JBoss Operations Network 2.4 Beta

Basic Admin Guide

for using the GUI to manage JBoss Operations Network

Ella Deon Lackey

Lee Carlon

Publication date: June 2, 2010 (Beta)

JBoss Operations Network 2.4 Beta Basic Admin Guide
for using the GUI to manage JBoss Operations Network
Edition 0

Author	Ella Deon Lackey
Author	Lee Carlon

This manual covers management tasks that are performed through the JBoss ON web UI. This includes managing inventory, configuring and provisioning resources, setting and acknowledging alerts, and monitoring resources. This also covers JBoss ON configuration like creating users and managing servers and agents.

Preface

JBoss Operations Network 2.4 provides an integrated solution for managing JBoss middleware, other network infrastructure, and applications built on Red Hat Enterprise Application Platform (EAP).

This manual covers all management tasks for JBoss ON, from configuring the server and agents to managing resources (servers and applications managed by JBoss ON). This *Basic Administration Guide* is intended for JBoss ON administrators and IT staff.

1. JBoss ON Overview

JBoss Operations Network has four major components, which work together to create the management platform:

- The JBoss ON servers, which centralize configuration and connect all other components
- A SQL database (PostgreSQL or Oracle) which stores both JBoss ON configuration settings and resource-related data, from packages to the resource inventory
- Local agents installed on managed platforms, which connect with servers to receive resource configuration updates and which collect and send monitoring data
- The JBoss ON GUI, which is a web-based interface that allows users to connect to any JBoss ON server, from any location, to view resource data and perform management tasks

These components each cover a functional area. All together, they make up JBoss ON.

A *resource* is any platform, server (application), or service that is monitored and managed by JBoss ON. The basic resource inventory is covered in *Chapter 2, Importing and Organizing Resources*, and all of the tasks in JBoss ON are related to managing those resources. Some of the key tasks are:

- Editing configuration files for resources (*Chapter 4, Managing Resource Configuration*)
- Updating packages or deploying new packages on resources (*Chapter 3, Setting up Content for Resources*)
- Monitoring resources (*Chapter 8, Monitoring Resources*) and sending alert notifications when events occur (*Chapter 9, Configuring and Managing Alerts*)
- Running operations or commands on resources remotely (*Chapter 7, Managing Operations*)

JBoss ON features and concepts are covered in more detail in the *Planning and Features Guide*.

JBoss ON itself has two key configuration areas that affect its performance:

- High availability for JBoss ON servers, which provides failover tolerance and improved resource maintenance performance (*Chapter 11, General Management*)
- Configuring roles, which assigns access permissions to both users and resources (*Chapter 10, Managing Users and Roles*)

2. Examples and Formatting

Each of the examples used in this guide, such as file locations and commands, have certain defined conventions.

2.1. Command and File Examples

All of the examples for JBoss ON commands, file locations, and other usage are given for Red Hat Enterprise Linux systems. Be certain to use the appropriate commands and files for your platform.

To start the JBoss ON server:

```
serverRoot/jon-server-2.4.0.Beta1/bin/rhq-server.sh start
```

Example 1. Example Command

2.2. Text Formatting and Styles

Certain words are represented in different fonts, styles, and weights. Different character formatting is used to indicate the function or purpose of the phrase being highlighted.

Formatting Style	Purpose
`Monospace font`	Monospace is used for commands, package names, files and directory paths, and any text displayed in a prompt.
`Monospace with a background`	This type of formatting is used for anything entered or returned in a command prompt.
Italicized text	Any text which is italicized is a variable, such as *instance_name* or *hostname*. Occasionally, this is also used to emphasize a new term or other phrase.
Bolded text	Most phrases which are in bold are application names, such as **Cygwin**, or are fields or options in a user interface, such as a **User Name Here:** field or **Save** button.

Other formatting styles draw attention to important text.

NOTE or TIP

A note provides additional information that can help illustrate the behavior of the system or provide more detail for a specific issue. Tips provide pointers to helpful information or to easy ways to accomplish something.

IMPORTANT

Important information is necessary, but possibly unexpected, such as a configuration change that will not persist after a reboot.

WARNING

A warning indicates potential data loss, as may happen when tuning hardware for maximum performance.

3. Additional Reading

JBoss ON has several guides and documents available that help people plan, install, and deploy JBoss ON.

Guide	
Release Notes	Contains important information on new features, fixed bugs, known issues and workarounds, and other important deployment information for this specific version of JBoss Operations Network.
Planning and Features Guide	Contains information on all the features and concepts in JBoss ON, with tips and outlines to help plan how to deploy JBoss ON.
Installation Guide	Contains complete information on how to set up databases, install JBoss ON servers and agents, and migrate existing deployments. This also contains platform support information.
Basic Admin Guide	Covers all management tasks for JBoss ON, both for managing resources and configuring the JBoss ON server and agents.
Plug-in Writing Guide	Contains procedures, references, and examples to write custom plug-ins for the server and for the agent.
Manging JBoss ON from the Command Line	Describes how to use the JBoss ON Java CLI to manage a server remotely, without the UI.
Resource Monitoring and Operations Guide	Contains a complete reference of each resource type in JBoss ON for monitoring metrics, available operations, and other configuration options.

For the latest information about JBoss ON, including current release notes, complete product documentation, technical notes, and deployment information, see the JBoss Operations Network documentation site at *http://www.redhat.com/docs/en-US/JBoss_ON/*.

4. Giving Feedback

If there is any error in this *Basic Admin Guide* or there is any way to improve the documentation, please let us know. Bugs can be filed against the documentation for the community-based RHQ Project in Bugzilla, *http://bugzilla.redhat.com/bugzilla*. Make the bug report as specific as possible, so we can be more effective in correcting any issues:

1. Select the **Other** products group.
2. Select **RHQ Project** from the list.
3. Set the component to **Documentation**.
4. Set the version number to 2.4.
5. For errors, give the page number (for the PDF) or URL (for the HTML), and give a succinct description of the problem, such as incorrect procedure or typo.

For enhancements, put in what information needs to be added and why.

6. Give a clear title for the bug. For example, **`"Incorrect command example for setup script options"`** is better than **`"Bad example"`**.

We appreciate receiving any feedback — requests for new sections, corrections, improvements, enhancements, even new ways of delivering the documentation or new styles of docs. You are welcome to contact Red Hat Content Services directly at *docs@redhat.com*[1].

5. Document History

Revision 0	**June 2, 2010**	**Ella Deon Lackey** ***`dlackey@redhat.com`***

Beta release of JBoss Operations Network.

[1] mailto:docs@redhat.com

A Tour of the JBoss ON Web Interface

JBoss Operations Network has a rich, layered UI which covers a broad range of functionality. This chapters gives a brief summary of the major sections of the UI so that users can more effectively perform management tasks.

1.1. Supported Browsers

The JBoss ON administrative GUI is accessed through a web browser. JBoss ON has been tested on these browsers:

- Firefox 2.x and 3.x
- Microsoft Internet Explorer 7 and 8
- Microsoft Internet Explorer 6 (deprecated)

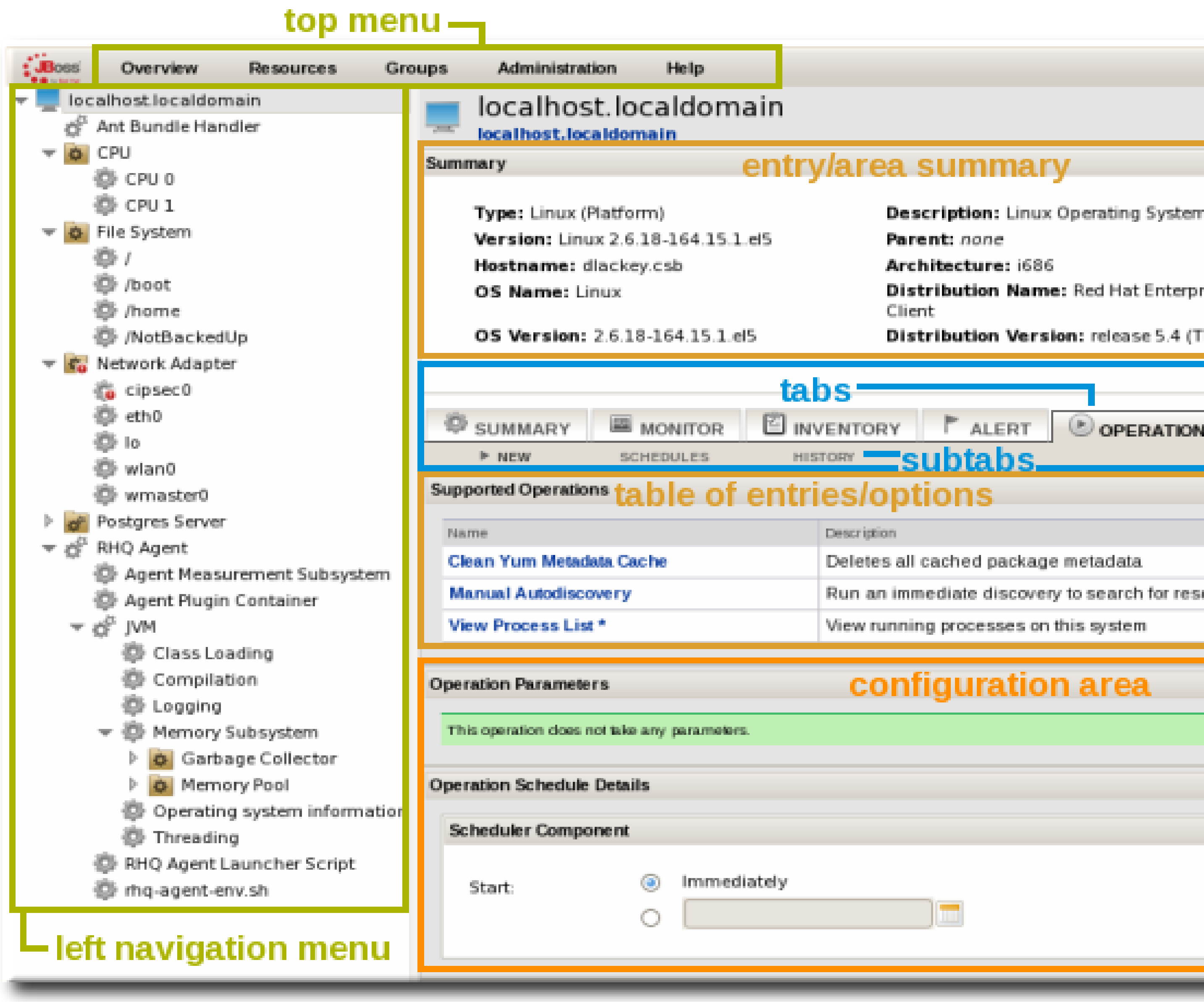
1.2. A High Level Walk-through

The JBoss ON UI is a very dense, layered interface. Most of the time, functionality is stacked on pages, so there are multiple areas that can be accessed or edited from any given page. To maximize its use of space, JBoss ON uses top navigation menus, tabbed browsing with subtabs, active links, and navigation trees.

There is a lot going on in the JBoss ON UI. Still, there are basically four main categories of UI locations in JBoss ON:

- The top menu
- The dashboard
- Inventory tables, which can be for the resource inventory, a summary report, or the results of a search
- Entry pages for any configuration entry in JBoss ON, including resources, groups, plug-ins, and JBoss ON server settings

All of these elements fit together in a repeated and reliable pattern.



Type: Linux (Platform)
Version: Linux 2.6.18-164.15.1.el5
Hostname: dlackey.csb
OS Name: Linux
OS Version: 2.6.18-164.15.1.el5
Description: Linux Operating System
Parent: *none*
Architecture: i686
Distribution Name: Red Hat Enterpr
Client
Distribution Version: release 5.4 (T

Name	Description
Clean Yum Metadata Cache	Deletes all cached package metadata
Manual Autodiscovery	Run an immediate discovery to search for res
View Process List *	View running processes on this system

This operation does not take any parameters.

Start: [✓ Immediately] ○

Figure 1.1. UI Elements All Together

Understanding the type of page that you're on can make it easier to navigate through the JBoss ON UI and can help you more completely understand what you can accomplish in JBoss ON.

1.2.1. The Top Menu

At the very top of the JBoss ON UI is a menu bar with four options that relate to the functionality of JBoss ON: **Overview**, **Resources**, **Groups**, and **Administration**.

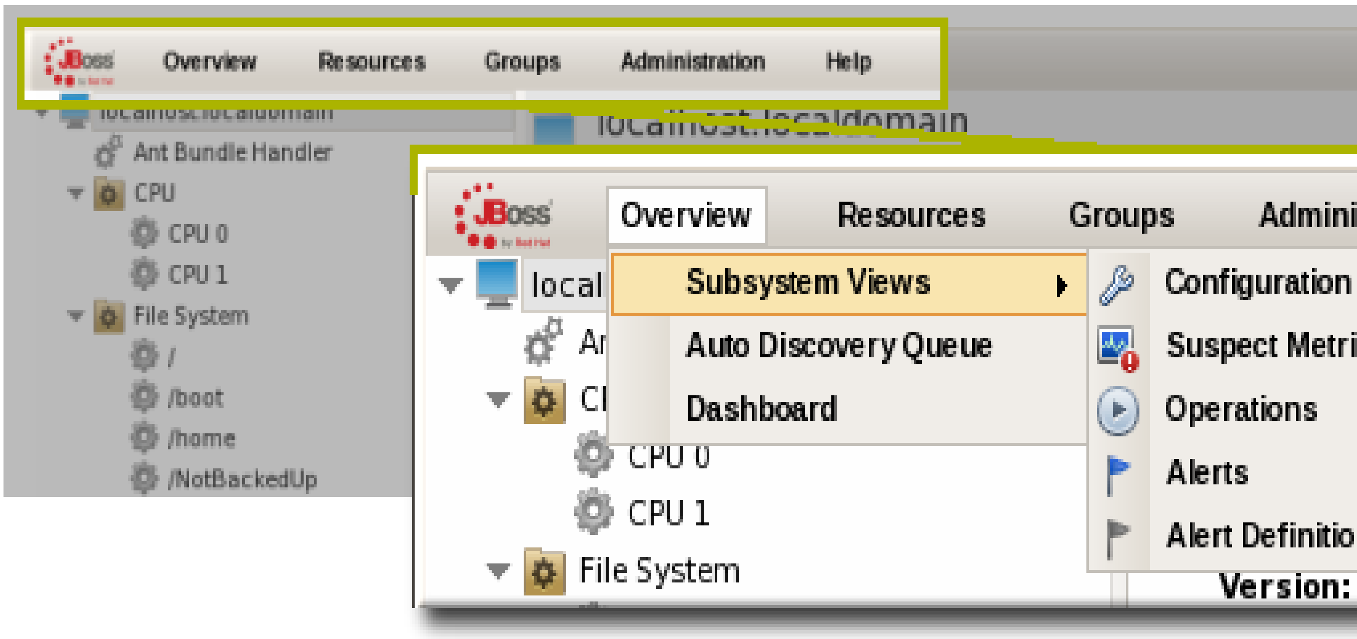


Figure 1.2. The Top Menu

Each menu item relates to a different functional aspect of JBoss ON.

- **Overview** links to the Dashboard, summaries of snapshot events with resource (like new alerts or unavailable resources), and the discovery queue. This is a collection of quick, brief, and global information for JBoss ON resources.
- **Resources** and **Groups** go to manage the inventory. **Resources** goes to the inventory and individual resource pages, while the **Groups** page goes to configured groups for resources.
- **Administration** goes to all areas related to configuring the JBoss ON server, instead of pages related to the resource inventory. This includes server settings, plug-ins, users and security, and agent settings.

The other elements in the top menu allow you to open a help ticket, set a refresh rate (*Section 1.6, “Refreshing the UI”*), or edit the information for the logged in user (*Section 10.2.2.1, “Editing Your Personal Entry”*).

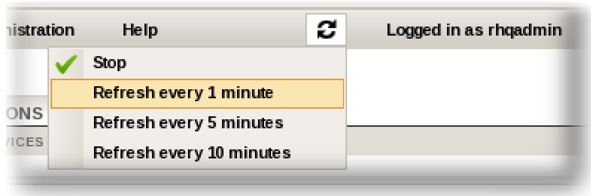
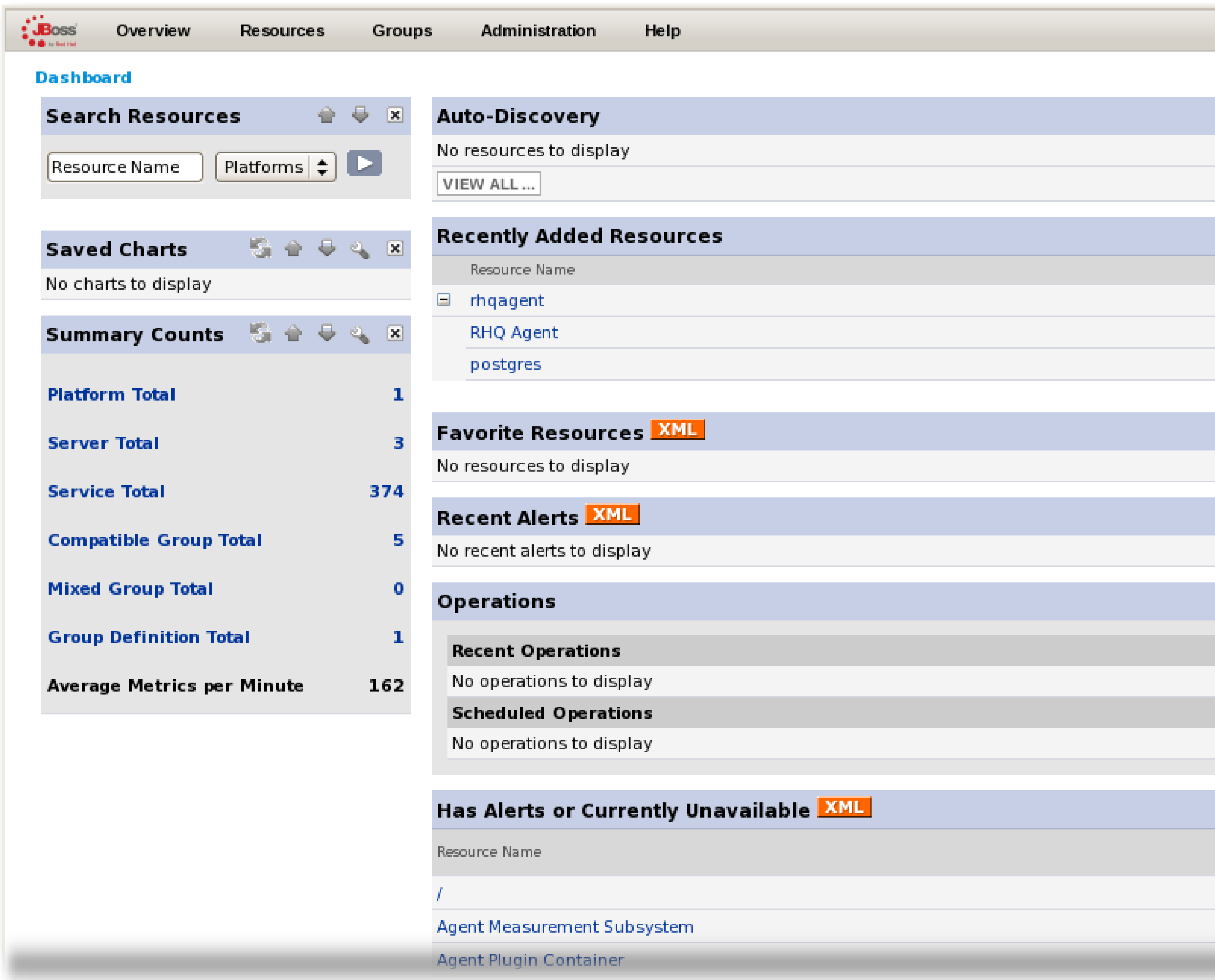


Figure 1.3. Other Top Menu Options

1.2.2. Dashboard

The Dashboard is an overview of everything in JBoss ON, from recent actions (fired alerts and operations) to availability reports to newly discovered and imported resources. The Dashboard is the landing page for JBoss ON, the first page that comes up after login.



Dashboard

Search Resources

Resource Name | Platforms

Saved Charts

No charts to display

Summary Counts

Platform Total	1
Server Total	3
Service Total	374
Compatible Group Total	5
Mixed Group Total	0
Group Definition Total	1
Average Metrics per Minute	162

Auto-Discovery

No resources to display

VIEW ALL ...

Recently Added Resources

Resource Name

rhqagent

RHQ Agent

postgres

Favorite Resources XML

No resources to display

Recent Alerts XML

No recent alerts to display

Operations

Recent Operations

No operations to display

Scheduled Operations

No operations to display

Has Alerts or Currently Unavailable XML

Resource Name

/

Agent Measurement Subsystem

Agent Plugin Container

Figure 1.4. Dashboard View

The Dashboard is the only page in JBoss ON that is configurable. It is composed of individual portlets. These portlets can be rearranged or independently refreshed through the icon menu displayed on each portlet. To move portlets, use the arrows; to get rid of the portlet, click the X icon. Some types of portlet allow customization, which can be accessed by clicking the wrench icon.

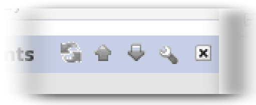


Figure 1.5. Portlet Icons

If a portlet is removed, it is automatically added to the **Select Portlet** drop-down menu. This lets portlets be added back easily.

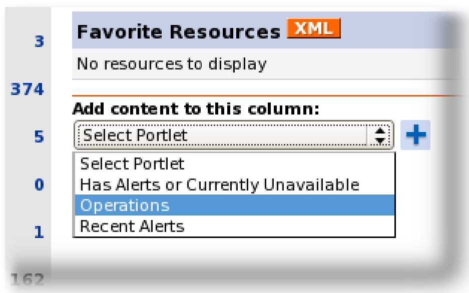


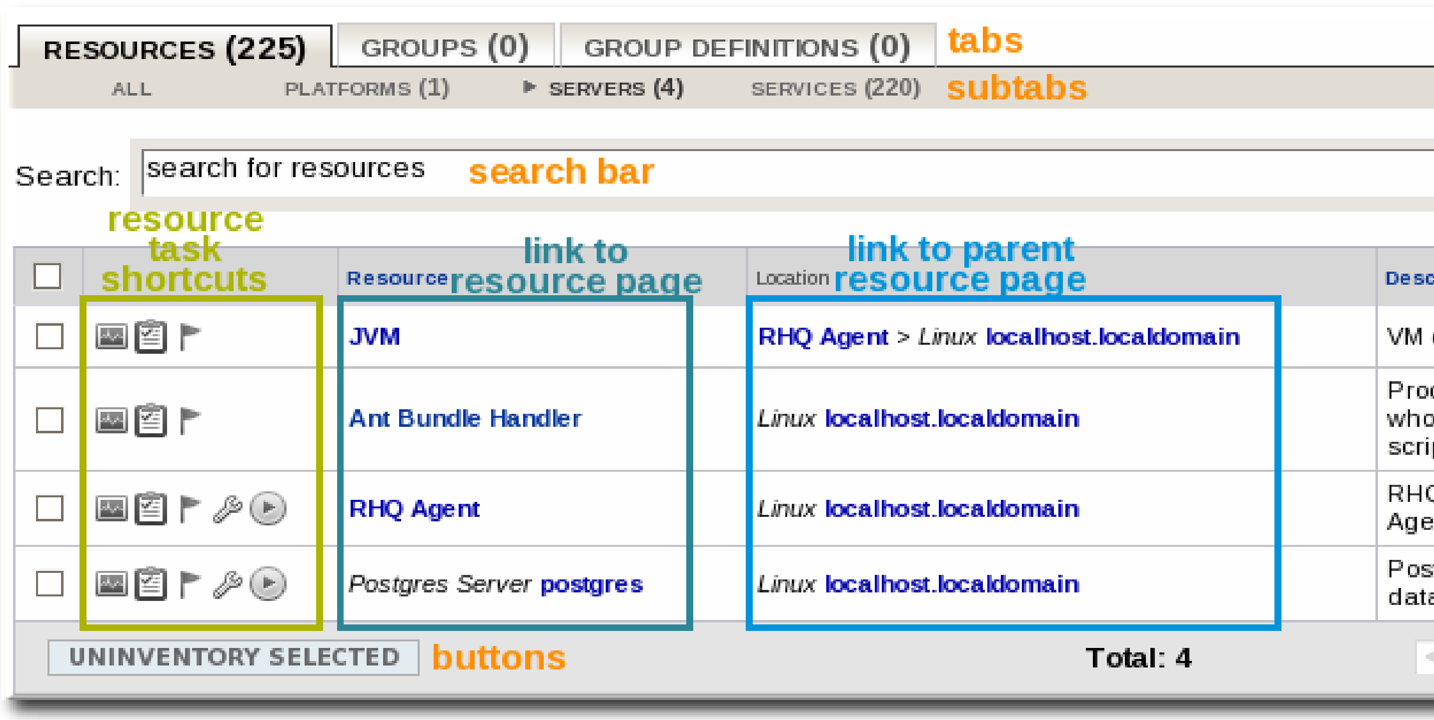
Figure 1.6. Adding a Portlet

1.2.3. Inventory Browsers and Summaries

Some pages are essentially long tables of information. All of the information in the **Overview > Subsystem Views**, **Resources**, and **Groups** areas are presented in basically the same way:

- A search form
- Tabs for different areas, with subtabs that further break down information
- A table of results
- Icons that open a configuration or task option for that specific entity
- Buttons that perform actions (create, delete, or some other specific action) on the entries; some of these buttons aren't active unless an entry is selected

Although the inventory in *Figure 1.7, "Inventory Browser"* shows only three resources, the interface itself is still rich with functionality. The search bar for resources and groups uses a specialized syntax and flexible dynamic search. The icons by each resource open up (depending on the resource type) the management tab for monitoring, configuration, alerts, content, inventory, and operations in that entry's details page. Clicking the name of the entry itself opens its default entry configuration page, while clicking the name of its parent opens up that parent resource's configuration page.



		Resource	Location	Desc
		JVM	RHQ Agent > *Linux* localhost.localdomain	VM
		Ant Bundle Handler	*Linux* localhost.localdomain	Pro who scri
		RHQ Agent	*Linux* localhost.localdomain	RHQ Age
		Postgres Server postgres	*Linux* localhost.localdomain	Pos data

Figure 1.7. Inventory Browser

In *Figure 1.7, “Inventory Browser”*, the **UNINVENTORY SELECTED** button is blue because no entries are selected, so the button is inactive. White buttons are active.

