobs

Jobs can be scheduled to periodically trigger specific system resources based on specific time constraints or events. As shown in the following figure, the Orchestration Server provides a built-in job scheduler that enables you or system administrators to flexibly deploy and run jobs.

Figure 1-4 *The Job Scheduler*



For more information, see "How Constraints Are Used" in "Job Development Concepts" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference* and "The Orchestration Server Job Scheduler" in the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*. See also "Job Scheduling" and "Job" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

Provisioning Jobs

Jobs also drive provisioning for virtual machines (VMs) and physical machines, such as blade servers. Provisioning adapter jobs for various VM hypervisors are deployed and organized into appropriate job groups for management convenience.

The provisioning jobs included in the Orchestration Server are used for interacting with VM hosts and repositories for VM life cycle management and for cloning, moving VMs, and other management tasks. These jobs are called "provisioning adapters" and are members of the job group called "provisionAdapters."

For more information, see the *NetIQ Cloud Manager 2.1 VM Orchestration Reference* and Section 1.2.3, "Resource Virtualization," on page 23 of this guide.

Auditing and Accounting Jobs

You can create Cloud Manager Orchestration jobs that perform reporting, auditing, and costing functions inside your data center. Your jobs can aggregate cost accounting for assigned resources and perform resource audit trails.

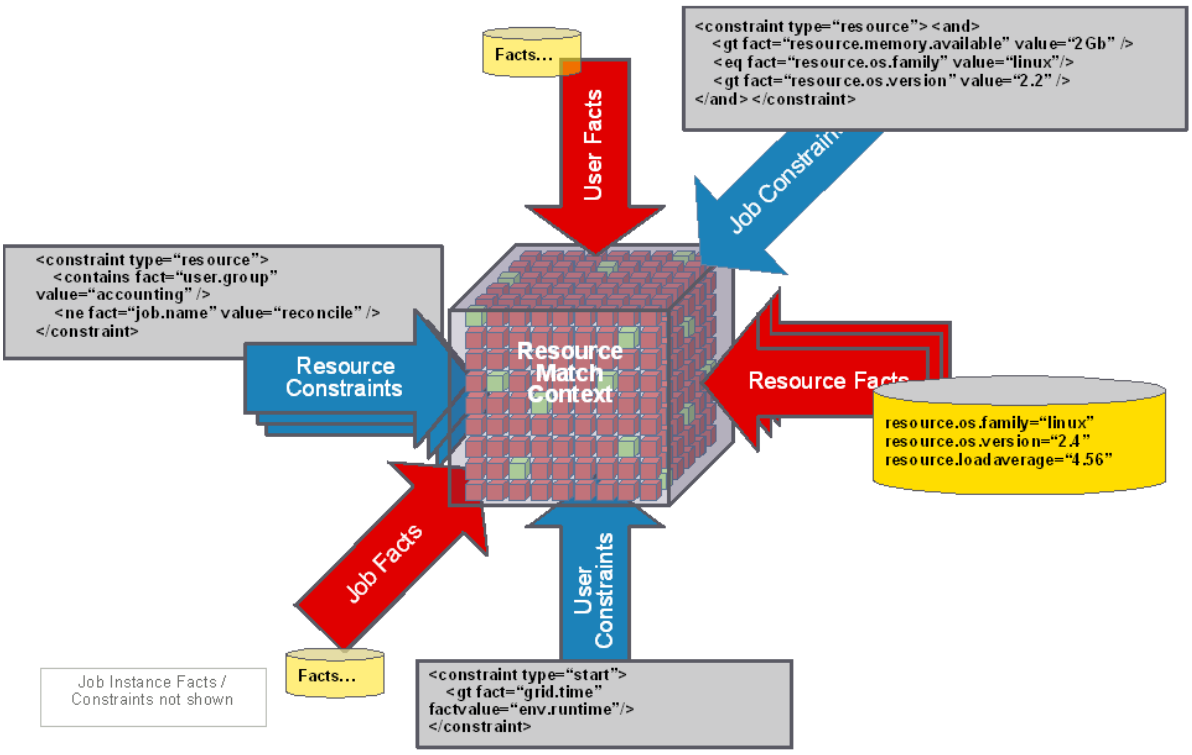
1.1.7 Constraint-Based Job Scheduling

The Orchestration Server is a "broker" that can distribute jobs to every "partner" agent on the grid. Based on assigned policies, jobs have priorities and are executed based on the following contexts:

- User Constraints
- User Facts
- Job Constraints
- Job Facts
- Job Instance
- Resource User Constraints
- Resource Facts
- Groups

Each object in a job context contains the following elements:

Figure 1-5 *Constraint-Based Resource Brokering*



For more information, see "Scheduling with Constraints"in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

1.2 Understanding Orchestration Functionality

- Section 1.2.1, "How Do I Interact with the Orchestration Server?," on page 20
- Section 1.2.2, "How Orchestration Components Communicate," on page 22
- Section 1.2.3, "Resource Virtualization," on page 23

1.2.1 How Do I Interact with the Orchestration Server?

Orchestration administrators and users perform their activities by using their own graphical tool or command line interface tools. In general, the same functions are available in either the graphical or the command line tools. The toolset is summarized in the chart below.

Table 1-1 *Summary of the Orchestration Toolset*

Role	Tool Type	Description	Common Function
Orchestration Administrator	Graphical Interface	The Orchestration Console	◆ Stops or starts the Orchestration Server(s) ◆ Deploys jobs. ◆ Manages Group and Policy associations. ◆ Monitors jobs. ◆ Helps troubleshoot jobs/ policies. ◆ Monitors computing resource usage. ◆ Creates and manages user accounts.
	Command Line Interface	Sample command line for help: `zosadmin` *`command`* `--help`	
	Graphical Interface	The Cloud Manager VM Client	◆ Discovers host servers in the Orchestration grid ◆ Discovers existing VMs ◆ Creates, edits, installs, and deletes VMs ◆ Manages VM repositories ◆ Stops, starts, pauses, or suspends VMs ◆ Migrates or moves VMs ◆ Installs the Orchestration Agent on VMs ◆ Creates and clones VM templates ◆ Provides group management of VMs, host servers, storage locations, and templates ◆ Resyncs state of VMs and hosts with Orchestration Server ◆ Provides access to VM and Host consoles ◆ Shows details of VM and host configurations ◆ Provides error log and progress views

Role	Tool Type	Description	Common Function
Orchestration User	Command Line Interface	Sample command line for help: `zos command --help`	◆ Displays deployed jobs. ◆ Displays available computing resources. ◆ Runs jobs. ◆ Monitors running jobs. ◆ Manages the user's own jobs. That is, a user can cancel, pause, restart, and change job priority.

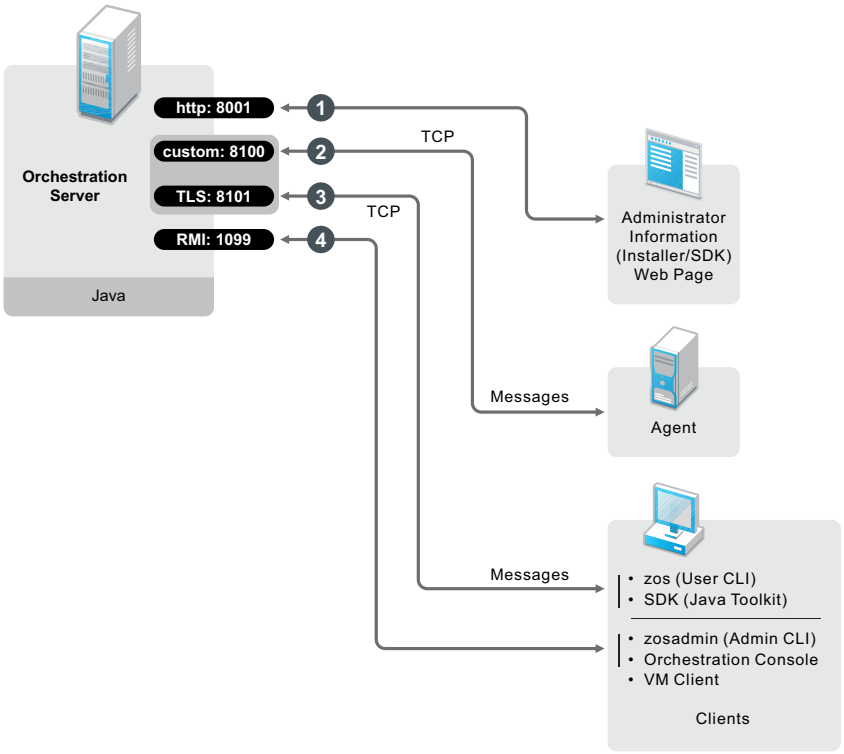
Other functions can also be performed by using either the graphical or command line tools. To help you understand how these tools can be used, you can find more information in the following sections:

- The *NetIQ Cloud Manager 2.1 Orchestration Console Reference*
- The *NetIQ Cloud Manager 2.1 VM Client Guide and Reference*
- "The zosadmin Command Line Tool" and "The zos Command Line Tool" in the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

1.2.2 How Orchestration Components Communicate

The following diagram illustrates how the various components of Cloud Manager Orchestration communicate with the Orchestration Server. An explanation for each communication link follows the diagram.

Figure 1-6 *Communication Ports Used By the Orchestration Server*



1. Administrators who want more information about Cloud Manager Orchestration and a method to access or install additional clients or agents can access the Administrator Information page. To do so, open a Web browser and enter the URL to the Orchestration Server, followed by the port designated for the Web Info page during installation. In a basic installation of Orchestration, the default is port 8001. The URL would therefore be entered as follows:

`http://`*`DNS_Name_or_IP_Address`*`:8001`

2. The Orchestration Server establishes and maintains contact with an installed Orchestration Agent on a computing resource through port 8100, using a custom protocol.
3. When a user invokes the `zos` command line interface (available after Orchestration clients are installed on a machine), or when using the Java toolkit SDK, those client tools communicate with the Orchestration Server over ports 8100 and 8101.
4. When the administrator invokes the `zosadmin` command line interface (available after Orchestration clients — including the Orchestration Console — are installed on a machine), or when using the Orchestration Console, those client tools communicate with the Orchestration Server over port 1099, which uses a Java RMI (Remote Method Invocation) protocol.

1.2.3 Resource Virtualization

Host machines or test targets managed by the Orchestration Server form nodes on the grid. All resources are virtualized for access by maintaining a capabilities database containing extensive information (facts) for each managed resource.

This information is automatically polled and obtained from each resource periodically or when it first comes online. The extent of the resource information the system can gather is customizable and highly extensible, controlled by the jobs you create and deploy.

1.2.4 Policy-Based Management

Policies are aggregations of facts and constraints that are used to enforce quotas, job queuing, resource restrictions, permissions, and other user and resource functions. Policies can be set on all objects and are inherited, which facilitates implementation within related resources.

Facts, which might be static, dynamic or computed for complex logic, are used when jobs or test scenarios require resources in order to select a resource that exactly matches the requirements of the test, and to control the access and assignment of resources to particular jobs, users, projects, etc. through policies.This abstraction keeps the infrastructure fluid and allows for easy resource substitution.

Of course, direct named access is also possible. An example of a policy that constrains the selection of a resource for a particular job or test is shown in the sample below. Although resource constraints can be applied at the policy level, they can also be described by the job itself or even dynamically composed at runtime.

```
<policy>
  <constraint type="resource">
    <and>
      <eq fact="resource.os.family" value="Linux"/>
      <gt fact="resource.os.version" value="2.2" />
    <and>
  </constraint>
</policy>
```

An example of a policy that constrains the start of a job or test because too many tests are already in progress is shown in the following sample:

```
<policy>
  <!-- Constrains the job to limit the number of running jobs to a
defined value but exempt certain users from this limit. All jobs
that attempt to exceed the limit are qued until the running jobs
count decreases and the constraint passes. -->

<constraint type="start" reason="Too busy">
    <or>
      <lt fact="job.instances.active" value="5"/>
      <eq fact="user.name" value="canary" />
    </or>
  </constraint>
</policy>
```

For more information about policies and constraints, see "Policies" in the "Job Development Concepts" section of the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

1.2.5 Grid Object Visualization

One of the greatest strengths of the Cloud Manager Orchestration solution is the ability to manage and visualize the entire grid. This is performed through the Cloud Manager Orchestration Console and the Cloud Manager VM Monitoring System.

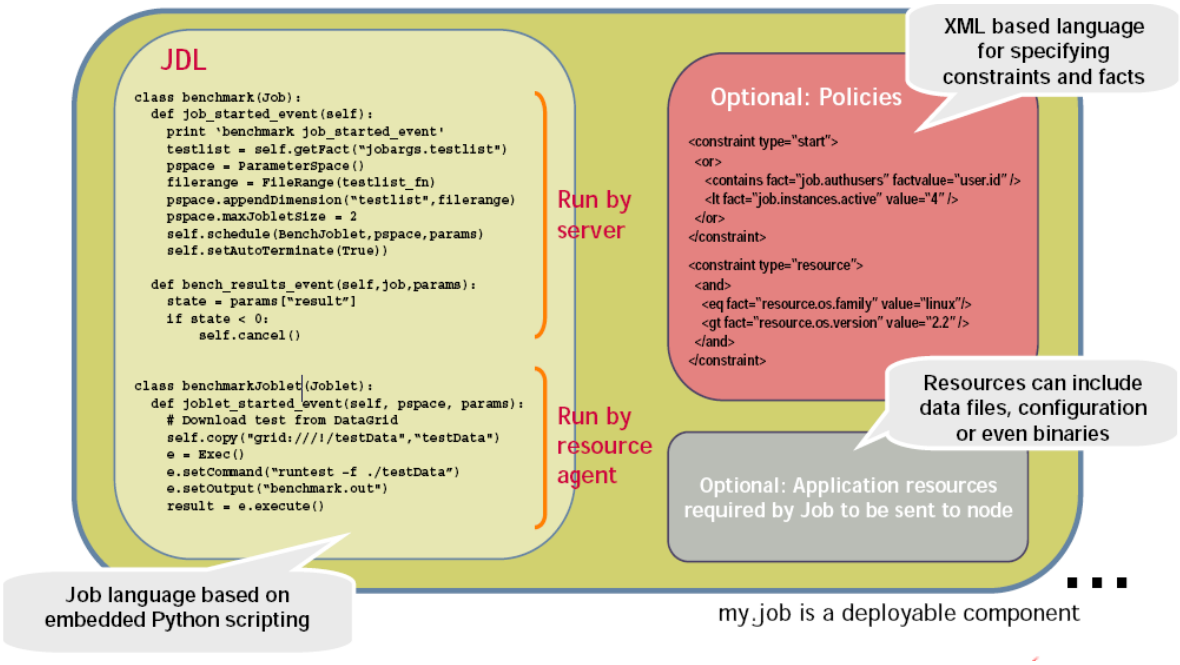
The desktop Orchestration Console is a Java application that has broad platform support and provides job, resource, and user views of activity as well as access to the historical audit database system, cost accounting, and other graphing features.

The Orchestration Console also applies policies that govern the use of shared infrastructure or simply create logical grouping of nodes on the grid. For more information about the console, see the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*.

1.2.6 Understanding Job Semantics

As mentioned earlier, the Orchestration Server runs jobs. A job is a container that can encapsulate several components including the Python-based logic for controlling the job life cycle (such as a test) through logic that accompanies any remote activity, task-related resources such as configuration files, binaries and any policies that should be associated with the job, as illustrated below.

Figure 1-7 *Components of a Job*



Workflows

Jobs can also invoke other jobs, creating hierarchies. Because of the communication between the job client (either a user/user client application or another job) it is easy to create complex workflows composed of discrete and separately versioned components.

When a job is executed and an instance is created, the class that extends job is run on the server and as that logic requests resources, the class(es) that extend the joblet are automatically shipped to the requested resource to manage the remote task. The communication mechanism between these distributed components manifests itself as event method calls on the corresponding piece.

For more information, see "Workflow Job Example" in "Job Examples", and "Job State Transition Events", or "Communicating Through Job Events" in "Job Architecture" in the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

1.2.7 Distributed Messaging and Failover

A job has control over all aspects of its failover semantics, which can be specified separately for conditions such as the loss of a resource, failure of an individual joblet, or joblet timeout.

The failover/health check mechanisms leverage the same communications mechanism that is available to job and joblet logic. Specifically, when a job is started and resources are employed, a message interface is established among all the components as shown in Figure 1-8 on page 26.

Optionally, a communication channel can also be kept open to the initiating client. This client communication channel can be closed and reopened later based on jobid. Messages can be sent with the command

```
sendEvent(foo_event, params, ...)
```

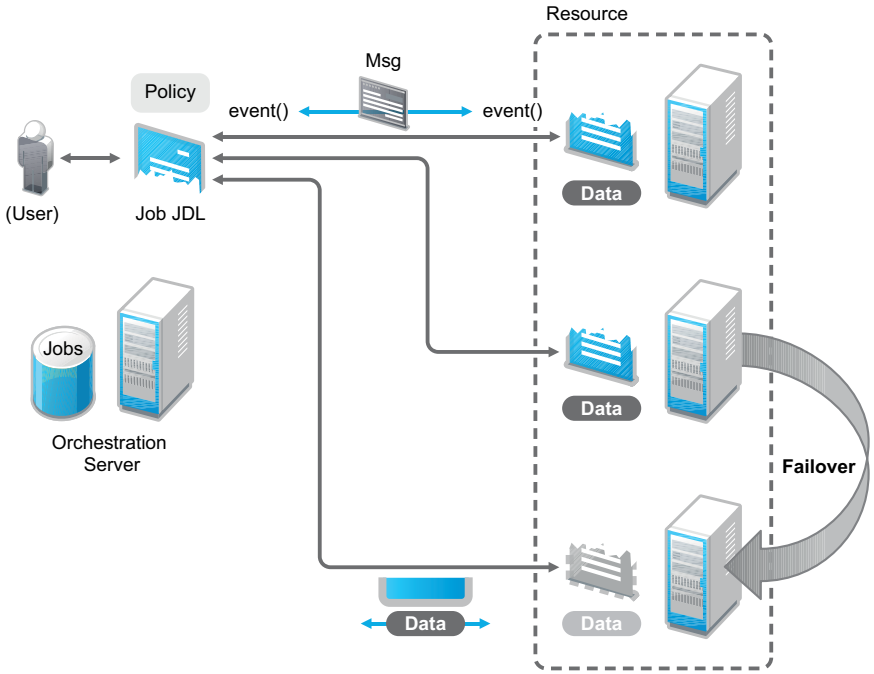
and received at the other end as a method invocation

```
def foo_event(self, params)
```

If a job allows it, a failure in any joblet causes the Orchestration Server to automatically find an alternative resource, copy over the joblet JDL code, and reestablish the communication connection. A job also can listen for such conditions simply by defining a method for one of the internally generated events, such as def joblet_failure_event(...).

Such failover allows, for example, for a large set of regression tests to be run (perhaps in parallel) and for a resource to die in the middle of the tests without the test run being rendered invalid. The figure below shows how job logic is distributed and failover achieved:

Figure 1-8 *A Job in Action*



2 Basic Orchestration Functions

After you install and configure the basic components of the NetIQ Cloud Manager Orchestration components, (that is, the Orchestration Server, the Orchestration Agent, and the Orchestration Clients, including the Orchestration Console), you will want to see them at work. The information in this section is organized sequentially (that is, in a "walkthrough" scenario) so that you can follow the process an administrator might use to begin applying Cloud Manager Orchestration capabilities in a production environment.

The first three subsections listed above are basic tasks you need to perform to make the Orchestration system perform at a basic level. The other sections include information to help you understand how Cloud Manager Orchestration can work in your production environment.

For information about the first use of the Cloud Manager VM Client components, see "Overview" in the *NetIQ Cloud Manager 2.1 VM Client Guide and Reference*.

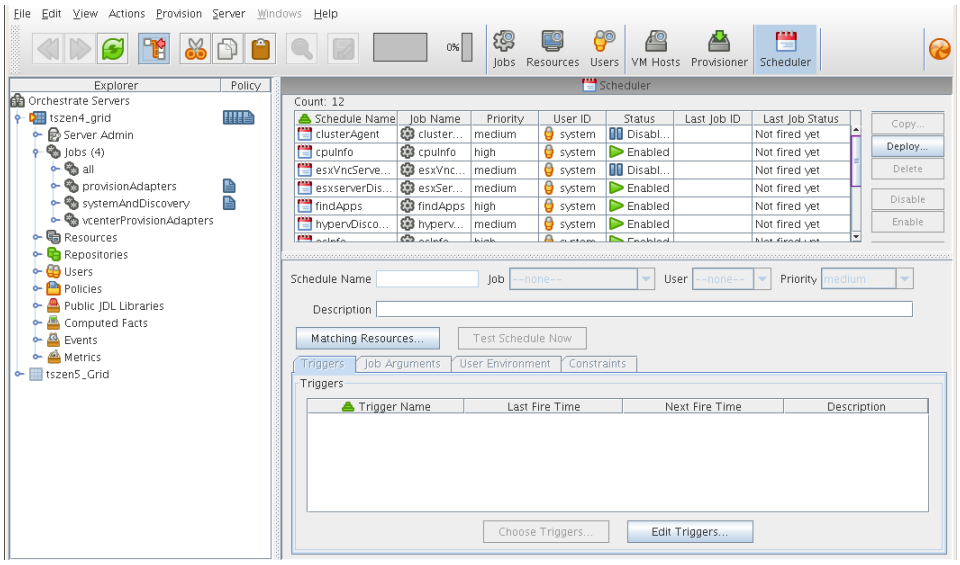
2.1 Observing Discovery Jobs

When you created a resource account for the first time, you might have noticed the status window of the Resource object change colors (see Step 2 in "Automatically Registering a Resource" in the *NetIQ Cloud Manager 2.1 Orchestration Installation Guide*) from blue to green. You might also notice new jobs

displayed as objects in the Explorer panel. What you are observing are the “discovery” jobs that are shipped with the Orchestration Server (different discovery jobs are shipped with the Orchestration Server, depending on which management pack you license).

To understand the reason why these jobs run:

1 In the Orchestration Console, click *Scheduler* to open the Job Schedule view in the workspace.

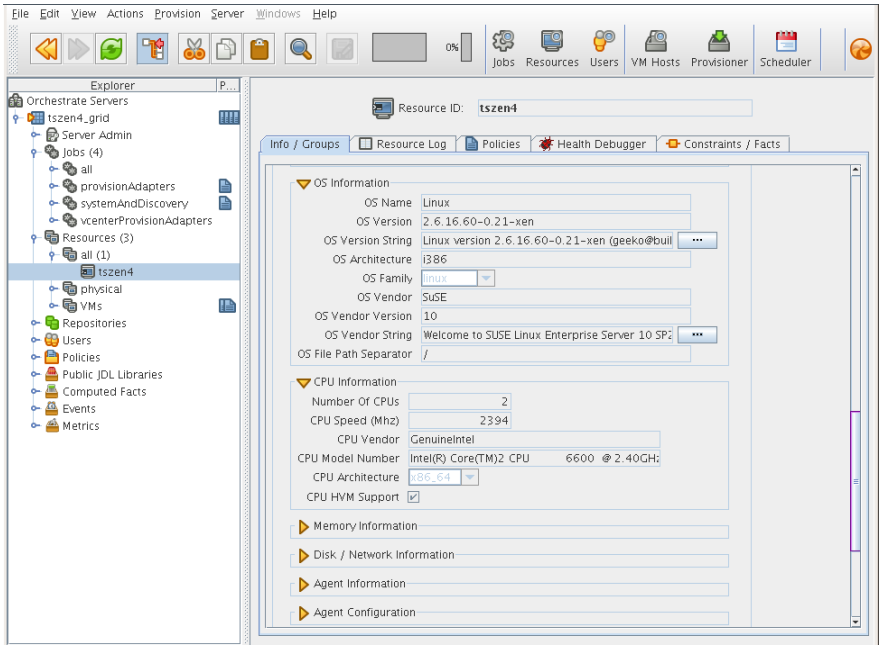


In this walkthrough of basic Orchestration Server functionality, you can see that several jobs are configured to run. If you select one of the jobs, such as `cpuInfo`, you will see that it was configured with a trigger called RESOURCE_ONLINE. All of the discovery jobs, like `cpuInfo`, are configured to run when the resource is online, that is, when the resource agent has logged into the Orchestration Server.

The discovery jobs, including provisioning adapter jobs, run basic operations at resource start as a convenience, to gather data that you or a job developer might need later when creating jobs, or that the Orchestration system might need as it allocates resources to run jobs. For example, the `cpuInfo` job and the `osInfo` job do some basic probing of the computing node (the machine where the agent is installed and has a resource account) for later reference.

To verify this, you can view the resource account that you created during the basic installation (as documented in “Creating a Resource Account” in the *NetIQ Cloud Manager 2.1 Orchestration Installation Guide*) by selecting its object in the Explorer tree. By default, the *Info/Groups* page for the resource opens in the Orchestration Console admin view.

Figure 2-1 *Resource Information Page After Discovery Jobs Run*



If you scroll down on the *Info/Groups* page, you see that the discovery jobs have gathered basic data about the processor and operating system of this computing node. If the jobs had not run at resource start, this information about the resource would not be ready for use.

Now that you have seen a how jobs are run by the Orchestration system on resource start, you can walk through the process of deploying and running a sample job on your own by proceeding with Section 2.2, “Deploying a Sample Job,” on page 29.

2.2 Deploying a Sample Job

One of the main functions of the Orchestration Server is to run application requests, called jobs, on grid resources. Because the Orchestration Server is capable of handling multiple application requests, it uses a policy-based broker and scheduler to decide when and how a job should run on the resources. These decisions are based on many controlled factors, including the number of resource nodes, their cost, and a variety of other factors as requested by the application, but managed under policy constraints set up by the administrator or the job developer.

Developing a job involves the creation of an application executable and a job file. See the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference* for more information on creating and building jobs by using the Orchestration Server Job Description Language (JDL) and the job policies.

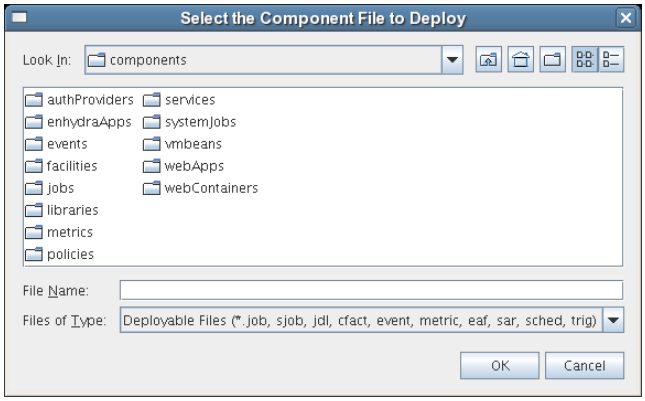
Before a job can run, the Orchestration Server administrator must deploy it, which involves moving it from a development state to a state where it is ready and available for users. Only the administrator has the necessary rights to deploy a job.

There are three methods you can use to deploy a job:

- Deploy from the Orchestration Console by right-clicking the *Jobs* container in the Explorer panel.
- Deploy from the Orchestration Console by selecting the *Actions* menu in the Orchestration Console.
- Deploy from the `zosadmin` command line (`zosadmin deploy path_to_job`).

For this walkthrough, we will deploy a simple job developed for Orchestration Server customers to demonstrate how jobs are deployed and run. Although the walkthrough shows only the first method for deploying, the other methods are relatively simple, so no further explanation is provided.

1. In the Explorer panel of the Orchestration Console, right-click the *Jobs* object, then click *Deploy Job* to open the Select the Component File to Deploy dialog box.

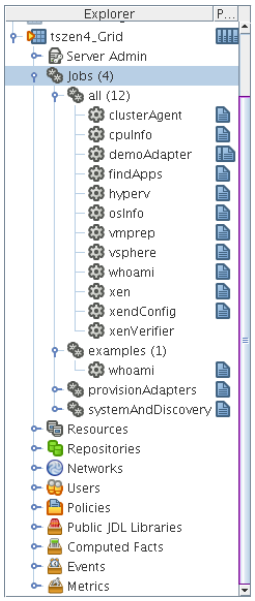


2. Open the *Look In* drop-down list, then navigate to the location of the job you want to deploy.

   Although a job developer can store Orchestration Server jobs at any location on the network, the sample jobs shipped with the Orchestration Server are limited to the directories where the product is installed. For this walkthrough, navigate to the `/opt/novell/zenworks/zos/server/examples` directory on the Orchestration Server. If the Orchestration Console is installed on a Windows machine, the default location is `c:\Program Files\Novell\zos\clients\examples`.

3. Select `whoami.job`, then click *OK* to deploy the job to the *Jobs* container.

   The *whoami* job appears in the *all* container and in the *examples* container in the tree.



   When deployed, the job is sent over the wire to the Orchestration Server with which it is associated. It is persisted there until undeployed.

When the job is available, you need to create a user who can run that job. For more information, see Section 2.3, “Creating a User Account,” on page 31

2.3 Creating a User Account

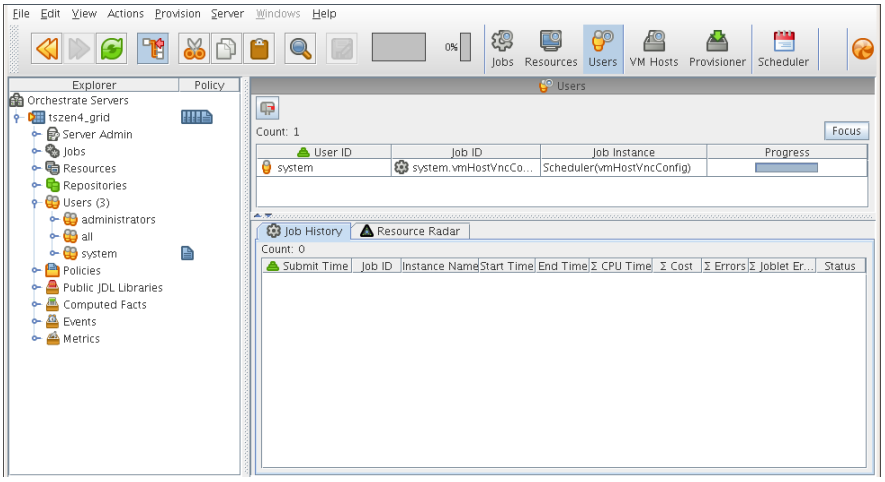
Although the Orchestration Server has some pre-assembled jobs, such as the `cpuInfo` discovery job that you learned about earlier, most jobs must be developed by a job developer, then be run and managed by a user (also called a job manager). Without an authorized individual who can log in to the Orchestration Server system to manage the use of a job, the product does not realize its potential.

This section of the walkthrough introduces the basics of creating a user account:

2.3.1 Opening the Users Monitor

Now that the Orchestration Server has run discovery jobs and you have deployed a sample job, you can begin to create user accounts. To do so, open the Orchestration Console and click *Users* in the toolbar to open the Users Monitor in the Workspace panel of the Orchestration Console.

Figure 2-2 *Users Monitor of the Orchestration Server Console*



In this monitor, you can see the users that are connected to the server and what they are doing in the grid.

If a user logs in but has not been registered (that is, no account is created for that user), the authentication to the server is retried every 90 seconds. If this is the case, the User Registration icon has a “flag up” status, meaning that a user is waiting to register. If the icon has a “flag down” status, either no user accounts have been created or all active users are logged in, so none are waiting to register.