1.2.4. Entry Details Pages

Possibly the most functionality-saturated area in JBoss ON is an entry's details page.

The left navigation area shows the hierarchy, both parents and children, of the selected resource. This makes it very easy to navigate among all of the different services and servers that affect a resource.

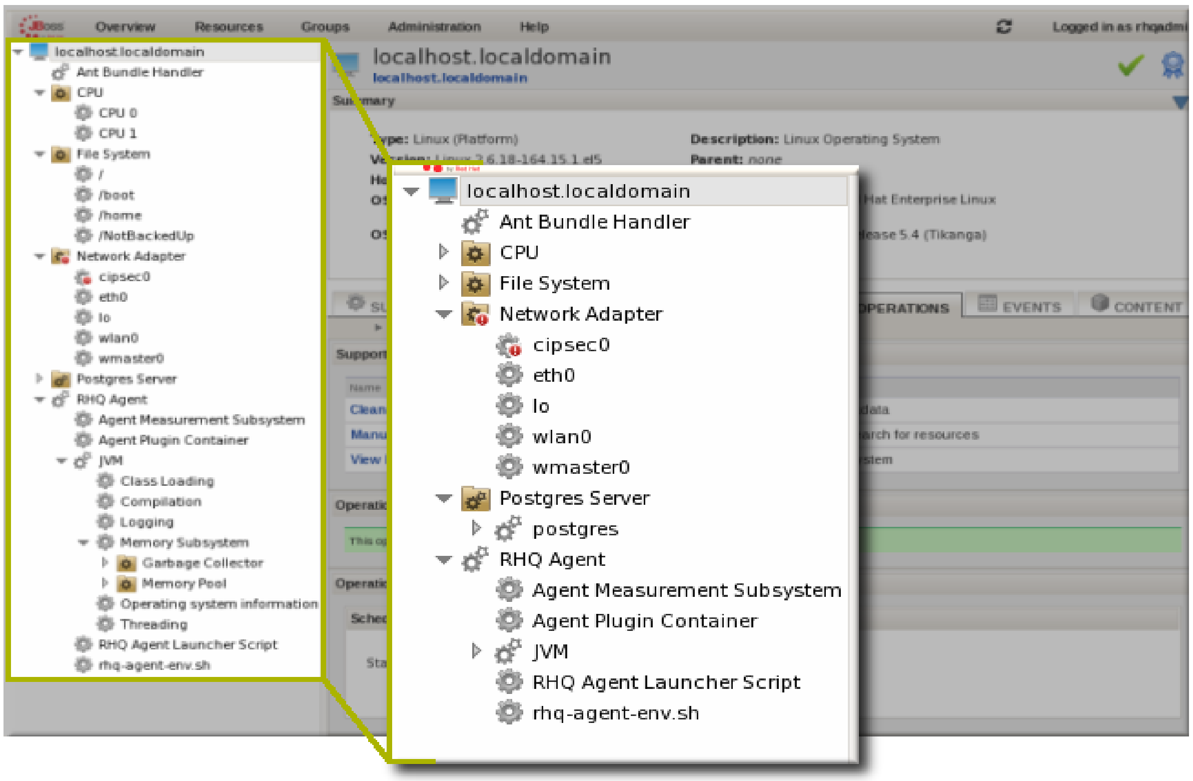


Figure 1.8. A Platform Resource Details Page

Right-clicking any of the resources in the left navigation opens shortcuts to that entry's configuration.

NOTE

Resources have short names that are automatically assigned based on their type, instance or system name, or IP address. These names are used in the inventory and in the tree navigation.

The configuration area of a resource entry page (and other JBoss ON entities, like plug-ins and templates) has three information layers that provide all of the possible functionality and tasks available for that entry.

The entry's configuration page is tabbed according to each area that can be configured, and frequently has subtabs for additional configuration options and to show the history of that area (like fired alerts, previous content updates, or monitoring data).

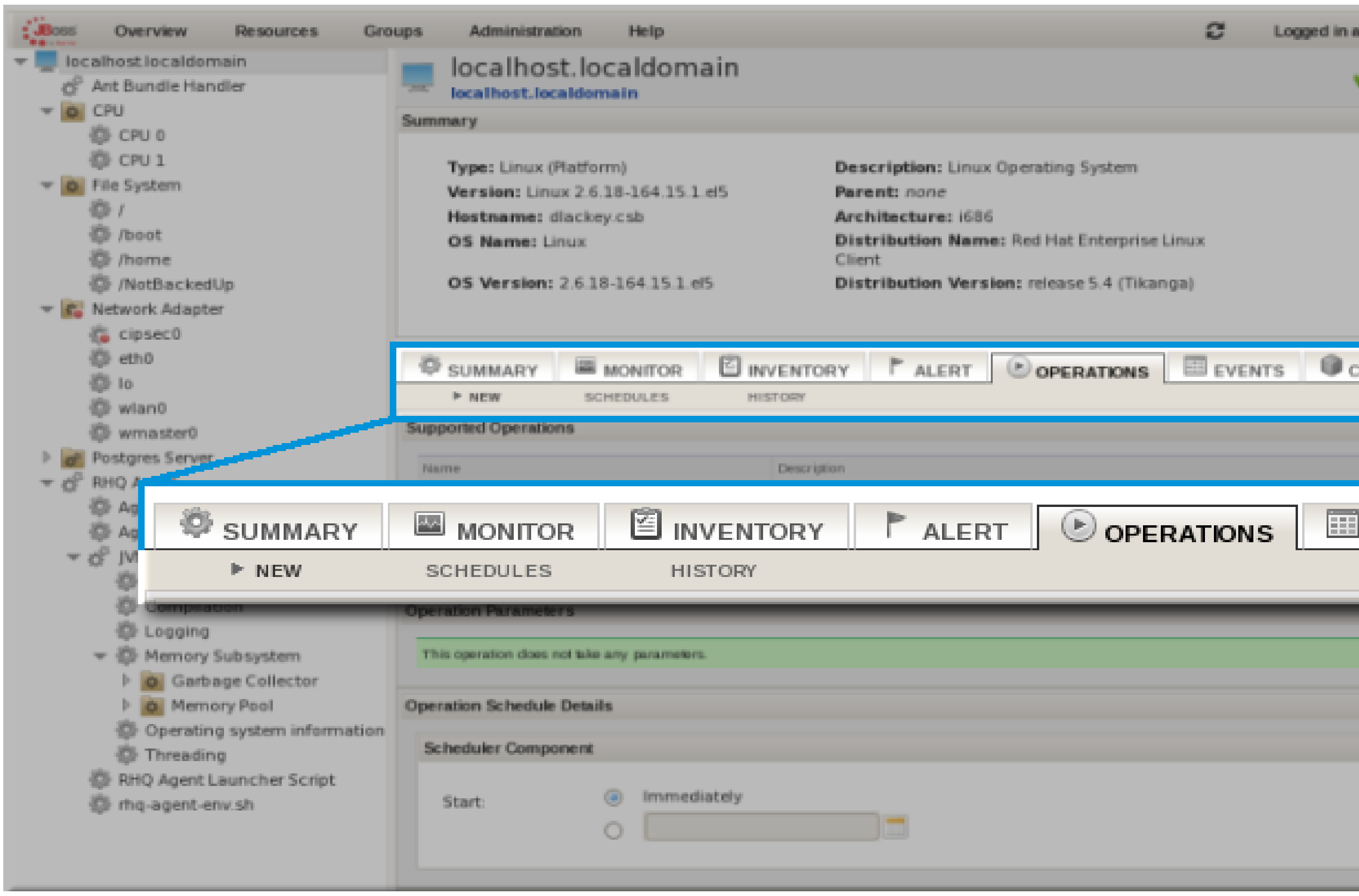
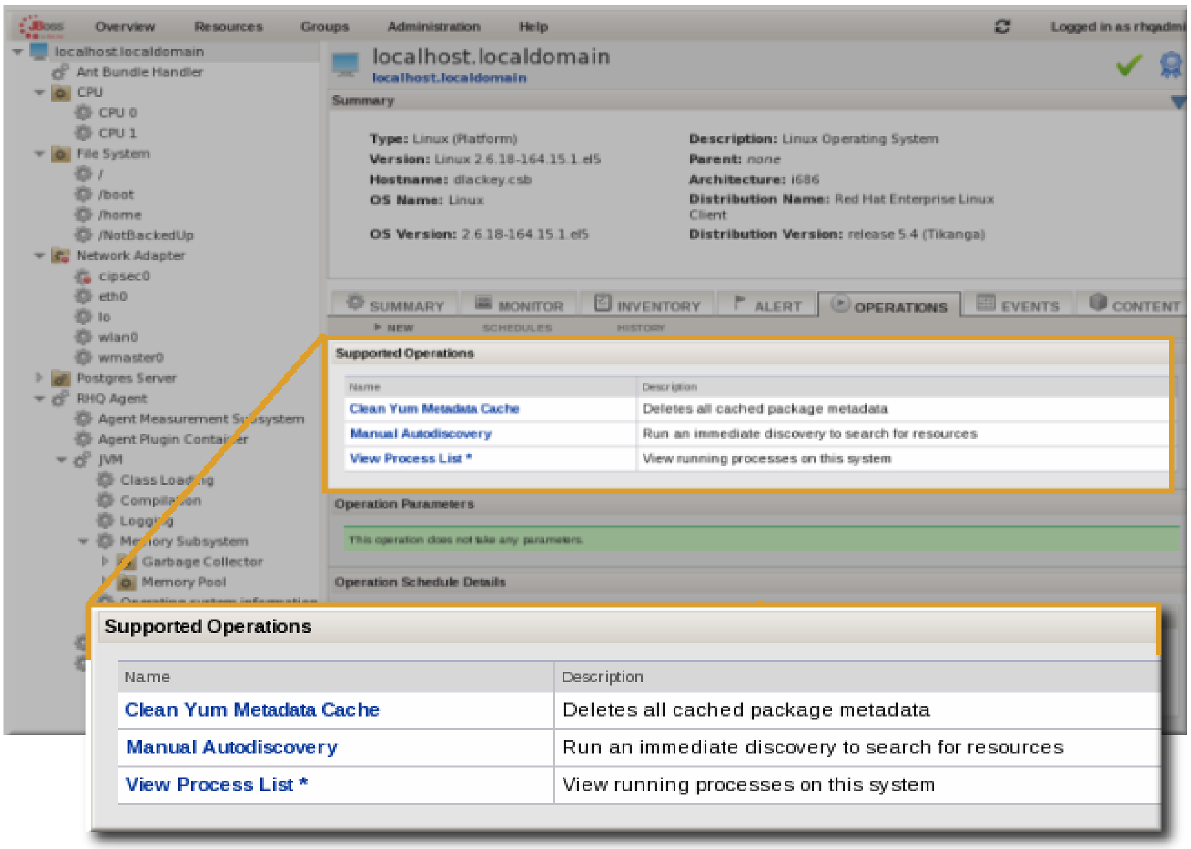


Figure 1.9. Tabs for a Resource Entry

The next section in the entry area shows lists of related configured entries for that task. For example, an **Operations** area will have a list of available operations in a table below the tabs. For **Inventory**, there is a list of configured child resources, while **Alerts** shows all of the configured alerts for that resource. All of those entries are listed in a table similar to the search results available in other parts of the JBoss ON UI.



localhost.localdomain
localhost.localdomain

Summary

Type: Linux (Platform)
Version: Linux 2.6.18-164.15.1.el5
Hostname: dlackey.csb
OS Name: Linux
OS Version: 2.6.18-164.15.1.el5

Description: Linux Operating System
Parent: *none*
Architecture: i686
Distribution Name: Red Hat Enterprise Linux Client
Distribution Version: release 5.4 (Tikanga)

Supported Operations

Name	Description
Clean Yum Metadata Cache	Deletes all cached package metadata
Manual Autodiscovery	Run an immediate discovery to search for resources
View Process List *	View running processes on this system

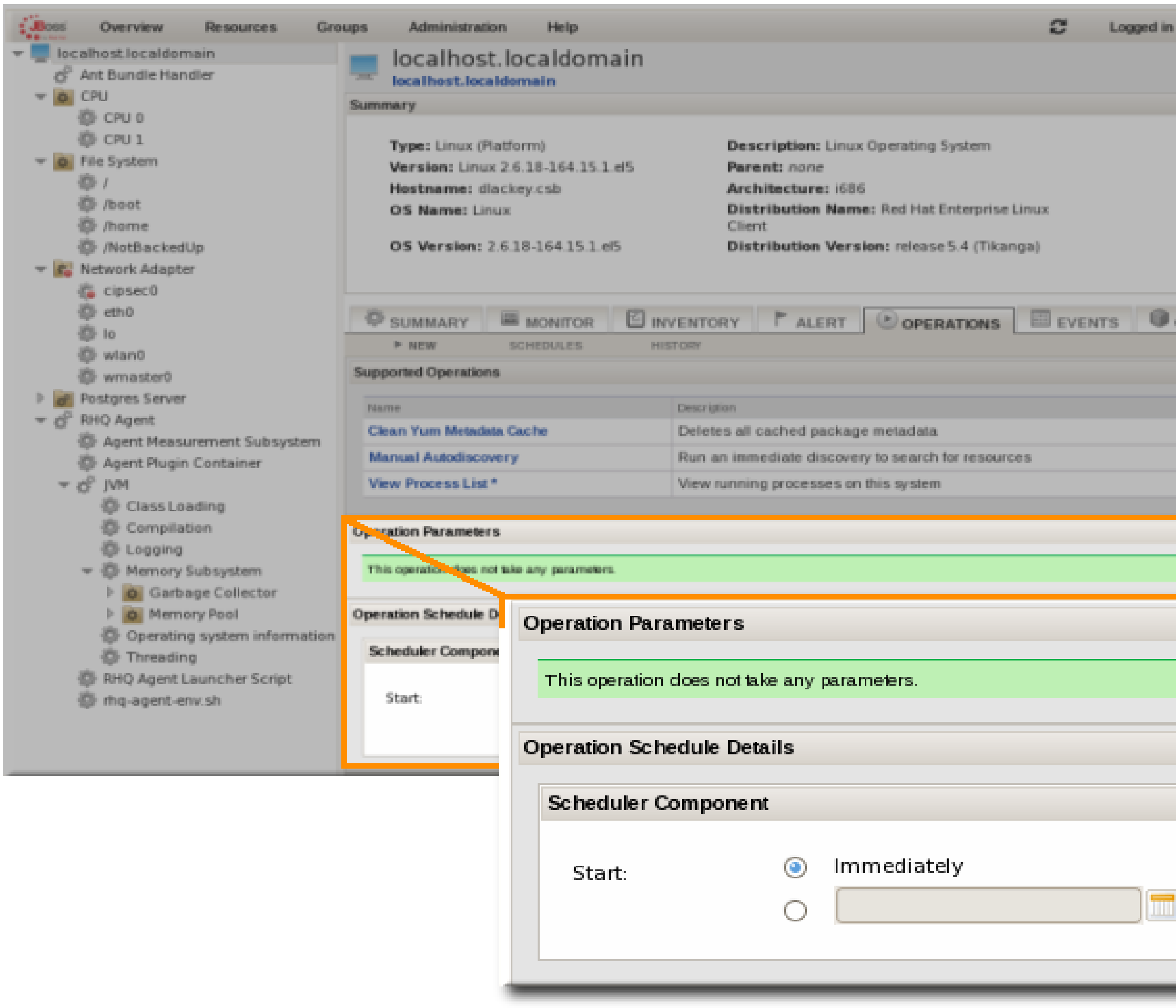
Operation Parameters

This operation does not take any parameters.

Operation Schedule Details

Figure 1.10. Configuration Entry Table

For entries that relate to resource configuration — like content, operations, and alerts — clicking on an item in the entries table in *Figure 1.10, “Configuration Entry Table”* will open an editable form below the entries table. This allows you to add a new configuration, edit existing configuration, view history, and perform other management tasks.



localhost.localdomain
- Ant Bundle Handler
- CPU
  - CPU 0
  - CPU 1
- File System
  - /
  - /boot
  - /home
  - /NotBackedUp
- Network Adapter
  - cipsec0
  - eth0
  - lo
  - wlan0
  - wmaster0
- Postgres Server
- RHQ Agent
  - Agent Measurement Subsystem
  - Agent Plugin Container
  - JVM
    - Class Loading
    - Compilation
    - Logging
    - Memory Subsystem
      - Garbage Collector
      - Memory Pool
    - Operating system information
    - Threading
  - RHQ Agent Launcher Script
  - rhq-agent-env.sh

localhost.localdomain
localhost.localdomain

Summary

Type: Linux (Platform)
Version: Linux 2.6.18-164.15.1.el5
Hostname: dlackey.csb
OS Name: Linux
OS Version: 2.6.18-164.15.1.el5
Description: Linux Operating System
Parent: none
Architecture: i686
Distribution Name: Red Hat Enterprise Linux Client
Distribution Version: release 5.4 (Tikanga)

Supported Operations

Name	Description
Clean Yum Metadata Cache	Deletes all cached package metadata
Manual Autodiscovery	Run an immediate discovery to search for resources
View Process List *	View running processes on this system

Operation Parameters

This operation does not take any parameters.

Operation Schedule Details

Scheduler Component

Start: [✓ Immediately] ○

Figure 1.11. Editable Areas for a Resource Entry

1.3. Setting Favorites

Setting favorites in the **Resource** top menu makes it easy to navigate to resources that administrators need to access routinely for configuration updates, monitoring, or alerting.

Each resource has a small ribbon icon in the upper right corner of its details page. Clicking that icon automatically adds it to the resource favorites list.

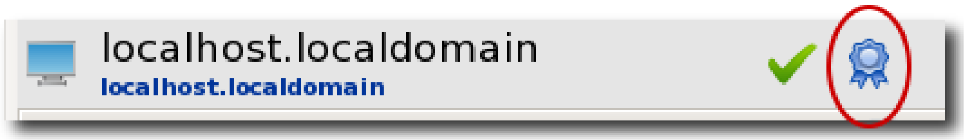


Figure 1.12. Favorites Icon

The resource favorites list is a menu item under the **Resources** tab in the main menu. Click a resource on that list automatically opens its details page without having to search for the resource.

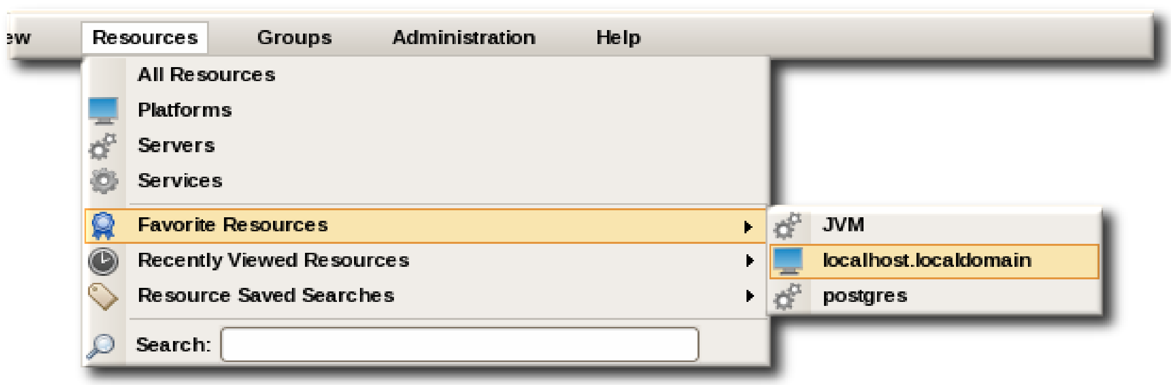
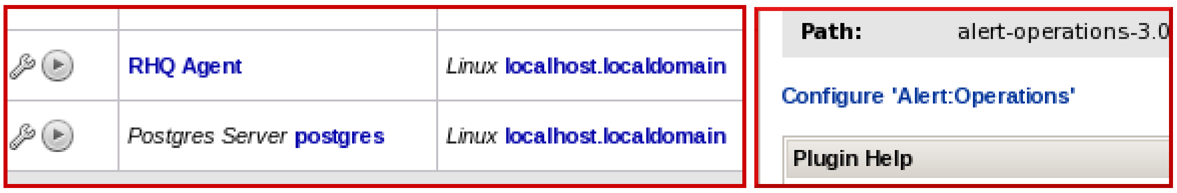


Figure 1.13. Favorites List

1.4. Editing and Deleting Entries

Both resource and JBoss ON configuration entries are edited in the same way. The name of the entry is an active link to that entity's edit page. Sometimes, the link is embedded in another configuration page (which can be because of a group configuration, resource hierarchy, or because sections of the entity are edited in different forms).



RHQ Agent
Linux localhost.localdomain
Postgres Server postgres
Linux localhost.localdomain
Path: alert-operations-3.0
Configure 'Alert:Operations'
Plugin Help

Figure 1.14. Edit Links

Resource-related entries can be deleted only through the inventory browser or group browser. Most JBoss ON server configuration entries cannot be deleted; only user-supplied elements, like plug-ins, roles, and users, can be deleted. As with resource entries, delete options are available in the browsers or summaries for those configuration areas.

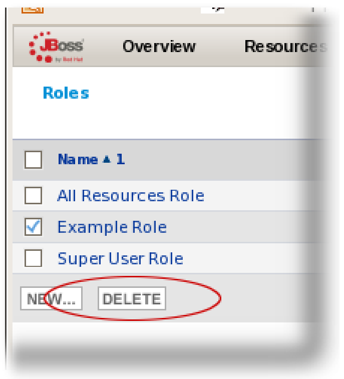


Figure 1.15. Delete Button in the Area Browser

1.5. Dynamic Searches for Resources and Groups

The dynamic inventory search for resources and groups provides an additional tool that can help manage your JBoss ON resources. Along with getting more accurate results based on more flexible or more precise search strings, dynamic searches in JBoss ON can saved to provide fast and reproducible snapshots of your JBoss ON deployment that match criteria that are relevant to your infrastructure.

1.5.1. Comparing Subsystem Search, Menu Search, and Dynamic Inventory Search

There are three areas in the JBoss ON UI that have some sort of search form: the subsystem views in the **Overview** menu, the quick search in the **Resources** and **Groups** menus, and the dynamic search bar in the **Resources** and **Groups** main pages.

The most basic searches are the quick searches in the **Resources** and **Groups** menus, which just make simple string matches.

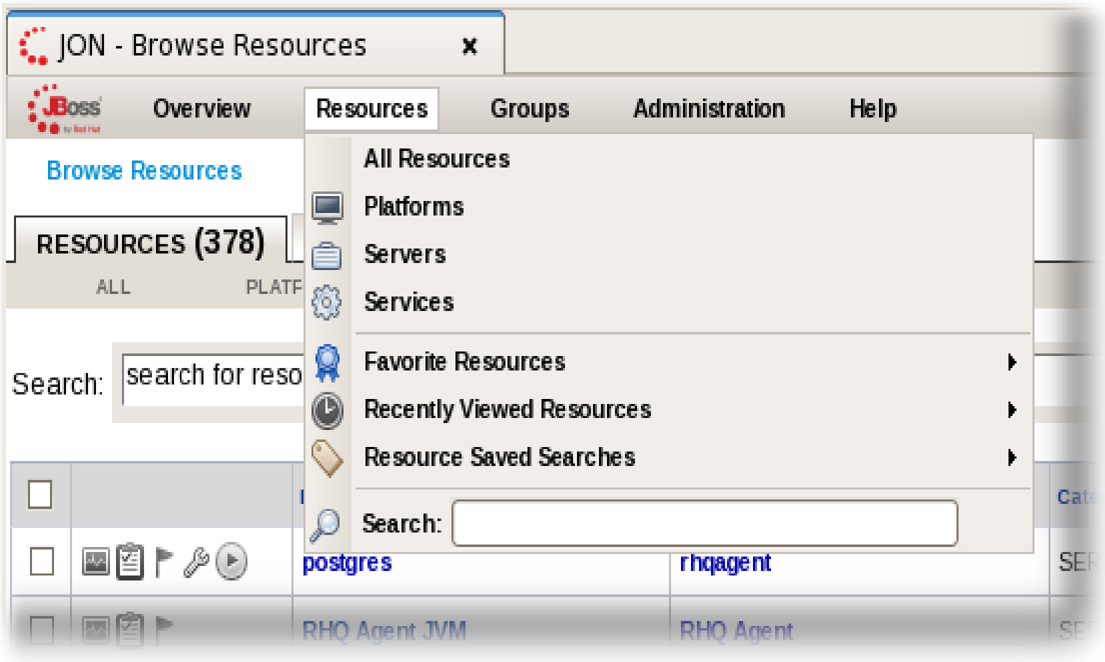
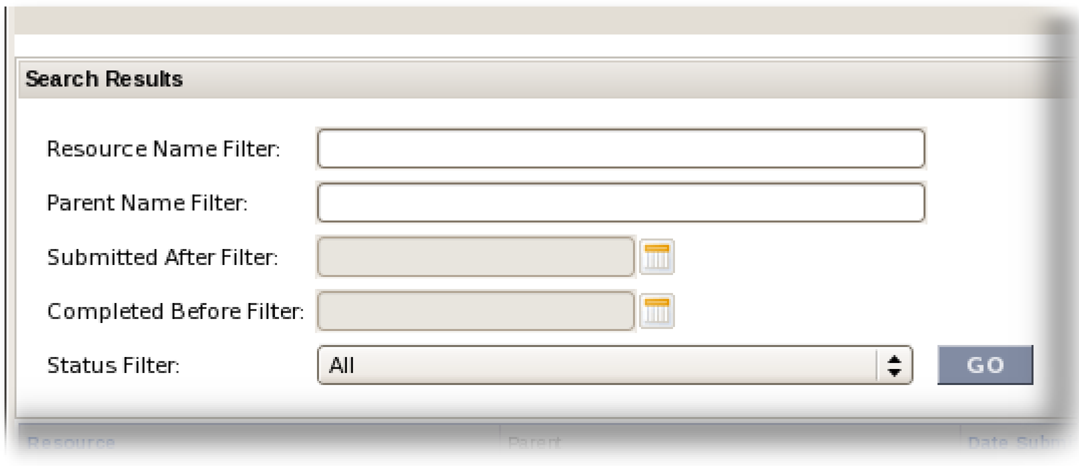


Figure 1.16. Quick Search Field

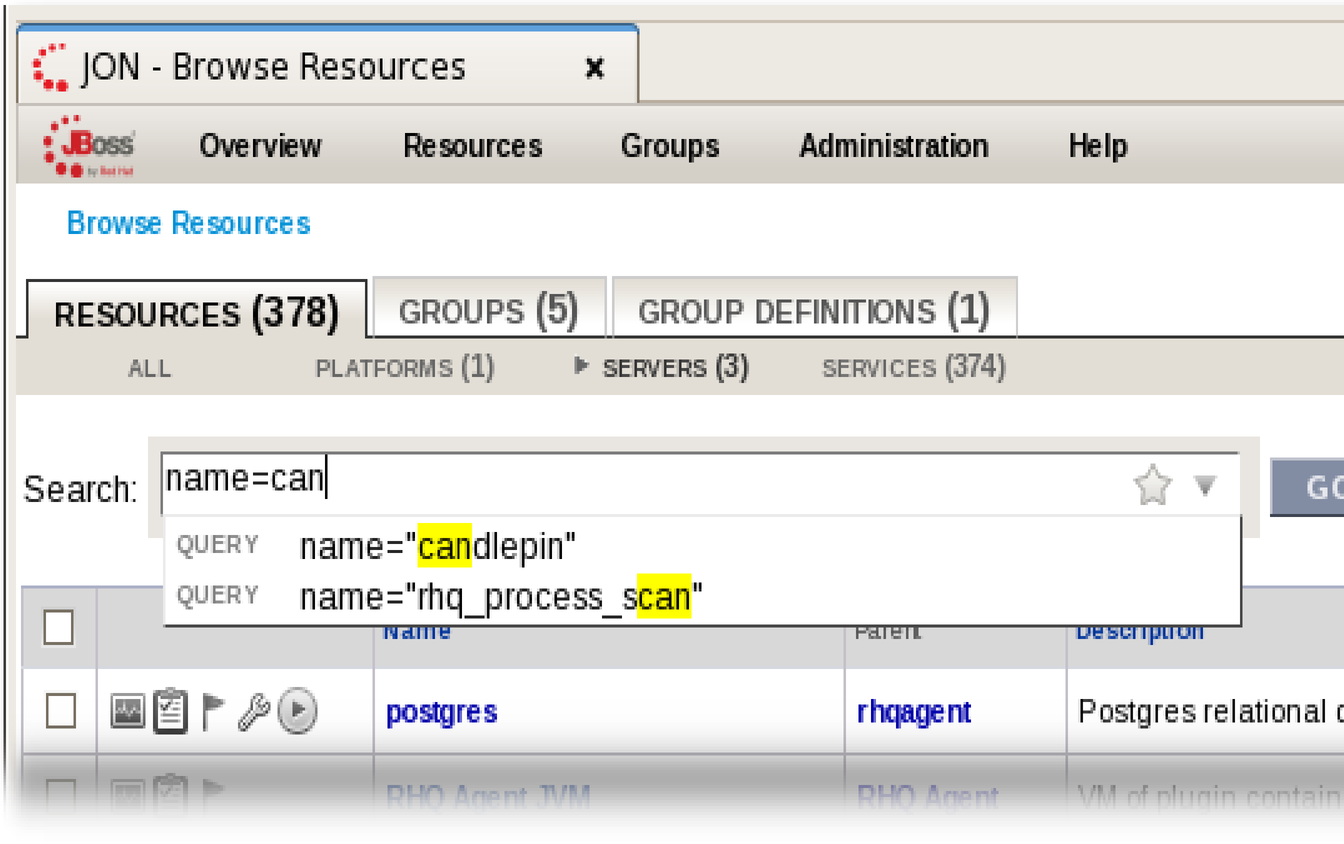
A slightly more complex search is in the subsystem views. Some basic information is used to filter results (the resource name, parent, or dates of changes) and then results are further filtered according to certain criteria that relate to the subsystem view. For example, the operations view offers only operation-related criteria, while the alert view has alert-related criteria. Any or all field options can be set, which offers the ability to be very precise or very general, depending on your need.