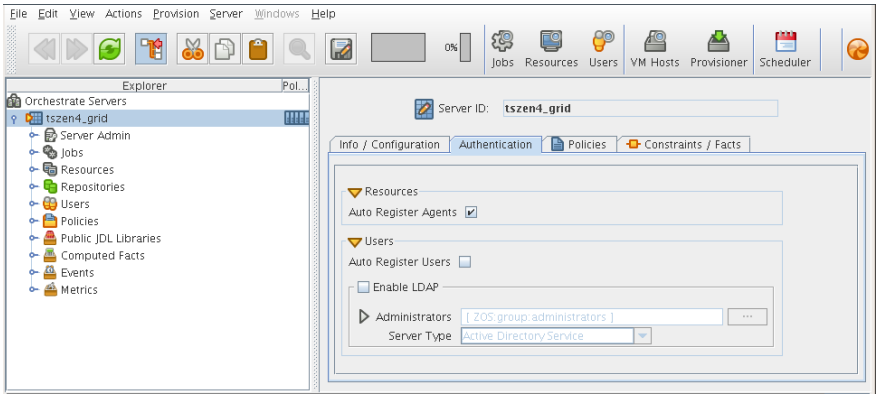
You can use the Orchestration Console to register a user automatically (see Section 2.3.2, "Automatically Registering a User," on page 32) or to register a user manually (see Section 2.3.3, "Manually Registering a User," on page 33). You can also select which users can log in to create accounts (see "Selecting a Resource for Manual Registration" in the *NetIQ Cloud Manager 2.1 Orchestration Installation Guide*).

The Users Monitor has many features to help you manage users when they are registered, including the jobs and joblets assigned to individual users. For more detailed information about the Users Monitor, see the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*.

2.3.2 Automatically Registering a User

If your network environment does not require a high level of security (such as in a development and testing environment) and you want a quick way to create a user account without a password, you can do so at the Orchestration Console.

1. In the Explorer tree of the Orchestration Console, select the grid object representing the Orchestration Server to open the *Info/Configuration* page of the grid object, then select the *Authentication* tab to open the Authentication page.
2. In the *Users* section of the page, select the *Auto Register Users* check box, then click the Save icon.



3. Use the `zos` command line interface to log in to the server.
   - **3a** From a system terminal, enter the following command:

     ```
     zos login -u user_ID
     ```

     If you are attempting to log in to a machine other than the local host, you can alter the command to the following:

     ```
     zos login Orchestration_Server_name -u user_ID
     ```

   - **3b** When prompted for the user password, press Enter.
   - **3c** (Conditional) If you are prompted for a decision regarding whether you want to accept the server certificate, enter `yes`.

NOTE: You can assign a password for the user at a later time in the *Info/Groups* page of the User Object.

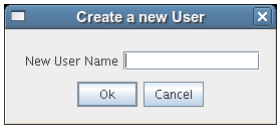
When a user logs out, the User object icon is dimmed in the Explorer tree or in the admin view of each User group to which it belongs.

2.3.3 Manually Registering a User

If you want a higher level of security for authorized users, you can manually create a user account in the Orchestration Console before the user logs in. When a user is created in the Orchestration Server Console, that user is ready to run jobs.

To create a new user in the Orchestration Console Explorer tree:

1 In the Explorer tree of the Orchestration Console, right-click *Users* > click *New User* to display the Create a New User dialog box.

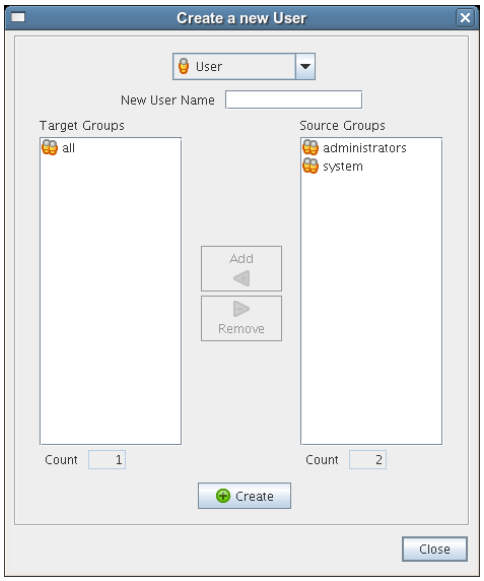


2 Specify the name of the new user you want to create in the *New User Name* field, then click *OK*.

The user account is created, but is not currently running jobs, as indicated by its object icon in the Explorer tree or in the admin view of each User group to which it belongs.

To create a new user through the *Actions* menu:

1 In the Orchestration Console, click *Actions* > *Create User* to display an expanded version of the Create a New User dialog box.



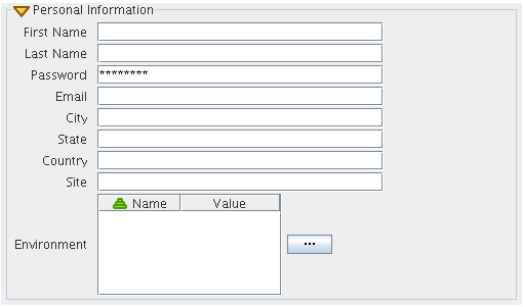
This dialog box includes a method for designating the user as a member of the *administrators* user group. In this walkthrough, we will create the user as a member of the *all* group, which does not place the user in the administrators group.

2 Specify the new username in the *New User Name* field, click *Create*, then click *Close*.

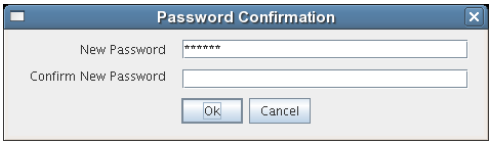
3 Define the user password.

3a In the Explorer tree of the Orchestration Console, select the new User in the Users object *all* group to open its Info/Groups page.

3b In the Info/Groups page, select the collapse/expand icon in the Personal Information section to open the fields of that section.



3c In the *Password* field, change the default password, then click the Save icon to display the Password Confirmation dialog box.



3d In the *Confirm New Password* field, enter the password you defined previously, click *OK*, then click the Save icon to save the password.

When a user logs out, the User object icon is dimmed in the Explorer tree or in the admin view of each User group to which it belongs.

2.3.4 Logging In a User for Manual Registration

If you do not select the *Auto Register Users* check box on the grid object's *Info/Configuration* page, you have the option of explicitly accepting or denying the login attempts of a user, thus preventing that user from creating an account.

1. Make sure that the *Auto Register Users* check box on the grid object's Authentication page is not selected (see Step 2 in "Automatically Registering a Resource" in the *NetIQ Cloud Manager 2.1 Orchestration Installation Guide*) and that you have created a new user.
2. Use the `zos` command line interface to log in to the server.
   - **2a** From a system terminal or from an Orchestration Server login in Windows, enter the following command:

     ```
     zos login --user=user
     ```

     If you are attempting to log in to a machine other than the local host, you can alter the command to the following:

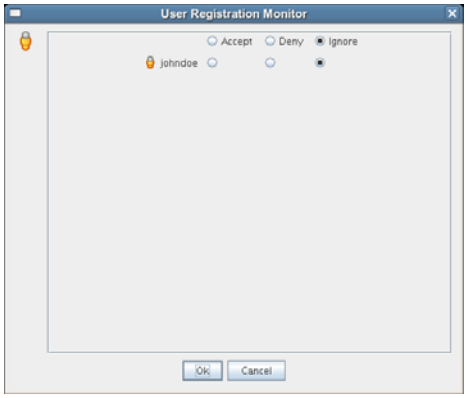
     ```
     zos login Orchestration_Server_name --user=user
     ```

   - **2b** Enter the password for the user credentials. For this walkthrough, you can simply press *Enter* to enter an empty password.
   - **2c** When prompted for a decision regarding whether you want to accept the server certificate, enter `yes`.

     An eror message is generated:

     ```
     ERROR: login failed: user name or password 'incorrect'
     ```

3 In the Users Monitor, click the User Registration icon to open the User Registration Monitor dialog box.



This dialog box lets you preview the users who are trying to log in to the server. The top row of radio buttons is a mass selector for all listed users, allowing you the choice to accept, deny, or ignore automatic registration for all listed agents.

If you want to choose the users that can be allowed to auto register, you can identify the user by name and select how you want to handle that agent's request for registration the next time it tries to log in.

4 For this example, select the *Accept* radio button adjacent to the user you want to register, then click *OK*.

The user account is created, but is not currently running jobs, as indicated by its object icon in the Explorer panel, or in the Information view of each User group to which it belongs.

2.3.5 Directory Service Authentication (Optional)

There are some configuration steps you need to follow in the Orchestration Console if you want to immediately configure the authentication of both users and resources to the Orchestration Server using a directory service like ADS or LDAP. For more information, see "Orchestration Server Authentication Page" in the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*.

2.4 Running a Sample Job

Deployed jobs can be run from the Scheduler utility inside of the Orchestration Server Console or from the `zos` command line. For the purpose of this walkthrough, we will run a sample job from the command line after logging in as a user.

1 From a system terminal, enter the following command:

```
zos run whoami
```

If you have more than one resource connected, you might want to run the job on one resource in particular. If you want do run the job this way, you can do so by adding an argument to the command line:

```
zos run whoami resource=name_of_resource
```

Now that you have run the sample job, you need to use some of the Orchestration Server tools to verify that it has run. For more information, see Section 2.5, “Looking at the Job After It Has Run,” on page 36

2.5 Looking at the Job After It Has Run

After you have run the job, there are several ways to verify that it has run. This section explains those methods.

2.5.1 Verification at the Command Line

The following sections explain some of the `zos` commands that you can use to verify that a job has run and monitor some of the results of the job:

zos jobs

You can use the `zos jobs` command to list all of the jobs that have run while you have been logged in as a user. Running this command yields an output like this:

```
Job         JobID           User       Started             Elapsed  State
whoami       userA.whoami.60  userA       12/24/2008 02:35:38  0:00:00   Completed
```

zos jobinfo job_name

You can use the `zos jobinfo -e` *`job_name`* command to display information for a named job the last time it was run. Running this command yields an output like this:

```
Jobname/Parameters    Attributes
------------------    ----------
whoami              Desc: This is a demo example of enhanced exec

    numJoblets        Desc: The number of joblets to schedule
                      Type: Integer
                   Default: 1

    resource          Desc: The resource id to run on
                      Type: String
                   Default: .*
```

zos status --detail

You can use the `zos status --detail` command to display the status of the most recently run job. Running this command yields an output like this:

```
Job Status for userA.whoami.60
------------------------------
              State: Completed
     Resource Count: 0              (0 this job)
   Percent Complete: 100%
          Queue Pos: n/a
    Child Job Count: 0              (0 this job)
      Joblet Counts: 1 (0)          (1 (0/0/1/0/0) this job)

      Instance Name: whoami
           Job Type: whoami
               Memo:
           Priority: medium
          Arguments: resource=tszen4_agent

        Submit Time: 12/24/2008 02:35:38
      Delayed Start: n/a
         Start Time: 12/24/2008 02:35:38
           End Time: 12/24/2008 02:35:39
       Elapsed Time: 0:00:00
         Queue Time: 0:00:00
         Pause Time: 0:00:00

     Total CPU Time: 0:00:00        (0:00:00 this job)
      Total GCycles: 0:00:00        (0:00:00 this job)
         Total Cost: $0.0002        ($0.0002 this job)
          Burn Rate: $0.8982/hr     ($0.8982/hr this job)

   Termination Type: n/a
          Job Error: <none>

 Joblet Error Count: 0              (0 this job)
   Node Error Count: 0              (0 this job)
     Excluded Nodes: 0              (0 this job)

Bad Provision Count: 0              (0 this job)
Excluded Provisions: 0              (0 this job)
```

zos log job_id --verbose

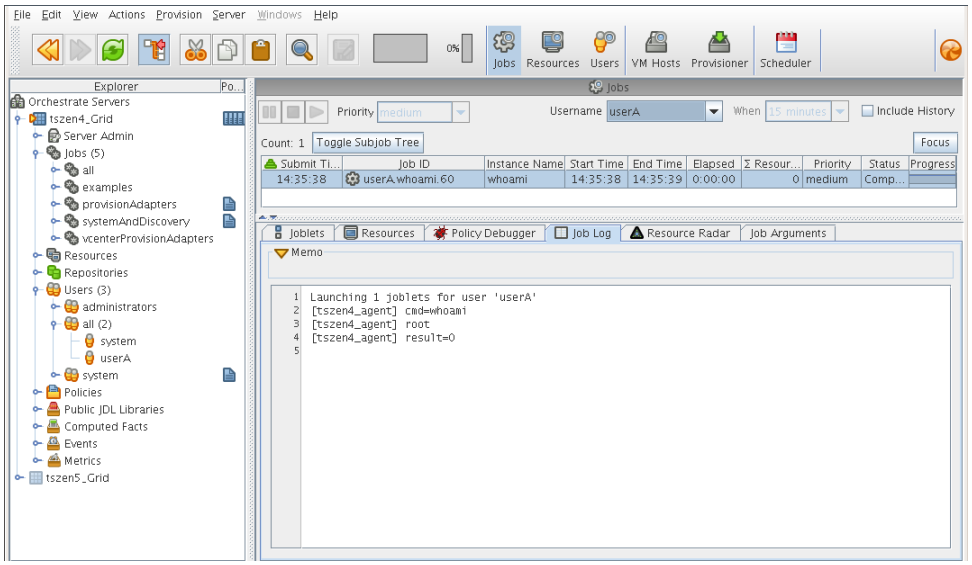
If you know the Job ID for a particular job that has run, you can use the `zos log job_id --verbose` command to display its detailed job log. Running this command yields an output like this:

```
Agent connected with ID: user_userA_64 (comms ok), Session ID: 52
Launching 1 joblets for user 'userA'
[tszen4_agent] cmd=whoami
[tszen4_agent] root
[tszen4_agent] result=0
Agent ID: null logged out.
```

2.5.2 Verification at the Jobs Monitor

If you want to use the Orchestration Server Console to verify that the job has run, you can open the *Jobs* Monitor to look at the Job Log and see the result of the job being run.

1. In the Orchestration Console, click *Jobs* in the main toolbar to open the Jobs Monitor view.
2. In the Jobs Monitor view, click the *Job Log* tab to open the Jobs page in the workspace panel.



3 In the *Username* drop-down list box, select the user who ran the `whoami` job.

Although there is much more you can learn about a job in the Jobs Monitor view, you can see by displaying a recent job that the Orchestration Console picks up the job activity and makes it available at the console.

2.6 Using the zosadmin Command to Gather Information

You can use the `zosadmin` command line to learn what users or nodes are defined in your Orchestration Server. Follow these steps to learn about the users and nodes in your system.

1 Log in to the Orchestration Server by using the following command:

```
zosadmin login
```

You can also use the server's host name as an argument when you log in.

If you use the grid= parameter, you can specify the grid name you want to log into. If other Orchestration Servers are installed on the local host, the system cannot log in to any of them unless you use this parameter. For more information, see "The zosadmin Command Line Tool" in the *NetIQ Cloud Manager 2.1 Orchestration Server Command Line Reference*.

2 Enter the administrator's user name and password.

If the login was successful, the command line tool returns a message like this:

```
Logged into tszen4_grid on server 'tszen4.provo.company.com'
```

3 Enter the following command to list the active users on your Orchestration Server:

```
zosadmin users
```

You can add the `--help` option at the command line to determine the run options for this command.

4 Enter the following command to list the active nodes on your Orchestration Server:

```
zosadmin nodes
```

You can add the `--help` option at the command line to determine the run options for this command.

With the completion of this part of the walkthrough, you have a good understanding of the parts of Orchestration Server and how they work. If you want more information about the details of the Orchestration Server Console, see the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*. If you want to learn about the job manager's role, see the *NetIQ Cloud Manager 2.1 Orchestration Developer Reference*.

2.7 Stopping and Starting Orchestration Components

Use the following methods for stopping and starting Cloud Manager Orchestration components.