Resource Name Filter:	
Parent Name Filter:	
Submitted After Filter:	
Completed Before Filter:	
Status Filter:	All

Figure 1.17. Subsystem Views Search Form

A dynamic search checks both resources and groups (recursively into group members, as well) much more effectively than either a subsystem views search or a quick search. A search can begin against a specific identifying attribute of a resource (such as its name, parent, type, or JBoss ON category) and then has rules that can set how the search handles the string. Multiple search parameters can be strung together to make precise and complex searches. Dynamic searches can be saved and reused later so their results are reliably reproducible. (*Section 1.5.2, "About the Dynamic Search Syntax"* covers the details more.)



Search: name=can

QUERY name="candlepin"
QUERY name="rhq_process_scan"

Name	Parent	Description
postgres	rhqagent	Postgres relational
RHQ Agent JVM	RHQ Agent	VM of plugin contain

Figure 1.18. Dynamic Resource Search

There are other aspects of dynamic searches like the autocomplete, hints, and highlight search strings that make it easier to use effectively than the limited substring and quick searches.

1.5.2. About the Dynamic Search Syntax

JBoss ON has its own search syntax for dynamic searches. The syntax is supposed to be relatively simple while covering a wide array of searchable items and allowing different phrases to be coupled together.

The basic structure of the search phrase is:

```
[search_area].[search_property] operator value operator additional_search
```

The *search_area* identifies what type of entry — resource or group — is being searched for. This is an optional value because the search area is implied by the location of the search; i.e., searching in the **Resources** area implies a resource search, so it's not necessary to include the **`resource.`** part of the search.

The search can be narrowed by looking for a specific value or type of attribute in the entry by using a search property. For example, looking for a resource with a CPU usage of 80% (**`trait`**) is different than looking for an entry with an ID that includes 80 (**`id`**). The available properties are listed in *Table 1.1, "Resource Search Contexts"* and *Table 1.2, "Group Search Contexts"*.

TIP

It's possible to search using group criteria in the resource search, and the reverse, by specifying the search area and the appropriate properties. For example, it's possible to do a search in the groups area to return the list of groups that a specific resource belongs to. This is done by explicitly passing the search context and search property. For example, in the **Groups** page, to list any group which contains a resource managed by the Postgres plug-in:

```
resource.type.plugin = Postgres
```

Property	Description
resource.id	The resource ID number assigned by JBoss ON.
resource.name	The resource name, which is displayed in the UI.
resource.version	The version number of the resource.
resource.type.plugin	The resource type, defined by the plug-in used to manage the resource.
resource.type.name	The resource type, by name.
resource.type.category	The resource type category (platform, server, or service).
resource.availability	The resource availability, either UP or DOWN.
resource.pluginConfiguration[*property-name*]	The value of any possible configuration entry in a plug-in.
resource.resourceConfiguration[*property-name*]	The value of any possible configuration entry in a resource.

Property	Description
resource.trait[*property-name*]	The value of any possible measurement trait for a resource.

Table 1.1. Resource Search Contexts

There are slightly fewer search properties for groups, since groups have simpler entries than resources.

Property	Description
group.name	The name of the group.
group.plug-in	For a compatible group, the plug-in which defines the resource type for this group.
group.type	For a compatible group, the resource type for this group.
group.category	The resource type category (platform, server, or service).
group.kind	The type of group, either mixed or compatible.
group.availability	The availability of resource in the group, either UP or DOWN.

Table 1.2. Group Search Contexts

The *operator* first refers to how the results should match the search string (*value*). This can require an exact match, every value but the one given in the search string. The *operator* then refers to how multiple search strings relate to each other (AND or OR); both explicit AND and OR statements and parenthetical statements are allowed.

Operator	Description
=	Case-insensitive match.
==	Case-exact match.
!=	Case-insensitive negative match (meaning, the value is *not* the string).
!==	Case-exact negative match (meaning, the value is *not* the string).

Table 1.3. Search String Operators

The last part of the syntax is how to match the string itself. With no other characters, the search is treated as a substring search, meaning the given value can be any part or all of the returned value. The search characters in *Table 1.4, "String Operators"* limit the search string by setting where in the result value the string appears.

Operator	Description
string	The string can occur anywhere in the result string.
^*string*	The given string must appear at the beginning of the result value.

Operator	Description
string$	The given string must appear at the end of the result value.
^*string*$	The result must be an exact match of the given string, with no leading or trailing characters.
!==	Case-exact negative match (meaning, the value is *not* the string).

Table 1.4. String Operators

NOTE

The dynamic search syntax treats null as an acceptable value for a search. Running a search which passes null will look for any resource or group with a null value for that property:

```
resource.trait[Database.startTime] = null
```

However, putting quotation marks around the term **`null`** will look for a resource or group with a value of the string null:

```
name = "null"
```

Complex searches can be written by stringing multiple search strings together. JBoss ON dynamic searches accept both explicit AND and OR operators for terms, with AND being given preference. Additionally, search phrases can be nested to multiple levels using parentheses. For example, this expression matches a OR b AND C OR D:

```
(a | b) (c | d)
```

Both OR and the pipe character (|) are allowed for OR operators.

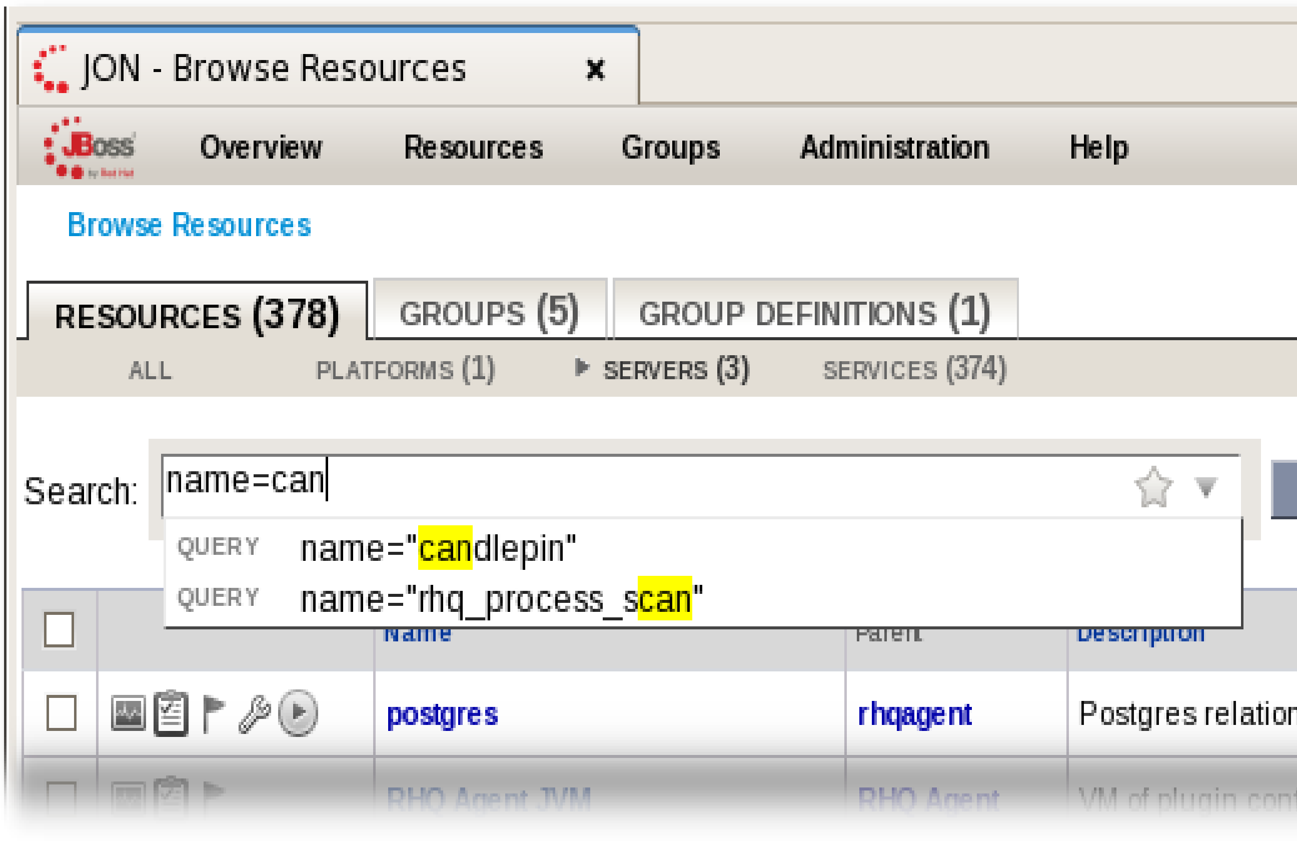
Multiple levels of nesting are allows. For example, this expression requires a AND either b OR c AND D:

```
(a) (b | (c d))
```

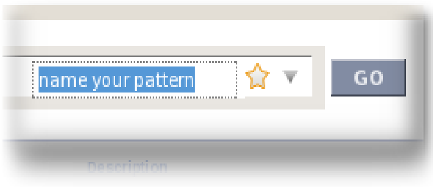
1.5.3. Saving, Reusing, and Deleting Dynamic Searches

Dynamic searches can be saved, which makes it much easier to reuse complex or common searches. To save a search:

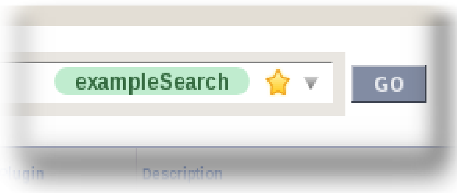
1. Run the search.



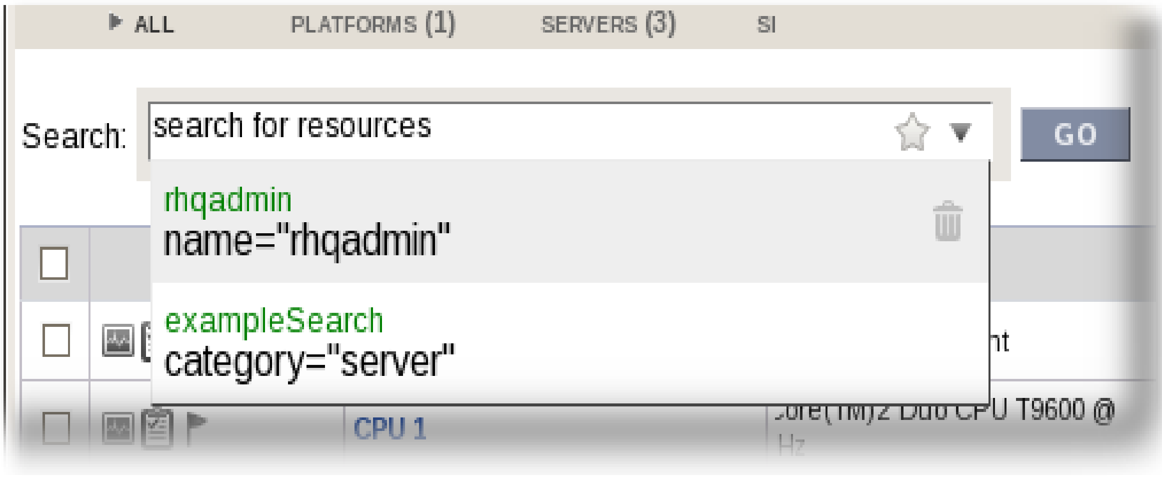
2. Click the star in the right of the search bar. When the field comes up, enter the name for the new search.



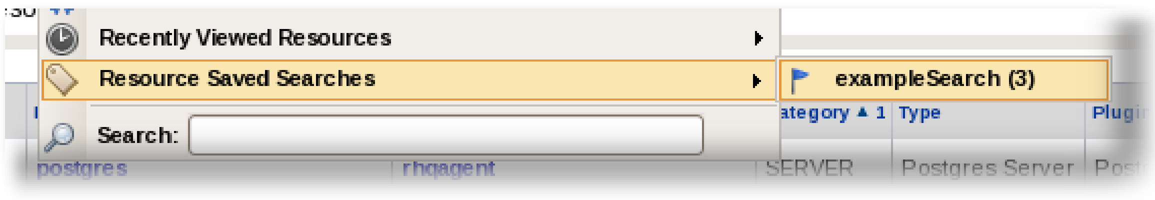
The search name is then displayed in green.



Saved searches are listed with other search results whenever the dynamic search criteria match any part of the saved search string. Saved searches are also listed immediately whenever the dynamic search field is active, before search parameters are entered.



The most recently created or active saved searches are listed in the **Resources** and **Groups** menus as another option. These menu items also show the number of resources that were last returned by the search.



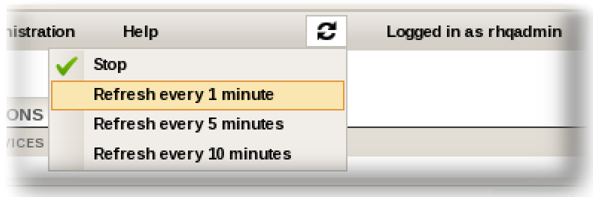
To delete a search, click the trash icon that is at the right of the drop-down menu when the search is highlighted. Deleting a saved search removes it from the suggested search terms and from the recently active searches menu.

1.6. Refreshing the UI

By default, the JBoss ON UI only refreshes when a page loads or when it is manually forced to do so. For viewing monitoring results, alerts, or other time-based elements, it can be useful to set an automatic refresh rate.

1. In the top menu, click the refresh icon, an image of two arrows in a circle.

2. Select the time interval for the UI to refresh.



Importing and Organizing Resources

The *inventory* in JBoss ON is the repository that contains all of the servers and applications that are managed or monitored by JBoss Operations Network. The inventory tells JBoss Operations Network which resources it can manage. Resources must must be imported into inventory before they can be managed.

Once in the inventory, resources can be organized in several different ways. Resources can be grouped automatically by their type in auto groups, resources can be added manually to user-defined groups, and they can be added manually to another resource as a child.

This chapter covers the process of identifying and importing resources through discovery, adding children, and managing groups.

2.1. Running Discovery

When an agent is installed, it *scans* the platform, and all applications on it, for any servers, services, or other items which can be included into the inventory. The process of finding potential resources is called *discovery*.

There are different scans for each type of resource: platform, server, and service. High level scans for servers and platforms are initiated by the agent every 15 minutes. A service scan detects lower-level services that are running in servers that have already been imported into the inventory. These scans run by default every 24 hours. Both of these intervals are configurable.

NOTE

A server must be imported into the inventory before any of its child processes, servers, or services can be detected by the discovery scan.

Although discovery is run automatically by the agent, discovery can also be initiated manually to capture infrastructure changed immediately.

2.1.1. Auto Discovery and Imports

Resources are moved into inventory using the auto discovery portlet. JBoss ON agents send information about the platform and servers it discovers back to the JBoss ON server, usually via a process table scan. This scan is performed when the JBoss ON agent first starts, and then runs periodically to discover newly added resources in the system.

The discovered resources will be shown in the auto discovery portlet on the dashboard.

Resources are committed to inventory by clicking the **IMPORT** button. For each committed platform or server, a recursive scan will search for nested services, and automatically committed each beneath its respective parent resource.

Clicking **View all...** in the auto discovery portlet lists all resources that are either new or ignored. Ignored services can be moved from ignored to a **NEW** state.

2.1.2. Manual Discovery

Resources that cannot be auto discovered can be manually added to inventory. To add a new resource, use the resource browser to find its logical parent resource. Since the resource browser only displays committed resources, you must first import the parent resource before manually adding a child resource to the inventory.

Click on the inventory tab of the parent resource. To manually move a resource into inventory, click on the drop-down menu item labeled **Manually add...**. Then select the type of resource you want to add.

The next page displays all the properties the plug-in requires in order to uniquely identify and successfully connect to this new resource. Each resource type in the system has a different set of required properties.

After submitting the form, the resource will start in the committed state and will be available for viewing in the resource browser.

Note

Manually adding a child resource to a parent resource requires that the parent resource be **UP** (green status icon).

The resource is now ready to be monitored and managed like any other auto discovered resource.

You can force an agent discovery by issuing the following command at the agent command prompt:

```
> discovery -f
```

2.1.3. Ignoring Discovered Resources

If your agent discovers a new platform and finds a few resources that you do not want to take into inventory, you have to tell the server to ignore those resources.

Select the resources to import in the auto discovery portlet and deselect the unwanted resources. As long as they are displayed in the portlet, they are not imported.

Alternatively, select the resource and click **Ignore**, this removes the resource from the auto discovery portlet, providing the parent resource has already been added to inventory.

You can ignore a server on a platform by performing the following steps:

1. Import the platform and leave that server unchecked
2. When the platform is successfully imported, select the server and click on **Ignore**

Note

It is not possible to just ignore an entire platform. If you want to ignore a platform, do not run an agent on it.

2.2. Creating New Resources

The **Create New** feature is used to create and deploy new child resources including EAR files, WAR files, data sources, connection properties, JMS queues and JMS Topics. The process for creating each of these resources is similar for all resource types. This process is outlined below.

2.2.1. Create

1. From the **Dashboard** select the parent resource, either from the resource tree on the left, the **Resources** menu item on the top navigation bar, or from the **Favorite Resources** area if the resources has been saved as a favorite.

2. Click the **Inventory** tab.

3. Scroll to the bottom of the Child Resources table to the **Manually Add:** drop down menu. Select the resource type you would like to create and click **OK**.

4. Enter the information for the resource to be deployed and click **OK**.

A system message will be displayed indicating whether the request to deploy the resource was successful. The success or failure of the deployment attempt will also be listed in the **History of Create Child Resource Requests** table at the bottom of the Child Resources table. It may be necessary to refresh the page to get an updated status of the deployment.

2.2.2. Delete

1. From the **Dashboard** select the parent resource, either from the resource tree on the left, the **Resources** menu item on the top navigation bar, or from the **Favorite Resources** area if the resources has been saved as a favorite.

2. Click the **Inventory** tab.

3. Check the box next to each WAR or EAR application that you would like to delete in the Child Resources table

4. Click the **DELETE** button.

Warning

Undeploying a resource with the **DELETE** button will actually remove the resource from the disk.

2.2.3. Managing Scripts as Resources

Scripts are auto detected on a server, as are other applications and services on the machine. Scripts can be configured and managed like any other resource, which means that JBoss ON allows you to both define configuration settings for and set up operations to run the scripts in inventory.

The environment variables that will be passed to the script must be on a new line and have the syntax **`name=value;`**. The variable's value can contain properties with the syntax **`%propertyName%`**, the script plug-in will interpolate these with the current values of the corresponding properties from the script's parent resource server's connection properties.

TIP

Add the line **`@echo off`** in Windows scripts to prevent echoing the executed commands along with the execution results.

Before setting environment variables in the JBoss ON configuration, make sure that the environment *on the resource* is already configured properly.

After resetting an environment variable, restart the JBoss ON agent to propagate the changes. If the agent isn't restarted, new variables will not be propagated and will not resolve during the operation execution, even if the configuration is correct.

2.3. Backup

You can back up existing EAR and WAR resources when deploying a new resource with the same name and type.

You can specify if you want to back up EAR and WAR resources in the create dialog box.

For example, if you had a file test.war and you selected **Create Backup**, and a file test.war already existed in the deploy folder, the existing file will be renamed to test.war.bak and the new version of test.war will be deployed.

To revert test.war to the previous version, go to the **OPERATIONS** tab and click on **revert**. If test.war.bak exists this will move test.war to a temporary backup and test.war.bak will be moved to test.war. If this was successful, test.war.bak will be deleted and only the rolled back version will exist.

If you try to revert again, or the backup file did not exist, the rollback will fail and you will get a fail message in the operations history.

2.4. Getting Inventory Reports

One quick management tool in JBoss ON is an inventory report. The report summarizes the resources currently in the inventory, grouped by resource type and five summaries:

- Resource type
- The JBoss ON server plug-in which manages the resource
- The JBoss ON category for the resource (platform, server, or service)
- The version number of the resource type; for example, for the Linux resource type, there can be a Linux 2.6.18-164.15.1.el5 version
- The total number of resources of that type in the inventory

To generate the inventory report:

1. In the top navigation, open the **Administration** menu, and select **Reports**.
2. Select the **Resource Version Inventory Report**.
3. On the reports page, select the **Group by Version** checkbox so that the version of each inventoried resource is displayed.
4. Click the name of any resource type to go to the inventory list for that type.

2.5. Organizing Resources in Groups

A group contains one or more resources and enables administrators to easily manage large numbers of resources across the entire enterprise, performing functions like checking group availability, deploying updates, and managing operations. Groups also make it possible for administrators to assign security permissions to users based a group of resources rather than assigning the permissions for individual resources one at a time. Administrator can then add new resources to the group without having to assign new security permissions to users who have access to the group.

There are two types of groups users can create:

- Mixed Group

  A mixed group contains resources of any resource type. There is no limit to how many or what types of resources can be placed into a mixed group. Mixed groups are used mainly for security purposes. If you have a set of resources that you want to give limited access to for a set of users, you should put those resources in a mixed group. For example, if you have an application that is running on a platform that consists of a JBoss Application Server and a Hibernate service, you can put the platform, JBossAS server and the Hibernate service in a single mixed group. You can then create a role and assign the mixed group a set of application administrator users to the role. If those users do not have access to any other roles in the JBoss ON system, those users will only be able to view or manage their applications; they will not have access to any other resources within JBoss ON.

- Compatible Group

  A compatible group is similar to a mixed group, except that it can only have resources that are of the **same** resource type. Because a compatible group limits membership to only resources of identical types, you can perform group operations on the members of the group. The JBoss ON UI console allows you to select a compatible group and execute an operation on that group. The available operations come from the list that are valid for the particular resource type associated with the compatible group.

2.5.1. Recursive Groups

Recursive groups simplify the authorization model by providing a convenient mechanism for providing access to a large number of resources quickly. When you create either a mixed or compatible group and mark it as recursive you are implicitly giving access to all descendants of that resource.

For instance, if you added to a JBossAS server a recursive group, that group's membership would then implicitly also include the EJB session beans, EARs, WARs, queues, datasources, connection factories, etc under that server.