2.7.1 Stopping and Starting the Orchestration Server

You can use the following methods for stopping and starting the Orchestration Server.

Stopping the Server

You need to shut down the Orchestration Server before you power off the computer where it is running. This routine prevents possible data corruption.You must be logged in to an Orchestration Server in order to stop it. There are two methods to stop the current server:

- If you are in the Orchestration Console, click *Server* > click *Shutdown ZOS*.
- If you are at the command line, enter the following command:

  ```
  /etc/init.d/novell-zosserver stop
  ```

Starting the Server

The Cloud Manager Orchestration components installation and configuration automatically starts the Orchestration Server. The Orchestration Server must be stopped before you can start it. You must be logged in to an Orchestration Server to start it. There are two methods you can use to start the current server:

- If you are at the command line, enter the following command:

  ```
  /etc/init.d/novell-zosserver start
  ```

- To restart the Orchestration Server from the command line, enter the following command:

  ```
  /etc/init.d/novell-zosserver restart
  ```

  This command stops the server before restarting it.

2.7.2 Stopping and Starting the Orchestration Agent

You can use the following methods for stopping and starting the Cloud Manager Orchestration Agent.

- "Stopping the Agent" on page 40
- "Starting the Agent" on page 40

Stopping the Agent

There are several methods you can use to stop the agent:

- If you are in the Orchestration Server Console, select the *Resources* Monitor, select a resource, then click the red icon in the work area to shut down that agent.
- If you are running the agent on a Windows machine, click *Start* > *Programs* > *Novell* > *ZOS* > *Agent* > *Shutdown ZOS Agent*.
- If you are at the Linux bash prompt, enter the following command:

  ```
  /etc/init.d/novell-zosagent stop
  ```

Starting the Agent

- To start the Orchestration Agent from a Windows machine, double-click the *ZOS Agent* shortcut on your desktop or click *Start* > *Programs* > *Novell* > *ZOS* > *Agent* > *Start ZOS Agent*.
- If you are at the Linux bash prompt, enter the following command:

  ```
  /etc/init.d/novell-zosagent start
  ```

2.7.3 Starting and Stopping the Orchestration Server Console

You can use the following methods for stopping and starting the Cloud Manager Orchestration Console.

- "Stopping the Orchestration Console" on page 40
- "Starting the Orchestration Console" on page 40

Stopping the Orchestration Console

To stop the Orchestration Server Console at the console itself, click *File* > *Exit*, or click the shutdown icon on the console window.

Starting the Orchestration Console

To start the Orchestration Console from a Linux machine: enter `./zoc`.

- On a SLES 11 machine, if you have installed the Orchestration Server along with the Orchestration Console, change to `/opt/novell/zenworks/zos/server/bin` and enter the following command

  ```
  ./zoc
  ```

- On a SLES 11 machine, if you have installed the Orchestration Console alone, change to `/opt/novell/zenworks/zos/clients/bin` and enter the following command

  ```
  ./zoc
  ```

To start the Orchestration Console on a Windows machine, double-click the *ZOS Clients* shortcut on your desktop or click *Start* > *Programs* > *Novell* > *ZOS* > *Clients* > *Orchestration Console.*

3 Server Discovery and Multicasting

The Cloud Manager Orchestration Server, the Orchestration Agent, and other Orchestration tools use IP multicast messages to locate servers and to announce when servers are started or shut down. If multicasting is not supported in your existing network environment, all Orchestration components allow a specific machine to be specified instead of using multicast discovery. Multicast support is not required to run the Orchestration Server, the Orchestration Agent, or any Orchestration tools.

This section includes the following multicast information:

3.1 Multicast Troubleshooting

An exhaustive tutorial of IP multicasting and troubleshooting is beyond the scope of this document. If you are having problems with multicast discovery, ensure that your operating system is configured to provide IP multicasting support. Most modern versions of Linux* and Windows* provide this support, however it might be disabled. Multicast discovery does not work unless IP multicasting is enabled by the operating system. Routing misconfiguration on the system can also lead to problems with multicast discovery.

3.2 Multicast Routes

A common problem with multicasting, particularly on Linux, is the lack of a default route or multicast network route. Most systems are configured to have at least a “default” route, and on such systems, multicast messages use the default route like any other network traffic. Systems do not necessarily require a default route. Multicasting might not function correctly on systems that lack a default route. Attempts to send messages on such systems fail with a `Network Unreachable` message because the operating system is unable to determine the correct network interface on which to send the message.

The quick solution is to add a default route on such systems. In some environments, however, it might not make sense to add a default route. In such cases, another solution is to add a network route for the 224.0.0.0/4 block representing the multicast IP address space. On Linux, for example, issue the following command as the `root` user:

```
route add -net 224.0.0.0 netmask 240.0.0.0 dev eth0
```

This command tells the system to send all multicast datagrams to the `eth0` network card by default. Substitute `eth0` with a different interface name if applicable.

3.3 Multi-homed Hosts

A multi-homed host is a machine with more than one network interface configured. This can be anything from a Linux system being used as a network router to a laptop computer with both an active Ethernet connection and an active wireless connection. If there are two or more network interfaces active at the same time (even if only one is actually being used) the system is "multi-homed."

Some operating systems like Linux provide only very rudimentary routing as a default part of the operating system. They rely on external routing software like "mrouted" to support full multicast routing. As a result, problems might arise in multi-homed machines because outgoing multicast messages are sent on only one interface in the absence of more sophisticated routing software. It is possible that the interface chosen by the operating system is incorrect. The Orchestration Server and its associated tools make a best effort to ensure that discovery queries and announcements are sent on all available interfaces. It should not be necessary to run an external routing program with the current Orchestration Server.

3.4 Multiple Subnets

By default, the Orchestration Server and its associated tools are configured to allow multicast messages to pass through up to two gateways. This allows discovery to work in multi-subnet environments, provided that the network routers on your network are properly configured to perform multicast routing. Consult the vendor's documentation for information on configuring multicast routing on your network routers.

3.5 Datagrid and Multicasting

The Cloud Manager Orchestration datagrid facility provides a multicast-based file distribution service that allows large multi-gigabyte files to be simultaneously delivered to a large number of recipient machines while using far less network bandwidth than would be used by copying the file individually to each node. This service is available only in network environments that support IP multicasting. Aside from the file multicast service, all other features of the datagrid use normal unicast network operations and do not require multicast support. The routing and troubleshooting pointers provided above for network discovery also apply to datagrid multicasting. In addition, due to the potentially large bandwidth used by file transfers, you might want to limit the set of interfaces on which files are multicasted.

3.6 Datagrid Multicast Interface Selection

On multi-homed servers, the datagrid multicast service sends outbound control and data packets on all available interfaces on the system. This allows datagrid multicasting to work "out of the box" with multi-homed servers. This behavior might not be optimal if you require multicasting of files only to a subset of the available interfaces. You can instruct the datagrid to multicast only to the desired interfaces by selecting the correct interfaces from the Orchestration Server Console on the Info/Configuration tab under *Data Grid Configuration*. Restricting the set of datagrid multicast interfaces prevents large amounts of file data from being sent to uninterested subnets.

4 Cloud Manager Orchestration and LDAP Authentication

Although the Cloud Manager Orchestration Server has its own user database and authentication mechanism, it also allows integration with an existing Lightweight Directory Access Protocol (LDAP) system for authenticating user credentials.

This section includes the following information:

4.1 What is LDAP?

At a high level, LDAP is a protocol designed to allow quick, efficient searches of directory services. Built around Internet technologies, LDAP makes it possible to easily update and query directory services over standard TCP/IP connections, and it includes many powerful features, including security, access control, data replication and support for Unicode.

LDAP is based on the Directory Access Protocol (DAP), which was designed for communication between directory servers and clients compliant to the X.500 standard. However, DAP can be difficult to implement and use, and is not suitable for use with Web applications. LDAP is a simpler, faster alternative, offering much of the same basic functionality without the performance overhead and deployment difficulties of DAP.

Because LDAP is built for a networked world, it is based on a client-server model.The system consists of one (or more) LDAP servers, which host the public directory service, and multiple clients, which connect to the server to perform queries and retrieve results. LDAP clients are built into most common address book applications, including e-mail clients like Microsoft Outlook and Qualcomm Eudora; however, since LDAP-compliant directories can store a diverse range of data (not just names and phone numbers), LDAP clients are also increasingly making an appearance in other applications. LDAP support is included in the Orchestration Server to allow it to integrate with existing user authentication mechanisms that are being used in many data centers.

There are many similarities between the Internet Domain Name System (DNS) model and LDAP: both are global directories that can be split across multiple hosts, both have built-in redundancy and replication features, and both include referral capabilities that make it possible to retrieve data that is not available locally from other hosts in the system.

4.2 Understanding LDAP Structure

An LDAP directory is usually structured hierarchically as a tree of nodes (the LDAP directory tree is sometimes referred to as the Directory Information Tree, or DIT). Each node represents a record, or "entry" in the LDAP database.

This section includes the following information:

4.2.1 The Distinguished Name

An LDAP entry consists of numerous attribute-value pairs. It is uniquely identified by what is known as a "distinguished name" (DN).

To draw a parallel with a relational database management system (RDBMS), an LDAP entry is analogous to a record, its attributes are the fields of that record, and a DN is a primary key that uniquely identifies each record.

Consider the following example of an LDAP entry:

```
dn: mail=joe@novell.com, dc=novell, dc=com
objectclass: inetOrgPerson
cn: Joe
sn: Somebody
mail: joe@novell.com
telephoneNumber: 1 234 567 8912
```

This is an entry for a single person, Joe Somebody, who works at Novell. The components of the entry – name, email address, telephone number – are split into attribute-value pairs, with the entire record identified by a unique DN (the first line of the entry). Some of these attributes are required and some are optional, depending on the object class being used for the entry; however, the entire set of data constitutes a single entry, or node, on the LDAP directory tree.

4.2.2 The Relative Distinguished Name

Every entry in the directory tree has a "relative distinguished name" (RDN) consisting of one or more attribute-value pairs. An RDN must be unique at that level in the directory hierarchy. In the example above, for instance, the following are all valid RDNs for the entry:

```
cn=Joe
```

or

```
cn=Joe+sn=Somebody
```

or

```
cn=Joe+sn=Somebody+telephoneNumber=12345678912
```

or

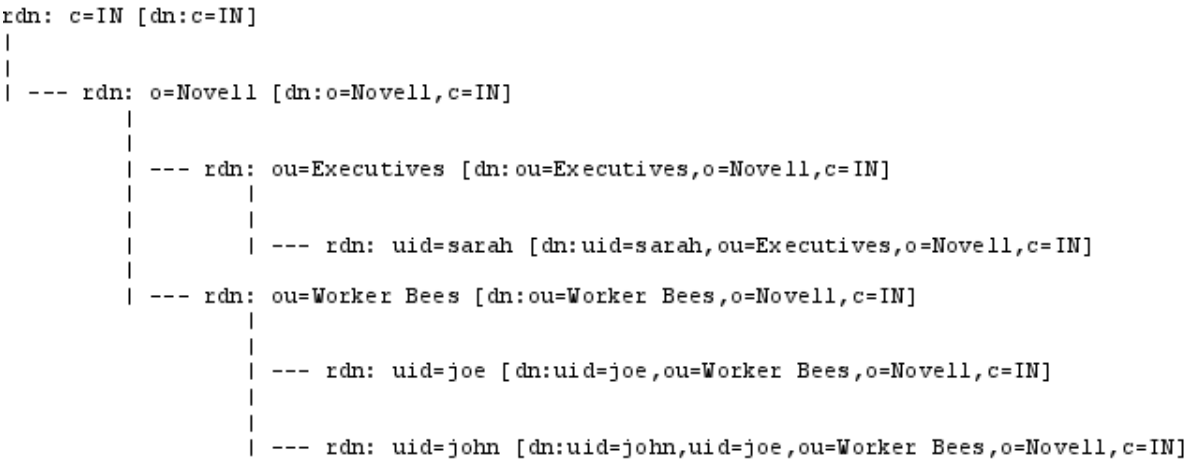
```
mail=joe@novell.com
```

There are no set rules regarding which attributes of a particular entry should be used for the RDN; the LDAP model leaves this decision to the directory designer, specifying only that the RDN of an entry must be such that it can uniquely identify that entry at that level in the DIT.

Because RDNs exist for every entry in the tree, the DN for any entry is formed by sequentially appending the RDNs of all the nodes between that entry and the root entry. In this way, you can use the DN to easily locate any node in the directory tree, regardless of its location or depth in the hierarchy.

For example, consider the following LDAP directory:

Figure 4-1 *Sample LDAP Directory*



To identify the node belonging to Joe Somebody (the DN for Joe Somebody's entry) you would add all the RDNs between that entry and the root of the tree:

```
uid=joe,ou=Worker Bees,o=Novell,c=IN
```

In a similar manner, the DN for the node belonging to Sarah would be

```
uid=sarah,ou=Executives,o=Novell,c=IN
```

while the DN for the Novell node would be

```
o=Novell,c=IN
```

Because LDAP entries are arranged in a hierarchical tree, and because each node on the tree can be uniquely identified by a DN, the LDAP model lends itself to sophisticated queries and powerful search filters.

4.3 Configuring the Orchestration Server for LDAP or ADS Authentication

There are some configuration steps you need to follow in the Orchestration Console if you want to immediately configure the authentication of both users and resources to the Orchestration Server through a directory service like ADS or LDAP.

The Cloud Manager Orchestration Server uses only one attribute of a given LDAP user: its group membership. For example, if the following settings were already configured in the Orchestration Server,

```
BaseDN 'dc=domain,dc=novell,dc=com'
UserAttribute 'uid'
UserPrefix 'ou=Users'
```

you could further configure the Orchestration Server to identify users belonging to an LDAP group using the setting `LDAP:groupnocase:administrators`.

You would do this by specifying a filter in the Orchestration Server using these settings:

```
GroupFilter 'memberUid=${USER_NAME}'
GroupPrefix 'ou=Groups'
GroupAttribute 'cn'
```

Applying these settings would let authenticated users belonging to the "administrators" LDAP group be added to the "administrators" user group in the Orchestration Server (and so allow them to log in to the Orchestration Console, for example).

For information on configuring these settings in the Orchestration Server, see "Orchestration Server Authentication Page" in the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*.

NOTE: Depending upon your selection at the *Server Type* drop down list on the *Enable LDAP* subpanel of the *Authentication* page of the Orchestration Console the configuration fields change to reflect the relevant settings. (One server type is *Active Directory Service*, the other is *Generic LDAP Directory Service*.)

---

5 Configuring the Orchestration Server to Use an Audit Database

When you install Cloud Manager Orchestration Server, you can optionally point it to a relational database that you can use to audit the work done by the product. There is no relational database management system bundled with the product, but because the Orchestration Server is supported by default on SLES 11 SP1, you can use a PostgreSQL database that ships with SLES 10 (or SLES 11) and configure it for use with Orchestration Server auditing. If you want to use another database, you must configure it separately for use with the Orchestration Server.

5.1 Installing the PostgreSQL Package and Dependencies on an Independent Host

When you enable and configure Orchestration Server auditing, you create a small custom database and a simple schema that persists all of the Orchestration Server jobs that have been run, along with their parameters. The database also maintains the login or logout activity of the Orchestration Server users and resources and includes an “actions” table that records provisioning actions and their status (started, failed, completed successfully, etc.).

NOTE: We recommend that you install the PostgreSQL packages on a SLES 10 SP3 server (or SLES 11 SP1 server) that is different from the server where you install the Orchestration Server. This ensures an adequate amount of space for running the server as the database is used.

We also recommend that you open TCP port 5432 (or whatever port you configure PostgreSQL to use—5432 is the PostgreSQL default) in the firewall of the RDBMS host. Without an open port in the host firewall, a remote Orchestration Server cannot access the audit database.

For high availability Orchestration Server configurations, you need to install the database outside of the high availability cluster.

If you want to run the database on the same host with the Orchestration Server, see Section 5.6, "Installing and Configuring the Orchestration Server for Use with a Local PostgreSQL Audit Database," on page 54.

If the SLES 10 SP3 machine (or the SLES 11 SP1 machine) does not have PostgreSQL packages installed and running, use YaST to search for `postgresql-server`, then install the package and its dependencies.

You can also run the following command from the bash prompt:

```
yast2 -i postgresql-server
```

When PostgreSQL is installed, you need to create the default database and start it. Use the following commands:

```
su - postgres

initdb

pg_ctl start
```

These commands create or update the PostgreSQL privilege database and installs the prepared tables. For more detail about what you will see when you run these commands, see "Detail" on page 50.

NOTE: You cannot run the `pg_ctl` command as `root`. You must first change to the superuser for PostgreSQL (`su - postgres`). Failure to issue this command first results in the following messages:

```
# pg_ctl start
pg_ctl: cannot be run as root
Please log in (using, e.g., "su") as the (unprivileged) user that will
own the server process.
```

5.1.1 Detail

```
postgres> initdb
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale en_US.UTF-8.
The default database encoding has accordingly been set to UTF8.

creating directory /var/lib/pgsql/data ... ok
creating directory /var/lib/pgsql/data/global ... ok
creating directory /var/lib/pgsql/data/pg_xlog ... ok
creating directory /var/lib/pgsql/data/pg_xlog/archive_status ... ok
creating directory /var/lib/pgsql/data/pg_clog ... ok
creating directory /var/lib/pgsql/data/pg_subtrans ... ok
creating directory /var/lib/pgsql/data/pg_twophase ... ok
creating directory /var/lib/pgsql/data/pg_multixact/members ... ok
creating directory /var/lib/pgsql/data/pg_multixact/offsets ... ok
creating directory /var/lib/pgsql/data/base ... ok
creating directory /var/lib/pgsql/data/base/1 ... ok
creating directory /var/lib/pgsql/data/pg_tblspc ... ok
```

```
selecting default max_connections ... 100
selecting default shared_buffers ... 1000
creating configuration files ... ok
creating template1 database in /var/lib/pgsql/data/base/1 ... ok
initializing pg_authid ... ok
enabling unlimited row size for system tables ... ok
initializing dependencies ... ok
creating system views ... ok
loading pg_description ... ok
creating conversions ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

WARNING: Enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the
next time you run initdb.

Success. You can now start the database server using:

    postmaster -D /var/lib/pgsql/data
or
    pg_ctl -D /var/lib/pgsql/data -l logfile start

postgres> postmaster -i
```

5.2 Configuring PostgreSQL to Accept Remote Database Connections

To configure the PostgreSQL database to accept remote database connections, you need to add the following line to the `/var/lib/pgsql/data/pg_hba.conf` file:

```
host    all      all         0.0.0.0/0      trust
```

NOTE: After initial configuration, you can replace the `0.0.0.0/0` with a more restrictive mask. In a high availability server configuration, make sure that each host in the high availability cluster is enabled as a remote host.

For added security, the `/var/lib/pgsql/data/pg_hba.conf` file should list only the desired hosts. For example, only the Orchestration Server would be included in the `trust` line.

After you make the change to the `pg_hba.conf` file, you need to specify the following command so that you do not receive an error when remote hosts try to connect:

```
pg_ctl reload
```

If `pg_hba.conf` is not configured andyou attempt to connect, an error similar to the following is displayed:

```
psql: FATAL:  no pg_hba.conf entry for host "164.99.15.64", user "postgres",
database "postgres", SSL off
```

Depending on the environment, you might need to perform some additional configuration for remote database setup. Editing the `listen_addresses` section of the `postgresql.conf` file enables the database server to listen for incoming connections on the specified IP addresses. The following is an excerpt from that section of the file:

```
listen_addresses = 'localhost'
                    # what IP address(es) to listen on;
                    # comma-separated list of addresses;
                    # defaults to 'localhost', '*' = all
```

After you modify the `listen_addresses` entry in `postgresql.conf`, use the following command to restart the PostgreSQL server (recommended in the PostgreSQL documentation):

```
pg_ctl restart
```

5.3 Logging in Locally to the PostgreSQL Database

When you have installed the database, the next step is to check that you can connect to the database on the database host. The default admin username is `postgres`. Use the following commands to set up a password for the postgres user on the database host machine:

```
psql
```

NOTE: Remember the password. You need to use it later to log in to the database.

Running this command results in a screen like this:

```
Welcome to psql 8.1.11, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

postgres=# alter user postgres password 'pass';
ALTER ROLE
postgres=#
```

5.4 Creating an Orchestration Server User for the PostgreSQL Database

Next, set up a PostgreSQL user to own the audit database schema before you run the server configuration script or the GUI Configuration Wizard.

1. On the database host machine, use the following commands to log in as `root` at the database host machine:

   ```
   su - postgres
   ```

   ```
   psql
   ```

2. At the `psql` prompt on the database host, use the following command to create an audit database schema user, for example:

   ```
   postgres=# create user zos password 'zos';
   ```

   ```
   CREATE ROLE
   ```

   Single quotes surrounding the password are required.

3. Enter the `\q` command at the `psql` prompt to exit the database.

5.5 Configuring the Orchestration Server Audit Database on a Separate Host

The easiest way to configure the audit database is to do so when you configure the Orchestration Server. Use the following procedure to configure the database.

NOTE: The questions presented in the text-based config script are shown here, but the questions presented in the graphical Configuration Wizard are similar.

---

1. After you have installed the Cloud Manager Orchestration packages you want, run the configuration (either the config script or the graphical Configuration Wizard) until you see the following question:

   ```
   Enable Auditing (y/n) [no]:
   ```

2. Enter `yes` to answer this question. The following question displays:

   ```
   Configure Audit DB (y/n) [no]:
   ```

3. Enter `yes` to answer this question. The following question displays:

   ```
   Jdbc URL [jdbc:postgresql://localhost/]:
   ```

4. Enter the URL of the server where PostgreSQL is running, then press Enter.

   `jdbc:postgresql://`*`IP_address_of_database_server`*`/`

   This is a standard JDBC URL because this is a Java server that uses JDBC for the interface database. The URL must be properly formed, with a slash and without a database name at the end. We do not recommend using "localhost" as the URL.

   The following prompt is displayed:

   ```
   DB Admin Username:
   ```

5. Specify the PostgreSQL database administrator username, then press Enter.

   This is the same username that was created when PostgreSQL was installed. In most instances, the username is `postgres`.

   The following prompt is displayed:

   ```
   DB Admin Password:
   ```

6. Specify the PostgreSQL database administrator password, then press Enter.

   The following prompt is displayed:

   ```
   Retype password:
   ```

7. Retype the database administrator password to verify it, then press Enter. The following prompt is displayed:

   ```
   ZOS Audit Database Name [zos_db]:
   ```

8. Specify the name of the database you want to create for Orchestration Server auditing, then press Enter. The following prompt is displayed:

   ```
   Audit DB Username:
   ```

9 Specify the name you want to use for the PostgreSQL database user that will be used by the Orchestration Server for auditing (that is, a user with Read and Write privileges, not the administrator), then press Enter. The following prompt is displayed:

```
Audit DB Password:
```

10 Specify the password you want to use for authentication by the designated PostgreSQL database user, then press Enter. The following prompt is displayed:

```
Retype password:
```

11 Retype the password, then press Enter.

After you retype the new audit database password, the configuration interview for the Orchestration Server continues normally.

5.6 Installing and Configuring the Orchestration Server for Use with a Local PostgreSQL Audit Database

When you install the Cloud Manager Orchestration Server, you can optionally point it to a relational database that you can use to audit the work done by the product. There is no relational database management system bundled with the product, but because the Orchestration Server is supported by default on SLES 10 SP3, SLES 10 SP4, or SLES 11 SP1, you can use a PostgreSQL database and configure it for use with Orchestration Server auditing. If you want to use some other database, you must configure it separately for use with Cloud Manager.

- Section 5.6.1, “Installing the PostgreSQL Package and Dependencies,” on page 54
- Section 5.6.2, “Configuring PostgreSQL to Accept Local Database Connections,” on page 55
- Section 5.6.3, “Logging in Locally to the PostgreSQL Database,” on page 55
- Section 5.6.4, “Installing and Configuring the Local Orchestration Server Audit Database,” on page 56

5.6.1 Installing the PostgreSQL Package and Dependencies

NOTE: We recommend that you install the PostgreSQL package on a SLES 10 SP3, SLES 10 SP4, or a SLES 11 SP1 server that is different from the server where you install the Cloud Manager Orchestration Server. This ensures an adequate amount of space for running the server as the database is used.

For more information, see Section 5.2, “Configuring PostgreSQL to Accept Remote Database Connections,” on page 51.

If your SLES 10 SP3.,SLES 10 SP4, or SLES 11 SP1 machine does not have the PostgreSQL package installed and running, use YaST to search for `postgresql-server`, then install the package and its dependencies.

You can also run the following command from the bash prompt:

```
yast2 -i postgresql-server
```

When PostgreSQL is installed, you need to create the default database and start it. Use the following commands:

```
su - postgres
```

```
initdb

pg_ctl start
```

These commands create or update the PostgreSQL privilege database and install the prepared tables. For more detail about what you will see when you run these commands, see "Detail" on page 50.

NOTE: You cannot run the `pg_ctl` command as `root`. You must first change to the superuser for PostgreSQL (`su - postgres`). Failure to issue this command first results in the following messages:

```
# pg_ctl start
pg_ctl: cannot be run as root
Please log in (using, e.g., "su") as the (unprivileged) user that will
own the server process.
```

5.6.2 Configuring PostgreSQL to Accept Local Database Connections

To configure the PostgreSQL database to accept remote database connections, you need to change the following line in the `/var/lib/pgsql/data/pg_hba.conf` file:

```
host    all      all         0.0.0.0/0      ident sameuser
```

The line should be changed as follows:

```
host    all      all         0.0.0.0/0      trust
```

5.6.3 Logging in Locally to the PostgreSQL Database

When you have installed the database, the next step is to check that you can connect to the database on the database host. The default admin username is `postgres`. Use the following commands to set up a password for the postgres user on the database host machine:

```
psql
```

NOTE: Remember the password. You need it to log in to the database later.

Running this command results in a screen like this:

```
Welcome to psql 8.1.11, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

postgres=# alter user postgres password 'pass';
ALTER ROLE
postgres=#
```

NOTE: This is the message you would see if you are logging in to PostgreSQL on a SLES 10 SP3 or SLES 10 SP4 server. If logging in to Postgres on a SLES 11 SP1 server, you would see a message indicating a login to psql 8.3.9.

5.6.4 Installing and Configuring the Local Orchestration Server Audit Database

When you enable and configure Orchestration Server auditing, you create a small custom database and a simple schema that persists all of the Orchestration Server jobs that have been run, along with their parameters.The database also maintains the login or logout activity of the Cloud Manager users and resources.

The easiest way to configure the audit database is to do so when you configure the Orchestration Server. Use the following procedure to configure the database.

NOTE: The questions presented in the text-based config script are shown here, but the questions presented in the graphical Configuration Wizard are similar.

---

1. After you have installed the Cloud Manager packages you want, run the configuration (either the config script or the graphical Configuration Wizard) until you see the following question:

   ```
   Enable Auditing (y/n) [no]:
   ```

2. Enter `yes` to answer this question. The following question displays:

   ```
   Configure Audit DB (y/n) [no]:
   ```

3. Enter `yes` to answer this question. The following question displays:

   ```
   Jdbc URL [jdbc:postgresql://localhost/]:
   ```

4. Press Enter to accept the default (`jdbc:postgresql://localhost/`).

   This is a standard JDBC URL because this is a Java server that uses JDBC for the interface database. The URL must be properly formed, with a slash and without a database name at the end.

   The following prompt is displayed:

   ```
   DB Admin Username:
   ```

5. Specify the PostgreSQL database administrator username, then press Enter.

   This is the same name that was specified when PostgreSQL was installed. In most instances, the username is `postgres`.

   The following prompt is displayed:

   ```
   DB Admin Password:
   ```

6. Specify the PostgreSQL database administrator password, then press Enter.

   The following prompt is displayed:

   ```
   Retype password:
   ```

7. Retype the database administrator password to verify it, then press Enter. The following prompt is displayed:

   ```
   ZOS Audit Database Name [zos_db]:
   ```

8. Specify the name of the database you want to create for Orchestration Server auditing, then press Enter. The following prompt is displayed:

   ```
   Audit DB Username:
   ```

9 Specify the name you want to use for the PostgreSQL database user that will be used by the Orchestration Server for auditing (that is, a user with Read and Write privileges, not the administrator), then press Enter. The following prompt is displayed:

```
Audit DB Password:
```

10 Specify the password you want to use for authentication by the designated PostgreSQL database user, then press Enter. The following prompt is displayed:

```
Retype password:
```

11 Retype the password, then press Enter.

After you retype the new audit database password, the configuration interview for the Orchestration Server continues normally.

5.7 Configuring the Audit Database after the Cloud Manager Orchestration Server Is Configured

If you have already installed and configured the Cloud Manager Orchestration Server, it is still possible to configure an audit database.

1 On the Orchestration Server host machine, use your favorite editor to edit the script `/opt/novell/zenworks/zos/server/conf/audit_db_prep.sql`:

1a Replace the `${DB_NAME}` variable with the PostgreSQL database name (for example, `zos_db`).

1b Replace the `${DB_USER}` variable with the PostgreSQL schema owner name (for example, `zos`).

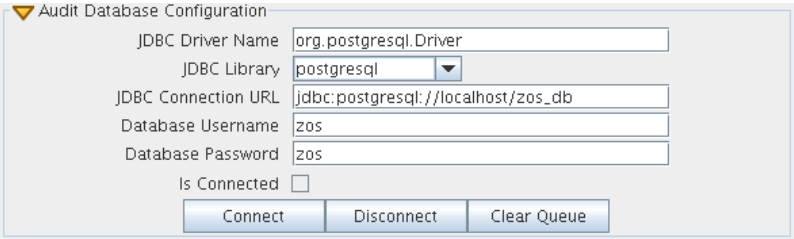
2 Use the following commands to run the modified script as the PostgreSQL database administrator:

```
su - postgres
psql -f audit_db_prep.sql
```

3 Use the following command to log into PostgreSQL, using the database name and schema owner substituted in Step 1 above:

```
su - postgres
psql -d zos_db -U zos -f audit_db_def.sql
```

4 Confirm that the database username and password match the values used when creating the schema owner database user in Section 5.4, “Creating an Orchestration Server User for the PostgreSQL Database,” on page 52. In this example, the username is `zos` and the password is `zos`.



5 Confirm that the database username and password match the values you replaced in the variables of the `.sql` script. In this example, the username is `zos` and the password is `zos`.

6 Click *Connect*.

The *Is Connected* check box is selected; the Orchestration Server is connected to the database so that any queued data and subsequent job, user, and resource events are written there.

5.8 Configuring the Remote Audit Database after the Cloud Manager Orchestration Server Is Configured

If you have already installed and configured the Cloud Manager Orchestration Server, it is still possible to configure an audit database.

1 On the Orchestration Server host machine, use your favorite editor to edit the script `/opt/novell/zenworks/zos/server/conf/audit_db_def.sql`:

1a Replace the `${DB_NAME}` variable with the PostgreSQL database name (for example, `zos_db`).

1b Replace the `${DB_USER}` variable with the PostgreSQL schema owner name (for example, `zos`).