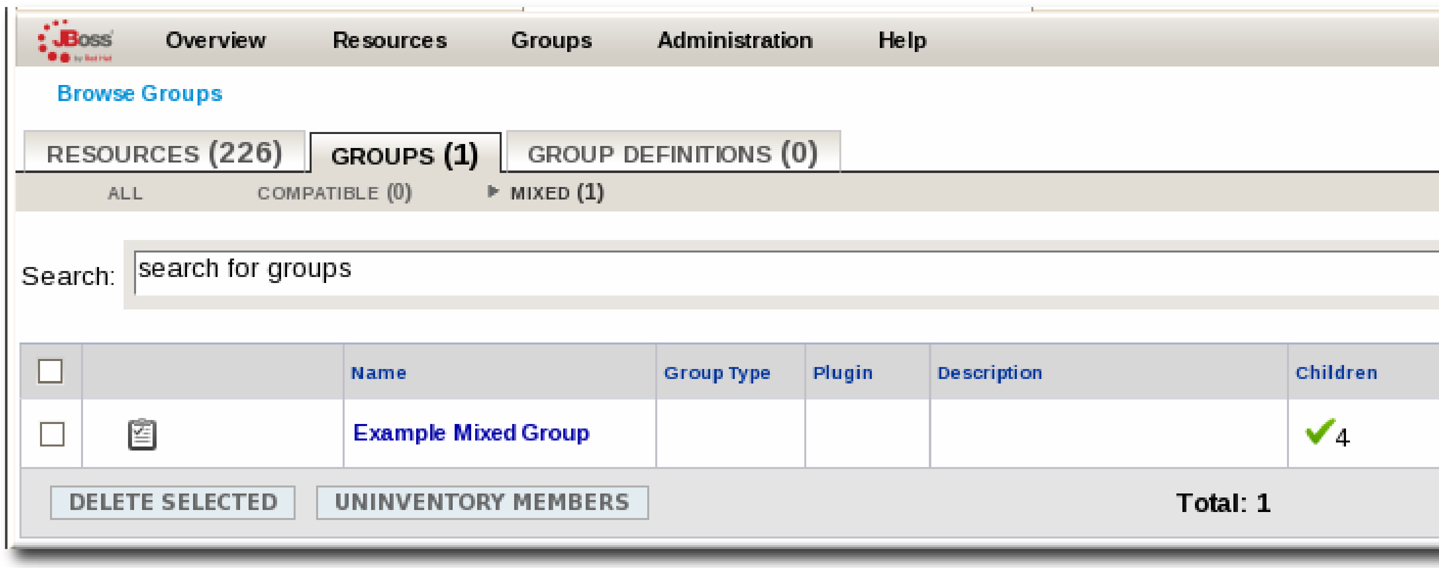
2.5.2. Common Uses

Mixed groups allow for resources of any type to be grouped. If a mixed group is marked as recursive and then added to various platforms and servers from across the inventory, it can very quickly give access to these heterogeneous resource types. Mixed groups are best suited for authorization purposes.

Compatible groups best suited for performing actions together against the inventory. Membership is restricted to resources of the same type. This allows aggregate metrics, operations, connection properties, configuration, and events to be shown or modified across all members of the group simultaneously. A recursive compatible group takes the concept one further and authorizes group-wise actions across direct children and descendant resources in the group.

2.5.3. Children Versus Descendants

When adding a resource to a recursive group, all descendant resources will be implicitly added to the group. The resources added to the group are referred to as the children resources. The resources that implicitly get added to the group automatically, are referred to as the descendant resources.



		Name	Group Type	Plugin	Description	Children
☐		Example Mixed Group				✓4

Total: 1

Figure 2.1. Mixed Group with Descendent and Children Resources

Anywhere in the user interface that displays lists of groups has two availability columns. This clarifies whether a primary or a descendent resource in the group is down.

2.5.4. AutoGroups

Every resource is part of a resource hierarchy with ancestry leading up to its platform, as well as child resources and other descendants. This is displayed in the Resource Tree Left Navigation area. A resource may have many child resources. For example, a database server resource may have several User and several database child resources. Each of these resources can be viewed independently by clicking on it. It may also be useful to view all of the children of a certain resource type together so that you get an aggregate view. To do this, JBoss ON automatically groups child resources of the same type into an *AutoGroup*.

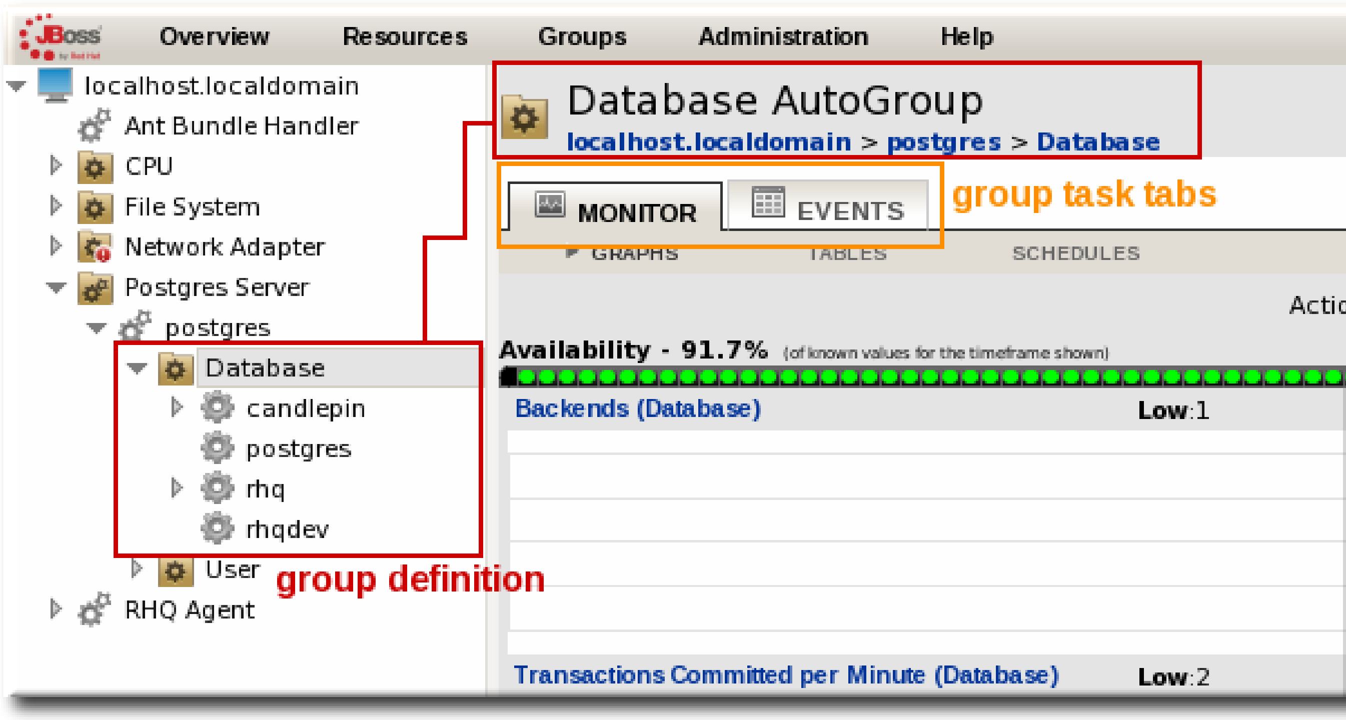


Figure 2.2. PostgreSQL Autogroup

Figure 2.2, “PostgreSQL Autogroup” shows all of the autogroups created in that JBoss ON deployment, with the PostgreSQL group selected. Other autgroups include services like CPU and system settings like Network Adapter. A red mark by the group name means that at least one resource within the group is unavailable.

Since autogroups are, by definition, groups of like resources, they are similar to compatible groups. They do not have all of the features of a compatible group because they are created dynamically, whereas true compatible groups are user created and persisted. But, since each of the autogroup's members offers the same metrics, those metrics can be combined to offer aggregated views.

The autogroup in *Figure 2.2, “PostgreSQL Autogroup”* has two tabs for both monitoring an events. Every autogroup has a monitoring tab, but the events tab is only available for an autogroup if the resource type in that group support event monitoring.

2.5.5. Autoclusters

Understanding cluster navigation first requires an understanding of compatible groups. A compatible group is a group of resources, all of the same type. A compatible group can be manually defined by a user via **Create Group** or generated as a dynamic group via **Create Group Definition**.

Once a compatible group, exists it can be selected by clicking on it from the list presented by navigating to **Resources** > **Compatible Groups**. Selecting a compatible group will then make it the root of the left navigation tree.

An autocluster groups instances of the same resources together, regardless of where they occur in the environment. Autocluster allows administrators to manage the same logical resources across different physical resources.

2.5.6. Managing Groups

2.5.6.1. Defining Groups

1. From the top navigation bar select **Groups** → **New Group**.
2. Enter a name for the group.
3. Select the group type, choosing from between **Compatible Resources** and **Mixed Resources**.
4. The recursive check box will be enabled for **Mixed Resources**.

NOTE

Only users with the global SECURITY or INVENTORY permission can create groups.

2.5.6.2. Removing Groups

1. From the top navigation bar select **Groups** → **All Groups**.
2. Select the group via its check box.
3. Select **DELETE SELECTED** at the bottom of the page.

NOTE

Removing the group does not remove the resource from inventory.

2.5.6.3. Groups Availability

On the **Browse Resources** page the number of up and down members for both the children and descendants of each group is displayed.

Clicking on the name of the group, the availability for that group will be visible at the top right-hand corner of the page.

Group availability is displayed as follows:

- **Green check**: icon representing members in the group which are up or available
- **Red exclamation point**: icon representing members in the group which are down or unavailable
- **Grey question mark**: the availability of the group members are unknown
- **Yellow triangle**: some of the resources in the group are up, some are down

2.5.6.4. Viewing Group Connection Properties

From the Inventory tab of a compatible group, there is a section labeled **Group Connection Properties**. Since a compatible group can only contain members of the same resource type, it is possible to see an aggregate view or average of all of their individual connection properties.

The rules are:

- If all of the resources in the group have identical values for a property, the group connection property will be **that** exact value
- If even one resource has a different value than the rest of the resources in the group, then that property will have a special marker value of **~ Mixed Values ~**

2.5.6.5. Editing Group Connection Properties

Editing the connection properties, via the **EDIT** button on the inventory tab, allows for one or more connection properties across the entire group to be updated. The **Override** check box determines whether or not a resource will be updated. If the override box is checked, that property will be passed to the resource for updating. If the override box is left blank, then that property will remain untouched on all resources, regardless of whether the value was already the same across all resources or had mixed values.

2.5.6.6. Asynchronous Updates

Updates to the connection properties for a single resource are done at the time the user enters the details, group updates are not. Group connection property updates are pushed across the enterprise asynchronously, this allows the user to continue using JBoss ON as the updates are processed and prevents the UI becoming unresponsive while the updates are processed.

Updates on the process can be obtained by accessing the inventory tab where a message located at the top of the page will provide status details. Refreshing the page will update the status message.

NOTE

Refreshing the inventory tab will cause the JBoss ON server to display the current values of the connection properties for each resource in the group. If the asynchronous update has not yet completed, but it has successfully changed some of the resource's connection properties, intermediate values will be displayed. **Please ignore these values**. Once the updates have completed, refresh the the page to view final results.

2.5.6.7. Update History

Updates can be in one of three states:

- **In Progress**: there is at least one resource in this group that has not completed the update to its connection properties.
- **Failed**: the group update is complete, but at least one resource's connection properties could not be updated.
- **Success**: the group update is complete, and the connection properties of all resources in the group were successfully updated.

2.5.6.8. Viewing Update Details

Update details can be viewed by clicking on the hyperlink in the **Date Created** column of the **Plug-in Configuration Update History** section.

The first section, labeled **Group Configuration Update Details**, displays the details of which resources have been updated.

The second section, labeled **Resource Configuration Updates Results**, displays which resource's connection properties failed to update and provides details. Clicking on the hyperlink in the **Date Created** column links to the inventory tab where any relevant error messages will be displayed.

2.5.7. Group Definitions and Dynamic Groups

Using group definitions and dynamic groups helps automate the management of large inventories.

A group definition is a rich, expressive short-hand for resource groups. At a high level, a definition tells the system how to create a group; the details are handled by specifying a list of rules that tell the system how to find and aggregate the appropriate resources to add as members of the group.

Enterprise resources can be grouped by cluster identifier, broadcast group, logical service layer, geographical location, security domain, or any any other logical grouping.

Individual resources can potentially belong to multiple groups, in large inventories it is important to know how difference group definitions will affect the enterprises resources.

2.5.7.1. Subsystem Integration

Subsystem	Major Type	Supported Attributes
Inventory	resource	id, name, version, parent-resource, grandparent-resource, children-resources
	resourceType	plug-in, name, category (platform, server, service)
Configuration	plug-inConfiguration	<any-plugin-configuration-property>
	resourceConfiguration	<any-resource-configuration-property>
Measurement	traits	<any-trait>
	availability	UP, DOWN

2.5.7.2. Group Definitions and Dynamic Groups Syntax

A group definition is a collection of one or more valid expressions, and each expression must follow one of the following two formats:

- `<resource-expression>` [`<string-match-expression>`] = `<value>,`
- "groupby" `<resource-expression>`

The first format is called a simple expression; the latter, a pivoted expression. A group definition that contains *only* simple expressions will only create one child group. A group definition that contains even *one* pivoted expression, has the potential to create many groups. See the other "Syntax Reference" sections for more information.

Although the above two formats form the basis for all of the functionality in group definitions, there is one small caveat. Since simple expressions are compared using '=' equality, it will not find records whose values are null. Since all information in JON is eventually stored into RDBMS, special handling is needed in order to compare values against null properly. Starting with JON 2.2 you can create a simple expression that matches against entries with null values:

- empty <*resource-expression*> [<*string-match-expression*>]

Notice how the "= <value>" at the end of the expression was removed and replaced with the "empty" keyword at the beginning of the expression. Conversely, you can create a simple expression that matches against entries whose values are NOT null:

- not empty <*resource-expression*> [<*string-match-expression*>]

In this case, the value can be anything as long as it is not null. This concept also works for pivoted expressions. The syntax for pivoted expressions has not changed, but if any of the expressions you are pivoting on has a value that is null, a special group will be created to capture the resources whose values are null.

2.5.7.3. Resource Expressions

This section describes the the various options for the <*resource-expression*> element introduced in the "Quick Summary" section.

```
<code>
resource.id
resource.name
resource.version
resource.type.plugin
resource.type.name
resource.type.category
resource.availability
resource.pluginConfiguration[<property-name>]
resource.resourceConfiguration[<property-name>]
resource.trait[<property-name>]
</code>
```

All 10 of the options you see above that start with **resource** are also available for the following prefixes:

```
<code>
resource.child
resource.parent
resource.grandParent
</code>
```

2.5.7.4. User Interface

The name of a group definition is important because it contributes to the calculated name of the resource groups it automatically creates, deletes, and updates with the results of its dynamic search. In the following examples we have named the group definition "GD1195":

- If you use simple expressions, the calculated name of the singular group created would be: DynaGroup – GD1195

- If you use a pivoted expression, the calculated name of the various groups created would appear as follows:
  - DynaGroup – GD1195 – {groupBy tokenValue1}
  - DynaGroup – GD1195 – {groupBy tokenValue2}

2.5.7.5. Expression Templates

Starting in JON 2.2, when you create a new group definition, or edit an existing one, you can quickly build your expressions set by choosing one of the following templates:

- The first section, labeled **One group for every...**, lists the options for pivoted expression sets.
- The second section, labeled **Exactly one group containing...**, lists the options for simple expression sets.

When you select a simple or pivoted template from the list, the text area will be overwritten with the set of expressions that implement that template.

2.5.7.6. Expression Builder

Alternatively, clicking on the blue plus sign will initiate the DynaGroup Expression Builder. The expression builder is a series of inter-related HTML input text fields, check boxes, and drop-down lists that hide the details behind the raw syntax.

As you change the various drop-down boxes you will notice that various other controls may be enabled or disabled. This is done purposefully to try and provide a smoother interaction and to prevent combinations of choices that would result in invalid expressions.

Each time you select something from a drop-down list, toggle a checkbox, or type a value into an input field, the area at the top will automatically render and show you the raw syntax that that particular set of options corresponds to.

Once you have selected the required options, click the **Add Expression** button. The expression builder window will close, and the expression will be added to the text area of the group definition you were editing.

2.5.7.7. Automatic recalculation

JON 2.2 introduced the ability to automatically recalculate on a periodic basis certain group definitions. When you create a new group definition, or edit an existing one, the following interface will appear:

By default, group definitions do not recalculate automatically (value of 0). If you want the system to periodically recalculate one or more of your definitions, set this to a non-zero value that represents the number of minutes in between recalculations.

Warning

Recalculating a group definition across large inventories could be expensive, so use this feature with caution. For example, you may want to recalculate a group definition once an hour as opposed to once a minute.

Note

If you urgently need to re-calculate the group definition, you do not have to wait for the recalculation interval to expire; simply go to the group definition and hit the **Calculation Groups** button. Since the recalculation interval is a relative value, as opposed to a date/time stamp, it is **always** based on the time when the group definition was last calculated. Recalculating group definitions immediately should not have a significant, adverse affect on the performance of recalculating other definitions in the background.

2.5.8. Tutorial

Expressions can take on one of two forms: a pivoted expression, or a simple expression. An example of this would be:

```
resource.parent.type.category = Platform
```

This means that you want to search your entire inventory to find any resources that have a parent that is a platform. In other words, you want to create a single group, and put all the direct children resources of any platform in your inventory into it. You can take a further step by adding another filter to the one above:

```
resource.parent.type.category = Platform
resource.name.contains = JBossAS
```

This compound expression, each part of which must be on a separate line, will search the entire inventory as above. This time, however, it will only add resources to the group that have "JBossAS" in their name.

For large inventories this would create a group with a very large membership. There are ways to avoid this. Pivoted expressions enable you to utilize group definitions more effectively. For example, the following pivoted expression:

```
groupby resource.parent.name
```

The above expression would first take the portion after the "groupby" text, and ask the inventory for ALL unique "resource.parent.name" results. It will find one result for each uniquely named resource in the system, as long as that resource has one or more children resources. For each unique result, it would create one group, and put any resource into it that matched that criteria.

For example, for this inventory:

- ResourceParentA
  - ResourceChildA1
  - ResourceChildA2
    - ResourceGrandChildA2a
    - ResourceGrandChildA2b
  - ResourceChildA3

- ResourceParentB
  - ResourceChildB1
  - ResourceChildB2
  - ResourceChildB3
  - ResourceChildB4

The first part of the expression will find all the unique names of the parent resources:

- ResourceParentA (which has 3 direct children)
- ResourceChildA2 (which has 2 direct children, which are the grandchildren of its parent)
- ResourceParentB (which has 4 direct children)

It will then create one group for each of them, and populate them as necessary to satisfy the expression. The group members would be as follows:

With only a few characters you have now automated the creation of three groups (but this would be more in larger inventories), **and** the memberships of those groups.

The above is only an introduction to pivoted expressions. JBoss ON also supports compound pivoted expressions, which look very similar to the compound simple expression discussed at the beginning of this section. An example of a compound pivoted expression might be:

```
groupby resource.type.plugin
groupby resource.type.name
groupby resource.parent.name
```

The "groupby" text is required in the front of each expression that you want to make a pivot. This compound expression creates a compatible group ensuring that we automatically get one group for each unique type currently known to the system.

The pivot concept can be taken one step further. Instead of having one gigantic compatible group of, for example, EJB3 Session Beans across your entire enterprise, you can also pivot on the parent. You could therefore get one group for each collection of Session Beans that share the same parent.

Pivoted expressions enable you to create extremely fine-grained, automatically managed, divisions of your resources. For a small inventory, this is a useful tool; for a large inventory, this is an invaluable tool.

The mixing and matching of pivoted and non-pivoted expressions in the same group definition is also supported. We will use the last example, but we will filter the results even more precisely as follows:

```
resource.type.category = server groupby
resource.type.plugin groupby
resource.type.name groupby
resource.parent.name
```

This compound, mixed expression would constrain the search. Instead of creating one compatible group for every single resource type known in the system, it would only create compatible groups for every unique resource type that is also a server.

Finally, we should make a quick note about the new “empty” keyword introduced in JON 2.2. What if you wanted to find all of your non-secured JBossAS servers in inventory? In other words, you want to find all of the instances that have a connection property called 'principal', but whose value must be null/unset. The following expression would achieve this:

```
<code>
resource.type.plugin = JBossAS
resource.type.name = JBossAS Server
empty resource.pluginConfiguration[principal]
</code>
```

2.5.9. Hints, tips, and tricks

2.5.9.1. Ordering Expressions

For example, the following two expressions are processed in exactly the same way:

```
<code>
exprA1
exprA2
groupby exprB1
groupby exprB2
</code>
```

```
<code>
exprA2
exprA1
groupby exprB2
groupby exprB1
</code>
```

But the following two are not:

```
<code>
exprA1
groupby exprB1
</code>

<code>
exprB1
groupby exprB1
</code>
```

2.5.9.1.1. Empty Lines in Expressions

For the purposes of readability, you are allowed to insert blank lines into a group definition to help visually separate or aggregate related expressions. For example, the following is permitted:

```
<code>
exprA1
```

```
exprA2

exprB
</code>
```

2.5.9.1.2. Expressions will not Create Empty Groups

If a definition is so restrictive that it creates groups with no resource members that match the filters, the group will not be created. This happens when you attempt to create fine-grained groups of resources that match a very specific set of properties. Instead of creating a few empty groups, which could consequently make managing large numbers of resource groups more difficult, the system will suppress their creation. This feature was added to make managing large inventories more intuitive.

2.5.9.2. Examples

2.5.9.2.1. JBoss Clusters

```
resource.type.plugin = JBossAS
resource.type.name = JBossAS Server groupby
resource.trait[partitionName]
```

2.5.9.2.2. Each Operating System Type

```
resource.type.plugin = Platforms
resource.type.category = PLATFORM groupby
resource.type.name
```

2.5.9.2.3. All Auto-Groups

```
groupby resource.type.plugin
groupby resource.type.name
groupby resource.parent.name
```

Note

This could create many groups in large inventories

2.5.9.2.4. Just Raw Measurement Tables

```
resource.type.plugin= Postgres
resource.type.name = Table
resource.parent.name = rhq Database
resource.name.contains = rhq_meas_data_num_
```

2.5.9.2.5. Just Agents Configured with Multi-Cast Server Detection

```
resource.type.plugin= RHQAgent
resource.type.name = RHQ Agent
resource.resourceConfiguration[rhq.communications.multicast-detector.enabled] =  true
```

2.5.9.2.6. Just Windows Platforms with Event Tracking

```
resource.type.plugin= Platforms
resource.type.name = Windows
resource.pluginConfiguration[eventTrackingEnabled] =  true
```

2.5.9.2.7. JBossAS Servers Grouped by Machine

```
groupby resource.parent.trait[Trait.hostname]
resource.type.plugin = JBossAS
resource.type.name = JBossAS Server
```

Setting up Content for Resources

JBoss Operations Network can be used to store and deploy content to resources. This can be done to apply updates and patches (as with JBoss AS servers) or to set up repositories used for provisioning applications and deploying custom software.

This chapter describes how to set up and manage content repositories. For information on applying patches or provisioning applications, see *Chapter 4, Managing Resource Configuration*.

3.1. Managing Content: Providers, Repos, and Packages

In JBoss Operations Network, the content system uses plug-ins to pull in specific pieces of content on a resource. This content typically consists of files related to the resource, including executable, configuration, or log files. The content system records these files in the inventory and captures metadata on different revisions.

JBoss ON can manage content by defining sources and collections of content (called content sources and repositories). Resources are

Content sources are developers and distributors of content. Sources can be external third party software developers (such as Red Hat) or internal development teams that create custom content. The type of content available from sources includes both software packages (such as RPMs or configuration scripts) and updates (version upgrades, patches, and errata).

A *repository* is a collection of software packages, which can come from one or multiple content sources. Repositories are user-defined collections. A repository may contain packages for an application or family of applications or for a specific purpose, like repositories for laptop configuration and repositories for installing web servers.

A *package* refers to any piece of content. A package is usually a file of some kind, like a JBoss AS JAR file or an RPM for a Linux platform. (It is also possible that a plug-in contains data not directly tied to a file.) A package type is defined in a plug-in's descriptor, which is associated with a particular resource type. Each package type is defined by certain attributes, listed in *Table 3.1, “Package Attributes”*.

NOTE

Package versions from different content sources can be associated with the same channel.

Attribute	Description	Optional or Required
Name	The programmatic name of the package type.	Required
Display Name	A user interface friendly name of the package type.	Optional
Description	Describes the type of content found in packages of this type.	Optional
Category	One of four enumerated options:	Required

Attribute	Description	Optional or Required
	• Executable Script (which is potentially editable) • Executable Binary • Configuration (a configuration file for the resource) • Deployable	
Discovery Interval	Defines the time between package discovery scans for this type; different package types can be configured with intervals to represent the likelihood of the package inventory changing.	Optional
Creation Type Flag	If set to true, a package of this type is used when creating resources of the enclosing resource type. An example of this situation is a Java EAR file. There is an EAR resource type that represents the enterprise application in JBoss ON. Under that resource type, there is a package type defined to represent the EAR file itself. This package type is flagged as a creation type; when creating a new EAR resource, the EAR file must be created at the same time. The default for this attribute is false, as packages will typically not represent the creation of a new resource.	Optional
Configuration	The configuration element allows the plug-in to define an open-ended set of attributes about the package type. These values will be populated during package discovery, and if not marked as read only, can be specified by the user at artifact creation time. An example of a property in this configuration element is a Boolean that describes if an EAR file is deployed as exploded or	Optional

Attribute	Description	Optional or Required
	zipped. When EAR files are discovered, this flag will be populated and carry package type specified information. Additionally, when deploying a new EAR file through JBoss ON, this flag can be set to indicate how the package should be deployed on the AS instance.	

Table 3.1. Package Attributes

Packages are versioned, meaning that there can be multiple iterations of the same packages maintained in the repository.

The actual content of a package version — the bytes themselves — are known as its package bits. Package bits are stored in the server by uploading them, having an agent auto-discover them, or downloading them through a content source to a remote repository.

Whenever a package is installed on a resource, it is recorded in the content history for the resource and the package. Since there can be multiple files associated with a single packe, then there can be multiple files recorded in the content history, all associated with that package version.

A package is inventoried in JBoss ON through a recurring *package discovery scan*. For each inventoried resource against which packages are defined, the plug-in will be queried for artifacts of a particular type. The interval at which this discovery occurs may be overridden in the package's definition in a plug-in's descriptor. The default value, if none is specified, can be found in the plug-in schema file.

Once inventoried in JBoss ON, all of the data which identifies each package is stored in the details page.