2 Use the following commands to run the modified script as the PostgreSQL database administrator for the remote database:

```
su - postgres
```

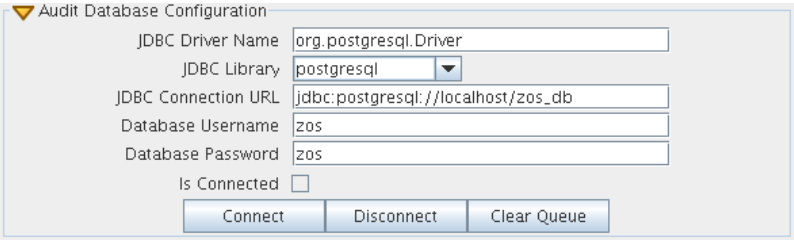
```
psql -h <psql-server-addr> -d postgres -U postgres -f audit_db_prep.sql
```

3 Use the following command to log into PostgreSQL, using the database name and schema owner substituted in Step 1 above:

```
su - postgres
```

```
psql -h <psql-server-addr> -d zos_db -U zos -f audit_db_def.sql
```

4 Confirm that the database username and password match the values used when creating the schema owner database user in Section 5.4, “Creating an Orchestration Server User for the PostgreSQL Database,” on page 52. In this example, the username is `zos` and the password is `zos`.



5 Confirm that the database username and password match the values you replaced in the variables of the `.sql` script. In this example, the username is `zos` and the password is `zos`.

6 Click *Connect*.

The *Is Connected* check box is selected; the Orchestration Server is connected to the database so that any queued data and subsequent job, user, and resource events are written there.

5.9 Modifying Audit Database Tables to Accommodate Long Names

If your installation of the Cloud Manager Orchestration Server uses Grid Object names that have an unusual number of characters, the server might lose its connection with the audit database.

If your Grid Objects are named with long names, you might need to configure some of the table columns in the audit database with different sizes. Here are some things you need to know about the database and how to make such changes:

- The default length of some names is predefined in the audit database. For example, the `username` and the `resource name` size in the audit database both default to 30 characters in allowable length.
- The workflow ID (`originWorkflowId`, `parentWorkflowId`) in the workflow table is constructed by concatenating the name of the user who invoked the job + the name of the deployed job + an instance number. The default size value is 100.
- The job instance ID (`jobinstanceid`) in the workflow table includes either the deployed name of the job or a server component name that invoked the job. For example, when the Scheduler invokes the job, then `Scheduler` is concatenated with the deployed job name. For example: `Scheduler(cpuinfo)`.
- The `name` column in the sessions table records both user and resource names.

The SQL table definition is found at `<server>/conf/audit_db_def.sql`.

Use SQL commands to change an existing table column. The following excerpts from the database show some table columns that you might need to change:

```
CREATE TABLE    actions          (

                targetobjectname        VARCHAR(50)      NOT NULL,
                username                VARCHAR(30)      NOT NULL,
                jobinstanceid           VARCHAR(100)


CREATE TABLE workflow (

       jobId VARCHAR(100) NOT NULL,
       jobInstanceName VARCHAR(100) NOT NULL,
       deployedJobName VARCHAR(30) NOT NULL,
       originWorkflowId VARCHAR(100) NOT NULL,
       parentWorkflowId VARCHAR(100),
       username VARCHAR(30) NOT NULL,


CREATE TABLE sessions (

       name VARCHAR(30) NOT NULL,
```

5.10 Understanding Grid ID Usage in the Audit Database

The Orchestration Grid ID is created using either the config or guiconfig configuration wizard operations or the `zosadmin create -g` command. The grid name you specify is displayed as the name for the container placed at the root of the tree in the Explorer panel of the Orchestration Console.The value for the Grid ID is saved in the `/var/opt/novell/zenworks/zos/server/zos.conf` file by the property `system.property.com.novell.zos.server.gridId`. Historical

records of job instances (also known as “workflows”) run on the Orchestration grid are stored in the audit database (if included in the Orchestration Server installation) and are indexed by Grid ID. If you change the value of the Grid ID, the Orchestration Console loses access to these records.

For instance, testing has shown that if you choose to upgrade the Orchestration Server using the `zosadmin create --upgrade -g` option (that is selecting a Grid ID) instead of the config or guiconfig operations, it is possible that you might not use the existing Grid ID value or that you might neglect to use a value with the command. In this case, the default (the fully-qualified domain name of the current host) is used, which could differ from the original value for the Grid ID.

If this happens, any workflows recorded in the audit database prior to the upgrade are not displayed in the Orchestration Console, but they are still recorded in the *gridid* column of the workflows table in the database.

NOTE: You can use an SQL query if you want to retrieve the workflows. The first part of such a query might look like this:

```
SELECT * FROM workflow WHERE gridId = 'labzos.pso.lab.novell.com_Grid' AND ...
```

Keep in mind that the original Grid ID is not lost. If you want a report on that ID, you can use the `zosadmin audit*` commands with a `-g` option to yield a report showing the old Grid ID.

If you want to change the Grid ID, you have several options:

- Edit `/var/opt/novell/zenworks/zos/server/zos.conf` and change the value of the Grid ID.
- Change the ID during an upgrade using `zosadmin create --upgrade -g` and with a new value for the Grid ID.
- Use SQL commands to change the Grid ID of existing records in the audit database. For example,

```
UPDATE actions SET gridid = 'newgridname' WHERE gridid = 'oldname';
UPDATE workflow SET gridid = 'newgridname' WHERE gridid = 'oldname';
UPDATE sessions SET gridid = 'newgridname' WHERE gridid = 'oldname';
```

6 Integrating the Orchestration Server with a Sentinel Collector

This section provides details about integrating the Novell Sentinel collector for the Cloud Manager Orchestration Server with your NetIQ Cloud Manager and Novell Sentinel installations to make the Orchestration Server logging easier to view and interpret.

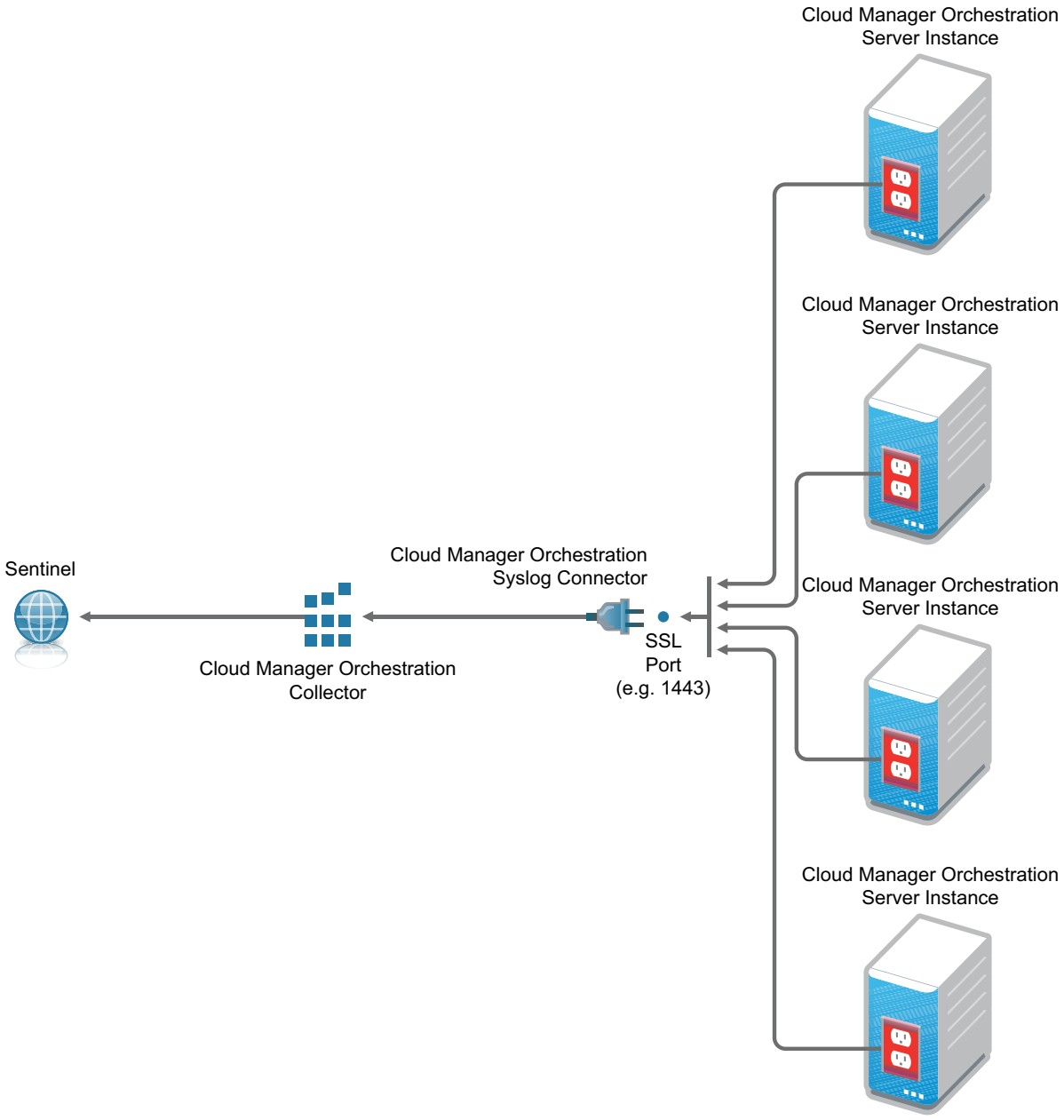
This integration is not required for the Orchestration Server to function. Logging with Novell Sentinel is entirely independent of the audit database feature. For more information about installing the audit database feature, see Section 5.1, "Installing the PostgreSQL Package and Dependencies on an Independent Host," on page 49.

6.1 Integration Architecture

Novell Sentinel is a security information and event management solution that receives information from many sources throughout an enterprise, then standardizes the information, prioritizes it, and presents it to you so that you can make threat, risk, and policy-related decisions. The Sentinel Control Center is the main user interface for viewing and interpreting this data. For overall information about Novell Sentinel, see the Novell Sentinel 6.1 product documentation Web site (http://www.novell.com/documentation/sentinel61/index.html).

The Orchestration Server can be configured to send log events to Sentinel over a single SSL connection (typically port 1443). The events are sent in RFC5424 (syslog) format, and are received by the Sentinel Event Source Server, which, for each event, parses the syslog header, and then hands the event over to the Orchestration Server Collector plug-in for Sentinel. The Sentinel collector parses the encapsulated Orchestration Server log event and performs normalization tasks before finally submitting it to the Sentinel event processing engine. These normalization tasks include mapping Orchestration Server log levels to Sentinel numerical event severities and extracting event metadata.

Figure 6-1 *Simplified Architecture for Orchestration Server Collector Integration*



NOTE: Multiple Orchestration Server instances can send syslog messages to a single Syslog Connector.

6.2 System Requirements

Integrating Sentinel and the Orchestration Server requires the following:

- Cloud Manager Orchestration Server 3.0 installed and running on a supported SUSE Linux Enterprise Server.

- A database (MS SQL or Oracle 10gR2 (http://www.novell.com/products/sentinel/techspecs.html)) installed as recommended in the *Database Installation* (http://www.novell.com/documentation/sentinel61/s61_install/data/bgmrzy2.html) section of the *Novell Sentinel 6.1 Installation Guide*.
- Sentinel 6.1.1.1 (and higher) installed and running.

NOTE: Sentinel 6.1*x* must be separately purchased from Novell.

Sentinel can be installed on the same server with the Orchestration Server, assuming that the server has sufficient RAM, processing power, and the disk space to accommodate running the two products side-by-side; otherwise they should be run on separate servers that communicate through TCP/IP.

For more information about the server requirements for the Orchestration Server, see "Cloud Manager Orchestration Server Requirements" in the *NetIQ Cloud Manager 2.1 Installation Planning Guide*.

For more information about the server requirements for Sentinel, see *System Requirements (http://www.novell.com/documentation/sentinel61/s61_install/data/bgmq7g2.html)* in the *Novell Sentinel 6.1 Installation Guide*.

- The appropriate Sentinel Syslog Connector, available for download from Novell Support (http://support.novell.com/products/sentinel/secure/sentinelplugins.html.) .
- The appropriate Orchestration Server/Sentinel Collector Plug-in, available for download on the Sentinel Collectors download page (http://support.novell.com/products/sentinel/secure/sentinelplugins.html.).

NOTE: Collector-specific information is available in `.pdf` format with the download above. Users should consult with Novell Support to determine the appropriate connector and collector for use with the Orchestration Server.

6.3 Importing and Deploying the Orchestration Server Sentinel Collector Plug-in

1. From the Sentinel Control Center, click *Event Source Management* > *Live View* to display the Event Source Management (Live View) window.
2. From the Event Source Management (Live View) window, click *Tools* > *Import plug-in* to display the Import Plug-in Wizard.
3. From the Import Plug-in Wizard, select *Import Collector Script or Connector plugin package file (.zip)*, then click *Next* to open a file browser.

IMPORTANT: When you try to import the syslog connector, you might see a message like this:

```
No universal SYSLOG collector found. Resolve this problem by importing either
the Novell Generic Collector or another Syslog collector that is designated as
a universal Syslog collector.
```

You can safely ignore this message. If you need more information, see the *Syslog Connector Installation Guide*.

---

4 From the file browser, locate and select the Orchestration Server Collector `.zip` file, `NetIQ_Cloud-Manager-Orchestration-Service_6.1r.clz.zip` (or similar), then click *OK* to display the Plugin Details page.

5 On the Plugin Details page, click *Finish* to display the Graph page of the Event Source Management Live View.

6 On the Graph page, right-click the Collector Manager icon, then click *Add Event Source Server* to display the Select Connector Plugin page of the Add Event Source Server Wizard.

7 From the Select Connector Plugin page, make sure that the Syslog connector is selected in the Installed Connectors table, then click *Next* to display Syslog Event Source Server page of the wizard.

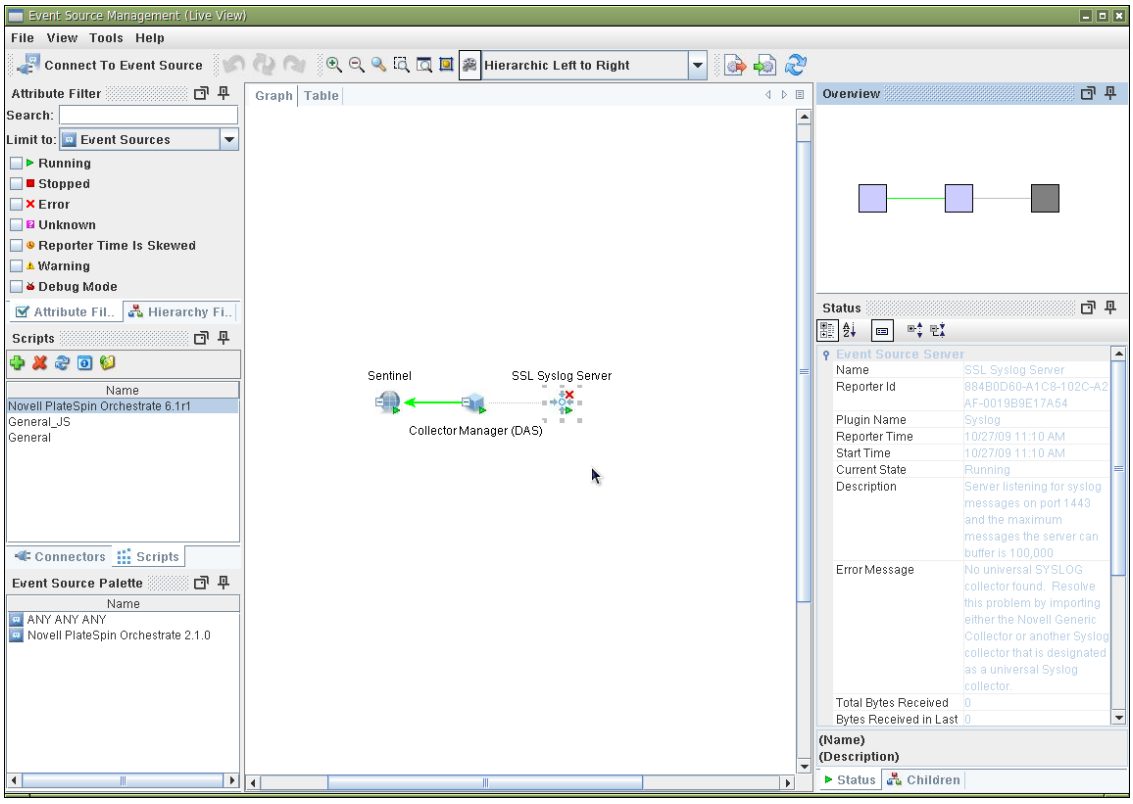
8 On the Syslog Event Source Server page of the tool, Select *SSL*, enter `1443` as the default port number, then click *Next* to display the *Message Handling* page of the wizard.