Detail	Description
Name	When a plug-in discovers a package, it assigns a descriptive name of what the package represents in the system.
Version	Identifies the specific version of the package.
Type	The package is defined as being of a particular type. Multiple packages can have the same name as long as they are of different package types.
Architecture	Identifies the architecture of the package, if applicable. There is a noarch option when the package is not built against a specific architecture.
Content Retrieval Time[1]	Time stamp at which the revision was discovered.
MD5[1]	MD5 checksum of the package file.

Detail	Description
SHA256[1]	SHA256 checksum of the package file.
Length[1]	Size of the package.
Owner[1]	Owner of the package (for the case of a file, the file system user who owns the file).
Creation Date[1]	Date the package was created.
Last Modified Date[1]	Date the package was last modified.
Last Accessed Date[1]	Date the package was last accessed.

This information is optional. The plug-in deteremines whether this information is available and whether to populate the package description with it.

Table 3.2. Package Discovery Details

3.2. Managing Content Sources

3.2.1. Creating a Content Source

1. In the top menu, click **Administration**, and open the **Content** menu.

2. Select the **Content Sources** item.

3. Below the list of current content sources, click the **CREATE NEW** button.

4. Select the content type. There are currently four options, depending on how the content is delivered from the source.

5. Fill in the basic details. These identify the content source in the JBoss ON server and are the same for each content source type.

   - Give a unique name and optional description for the content source provider.

   - The schedule sets how frequently the content in the JBoss ON database is updated by the content source; this uses a standard *Quartz Cron Synax*[1].

   - The lazy load setting sets whether to download packages only when they are installed (**Yes**) or if all packages should be download immediately.

   - The download mode sets how the content is stored in JBoss ON. The default is **DATABASE**, which stores all packages in the JBoss ON database instance. The other options are to store the packages on a network filesystem or not to store them at all.

6. Fill in the other configuration information for the content source. The required information varies depending on the content source type. This is going to require some kind of connection information, such as a URL or directory path, and possibly authentication information, like a username and password.

3.2.2. Synchonizing Content Sources and JBoss ON

The original source of content is external to JBoss ON, and the content packages are pulled into JBoss ON and stored. Any changes that are made at the original content source need to be pulled into JBoss ON by *synchronizing* the two sources.

3.2.2.1. Scheduling Synchronization

Synchronization is already scheduled in the content source entry in JBoss ON. This schedule has the standard cron format.

```
*     *    *    *     *  [sync-command]
-     -     -     -     -
|     |     |     |     |
|     |     |     |     +----- Day of Week (0=Sunday ... 6=Saturday)
|     |     |     +------- Month (1 - 12)
|     |     +--------- Date (1 - 31)
|     +----------- Hour (0 - 23)
+------------- Minute (0 - 59)
```

For example, to synchronize the source with JBoss ON on Tuesday and Friday at 3am:

```
0 3 * * 2,5
```

The *Quartz documentation*[2] explains the cron syntax in more detail.

To edit the schedule synchronization times for a source:

1. In the top menu, click **Administration**, and open the **Content** menu.
2. Select the **Content Sources** item.
3. Click the name of the content source in the list of sources.
4. Click the **Edit** button in the **Basic Details** section.
5. Reset the cron schedule in the **Sync Schedule** field.
6. Click **Save**.

3.2.2.2. Manually Synchronizing Content Sources and JBoss ON

If a major change happens to the content source, then the changes can be manually sent over to the JBoss ON server by initiating a synchronization manually.

1. In the top menu, click **Administration**, and open the **Content** menu.
2. Select the **Content Sources** item.
3. Select the checkbox by the names of all of the content sources to synchronize.
4. Click **SYNC SELECTED**.

3.3. Managing Repositories

A repository is essentially a mapping between the data in a content source and specific resources in the JBoss ON inventory.

3.3.1. Creating a Repository

1. In the top menu, click **Administration**, and open the **Content** menu.

[2] http://www.opensymphony.com/quartz/wikidocs/CronTriggers%20Tutorial.html

2. Select the **Repositories** item.
3. Below the list of current repositories, click the **CREATE NEW** button.
4. Fill in the name and a description.
5. Click **Save**.
6. On the **Repositories** page, click the name of the new repo in the list.
7. *Optional*. To change the default synchronization schedule, click the **Edit** button and enter a new schedule, in a cron format, in the **Sync Schedule** field.
8. Add content sources to supply content to the repository.

   a. In the **Content Sources** section, click the **SUBSCRIBE** button to add existing content sources to the repository.

   b. Select checkboxes next to the content sources to associate with the repository.

   c. Click the **SUBSCRIBE SELECTED** button.

   More than one content source can supply content to a repository.
9. Associate resources with the repository. A resource can only be updated by the repository if it is associated with the repo.

   a. In the **Resources** section, click the **SUBSCRIBE** button to add resources to the repository.

   b. Select checkboxes next to the resources to associate with the repository.

   c. Click the **SUBSCRIBE SELECTED** button.

TIP

You can search for specific resources or types of resources and subscribe multiple resources at once.

3.3.2. Importing Content Sources

There are a couple of ways to map the repositories to the right content sources. A repository can be subscribed to multiple content sources by editing the repo configuration. A content source can be added to multiple repos simultaneously by importing the content source.

3.3.2.1. Associating Content Sources with a Repository

1. In the top menu, click **Administration**, and open the **Content** menu.
2. Select the **Repositories** item.
3. On the **Repositories** page, click the name of the repo in the list.
4. In the **Content Sources** section of the repo's details page, click the **SUBSCRIBE** button to add existing content sources to the repository.

5. Select checkboxes next to the content sources to associate with the repository.

6. Click the **SUBSCRIBE SELECTED** button.

More than one content source can supply content to a repository.

3.3.2.2. Importing a Content Source into Multiple Repository

If the same content source will be associated with multiple repositories, the content source can be imported into all of them simultaneously.

1. In the top menu, click **Administration**, and open the **Content** menu.

2. Select the **Repositories** item.

3. On the **Repositories** page, click the **IMPORT** button.

4. Select the radio button by the name of the content source to import.

5. In the **Available repositories....** area, select the checkbox by the name of each repository to associate with the content source.

6. Click the **IMPORT SELECTED** button.

3.3.3. Associating Resources with the Repo

Content can only be sent to a resource if that resource is first associated with a repository.

3.3.3.1. Editing the Resources in a Repository

1. In the top menu, click **Administration**, and open the **Content** menu.

2. Select the **Repositories** item.

3. In the **Resources** section, click the **SUBSCRIBE** button to add resources to the repository.

4. Select checkboxes next to the resources to associate with the repository.

5. Click the **SUBSCRIBE SELECTED** button.

3.3.3.2. Managing Repos on a Resource

A few resource types, like platforms, have content tabs in their configuration which allows them to control their content subscriptions.

1. Search for the resource.

   Click the **Resources** tab and go to the resource category to browse for the resource or simply search for the resource in the **Resources** tab search box.

2. Click the **Content** tab of the resource.

3. Open the **Subscriptions** subtab.

4. The **Available Repositories** section has a list of repositories that the resource isn't subscribed to. Click the checkboxes by all of the repos to subscribe the resource to.

5. Click **SUBSCRIBE**.

The same process can be used to unsubscribe a resource from content repositories.

3.3.4. Synchronizing Repos and Content Sources

Any changes in the content source are carried voer to the repository when the source and repo are synchronized.

3.3.4.1. Scheduling Synchronization

Synchronization is already scheduled in the repostiroty entry. This schedule has the standard cron format.

```
*     *    *    *     *  [sync-command]
-     -     -     -     -
|     |     |     |     |
|     |     |     |     +----- Day of Week (0=Sunday ... 6=Saturday)
|     |     |     +------- Month (1 - 12)
|     |     +--------- Date (1 - 31)
|     +----------- Hour (0 - 23)
+------------- Minute (0 - 59)
```

For example, to synchronize the repo with its content sources on Tuesday and Friday at 3am:

```
0 3 * * 2,5
```

The *Quartz documentation*[3] explains the cron syntax in more detail.

To edit the schedule synchronization times for a repo:

1. In the top menu, click **Administration**, and open the **Content** menu.
2. Select the **Repositories** item.
3. Click the name of the repo to edit in the list.
4. Click the **Edit** button in the **Basic Details** section.
5. Reset the cron schedule in the **Sync Schedule** field.
6. Click **Save**.

3.3.4.2. Manually Synchronizing Repos and Content Sources

If a major change happens to the content source, then the changes can be manually sent over to the the repositories by initiating a synchronization manually.

1. In the top menu, click **Administration**, and open the **Content** menu.
2. Select the **Repositories** item.
3. Select the checkbox by the names of all of the repositories to synchronize.

[3] http://www.opensymphony.com/quartz/wikidocs/CronTriggers%20Tutorial.html

4. Click **SYNC SELECTED**.

Managing Resource Configuration

Once a resource is added into the JBoss ON inventory, JBoss ON can actually manage the resource configuration in two ways:

- Editing resource configuration files and settings through the JBoss ON GUI
- Provisioning new applications or pushing updates through the provisioning system

4.1. Changing Resource Configuration

Along with monitoring resource performance, JBoss ON can implement configuration changes. As with monitoring, the resources which can be modified and the types of modifications that can be made depend on the resource type. (Not all resources can modify their configuration through JBoss ON.) Basic resource configuration uses a structured form that makes it simple to view and change current settings on currently-deployed servers or services.

Servers, databases, even shell scripts can have configurable settings that are managed by JBoss ON. The configuration tab on resources displays the current configuration for those files or scripts. From there, the configuration can be edited or previous changes can be tracked or reverted.

NOTE

Not all resource types can be configured through JBoss ON. Any resource which can have its configuration edited through JBoss ON will display a small wrench by its name in the inventory browser and will have a **Configuration** tab in the resource details page.

The configurable properties for each resource are listed with the other monitoring and management properties in the *Resource Monitoring Reference*.

The configuration management integrates several functions:

- Make configuration changes to resource configuration files or set environment variables for the execution environment
- Apply configuration changes to groups of resource of the same type
- View audit trails in the history of both external and JBoss ON-initiated configuration changes
- Use alert notifications to send automatic announcements of any configuration changes
- Revert configuration changes

The basic element of configuration ia a *property*. A property in JBoss ON can have almost value. *Simple properties* can have simple string or numeric values, while *complex properties* can have more intricate lists or mappings. JBoss ON detects the type of property in the configuration and renders it in the JBoss ON GUI as appropriate (text fields, selection buttons, drop-downs, or collections of properties).

JBoss ON validates every change to configuration according to the interpreted format of the configuration value.

When editing the configuration, some elements have to be *set* before they can be edited. The **Unset** checkbox means that JBoss ON won't submit any values for that resource and any values are taken from the resource itself.

4.2. Changing the Configuration on a Single Resource

1. Search for the resource.

   Click the **Resources** tab and go to the resource category to browse for the resource or simply search for the resource in the **Resources** tab search box.

2. Open the **Configuration** tab.

3. Click the **Current** subtab.

4. Click the **EDIT** button to make the fields editable.

5. Make any changes to the configuration; if necessary, uncheck the **Unset** selection box so that the values are taken from the JBoss ON configuration rather than the local configuration.

   The list of available configuration properties, and their descriptions, are listed for each resource type in the *Resource Monitoring Reference*.

6. Click the **COMMIT** button.

4.3. Changing the Configuration through a Group

Similar to other templating functions in JBoss ON, like alert templates, configuration changes can be made on a compatible or autogroup, so that all of the members of that group can be up updated simultaneously with the same settings.

To change the current configuration for a group, a few conditions must be true so that the current group configuration can be reliably calculated for the member resource configurations:

- The group members must all be the same resource type.

- All group member resources must be available (`UP`).

- No other configuration update requests can be in progress for the group or any of its member resources.

- The current member configurations must be successfully retrieved from the agents.

To set configuration for a group:

1. Search for the group.

   Click the **Groups** tab and go to the **All Groups** or **Compatible Groups** area to browse for the resource or simply search for the resource in the **Resources** tab search box.

2. Select the group.

3. Open the **Configuration** tab.

4.

5. Click the **EDIT** button to make the fields editable.

6. Make any changes to the configuration. To change the value for all members to a single value, fill in the value in the field. To change the group member values to different values, click the pages icon by the property to open the member configuration edit form.

   The list of available configuration properties, and their descriptions, are listed for each resource type in the *Resource Monitoring Reference*.

7. Click the **SAVE** button.

4.4. Tracking and Reverting Configuration Changes

Every time a change is made to the resource configuration, whether through JBoss ON GUI or on the resource itself, the change is detected by JBoss ON and logged with a revision number.

The revision numbers are global number across the JBoss ON server. For example, if Resource A is edited, then it gets revision #1. Then, when Resource B is edited, it gets revision #2, and the next edit gets #3.

The current configuration has an asterisk (*****) by the ID number. (The current version may not be the latest configuration change; it is the latest *successful* configuration change.)

To view the change history or to revert to a previous configuration state:

1. Search for the resource.

   Click the **Resources** tab and go to the resource category to browse for the resource or simply search for the resource in the **Resources** tab search box.

2. Open the **Configuration** tab.

3. Click the **History** subtab.

4. All of the configuration edits are listed in the history. Click the checkbox by the revision number of the configuration version to roll back to.

   TIP

   To view the previous configuration version, click the version ID number. The previous configuration is displayed in the area below the history table.

5. Click the **ROLLBACK TO SELECTED** button.

Provisioning Applications

Provisioning is a more advanced form of configuration management. Provisioning takes uploaded packages and deploys them on targeted resources with instance-specific configuration. This makes it easier to deploy applications like JBoss, as well as patches and updates, consistently.

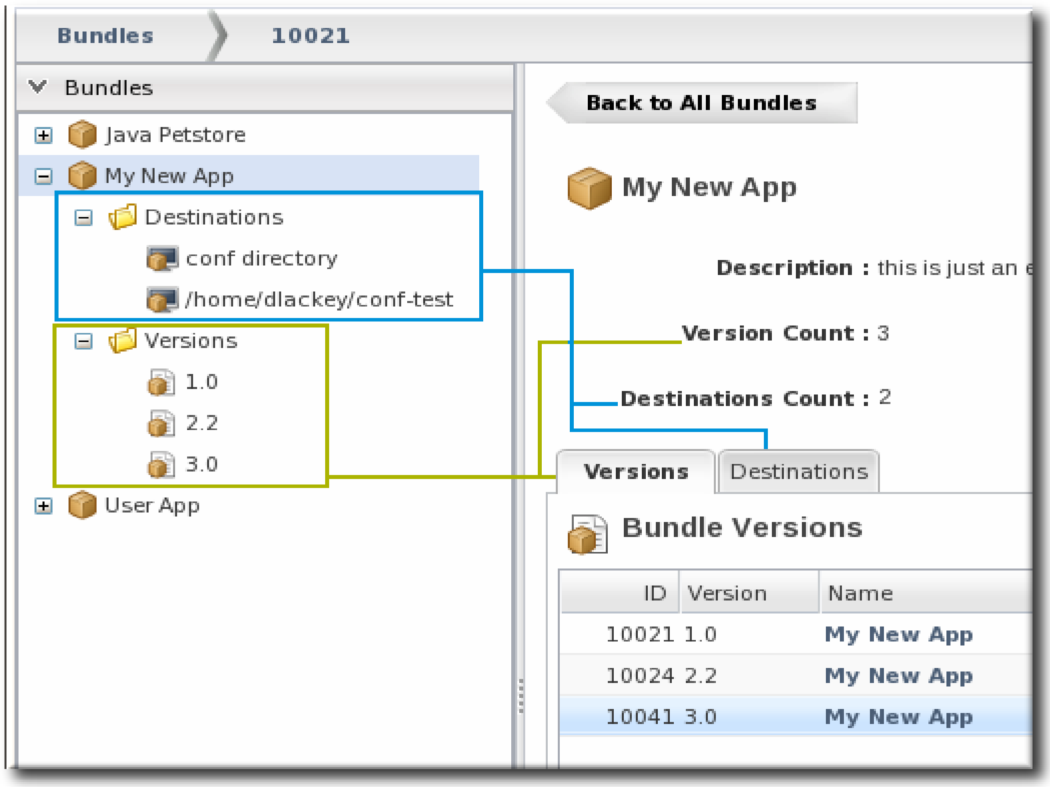
5.1. About Provisioning

Most types of applications can be deployed from a simple archive file. JBoss ON provides a way to upload or access those archive files and deploy them all at once, to all the platforms in a group.

The provisioning system in JBoss ON has basically two elements:

- An entity that is made of the files and specs of the application — the *bundle*
- The place where the application is installed (deployed) — the *destination*

Each version of a bundle and each destination where the bundle is deployed is collected into the bundle definition. From the top level, viewing a bundle shows all of the different versions of the bundle and all of the deployment destinations. Looking at a destination folder shows all of the versions that are deployed there, while looking at a version entry shows all of the files and recipes in that version and a list of all of the destinations where it is deployed.



Description : this is just an

Version Count : 3

Destinations Count : 2

ID	Version	Name
10021	1.0	My New App
10024	2.2	My New App
10041	3.0	My New App

Figure 5.1. Bundles, Versions, and Destinations

A bundle is composed of file that defines the metadata and processing rules for the bundle and then all of the files required for the applicaiton. These elements can be combined into a single *bundle distribution file*, which is an archive file.

Naturally, the first part of provisioning an application is actually writing the application. From there, the application files are gathered together into some kind of archive. To complete the bundle, write the recipe and any configuration files or scripts that are used by the application.

When a bundle is deployed, all of the files defined in that bundle are first copied over to the destination servers. Then, any templatized configuration files are *realized* by having the real values written in for the tokens. The recipe can also contain scripts which are invoked during provisioning to complete setting up the application.

5.1.1. About Bundles

The application that is deployed is defined in a *bundle package*. The bundle contains a *recipe* which sets the metadata, tokens that represent user- or system-defined values, and list of files that are covered in the application. The bundle also contains the files referenced in the recipe.

The recipe sets a version number for the bundle, so that the same application can have multiple versions available through the same bundle definition. Ultimately, versioning bundles simplifies application maintance by providing an audit trail of changes and a path to rolloback changes and apply updates smoothly. The recipe must follow a format that the server recognizes for it to be validated and loaded. JBoss ON; recognizes Ant recipes by default. Additional bundle types can be added by creating the requisite agent plug-ins.

The application bundle files can be both archive files that contain the application and configuration files. The provisioning process doesn't have any kind of file requirements; there can be multiple archive files, multiple configuration files, or only one file associate with the bundle.

The bundle files can be uploaded to JBoss ON itself, similar to the content management and repository systems in JBoss ON, or can be accessed through a given URL. This allows packages to be downloaded directly from GIT or SVN repositories, FTP sites, or other centralized locations. Either way, the provisioning system copies and unpacks a bundle archive file on the destination platform. The provisioning process has default commands that it uses to unpack each type of archive and deploy the bundle archive, depending on what is detected when the bundle is deployed. The Ant provisioning system can process both JAR and ZIP archive files.

5.1.2. About Destinations

A bundle is copied over, unzipped, and any templatized values are filled in on a specified platform. The place where a bundle is deployed is a *destination*. A destination has two parts: a group and a defined directory location.

JBoss ON groups are automatically linked to the bundles system as long as those groups only contain platforms. A bundle can only be deployed to a platform. The second part of the destination gives the specific directory where the application is deployed, and that provides some flexibility. A bundle can be deployed in a general directory like **`/opt`** or can be deployed in a JBoss AS container.

5.2. Creating Bundles

A bundle is made of two elements:

- A recipe file, which can either be an Ant recipe or a simple text file
- Associated bundle files, which can include archive files (JAR or ZIP files) or raw text configuration files

The bundle type is determined by its recipe type because the recipe file is the one interpreted by JBoss ON when the bundle is provisioned.

The Ant recipe is the one that will be used most commonly in production deployments because it offers better flexibility and control. File template bundles will probably be more common for testing or internal deployments because of their simplicity.

When a bundle is created, then either the recipe and all associated files can be loaded manually to the JBoss ON server, or the recipe and all associated files can be ziped into a single distribution zip file which can be uploaded or accessed through a URL. If there is a single distribution file, then the recipe file must be in the top directory and all files should be available in the locations they are referenced in in the recipe. This is illustrated in *Figure 5.2, “Bundle Layout”*, which has a recipe and associated files in a flat directory, packaged in **`distribution.zip`** (although the name doesn't matter).

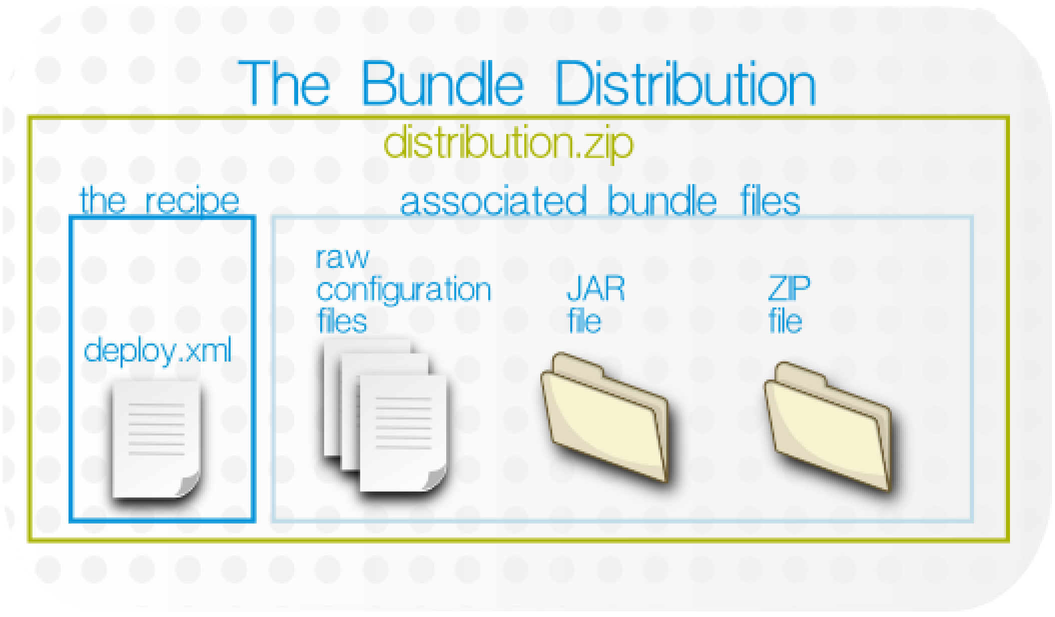


Figure 5.2. Bundle Layout

5.2.1. Creating an Associated Archive File

The application that is being deployed itself has to be built into an archive file of some kind. JBoss ON allows JAR and ZIP formats. The bundle archive file can also include raw files that are used to configuration the application, such as XML, **`.conf`**, and text files. These can be templatized to supply user- and system-specific information (as described in *Section 5.2.2, “Using Template Configuration Files in Bundles”*).

Any required archive or file must be referenced in the recipe so that the server knows to copy it during deployment.

The bundle files can be uploaded and stored in the JBoss ON server or they can be zipped up, with the recipe files, into a single distribution file.

5.2.2. Using Template Configuration Files in Bundles

A bundle can contain configuration files for an application. These configuration files can use hardcoded values or they can use *tokens* that are automatically realized when the bundle is actually deployed.

A token value can be supplied by a user or derived from the specific host system.

NOTE

For a user-defined token to be realized, it must be referenced in the recipe so that the bundle deployment wizard will prompt for the value. For an Ant recipe, this requires the **`<rhq:input-property>`** key.

User-defined tokens can be anything; the values are supplied through the provisioning UI and inserted into the templatized configuration file. The only restriction is that user-defined tokens must contain only alphanumeric characters, an underscore (_), or a period (.); no other characters are allowed. For example, to set a port number in a configuration file:

```
port=@@listener.port@@
```

The user-defined token then must be noted in the recipe, so that the provisioning process knows to realize the phrase. The provisioning wizard will prompt for a value for all of the user-defined tokens referenced in the recipe. To configure a property in an Ant recipe, add a **`<rhq:input-property>`** key in the Ant XML file.

For example:

```
<rhq:input-property
    name="listener.port"
    ... />
```

Along with user-defined variables that can be specified in the recipe file, there are variables that are made implicitly available to each recipe type. These tokens can be used in a templatized file as a user-defined variable without having to define the token template in the recipe itself.

Token	Description
rhq.deploy.dir	The directory location where the bundle will be installed.
rhq.deploy.id	A unique ID assigned to the specific bundle deployment.
rhq.deploy.name	The name of the bundle deployment.

Table 5.1. Variables Defined by JBoss ON

Additionally, some tokens can be realized by the provisioning process pulling information from the local system. These values, listed in *Table 5.2, “System-Defined Tokens”*, are taken either from the Java API or from Java system properties. They can be inserted directly in the templatized configuration file without having to put a corresponding entry in the recipe. For example:

```
@@rhq.system.hostname@@
```

Token Name	Taken From...	Java API
rhq.system.hostname	Java API	SystemInfo.getHostname()

Token Name	Taken From...	Java API
rhq.system.os.name	Java API	SystemInfo.getOperatingSystemName()
rhq.system.os.version	Java API	SystemInfo.getOperatingSystemVersion()
rhq.system.os.type	Java API	SystemInfo.getOperatingSystemType().toStrin
rhq.system.architecture	Java API	SystemInfo.getSystemArchitecture()
rhq.system.cpu.count	Java API	SystemInfo.getNumberOfCpus()
rhq.system.interfaces.java.address	Java API	InetAddress.getByName(SystemInfo.getHostn
rhq.system.interfaces.*network_adapter_name*.mac	Java API	NetworkAdapterInfo.getMacAddress()
rhq.system.interfaces.*network_adapter_name*.type	Java API	NetworkAdapterInfo.getType()
rhq.system.interfaces.*network_adapter_name*.flags	Java API	NetworkAdapterInfo.getAllFlags()
rhq.system.interfaces.*network_adapter_name*.address	Java API	NetworkAdapterInfo.getUnicastAddresses().ge
rhq.system.interfaces.*network_adapter_name*.multicast.address	Java API	NetworkAdapterInfo.getMulticastAddresses().g
rhq.system.sysprop.java.io.tmpdir	Java system property	
rhq.system.sysprop.file.separator	Java system property	
rhq.system.sysprop.line.separator	Java system property	
rhq.system.sysprop.path.separator	Java system property	
rhq.system.sysprop.java.home	Java system property	
rhq.system.sysprop.java.version	Java system property	
rhq.system.sysprop.user.timezone	Java system property	
rhq.system.sysprop.user.region	Java system property	
rhq.system.sysprop.user.country	Java system property	
rhq.system.sysprop.user.language	Java system property	

Table 5.2. System-Defined Tokens

5.2.3. Creating Ant Bundles

Most bundle deployments will use an Ant recipe bundle because of the inherent flexibility and extensibility of using the Ant build system to deploy bundles.

Section 5.2.3.1, “Requirements for an Ant Bundle” and *Section 5.2.3.2, “A Reference of JBoss ON Ant Recipe Elements”* both cover the details of creating an Ant recipe. The Ant recipe and bundle can be deployed using the JBoss ON server UI. There is also an additional tool that allows developers to test-deploy Ant bundles from the command line; using the tool can help automate building and testing custom software. This tool (described in *Section 5.3, “Testing Bundle Deployment Using the Command Line”*) is only intended for testing Ant bundles, which can be very complex.

5.2.3.1. Requirements for an Ant Bundle

The Ant recipe for JBoss ON bundles is the same basic file as a standard Apache Ant file and is processed by an integrated Ant build system in JBoss ON. This Ant recipe file must be bundled in the top directory of the distribution ZIP file and be named **`deploy.xml`**.

The JBoss ON Ant recipes allows all of the standard tasks that are available for Ant builds, which provides flexibility in scripting a deployment for a complex application. The JBoss ON Ant recipe *must* also provide additional information about the deployment that will be used by the provisioning process; this includes information about the destination and, essentially, metadata about the application itself.

Example 5.1, “Example Ant Recipe” shows a simplified recipe with several basic characteristics:

- A version number of 2.4
- Two required application files, an archive file called **`MyApp.zip`** and a configuration file called **`test-v2.properties`** (in the bundle)
- One user-defined token, **`listener.port`**
- An application log directory to be created in *appRoot*/**`logs`**

As with other Ant scripts, the JBoss ON Ant recipe uses a standard XML file with a **`<project>`** root element and defined targets and tasks. The elements defined in the **`<rhq:bundle>`** area pass metadata to the JBoss ON provisioning system when the project is built. For provisioning, the Ant recipe is more of a definition file than a true script file.

```
<?xml version="1.0"?>
<project name="test-bundle" default="main"
    xmlns:rhq="antlib:org.rhq.bundle">

    <rhq:bundle name="Example App" version="2.4" description="an example bundle">
        <rhq:input-property
            name="listener.port"
            description="This is where the product will listen for incoming messages"
            required="true"
            defaultValue="8080"
            type="integer"/>

        <rhq:deployment-unit name="appserver" preinstallTarget="preinstall"
 postinstallTarget="postinstall">
            <rhq:file name="test-v2.properties" destinationFile="subdir/test.properties"
 replace="true"/>
            <rhq:archive name="MyApp.zip">
                <rhq:replace>
                    <rhq:fileset>
                        <include name="**/*.properties"/>
                    </rhq:fileset>
                </rhq:replace>
            </rhq:archive>
            <rhq:ignore>
                <rhq:fileset>
                    <include name="logs/*.log"/>
                </rhq:fileset>
            </rhq:ignore>
        </rhq:deployment-unit>
    </rhq:bundle>

    <target name="main" />

    <target name="preinstall">
        <echo>Deploying Test Bundle v2.4 to ${rhq.deploy.dir}...</echo>
        <property name="preinstallTargetExecuted" value="true"/>
    </target>

    <target name="postinstall">
        <echo>Done deploying Test Bundle v2.4 to ${rhq.deploy.dir}.</echo>
        <property name="postinstallTargetExecuted" value="true"/>
    </target>
```

```
</project>
```

Example 5.1. Example Ant Recipe

Section 5.2.3.2, "A Reference of JBoss ON Ant Recipe Elements" explores the different JBoss ON elements in the Ant recipe file. For information on standard Ant tasks, see the *Apache Ant documentation*[1].

5.2.3.2. A Reference of JBoss ON Ant Recipe Elements

- *Section 5.2.3.2.1, "rhq:bundle"*
- *Section 5.2.3.2.2, "rhq:input-property"*
- *Section 5.2.3.2.3, "rhq:deployment-unit"*
- *Section 5.2.3.2.4, "rhq:archive"*
- *Section 5.2.3.2.5, "rhq:file"*
- *Section 5.2.3.2.6, "rhq:replace"*
- *Section 5.2.3.2.7, "rhq:ignore"*
- *Section 5.2.3.2.8, "rhq:fileset"*
- *Section 5.2.3.2.9, "rhq:system-service"*
- *Section 5.2.3.2.10, "Default Target"*
- *Section 5.2.3.2.11, "Pre-Install and Post-Install Targets"*

5.2.3.2.1. rhq:bundle

Contains the definition for the main JBoss ON-related Ant task that is required for any Ant bundle recipe. This element defines basic information about the bundle and is the parent element for all of the specific details about what is in the bundle and how it should be provisioned.

Element Attributes

Attribute	Description	Optional or Required
name	The name given to the bundle.	Required
version	The version string for this specific bundle. Bundles can have the same name, but each bundle of that name must have a unique version string. These version strings normally conform to an OSGi style of versioning, such as **`1.0`** or **`1.2.FINAL`**.	Required

[1] http://ant.apache.org/manual/using.html#buildfile

Attribute	Description	Optional or Required
description	A readable description of this specific bundle version.	Optional

Example

```
<rhq:bundle name="example" version="1.0" description="an example bundle">
```

5.2.3.2.2. rhq:input-property

Adds a property to the bundle task that defines a template token that must have its value supplied by a user at the time the bundle is deployed. This is similar to standard Ant properties.

NOTE

All of the system properties listed in *Table 5.2, "System-Defined Tokens"* and the Ant-specific tokens in *Table 5.1, "Variables Defined by JBoss ON"* are available to be used as templatized tokens in bundle configuration *without* having to set a **`<rhq:input-property>`** definition.

All input properties set some parameter that must have its value defined by a user when the bundle is provisioned on a platform, and the fields to enter those values are automatically generated in the JBoss ON UI bundle deployment wizard.

Element Attributes

Attribute	Description	Optional or Required
name	The name of the user-defined property. Within the recipe, this property can be referred to by this name, in the format **`${`***`property_name`***`}`**.	Required
description	A readable description of the property. This is the text string displayed in the JBoss ON bundle UI when the bundle is deployed.	Required
type	Sets the syntax accepted for the user-defined value. There are several different options: • string • longString • long • password • file	Required

Attribute	Description	Optional or Required
	• directory • boolean • integer • float • double	
required	Sets whether the property is required or optional for configuration. The default value is **`false`**, which means the property is optional. If this argument isn't given, then it is assumed that the property is optional.	Optional
defaultValue	Gives a value for the property to use if the user does not define a value when the bundle is deployed.	Optional

Example

```
<rhq:input-property
    name="listener.port"
    description="This is where the product will listen for incoming messages"
    required="true"
    defaultValue="8080"
    type="integer"/>
```

See Also

- *Section 5.2.3.2.4, “rhq:archive”*
- *Section 5.2.3.2.5, “rhq:file”*

5.2.3.2.3. rhq:deployment-unit

Defines the application being deployed by the bundle. A deployment unit is a software product, in any form, including an application server, web server, or database. A deployment unit (application) can have multiple archive and configuration files associated with it.

Only a single deployment unit is provisioned at a time by the provisioning process, so there can be only one **`<rhq:deployment-unit>`** element in a bundle recipe.

Element Attributes

Attribute	Description	Optional or Required
name	The name of the application.	Required

Attribute	Description	Optional or Required
preinstallTarget	An Ant target that is invoked before the deployment unit is installed.	Optional
postinstallTarget	An Ant target that is invoked after the deployment unit is installed.	Optional

Example

```
<rhq:deployment-unit name="appserver" preinstallTarget="preinstall"
 postinstallTarget="postinstall">
```

See Also

- *Section 5.2.3.2.11, "Pre-Install and Post-Install Targets"*

5.2.3.2.4. rhq:archive

Defines any archive file that is associated with deploying the application. An archive can be a ZIP or JAR file. A bundle doesn't require an archive file, so this element is optional.

Element Attributes

Attribute	Description	Optional or Required
name	The filename of the archive file to include in the bundle. **IMPORTANT** If the archive file is packaged with the Ant recipe file inside the bundle distribution ZIP file, then the **name** must contain the *relative path* to the location of the archive file in the ZIP file.	Required

Example

```
<rhq:archive name="file.zip">
```

```
    <rhq:replace>
         <rhq:fileset>
               <include name="**/*.properties"/>
         </rhq:fileset>
    </rhq:replace>
</rhq:archive>
```

See Also

- *Section 5.2.3.2.2, “rhq:input-property”*
- *Section 5.2.3.2.8, “rhq:fileset”*
- *Section 5.2.3.2.6, “rhq:replace”*

5.2.3.2.5. rhq:file

Contains the information to identify and process configuration files for the application which have token values that must be realized. Normally, configuration files are copied directly from the bundle package into the destination directory. The **`<rhq:file>`** element calls out files that require processing before they should be copied to the destination. The attributes on the **`<rhq:file>`** element set the name of the raw file in the bundle distribution ZIP file and the name of the target file that it should be copied to.

Raw files can be included with the archive files that contain properties or configuration for the application. These configuration files can be templatized with user-defined or system-defined tokens, like those listed in *Section 5.2.2, “Using Template Configuration Files in Bundles”*. Any templatized files that are included in the bundle distribution file that are templatized must be listed in the Ant recipe so that they are processed and the tokens are realized.

Element Attributes

Attribute	Description	Optional or Required
name	The name of the raw configuration file. **IMPORTANT** If the configuration file is packaged with the Ant recipe file inside the bundle distribution ZIP file, then the **name** must contain the *relative path* to the location of the file within the ZIP file.	Required

Attribute	Description	Optional or Required
destinationFile	The full path and filename for the file on the destination platform. Relative paths must be relative to the final destination directory (defined in the **`rhq.deploy.dir`** parameter when the bundle is deployed). It is also possible to use absolute paths, as long as both the directory and the filename are specified.	Required
replace	Indicates whether the file is templatized and requires additional processing to realize the token values.	Required

Example

```
<rhq:file name="test-v2.properties" destinationFile="subdir/test.properties" replace="true"/>
```

5.2.3.2.6. rhq:replace

Lists templatized files, in children **`<rhq:fileset>`** elements, contained in the archive which need to have token values realized when the archive is deployed.

Any file which uses a token that must be replaced with a real value is a templatized file. When the provisioning process runs, the token value is substituted with the defined value. This element lists all of the files which are templatized; the only files which are processed by the provisioning system for token substitution are the ones lised in the **`<rhq:replace>`** element.

Example

```
<rhq:archive name="file.zip">
    <rhq:replace>
        <rhq:fileset>
            <include name="**/*.properties"/>
        </rhq:fileset>
    </rhq:replace>
</rhq:archive>
```

See Also

- *Section 5.2.3.2.8, “rhq:fileset”*
- *Section 5.2.3.2.4, “rhq:archive”*

5.2.3.2.7. rhq:ignore

Lists files from a previous bundle deployment that should be ignored when the bundle is deployed.

Once an application is deployed, instance-specific files — like data files or logs — can be created and should be retained if the application is ever upgraded. This element, much like **`<rhq:replace>`**, contains a list of files or directories in the instance to save.

NOTE

If a file is ignored in the recipe, then the file is left unchanged. *Never* ignore files packaged in the bundle. Only files generated by the applocations, such as log and data files, should be ignored by the provisioning process since they should be preserved for the upgraded instance.

Example

```
<rhq:ignore>
    <rhq:fileset>
        <include name="logs/*.log"/>
    </rhq:fileset>
</rhq:ignore>
```

See Also

- *Section 5.2.3.2.8, “rhq:fileset”*

5.2.3.2.8. rhq:fileset

Provides a list of files.

Two JBoss ON elements — **`<rhq:replace>`** and **`<rhq:ignore>`** — define file lists in either the archive file or the destination directory. This element contains the list of files.

Child Element

Child Element	Description
<include name=*filename* />	The filename of the file. For **`<rhq:replace>`**, this is a file within the archive (JAR or ZIP) file which is templatized and must have its token values realized. For **`<rhq:ignore>`**, this is a file in the application's deployment directory which should be ignored and preserved when the bundle is upgraded.

Example

```
<rhq:replace>
     <rhq:fileset>
         <include name="**/*.properties"/>
     </rhq:fileset>
</rhq:replace>
```

See Also

- *Section 5.2.3.2.7, “rhq:ignore”*
- *Section 5.2.3.2.6, “rhq:replace”*

5.2.3.2.9. rhq:system-service

Points to a script file to launch as part of the provisioning process. This is usually an init file or similar file that can be used by the deployed application to set the application up as a system service.

Element Attributes

Attribute	Description	Optional or Required
name	The name of the script.	Required
scriptFile	The filename of the script. If the script file is packaged with the Ant recipe file inside the bundle distribution ZIP file, then the **`scriptFile`** must contain the *relative path* to the location of the file in the ZIP file.	Required
configFile	The name of any configuration or properties file used by the script. If the configuration file is packaged with the Ant recipe file inside the bundle distribution ZIP file, then the **`configFile`** must contain the *relative path* to the location of the file in the ZIP file.	Optional
overwriteScript	Sets whether to overwrite any existing init file to configure the application as a system service.	Optional
startLevels	Sets the runlevel for the application service.	Optional
startPriority	Sets the start order or priority for the application service.	Optional
stopPriority	Sets the stop order or priority for the application service.	Optional

Example

```
<rhq:system-service name="example-bundle-init" scriptFile="example-init-script"
      configFile="example-init-config" overwriteScript="true"
      startLevels="3,4,5" startPriority="80" stopPriority="20"/>
```

5.2.3.2.10. Default Target

As with other Ant tasks, the **`<project>`** allows a default target, which is required by the provisioning system. This target can be used to do something for the application or can be a no-op. The most common thing is for the default target to be a no-op because the Ant recipe mainly defines the metadata for and identifies files used by the provisioning process. Other operations aren't necessary.

```
<target name="main" />
```

5.2.3.2.11. Pre-Install and Post-Install Targets

JBoss ON provisioning tasks, like other Ant tasks, can define both pre- and post-install targets. This allows custom tasks, like simple progress messages or setting properties.

5.2.3.3. Upgrading Ant Bundles

The bundle upgrade process decides whether to upgrade (meaning, overwrite) files within the application's deployment directory by comparing the MD5 hashcodes on the files. There are several different upgrade scenarios:

- If the hashcode on the new file is different than the original file and there are no local modifications, then JBoss ON installs the new file over the existing file.
- If the hashcode on the new file is different than the original file and there *are* local modifications, then JBoss ON backs up the original file and installs the new file.
- If the hashcode on the new file and the original file is the same and there *are* local modifications on the original file, then the provisioning process preserves the original file, in place.
- If there was no file in the previous bundle but there is one in the new bundle, then the new file is used and any file that was added manually is backed up.

Backed up files are saved to a **`backup/`** directory within the deployment's destination directory. If the original file was located outside the application's directory (like, it was stored according to an absolute location rather than a relative location), then it is saved in an **`ext-backup/`** directory within the deployment's destination directory.

NOTE

If a file is ignored in the recipe, then the file is left unchanged. *Never* ignore files packaged in the bundle. Only files generated by the applocations, such as log and data files, should be ignored by the provisioning process since they should be preserved for the upgraded instance.

If a completely fresh installation is required, then it is possible to run a clean deployment. This is described in *Section 5.8, “Deploying a Bundle to a Clean Destination”*.

5.3. Testing Bundle Deployment Using the Command Line

Ant recipes can be complex, so it's important (and useful) to test a bundle before deploying it. JBoss ON includes a command-line tool that can be used to test Ant provisioning bundles quickly.

5.3.1. Installing the Bundle Deployer Tool

This tool can be downloaded and installed on any machine, independent of any JBoss ON server or agent.

1. Click the **Administration** link in the main menu.
2. Select the **Downloads** menu item.
3. Scroll to the **Bundle Deployer Download** section, and click **Download Bundle Deployer**.
4. Save the **`.zip`** file into the directory where the bundle tool should be installed. For example:

   ```
   /opt/
   ```

5. Unzip the packages.

   ```
   cd /opt/
   unzip rhq-bundle-deployer-version.zip
   ```

5.3.2. Using the Bundle Deployer Tool

IMPORTANT

This bundle deployment tool is *only* to test the provisioning process and deployed application. This tool does not interact with the JBoss ON server or agent, so JBoss ON is unaware of any applications deployed with this tool and cannot manage them.

1. Unzip the bundle distribution package to check (or copy an unzipped directory that contains the application files). For example:

   ```
   mkdir /tmp/test-bundle
   cd /tmp/test-bundle
   unzip MyBundle.zip
   ```

2. Open the top directory of the bundle distribution, where the **`deploy.xml`** Ant recipe file is.
3. Set the bundle deployer tool location in the PATH.

   ```
   PATH="/opt/rhq-bundle-deployer-3.0.0/bin:$PATH"
   ```

4. Run the bundle deploy tool, and use the format **`-D`***input_properties* to pass the values to user-defined tokens in the templatized files. For example:

   ```
   rhq-ant -Drhq.deploy.dir=/opt/exampleApp -Dlistener.port=7081
   ```

   This installs the application in **`/opt/exampleApp`** and sets a port value of 7081.

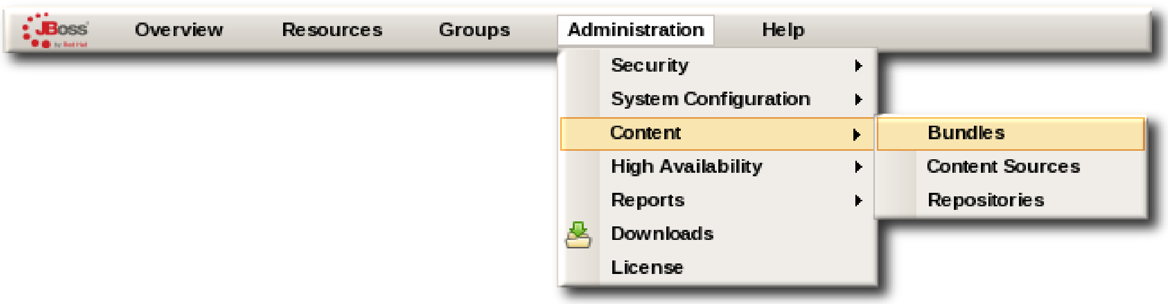
5.4. Uploading Bundles

All of the files associated with a distribution — the recipe, any JARs or ZIPs, and any configuration files — have to be accessible to JBoss ON. Either the files need to be uploaded and stored in the JBoss ON database or a URL to the packages needs to be configured.

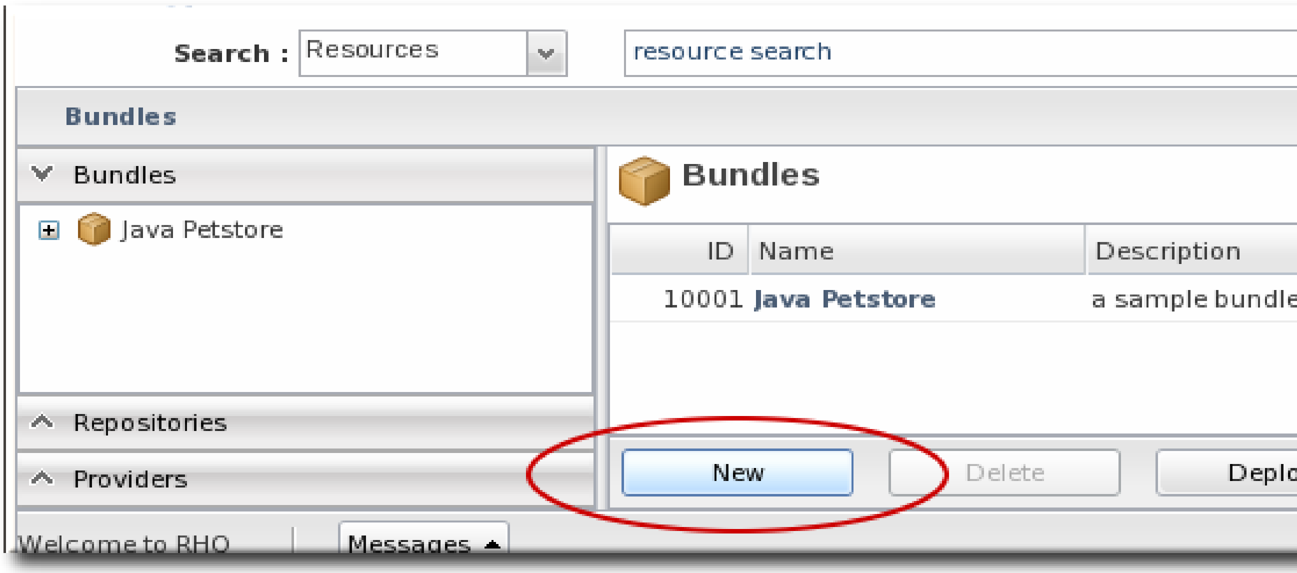
NOTE

If the files are all combined in a single ZIP file to upload, then the recipe file must be in the top level of the package.

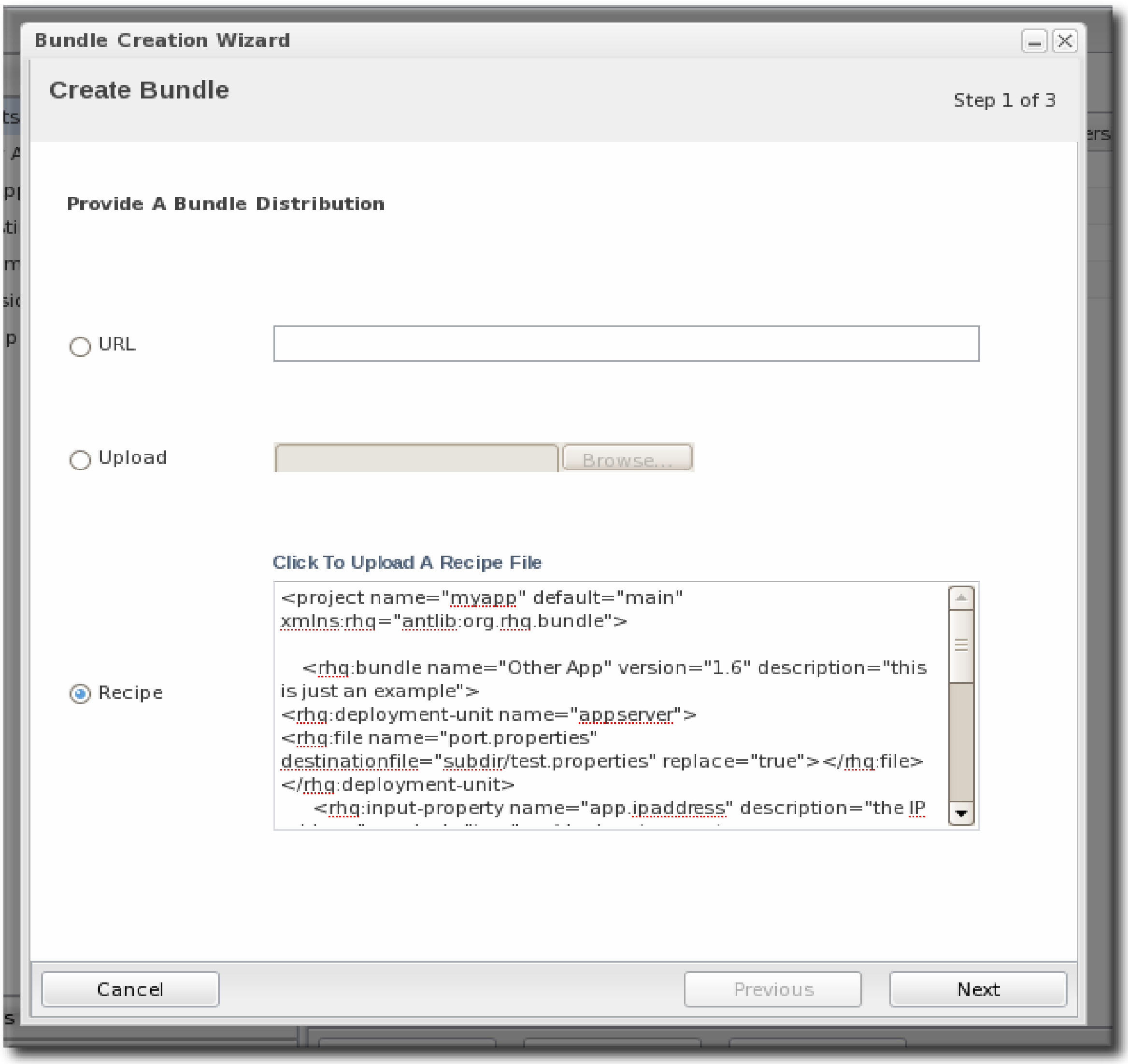
1. In the top menu, click the **Administration** tab, open the **Content** menu, and select the **Bundles** item.