Port 1443 is used in this example to eliminate conflict with other Novell products.

It is not required that you use `1443` in this field, but any port number that you configure here must match the port number you assign for Sentinel integration in the Orchestration Console. For more information, see "Sentinel Server Configuration Panel" in the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*.

9 Click *Next* on the Message Handling page and succeeding pages of the wizard until you reach the General page.

10 On the *General* page of the wizard, make sure that the *Run* check box is selected, then click *Finish* to display the revised Graph page of the Event Source Management Live View.



In this graph page, the Event Source Server icon displays a superimposed red cross. The view also displays a warning (`No universal SYSLOG connector found...`) is visible in the right hand pane. You can safely disregard these warnings.

6.4 Connecting the Orchestration Server to the Sentinel Collector Plug-In

After you deploy the the Orchestration Server/Sentinel Collector Plug-in, use the following steps to configure the connection between the plug-in and the the Orchestration Server.

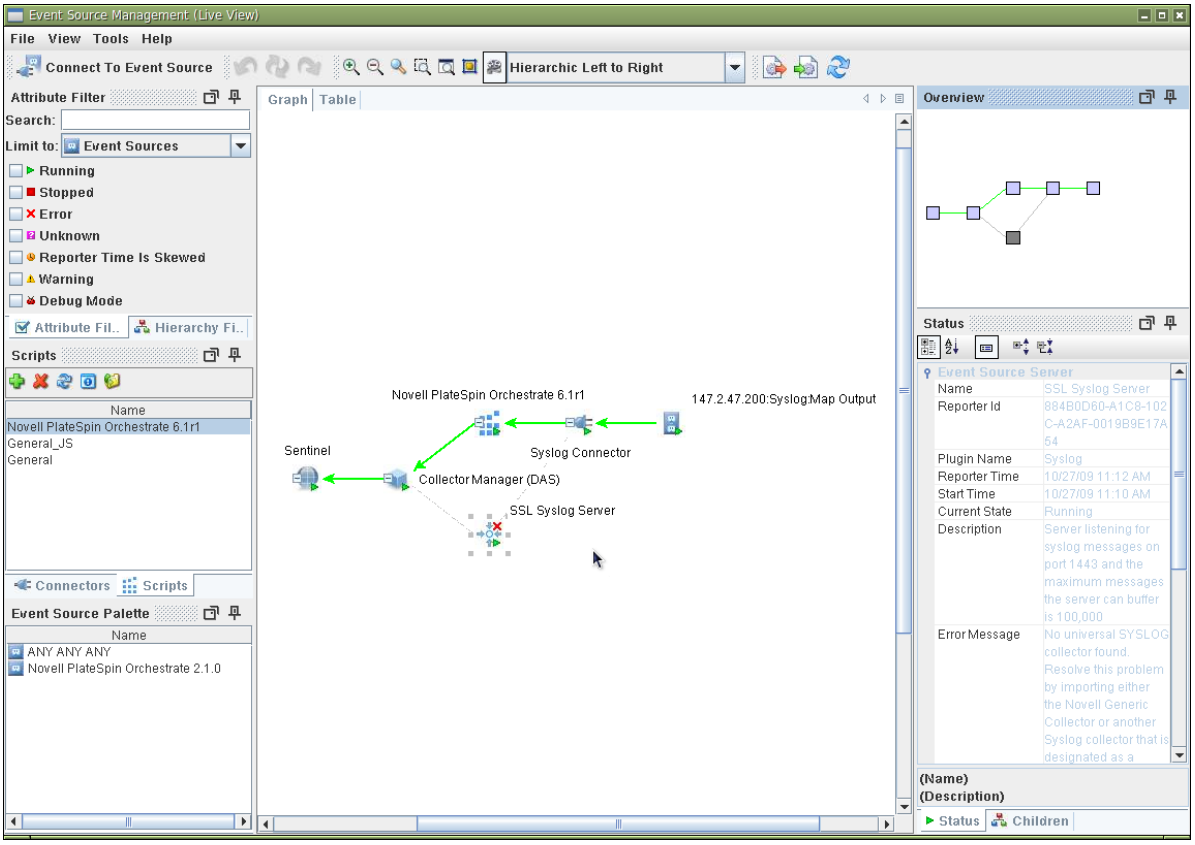
1. From a running Orchestration Console, select the Grid Server object that you want to connect to the Sentinel Collector Plug-in, then log in to that grid.
2. In Admin View of the object, select *Info*/*Configuration*, then scroll to the *Sentinel Server Configuration* panel in this view to configure the collector.
3. In the *Server Hostname* and *Server Port Number* fields, ensure that the server hostname points to the host running the Sentinel Event Source Server and that the port value is listed as `1443`, then click *Connect*.
4. Verify that the *Is Connected* check box is selected.

For more information about these fields in the Orchestration Console Admin View, see "Sentinel Server Configuration Panel" in the *NetIQ Cloud Manager 2.1 Orchestration Console Reference*.

6.5 Verifying the Sentinel Configuration After Connecting to the Orchestration Server

The first time that the Orchestration Server connects to Sentinel through the SSL port, Sentinel automatically creates and configures its Connector, Collector, and Event Source Server. You can verify this on the Graph page of the Event Source Management Live View.

Figure 6-2 *Event Source Server Diagram Showing the the Orchestration Server and Sentinel Server Connections*



6.6 Viewing Orchestration Server Data in the Sentinel Event Source Server

The Sentinel Event Source Server displays data from the Orchestration Server.

For each logging event, the Orchestration Server passes a free form text log message and log event metadata to the Sentinel Event Source Server. The following table shows how the data are mapped:

Table 6-1 *Orchestration Server Logged Data Mapped to Sentinel Event Source Server Fields*

Sentinel Event Field Name	Orchestration Server Data Represented
Message	A free-form text log message.
TargetServiceName	The Orchestration Server grid name.
ReporterIP	The IP address of the Orchestration Server.
TargetServiceComp	The originating facility of the Orchestration Server event.
EventName	A name formatted as `gridname: taxonomy_key`. For more information, see Section 6.8, "Event Classification and Taxonomy Keys," on page 69.
ProductName	The product is always `Cloud Manager Orchestration` for convenient event filtering.
InitUserName *InitUserID*	The name of the Orchestration Server user. Used for user-oriented events.

For some events, the Orchestration Server Sentinel Collector passes some structured data related to the log event. When available, this data appears in the *ExtendedInformation* Sentinel Event field, which is a list of key/value pairs delimited by semicolons. Some keys are always present in *ExtendedInformation*, but others appear only when relevant. The following table lists these keys:

Table 6-2 *Orchestration Server Key/Value Pairs Displayed in the Sentinel Event Source Server*

Key	Displayed	Value
`loglevel`	always	The original Orchestration Server log level, for example, `NOTICE`, or `INFO`.
`eventclass`	always	A symbolic identifier for this class of event, e.g. `admin_login`, `policy_association`. This also serves as the taxonomy key. The value is unclassified if the event does not contain any additional structured data.
`group`	when relevant	The Grid object group related to this log event.
`job`	when relevant	The job related to this log event.
`jobinstance`	when relevant	The job instance ID related to this log event.
`member`	when relevant	The Grid object group member related to this log event.
`policy`	when relevant	The policy related to this log event.
`repository`	when relevant	The repository related to this log event.
`resource`	when relevant	The resource related to this log event.
`schedule`	when relevant	The schedule related to this log event.
`trigger`	when relevant	The trigger related to this log event.
`type`	when relevant	The Grid object type related to this log event.
`user`	when relevant	The name of the user related to this log event. This key also appears in the *InitUserName* and *InitUserID* fields.

Key	Displayed	Value
vbridge	when relevant	The Vbridge related to this log event.
vdisk	when relevant	The Vdisk related to this log event.
vm	when relevant	The VM related to this log event.
vmhost	when relevant	The VM host server related to this log event.
vnic	when relevant	The VNIC related to this log event.
target	when relevant	The name of the Grid object related to this log event.
action	when relevant	The action related to this log event.

6.7 Orchestration Server Log Levels Mapped to Sentinel Log Levels

The information in the following table shows the correlation between the Orchestration Server logging levels and those you are likely to see in Novell Sentinel.

Table 6-3 *Orchestration Server Log Levels Mapped to Sentinel Log Levels*

Orchestration Server Log Level (alpha)	Sentinel Collector Log Level (numeric)	Comments	Usage Examples
EMERGENCY	5	Urgent conditions that require immediate attention. Indicates that the system is no longer functioning.	
ALERT	5	Conditions that should be corrected immediately.	
CRITICAL	5	Critical conditions.	
ERROR	4	Errors that have been correctly handled.	Unsuccessful job runs and provisioner/VM actions that have been correctly handled but might require manual attention.
WARNING	3	Warning messages.	Unexpected but recoverable events indicating degraded operational status, fsuch as grid objects becoming unhealthy.
NOTICE	2	Conditions that are not error conditions, but should possibly be handled specially.	Events expected occasionally as part of usual business operations, such as password changes, failed authentications and authorizations, grid objects returning to good health, server shutdown.

Orchestration Server Log Level (alpha)	Sentinel Collector Log Level (numeric)	Comments	Usage Examples
STATUS	1	Conditions that report on changes in operational conditions.	Events expected frequently as part of usual business operations, such as successful authentications (logins and logouts), deletion of grid objects, changes in group membership, deployment of jobs/policies/schedules/triggers, policy association.
INFO	1	Informational messages.	Non-security-sensitive events expected very frequently as part of usual business operations, such as successful job runs and provisioner/VM actions, resources going online/offline, session expiration/timeout.

6.8 Event Classification and Taxonomy Keys

The value of the `eventclass` key in Sentinel's *ExtendedInformation* event field specifies a class of events where this particular event belongs. This mechanism allows classification of multiple events into a single event class, even when their message bodies and/or other `ExtendedInformation` key/value pairs differ. For example, any event detailing the logout of a user from an Orchestration Server administrator account is classified with the event class `admin_logout`. This event class also serves as the taxonomy key for use by Sentinel.

Many log events are currently unclassified, so their event classes are also unclassified. We anticipate the number of unclassified log events to decrease incrementally over subsequent Cloud Manager Orchestration Server releases.

All non-failure events are logged on success rather than on action initiation. For example, an event with the event class `job_deployed` would only be seen after the job had been successfully deployed, not when the deployment attempt was initiated.

Table 6-4 *eventclass Taxonomy Keys*

Value of the eventclass Taxonomy Key	Events in This Classification	Other Keys Typically Used with This Class of Events
unclassified	Any event not yet assigned an event class.	
sentinel	Events relating to integration of Cloud Manager Orchestration Server with Sentinel.	
authorization_failure	Any kind of authorization failure.	action
authentication_failure	Any kind of authentication failure.	action
admin_login	Login to a Orchestration Server administrator account (for example, through zosadmin or through the Orchestration Console).	user

Value of the eventclass Taxonomy Key	Events in This Classification	Other Keys Typically Used with This Class of Events
admin_logout	Logout from a Orchestration Server administrator account (for example, through zosadmin or through the Orchestration Console).	user
user_login	Logout from a grid user account.	user
user_logout	Login of a resource to the grid.	user
user_password_change	Password change for grid user account.	user
resource_login	Logout of a resource from the grid.	resource
resource_logout	Logout of a resource from the grid.	resource
repository_created	New repository created.	repository
repository_deleted	Repository deleted.	repository
resource_created	New resource created.	resource, type
resource_deleted	Resource deleted.	resource
user_created	New user created.	user
user_deleted	User deleted.	user
session_not_found	User or agent session not found.	session
vbridge_created	New vBridge created.	vbridge, vmhost
vbridge_deleted	vBridge deleted.	vbridge
vdisk_created	New vDisk created.	vdisk, vm
vdisk_deleted	vDisk deleted.	vbridge
vnic_created	New vNIC created.	vnic, vm
vnic_deleted	vNIC deleted.	vnic
group_created	New group created.	group, type
group_deleted	Group deleted.	group, type
group_member_added	New member added to the Grid object group.	group, member, type
group_member_removed	Member removed from the Grid object group.	group, member, type
job_deployed	New job deployed to the grid.	job
job_undeployed	Job undeployed from the grid.	job
schedule_deployed	New schedule deployed to the grid.	schedule
schedule_undeployed	Schedule undeployed from the grid.	schedule
trigger_deployed	New trigger deployed to the grid.	trigger
trigger_undeployed	Trigger undeployed from the grid.	trigger

Value of the eventclass Taxonomy Key	Events in This Classification	Other Keys Typically Used with This Class of Events
policy_association	Association of an Orchestration Server policy with a Grid object.	policy, target
policy_disassociation	Disassociation of an Orchestration Server policy with a Grid object.	policy, target
policy_created	Policy added to the grid.	policy
policy_removed	Policy removed from the grid.	policy
job_started	Job (instance) started.	jobinstance
job_finished	Job (instance) finished.	jobinstance, outcome (completed, canceled, or failed), reason (when canceled)
vm_applyconfig	VM apply config action initiated.	vm, user
vm_build	VM build action initiated.	vm, user
vm_check_status	VM check status action initiated.	vm, user
vm_checkpoint	VM checkpoint action initiated.	vm, user, checkpoint
vm_clone	VM clone action initiated.	vm, user
vm_create_template	VM create_template action initiated.	vm, user
vm_delete	VM delete action initiated.	vm, user
vm_destroy	VM destroy action initiated.	vm, user
vm_install_agent	VM install agent action initiated.	vm, user
vm_make_standalone	VM make standalone action initiated.	vm, user
vm_migrate	VM migrate action initiated.	vm, user
vm_move	VM move action initiated.	vm, user
vm_pause	VM pause action initiated.	vm, user
vm_personalize	VM personalize action initiated.	vm, user
vm_provision	VM provision action initiated.	vm, user
vm_restart	VM restart action initiated.	vm, user
vm_restore	VM restore action initiated.	vm, user, checkpoint
vm_resume	VM resume action initiated.	vm, user
vm_saveconfig	VM save config action initiated	vm, user
vm_shutdown	VM shutdown action initiated.	vm, user
vm_suspend	VM suspend action initiated.	vm, user
resource_healthy	The Resource became healthy.	resource
resource_unhealthy	The Resource became unhealthy.	resource

Value of the eventclass Taxonomy Key	Events in This Classification	Other Keys Typically Used with This Class of Events
repository_healthy	The Repository became healthy.	repository
repository_unhealthy	The Repository became unhealthy.	repository
vbridge_healthy	The vBridge became healthy.	vbridge
vbridge_unhealthy	The vBridge became unhealthy.	vbridge
vnic_healthy	The vNIC became healthy.	vnic
vnic_unhealthy	The vNIC became unhealthy.	vnic
vdisk_healthy	The vDisk became healthy.	vdisk
vdisk_unhealthy	The vDisk became unhealthy.	vdisk
vmhost_healthy	The VM host became healthy.	vmhost
vmhost_unhealthy	The VM host became unhealthy.	vmhost

7 Cloud Manager Orchestration Security

This section explains various security issues related to NetIQ Cloud Manager Orchestration:

7.1 User and Administrator Password Hashing Methods

All passwords stored in the Orchestration Server are hashed using Secure Hash Algorithm-1 (SHA-1). However, user passwords are no longer hashed when sent from the client to the server. Instead, the plain text password entered by the user is sent over an encrypted authentication connection to the server to obtain a unique per-session credential issued by the server. This allows the server to “plug in” to alternative user directories such as Active Directory or OpenLDAP. Agent credentials are still stored, singly hashed, on the disk on the agent machine. The first pass hashing prevents “user friendly” passwords entered by administrators from being compromised by storing them on the agent machines. The server’s password database (for agents and for users not using an alternative user directory) stores all passwords in a double-hashed form to prevent a stolen password database from being used to obtain passwords.