2. Scroll to the bottom of the window and click the **New** button.



3. Upload the distribution package or the recipe file.



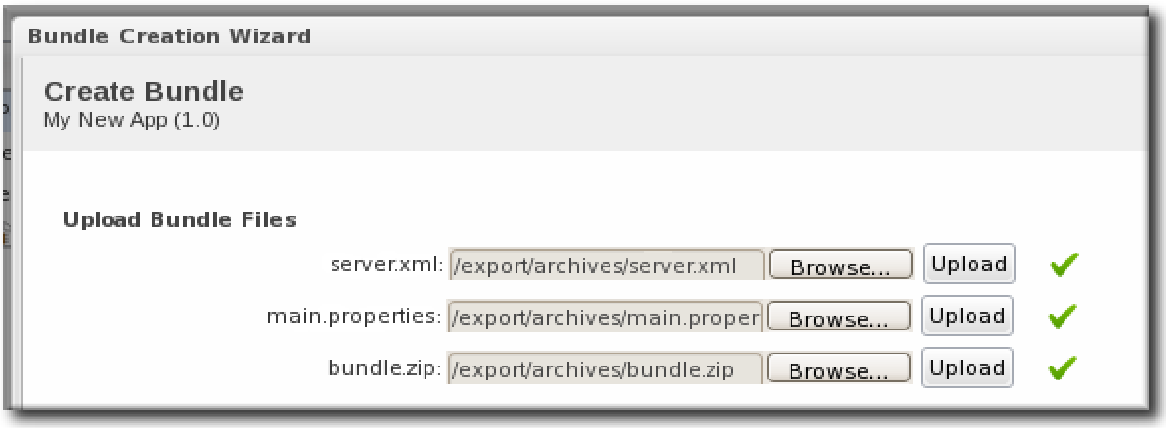
```
<project name="myapp" default="main"
xmlns:rhq="antlib:org.rhq.bundle">

   <rhq:bundle name="Other App" version="1.6" description="this
is just an example">
<rhq:deployment-unit name="appserver">
<rhq:file name="port.properties"
destinationfile="subdir/test.properties" replace="true"></rhq:file>
</rhq:deployment-unit>
     <rhq:input-property name="app.ipaddress" description="the IP
```

There are three options on how the bundle distribution is made available to the JBoss ON server:

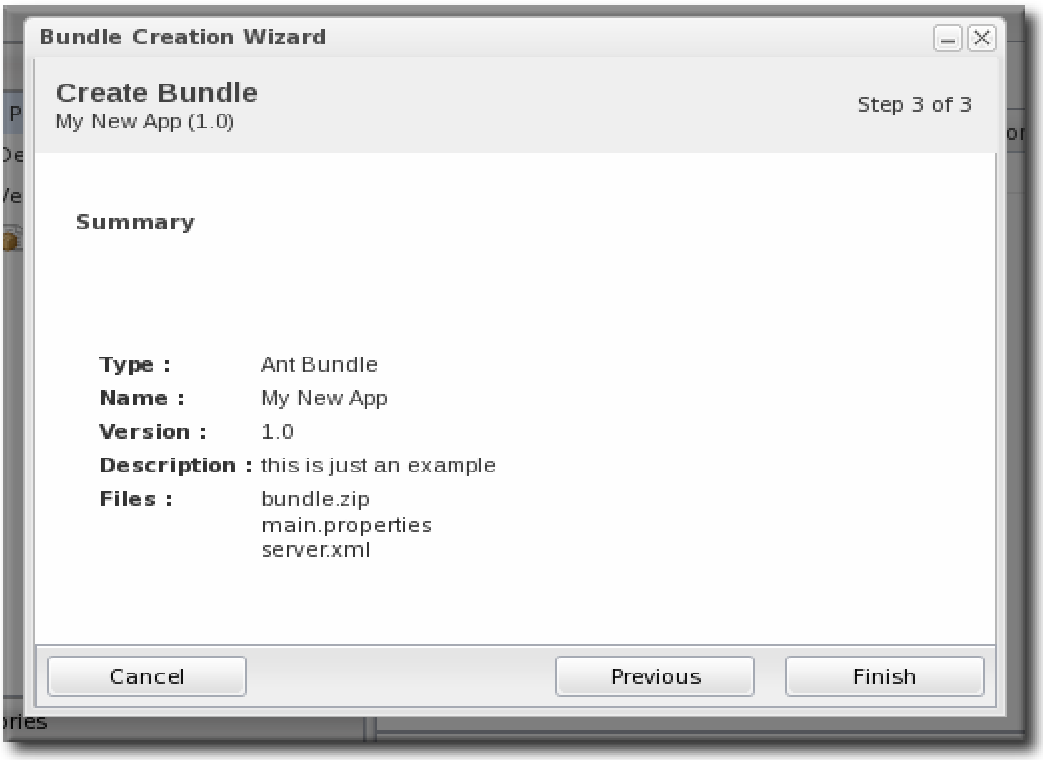
- **URL** points to any URL, such as an FTP site or GIT repo, where there is a complete bundle distribution file available.
- **Upload** uploads a single bundle distribution file (which includes both the recipe an all associated files) from the local system to the JBoss ON server.
- **Recipe** uploads a recipe file only, and then any additional files required for the bundle are uploaded separately. This option includes an edit field where the uploaded recipe can be edited before it is sent to the server.

4. In the next screen, upload any associated files that were not uploaded previously. For the **URL** and **Upload**, all of the files are usually uploaded in a single file, so there is nothing to do on this screen. For the **Recipe** option, all of the files listed in the recipe must be uploaded manually at this step.



server.xml: /export/archives/server.xml
main.properties: /export/archives/main.proper
bundle.zip: /export/archives/bundle.zip

5. The final screen shows all of the information for the new bundle. Click **Finish** to save the new bundle.

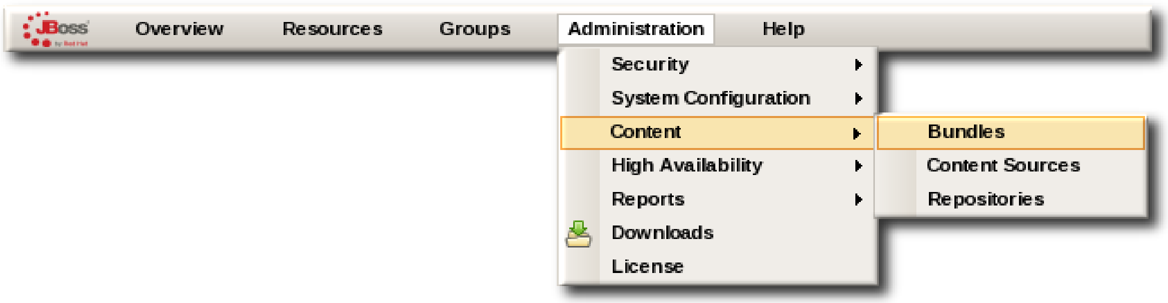


Type :	Ant Bundle
Name :	My New App
Version :	1.0
Description :	this is just an example
Files :	bundle.zip main.properties server.xml

5.5. Deploying Bundles to a Platform

Bundles are deployed on a platform by deploying the bundle to a JBoss ON group. Any group that contains only platforms is automatically listed as an option for the destination. The groups can be mixed, too; the provisioning process will automatically format the destination directory and provisioned files to match the platform's architecture.

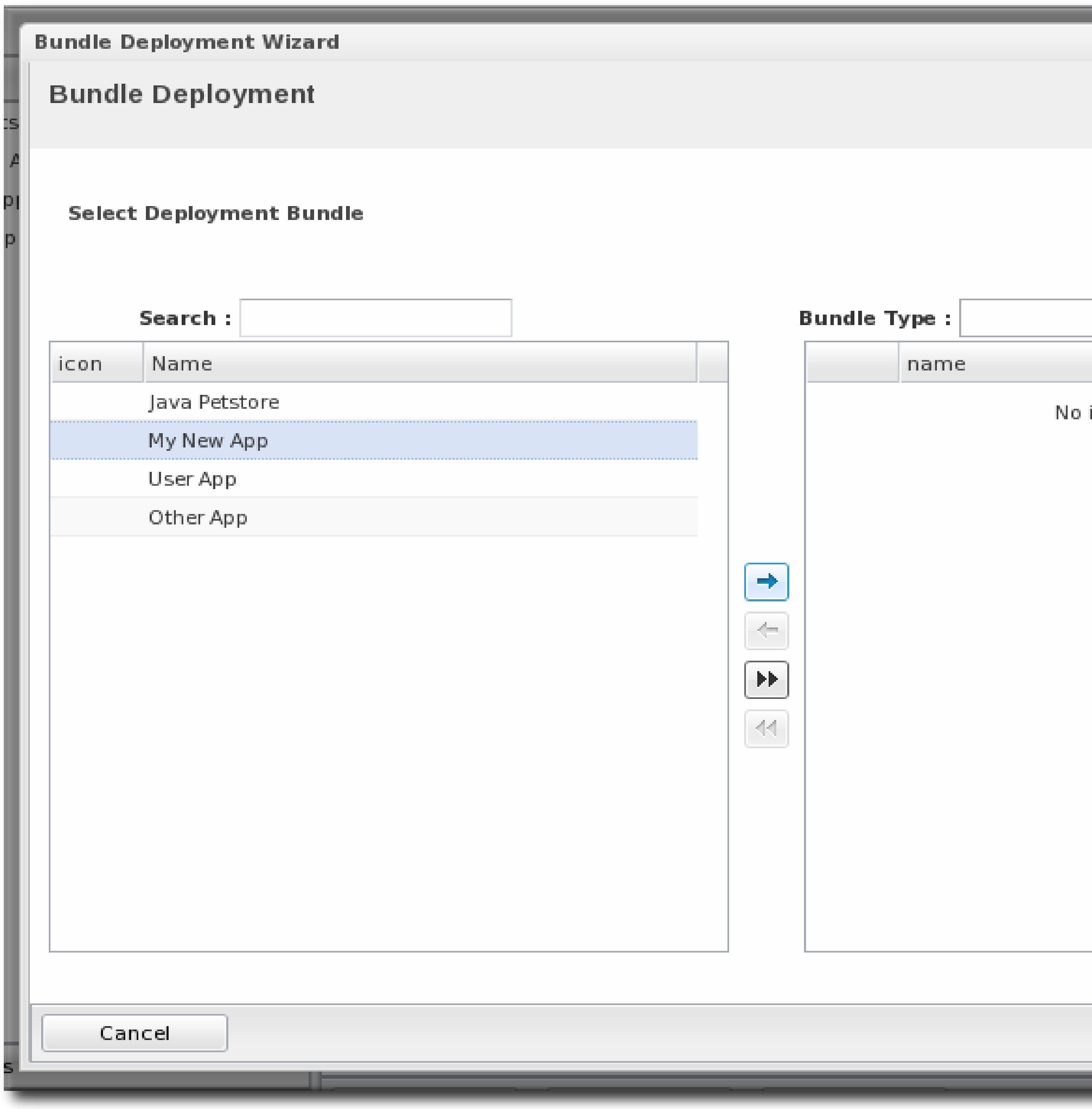
1. In the top menu, click the **Administration** tab, open the **Content** menu, and select the **Bundles** item.



2. Scroll to the bottom of the window and click the **Deploy** button.

   Alternatively, click the name of the bundle in the list, and then click the deploy button at the top of the bundle page.

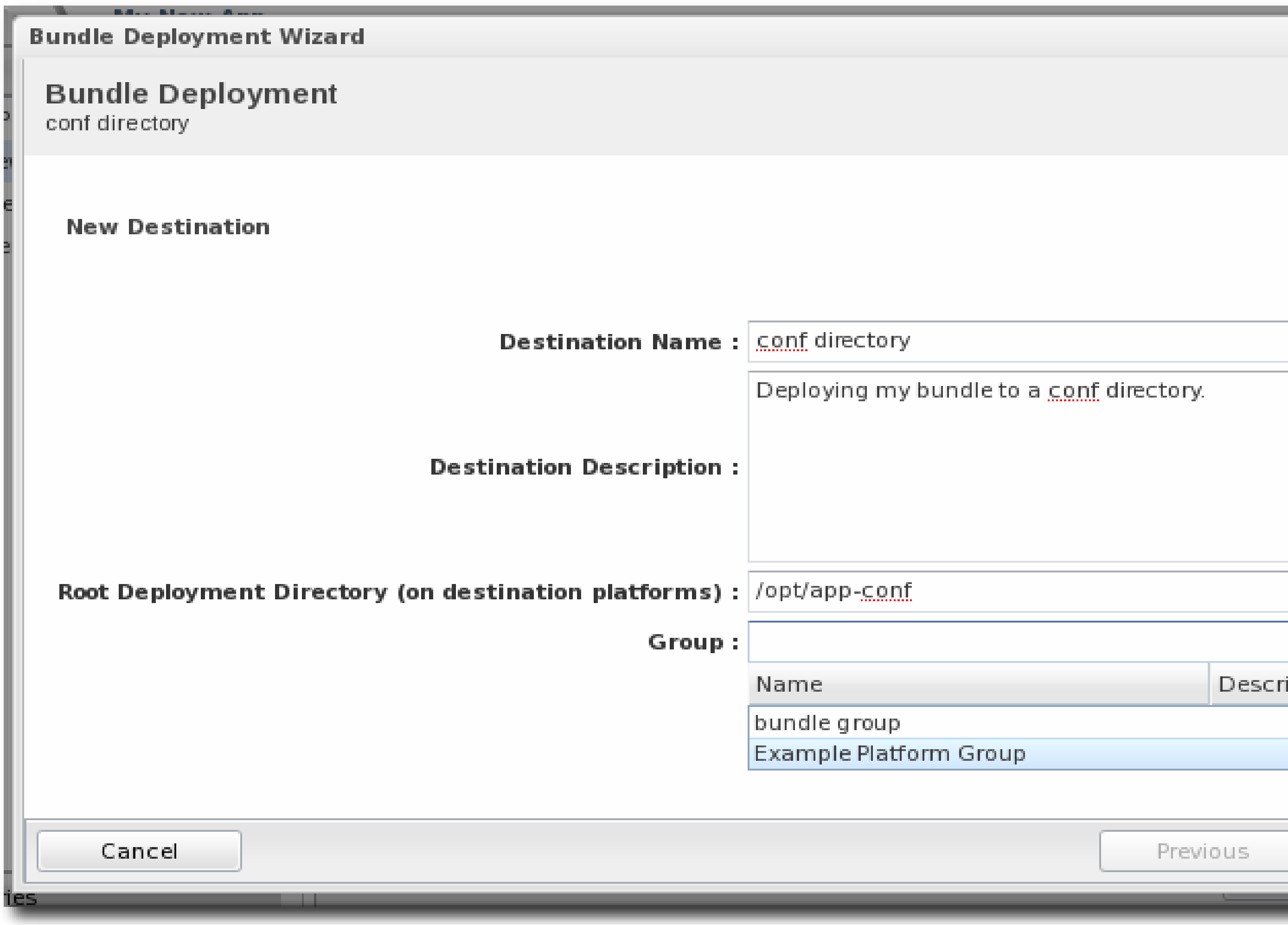
3. Select the bundles to deploy from the list on the left and use the arrows to move them to the box on the right.



Java Petstore
My New App
User App
Other App

4. Once the bundles are selected, define the destination information.

   The destination is a combination of the platforms the bundle is deployed on and the directory to which is it deployed. Each destination is uniquely defined for each bundle. To define the destination, set the deployment directory in the **Root Deployment Directory** field and select the group from the **Group** drop-down menu.



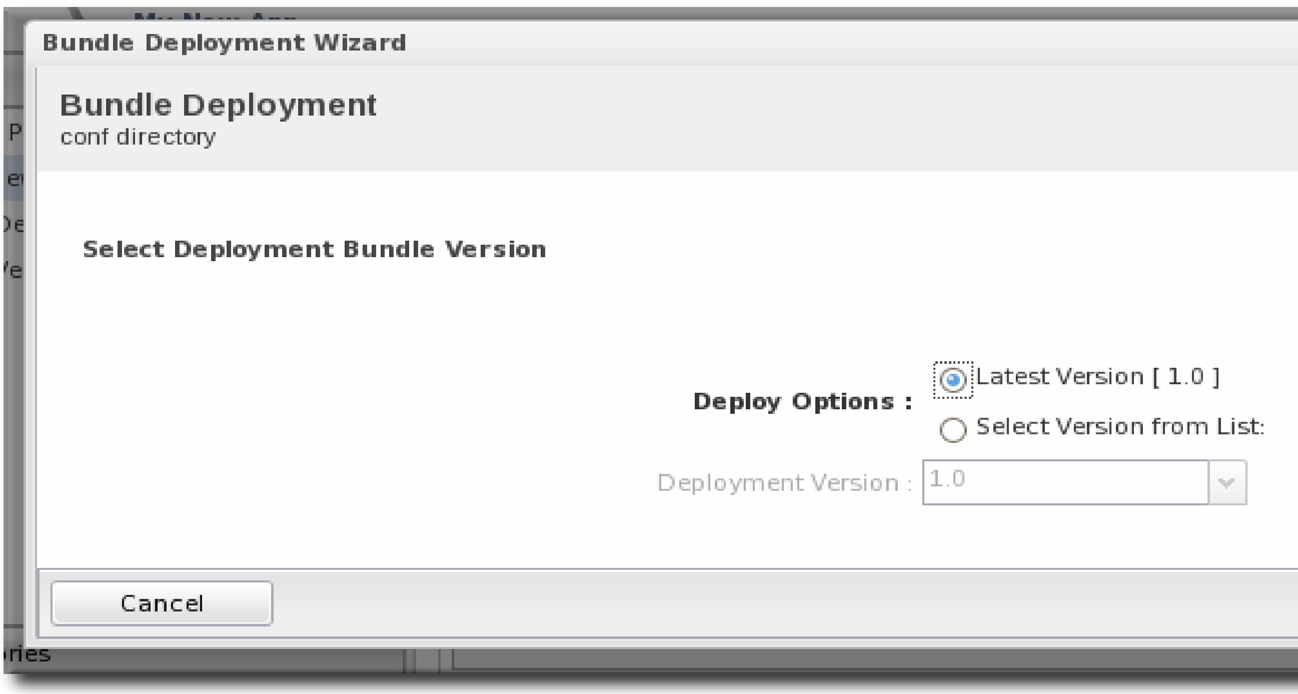
Bundle Deployment
conf directory

New Destination

Destination Name :	conf directory
Destination Description :	Deploying my bundle to a conf directory.
Root Deployment Directory (on destination platforms) :	/opt/app-conf
Group :	

Name	Descri
bundle group	
Example Platform Group	

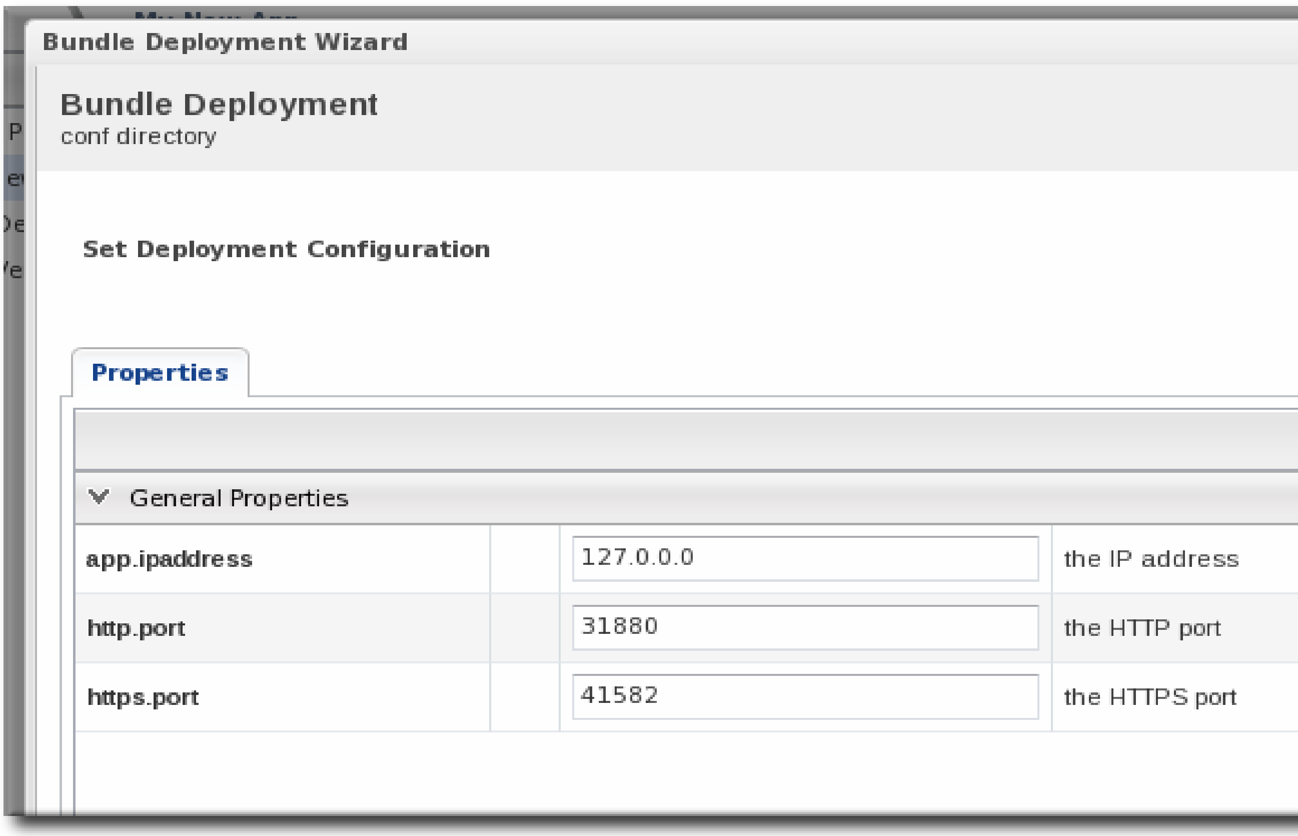
5. Select the version of the bundle to deploy. If there are multiple versions of a bundle available, then any of those versions can be selected. There are also quick options to deploy the latest version or the currently deployed version.



Deploy Options : [✓ Latest Version [1.0]] Select Version from List:

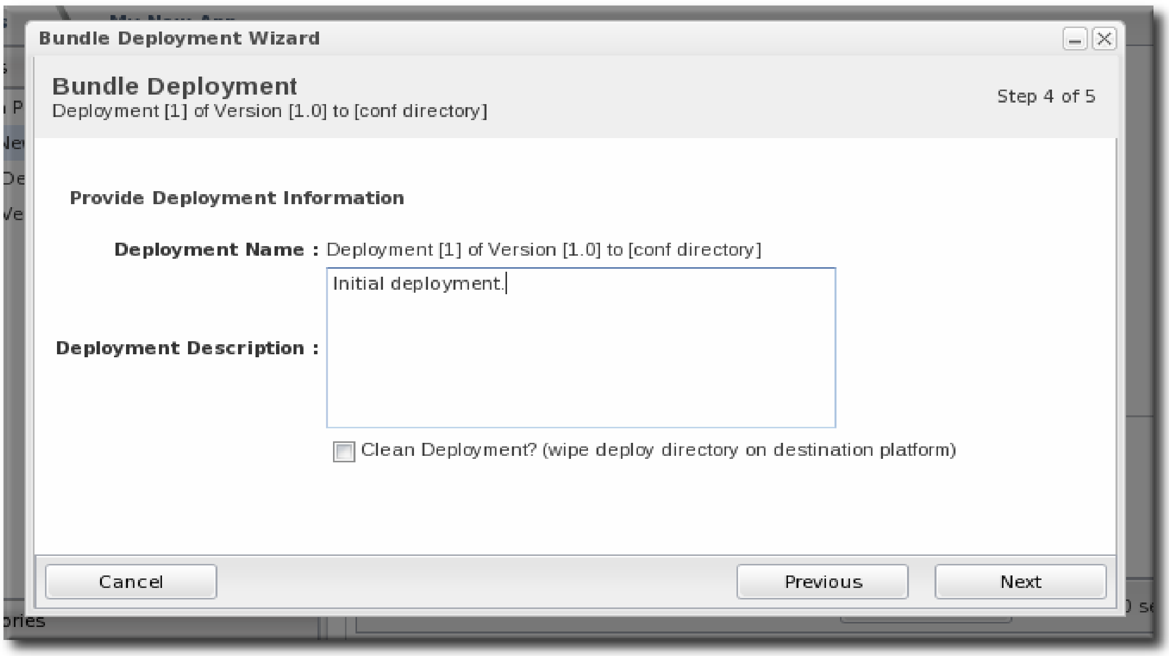
Deployment Version : 1.0

6. If there are any user-defined properties, then they are entered in the fields in the next page. User-defined properties are configured in the bundle recipe using tokens.



app.ipaddress		127.0.0.0	the IP address
http.port		31880	the HTTP port
https.port		41582	the HTTPS port

7. Fill in the information about the specific deployment instance. The checkbox sets the option on whether to overwrite anything in the exisitng destination directory or whether to preserve any existing files.



Bundle Deployment
Deployment [1] of Version [1.0] to [conf directory]
Step 4 of 5

Provide Deployment Information

Deployment Name : Deployment [1] of Version [1.0] to [conf directory]

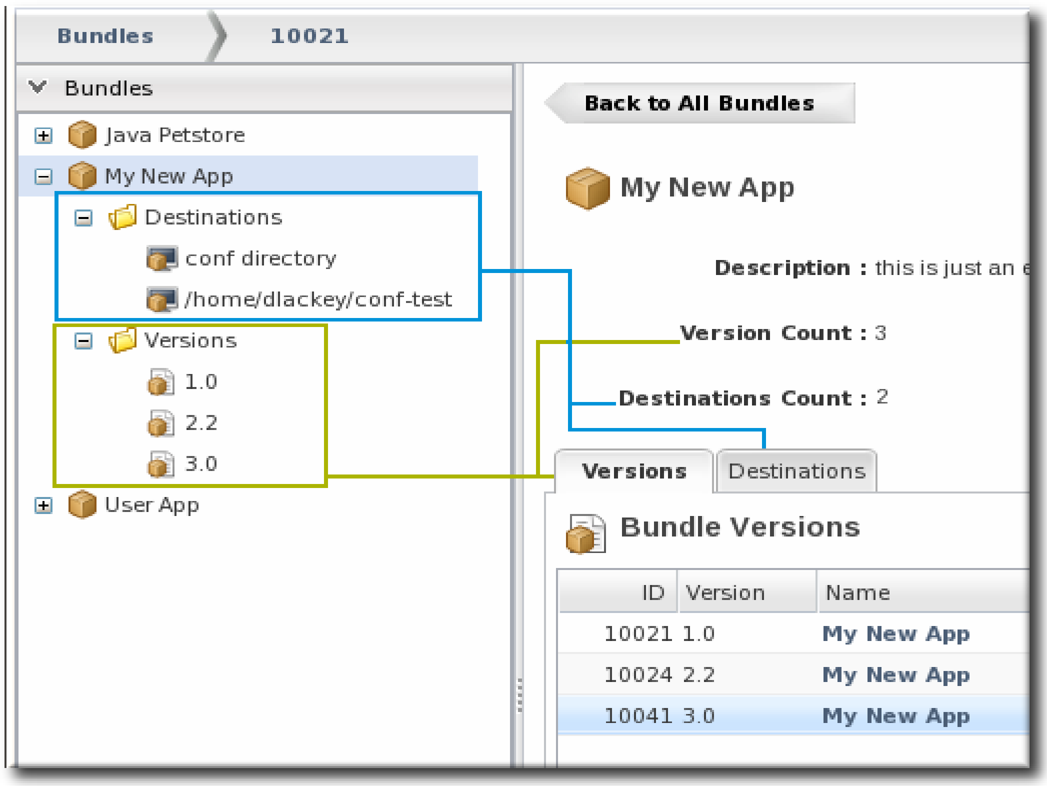
Deployment Description : Initial deployment.

Clean Deployment? (wipe deploy directory on destination platform)

8. The final screen shows the progress for deploying the packages. Click **Finish** to complete the deployment.

5.6. Viewing the Bundle Deployment History

A bundle has two areas of information: one for its versions and one for its destinations (places where it is deployed). The main bundle entry shows only those two things, the versions and the destinations. The version area is a way to track and control the *content of the bundle*, while the destinations area is a way to track and control *the process of deploying bundles*.



Description : this is just an e

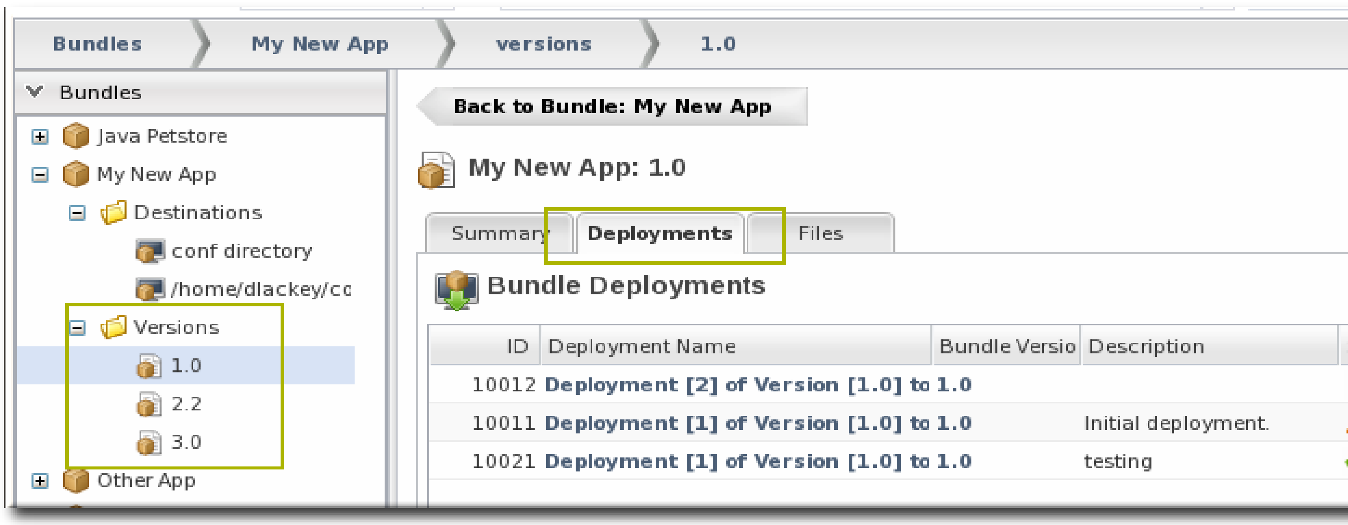
Version Count : 3

Destinations Count : 2

ID	Version	Name
10021	1.0	My New App
10024	2.2	My New App
10041	3.0	My New App

Figure 5.3. Bundles, Versions, and Destinations

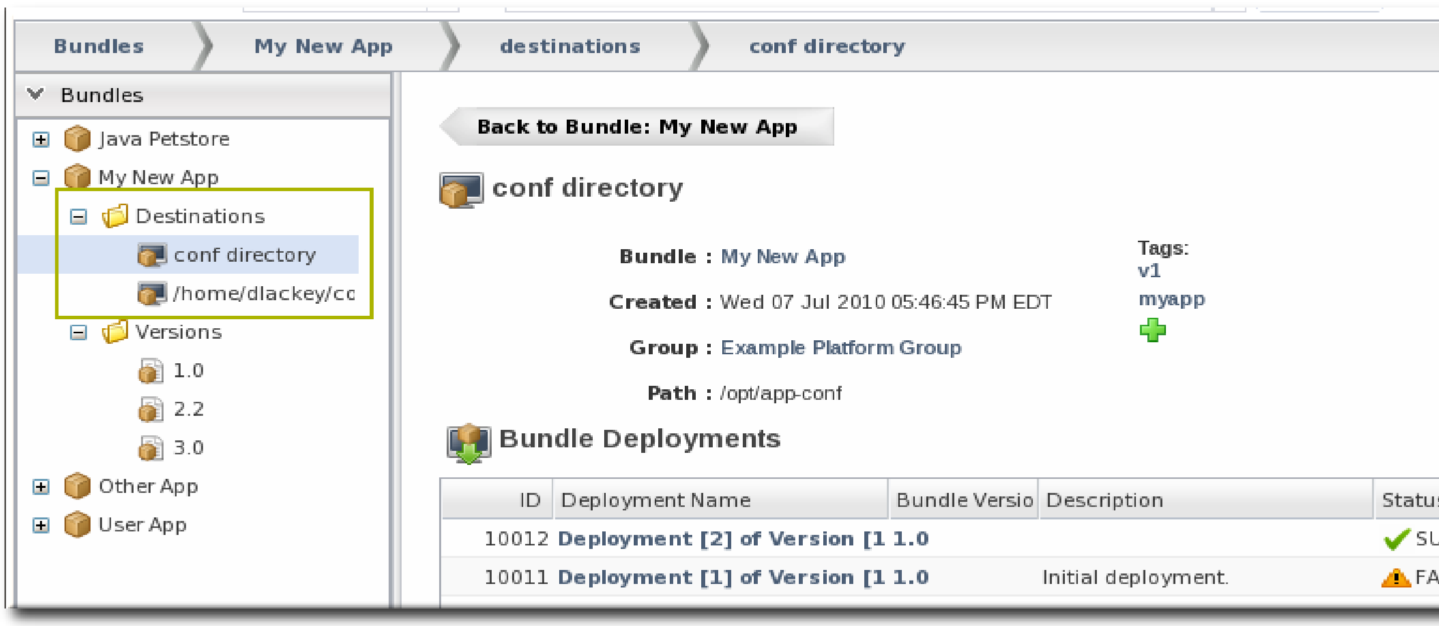
Selecting a version under the main bundle entry shows its recipe (on the **Summary** tab) and a list of all of the files associated with that particular version (on the **Files** tab). The **Deployments** tab shows every destination, with timestamps and comments, that that particular version of the bundle has been deployed to.



ID	Deployment Name	Bundle Versio	Description
10012	Deployment [2] of Version [1.0] to	1.0	
10011	Deployment [1] of Version [1.0] to	1.0	Initial deployment.
10021	Deployment [1] of Version [1.0] to	1.0	testing

Figure 5.4. Deployment Information for a Version

A destination entry shows only a list of versions that have been deployed to that destination. In a sense, the destination area is the best areas to track the audit history of an application. Along with shows the history of deployments and updates, the destinations area is the place where new versions can be deployed or reverted most directly.



conf directory

Bundle : My New App
Created : Wed 07 Jul 2010 05:46:45 PM EDT
Group : Example Platform Group
Path : /opt/app-conf

Tags:
v1
myapp

Bundle Deployments

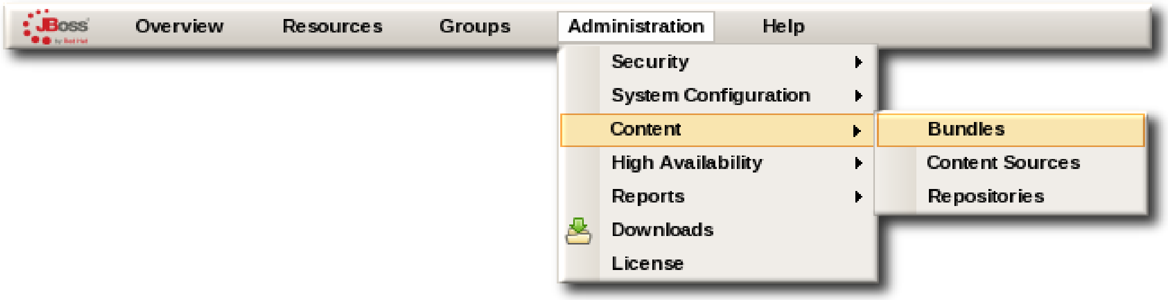
ID	Deployment Name	Bundle Versio	Description	Statu
10012	Deployment [2] of Version [1	1.0		✓ SU
10011	Deployment [1] of Version [1	1.0	Initial deployment.	⚠ FA

Figure 5.5. Deployment History for a Destinations

5.7. Reverting a Deployed Bundle

Ant bundles can be rolled back to a previous version number or a previous deployment of that bundle. This provides some extra protection and flexiblity when deploying and managing applications, particularly for testing and production systems.

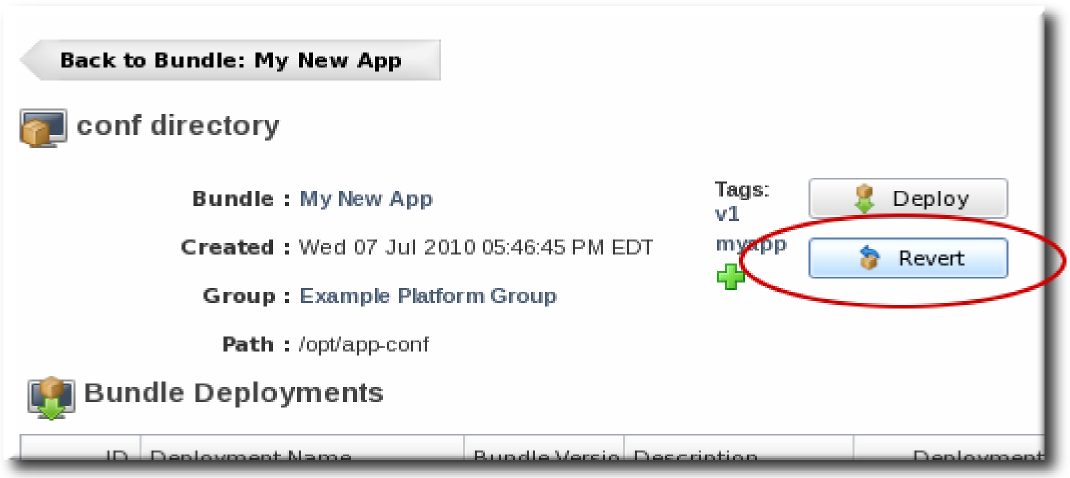
1. In the top menu, click the **Administration** tab, open the **Content** menu, and select the **Bundles** item.



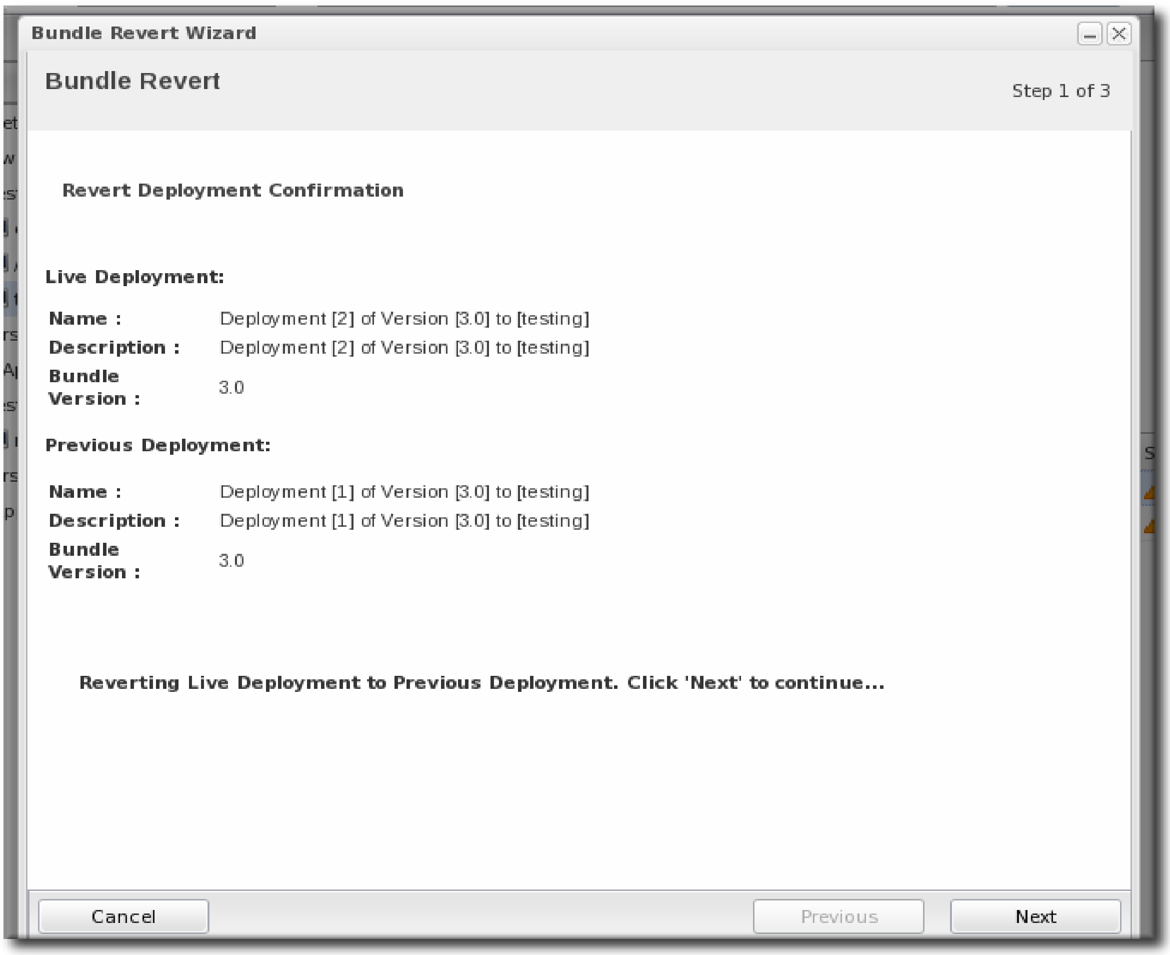
2. In the left navigation window, expand the bundle node, and then open the **Destinations** folder beneath it.

3. Select the destination from the left navigation.

4. In the main window for the destination, click the **Revert** button.


Back to Bundle: My New App
conf directory
Bundle : My New App
Created : Wed 07 Jul 2010 05:46:45 PM EDT
Group : Example Platform Group
Path : /opt/app-conf
Tags: v1 myapp
Deploy
Revert
Bundle Deployments

5. The next page shows the summary of the current deployment and the immediate previous deployment, which it will be reverted to.


Bundle Revert Wizard

Bundle Revert

Step 1 of 3

Revert Deployment Confirmation

Live Deployment:

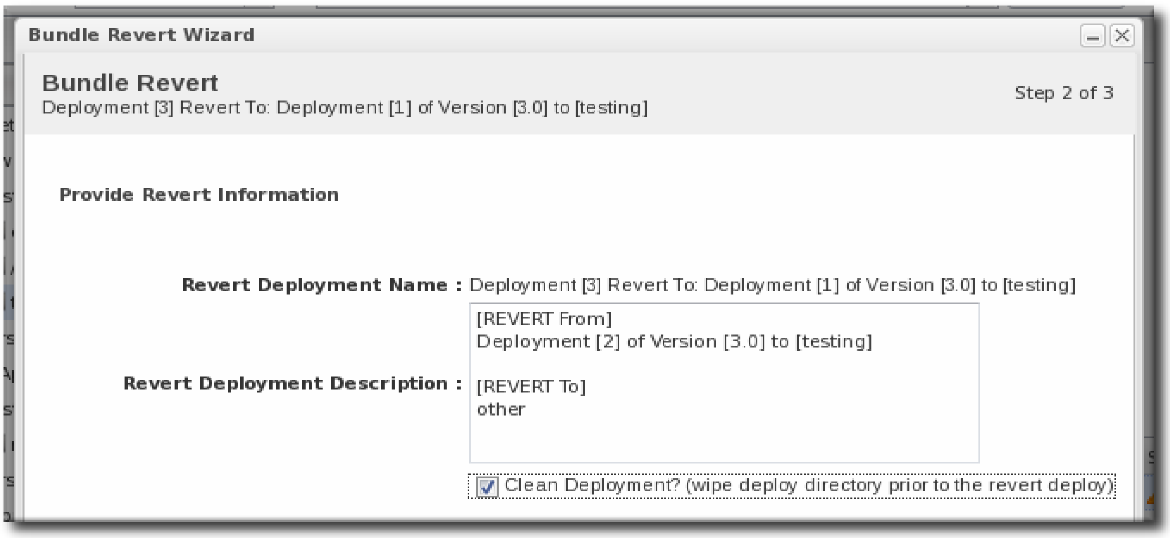
Name :	Deployment [2] of Version [3.0] to [testing]
Description :	Deployment [2] of Version [3.0] to [testing]
Bundle Version :	3.0

Previous Deployment:

Name :	Deployment [1] of Version [3.0] to [testing]
Description :	Deployment [1] of Version [3.0] to [testing]
Bundle Version :	3.0

Reverting Live Deployment to Previous Deployment. Click 'Next' to continue...

6. Add any notes to the revert action. Optionally, select the checkbox to clean the destination directory and install the previous version fresh.


Bundle Revert Wizard

Bundle Revert
Deployment [3] Revert To: Deployment [1] of Version [3.0] to [testing]

Step 2 of 3

Provide Revert Information

Revert Deployment Name : Deployment [3] Revert To: Deployment [1] of Version [3.0] to [testing]

Revert Deployment Description : [REVERT From]
Deployment [2] of Version [3.0] to [testing]

[REVERT To]
other

☑ Clean Deployment? (wipe deploy directory prior to the revert deploy)

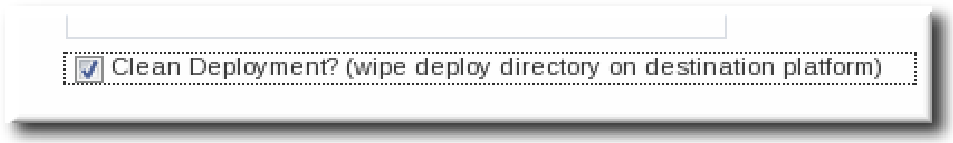
7. Click **Finish** on the final screen to complete the rollback.

5.8. Deploying a Bundle to a Clean Destination

A bundle can be deployed to a destination where there may already be an application, files, or even a previous bundle deployment. When deploying a new bundle, there are two options for how the provisioning process handles the

- Preserve the existing files and directories, with appropriate upgrades
- Completely overwrite the existing files and deploy the bundle cleanly

To deploy the bundle in a clean directory, then select the **Clean Deploy** checkbox when running through the deployment wizard in *Section 5.5, “Deploying Bundles to a Platform”*.


☑ Clean Deployment? (wipe deploy directory on destination platform)

5.9. Tagging Bundles

Most entries in the bundles area have a **Tags** label. The tag on these entries provides an ad hoc and flexible way to group entries by any user-defined terms: version numbers, keywords, application names.

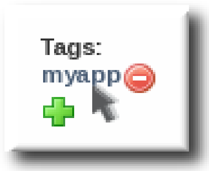
Any bundle-related entry can be tagged:

- Main bundle entries
- Individual versions, by tagging the recipe
- Individual destinations
- Specific deployments

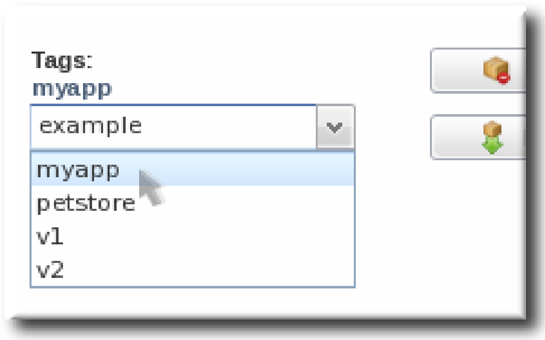
5.9.1. Adding Tags

1. Open the bundles entry to tag.

2. Click the cross icon under the **Tags** label.



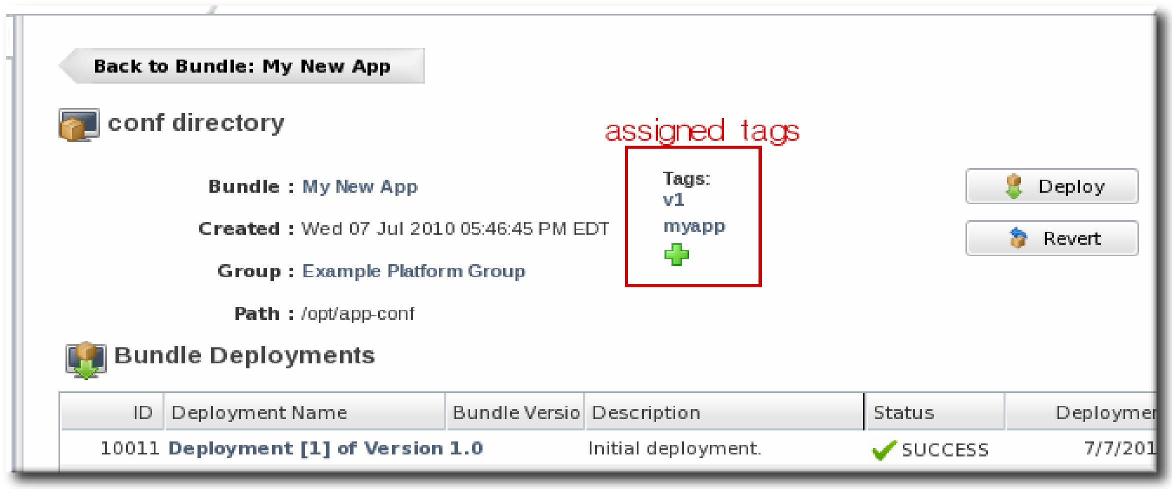
3. Select a tag from the drop-down menu or type in the text for a new label.



4. Hit `Enter` to apply the new tag.

5.9.2. Viewing Tags

The tags for a single entry are listed at the top of the entry's display page, in the **Tags** area. (On a version, the tags are listed on the **SUmmary** tab.)



conf directory

Bundle : My New App

Created : Wed 07 Jul 2010 05:46:45 PM EDT

Group : Example Platform Group

Path : /opt/app-conf

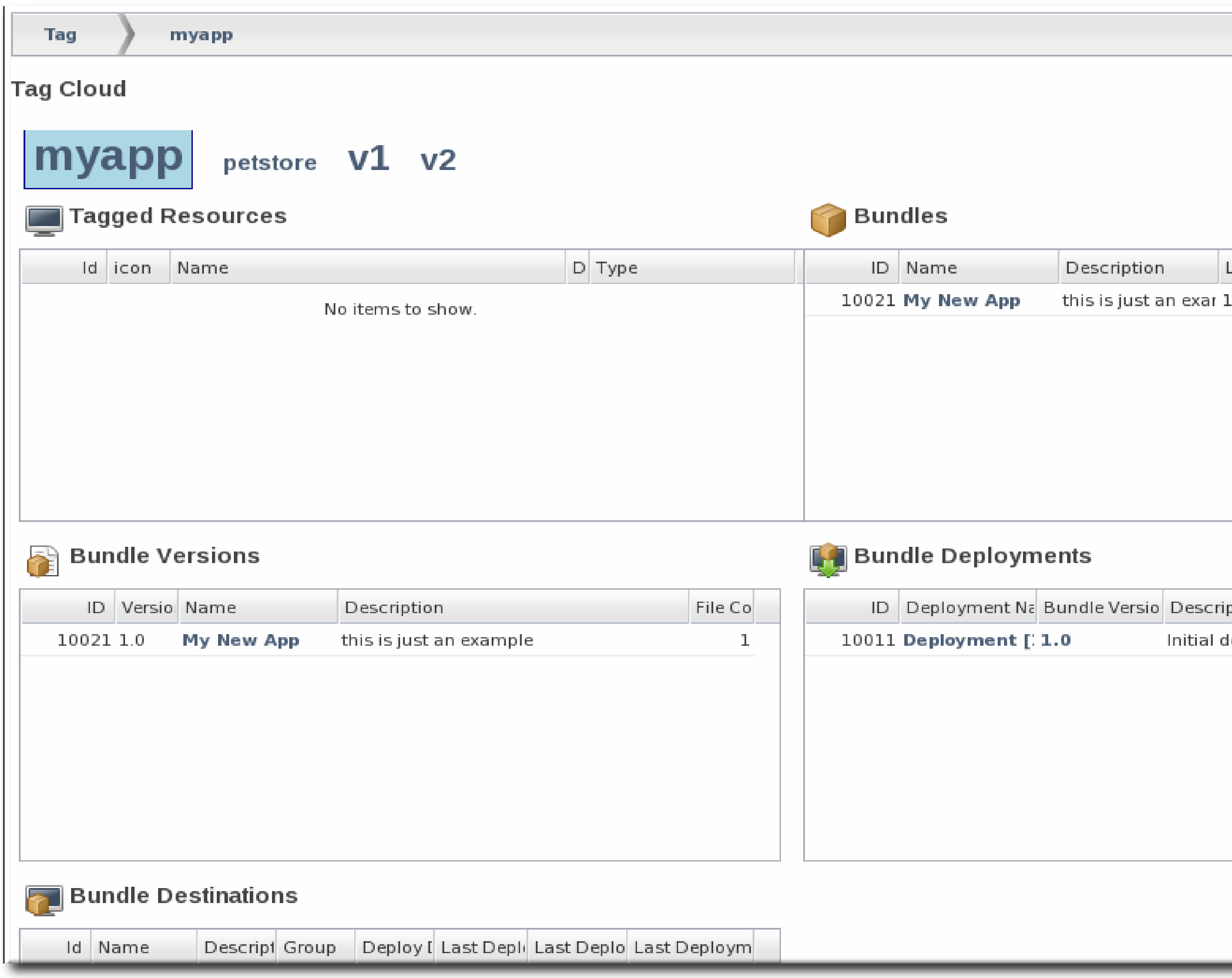
Tags:
v1
myapp

Bundle Deployments

ID	Deployment Name	Bundle Versio	Description	Status	Deploymer
10011	Deployment [1] of Version 1.0		Initial deployment.	✓ SUCCESS	7/7/201

Figure 5.6. Tags on a Bundle Entry

Clicking on a tag opens a tag area that shows all of the entries which use that particular tag, organized by entry type. There's a cloud at the top of all configured tags in JBoss ON, and clicking on any of the tags at the top opens the tag cloud for that tag.



Tagged Resources

Id	icon	Name	D	Type

No items to show.

Bundles

ID	Name	Description	L
10021	My New App	this is just an exar	1

Bundle Versions

ID	Versio	