WARNING: The zosadmin command line and the Orchestration 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 Orchestration Agent.

7.2 User and Agent Password Authentication

The Orchestration Server stores all user and agent passwords in its data store as double-hashed strings. User clients such as the zos command send the plain text password over a TLS encrypted authentication connection to obtain a randomly generated per-session credential issued by the server. This session credential is retained by the client, either in memory or in a temporary disk file for the duration of the session.

It is not possible to obtain the user's password from the session credential, however. It should be protected to prevent unauthorized users from taking over the session. Agents send a singly hashed password as their login credential, which is in turn hashed once more on the server to authenticate new agent connections. Upon authentication, agents receive the same type of session credential as user clients.

Singly-hashed password strings are used as a special case for agents, because agents typically must store their plain text credentials to disk to allow the agents to start up on host or VM reboot. The use of a once hashed version of the password on the agent prevents administrators from compromising "user friendly" text passwords by storing them unhashed on agents. The use of single hashing on the agents and double hashing on the server database prevents stolen credential data from being used to obtain actual user or administrator-entered passwords

7.3 Password Protection

You should take measures to protect the passwords and credentials on both the Cloud Manager Orchestration Server and the Cloud Manager Orchestration Agents by ensuring that only the user account of the Orchestration 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. On agents, allow only the agent users (normally `root` or `Administrator`) to have access to the `agent.properties` file, which contains the agent's authentication credential.

Currently, the Orchestration Server restricts file access on the server, but we recommend that you disallow shell accounts on server machines for general users as a precaution.

For users, none of the NetIQ-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.

Orchestration 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.

Administrators can enhance security when configuring new agents by setting the `zos.agent.password` property to the asterisk character (*). This causes the agent to automatically generate a new random credential not based on any easily guessable plain text word. When the new agent is "accepted" by the administrator, the newly generated credential is stored by the server. This is the default behavior when the Orchestration Agent is first installed.

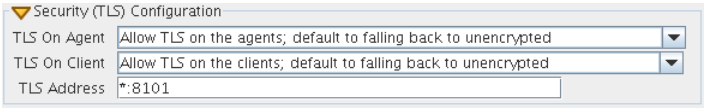
n addition, the `zos.agent.password` property can be set to a plain text password in `agent.properties`. If this is done, the agent automatically replaces the plain text password with the hashed version when it next starts. This allows administrators to more easily set up an initial password for agents.

7.4 TLS Encryption

7.4.1 Setting TLS Options

Cloud Manager Orchestration uses Transport Layer Security (TLS) to provide encryption for both user and agent connections. The Orchestration Agent, the Orchestration Console and other Orchestration clients use TLS to initiate their connections to the Orchestration 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 Orchestration 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 Orchestration Console.

Figure 7-1 *TLS Options in the Cloud Manager Orchestration Server 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 PlateSpin Orchestrate* article at the *Novell Cool Solutions Community (http://www.novell.com/communities/coolsolutions/)*.

NOTE: The principles of this article are still technically correct, although the product branding and some terms have been updated in the product since the original posting of this article.

7.4.2 Updating the TLS Server Certificate

Understanding Transport Layer Security (TLS) encryption is particularly important if you reinstall the server and have an old server certificate in either your agent or client user profile similar to `ssh` shared keys. If you have an old certificate, you need to either manually replace it or delete it and allow the client or agent to download the new one from the server using one of the following procedures:

- **For the Agent:** The TLS certificate is in `<agentdir>/tls/server.pem`. Deleting this certificate will cause the agent, by default, to log a minor warning message and download a new one the next time it tries to connect to the server. This is technically not secure, since the server could be an impersonator. If security is required for this small window of time, then the real server's `<serverdir>/<instancedir>/tls/cert.pem` can be copied to the above server.pem file.
- **For the Client:** The easiest way to update the certificate from the command line tools is to simply answer "yes" both times when prompted about the out-of date certificate. This is, again, not 100% secure, but is suitable for most situations. For absolute security, hand copy the server's cert.pem (see above) to `~/.novell/zos/client/tls/<serverIPAddr:Port>.pem`.
- **For Java SDK clients:** Follow the manual copy technique above to replace the certificate. If the local network is fairly trustworthy, you can also delete the above `~/.novell/.../*.pem` files, which will cause the client to auto-download a new certificate.

7.5 Security for Administrative Services

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

When the user logs in using either `zosadmin login` or the Orchestration Console, the user's password is sent to the server, and then the server issues a per-session credential to be used for further operations. The user's cleartext password is never stored to disk; however, it is currently sent "over the wire" in plain text form. For this reason, the administrative clients should only be used in a secure, trusted environment.

The zosadmin client stores the session credential obtained from a `zosadmin login` request in a temporary file for use by subsequent operations. This credential cannot be used to obtain the user's password, but it could be used to take over the user's current session until it times out or expires. For this reason, the files in the user's `.novell/zoc/` directory should be configured to disallow access by other users.

7.6 Plain Text Visibility of Sensitive Information

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

Table 7-1 *Locations Where Sensitive Information Might Be Stored As Plain Text*

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

A Determining the Version of an Orchestration Component

Cloud Manager Orchestration is made up of many components, which you choose whether to install when you first install the product. If you are new to Cloud Manager Orchestration, if you need to determine component compatibility, or if you need to confer with NetIQ Support, it is useful to know how to obtain the version numbers of the different components.

One way to determine which version of the Cloud Manager RPM packages you have installed is to use the following command on the machine where a Cloud Manager Orchestration component is installed:

```
rpm -qa | grep novell
```

The table below provides additional methods you can use to determine the version number for Orchestration components.

Table A-1 *Determining the Version Number for Cloud Manager Orchestration Components*

Component	How to Determine Version Number
Orchestration Server	At the Orchestration Console Explorer view, select the grid object, then open the Info/Configuration page of the workspace panel. The version number is listed in the *Server Version* field. Advanced Orchestration Server users can also find the version value in the `matrix.version` fact.
Orchestration Agent	After the agent is registered at the Orchestration Console Explorer view, select its resource object listed under *Resources*, then open the Info/Groups page of the workspace panel. On the Info/Groups page, select *Agent Information*. The version number is listed in the *Agent Version* field. Advanced Orchestration Server users can also find the version value in the `resources.agent.version` fact.
Orchestration Console	At the console, click *Help* > *About Cloud Manager Orchestration Console*. The console version number and license expiration date is listed on the About Cloud Manager Orchestration Console dialog box.
Command Line Tools (zos, zosadmin)	No method is currently available.

Component	How to Determine Version Number

Cloud Manager Monitoring Server	On the command line of the machine where the Monitoring Server is running, enter (at `/opt/novell/zenworks/monitor/sbin`) the following command:
	
	`gmetad -V`
	
	`gmetad -- version`

Cloud Manager Monitoring Agent	On the command line of the machine where the agent is running, enter (at `/opt/novell/zenworks/monitor/sbin`) the following command:
	
	`gmond -V`
	
	`gmond -- version`

B Changing Orchestration Server Default Parameters and Values

The following table provides the current default values for some key performance parameters of the Cloud Manager Orchestration Server. Although the server is fine-tuned by default for optimal performance at normal loads, if you want to perform hundreds of provisioning actions simultaneously you can change some of the default settings for increased server performance in such a scenario.

Table B-1 *Default Parameters of the Cloud Manager Orchestration Server*

Parameter Name	Shipping Default Value	Changing or Displaying the Parameter Configuration
File Descriptors Limit	2048	(Optional) You can change the value as shown below: `/etc/init.d/novell-zos-server: ulimit -n <new_limit>`
Java Heap Space	2048 MB	Create a file of the following name: `/opt/novell/zenworks/zos/server/conf/cmosadmin.properties` Add the following line to the newly created file: `default.jvmargs=-Xmx2560m`
PermGen Space	512 MB	Create a file of the following name: `/opt/novell/zenworks/zos/server/conf/cmosadmin.properties` Add the following line to the newly created file: `default.jvmargs=-XX:MaxPermSize=1536m`
Audit Queue Size Max	200	Increase the value of this parameter by using the following command: `zosadmin set --mbean="local:facility=audit" --attr=QueueSizeMax --type=Integer --value=1000`

Parameter Name	Shipping Default Value	Changing or Displaying the Parameter Configuration
MaxRunJobWaitTimeout	120000	You can change the value of this parameter as shown below: `zosadmin set --mbean="local:facility=broker" --attr=MaxRunJobWaitTimeout --type=Integer --value=<time_in_milliseconds>`
MatchingResourcesCheckinterval	30000	Increase the value of this parameter by using the following command: `zosadmin set --mbean="local:facility=broker" --attr=MatchingResourcesCheckInterval --type=Integer --value=600000`

Kernel ARP Threshold Values	◆ thresh1 = 128
	◆ thresh2 = 512
	◆ thresh3 = 1024
		Set these values higher than the default. For example:
		`cat /proc/sys/net/ipv4/neigh/default/gc_thresh1 = 256`
		`cat /proc/sys/net/ipv4/neigh/default/gc_thresh2 = 1024`
		`cat /proc/sys/net/ipv4/neigh/default/gc_thresh3 = 2048`

Job Limits: ◆ Soft Top Level Job Limit ◆ Max Queued Jobs ◆ Absolut Max Active Jobs	`matrix.maxtopjobs` = 200 `matrix.maxqueued` = 300 `matrix.maxactive` = 400	Change the Grid Object default values in the Orchestration Console as follows: `matrix.maxtopjobs` = 600 `matrix.maxqueued` = 700 `matrix.maxactive` = 1000

C Increasing the Kernel ARP Threshold Value on the Orchestration Server

Testing has shown that Cloud Manager Orchestration grids that have more than 1210 VMs (each with an installed Orchestration Agent) and 124 Resource objects, the Orchestration Server (installed on supported SUSE Linux Enterprise Server computers) in the grid loses connection to the Audit Database Server or other devices installed on other machines. The failure is manifest following the ping command when the following error is displayed:

```
connect : No buffer space is available
```

To correct this problem, it is necessary to increase the kernel ARP threshold on the Orchestration Server (SL. This section contains the following information to help you perform this increase:

C.1 Threshold Definitions

The following definitions for kernel ARP levels is referenced from the Linux ARP man page (http://linux.die.net/man/7/arp):

- **gc_thresh1:** The minimum number of entries to keep in the ARP cache. The garbage collector will not run if there are fewer than this number of entries in the cache. Defaults to 128.
- **gc_thresh2:** The soft maximum number of entries to keep in the ARP cache. The garbage collector will allow the number of entries to exceed this for 5 seconds before collection will be performed. Defaults to 512.
- **gc_thresh3:** The hard maximum number of entries to keep in the ARP cache. The garbage collector will always run if there are more than this number of entries in the cache. Defaults to 1024.

C.2 Determining the Current Kernel Threshold Value

To determine the current kernel threshold value on the Orchestration Server, use the following commands to determine the values for each threshold:

- `# cat /proc/sys/net/ipv4/neigh/default/gc_thresh1`

  The command might result in a displayed value like this:

```
# cat /proc/sys/net/ipv4/neigh/default/gc_thresh1
128
```

- `# cat /proc/sys/net/ipv4/neigh/default/gc_thresh2 512`

  The command might result in a displayed value like this:

  ```
  # cat /proc/sys/net/ipv4/neigh/default/gc_thresh2
  512
  ```

- `# cat /proc/sys/net/ipv4/neigh/default/gc_thresh3`

  The command might result in a displayed value like this:

  ```
  # cat /proc/sys/net/ipv4/neigh/default/gc_thresh3
  1024
  ```

You can also use this command:

```
#sysctl -A|grep ipv4|grep default |grep gc_thresh
```

The result is a listing similar to the following:

```
net.ipv4.neigh.default.gc_thresh3 = 1024
net.ipv4.neigh.default.gc_thresh2 = 512
net.ipv4.neigh.default.gc_thresh1 = 128
```

C.3 Changing the Current Kernel Threshold Value

When you know the current threshold values, you can change them using one of two methods:

- Section C.3.1, “Editing the /etc/sysctl.conf File,” on page 84
- Section C.3.2, “Making Live Changes to the Threshold Values,” on page 84

C.3.1 Editing the /etc/sysctl.conf File

1. Open `/etc/sysctl.conf` in a text editor.
2. Add the following lines to the `.conf` file:

   ```
   net.ipv4.neigh.default.gc_thresh1 = 256

   net.ipv4.neigh.default.gc_thresh2 = 1024

   net.ipv4.neigh.default.gc_thresh3 = 2048
   ```

3. Reboot the server.

C.3.2 Making Live Changes to the Threshold Values

To make changes to a given threshold on the Orchestration Server you can run a command for each threshold that you want to change, for example:

```
# echo '256' > /proc/sys/net/ipv4/neigh/default/gc_thresh1
```

After you run the command, perform a `/etc/init.d/network restart` command to restart the Orchestration Server and put the changes in place.

NOTE: This method of changing the threshold values is volatile: if you reboot the SLES server, the changes are lost.

D Caching Computed Facts in a Grid with Large Numbers of Resources

If your Orchestration grid includes a large number of resources with associated Computed Facts, it is likely that these computed facts are evaluated with each Ready for Work message received by the broker from the Orchestration Agent. These evaluations can cause an excessive load on the Orchestration Server, causing a decrease in performance. You might see warnings in the server log similar to the following:

```
07.07 18:27:54: Broker,WARNING: ----- Long scheduler cycle time detected -----
07.07 18:27:54: Broker,WARNING: Total:3204ms, JDL thrds:8, TooLong:false
07.07 18:27:54: Broker,WARNING: Allocate:0ms [P1:0,P2:0,P3:0], Big:488
07.07 18:27:54: Broker,WARNING: Provision:4ms [P1:0,P2:0,P3:0], Big:253
07.07 18:27:54: Broker,WARNING: Msgs:3204ms [50 msg, max: 3056ms (3:RFW)]
07.07 18:27:54: Broker,WARNING: Workflow:[Timeout:0ms, Stop:0ms]
07.07 18:27:54: Broker,WARNING: Line:0ms, Preemption:0ms, (Big: 3), Mem:0ms
07.07 18:27:54: Broker,WARNING: Jobs:15/0/16, Contracts:10, AvailNodes:628
07.07 18:27:54: Broker,WARNING: PermGen: Usage [214Mb] Max [2048Mb] Peak
[543Mb]
07.07 18:27:54: Broker,WARNING: Memory: free [1555Mb]  max [3640Mb]
07.07 18:27:54: Broker,WARNING: Msgs:483/50000 (recv:128692,sent:14202),
More:true
07.07 18:27:54: Broker,WARNING: ---------------------------------------------
```

To work around this issue, we recommend that you cache the Computed Facts.

1. In the Explorer tree of the Orchestration Console, expand the Computed Facts object, then select *vmbuilderPXEBoot*.

   The `vmbuilderPXEBoot` fact does not change, so setting the cache here is safe from any automatic modifications.

2. In the Computed Facts admin view, select the *Attributes* tab to open the Attributes page.
3. In the Attributes page, select the *Cache Result for* check box, then in the newly active field, enter 10 minutes (remember to change the drop-down list to indicate *Minutes*).

   This value must be greater than the default of 30 seconds.

4. Click the *Save* icon to save the new configuration.

NOTE: If necessary, you can also cache other computed facts to improve server performance.
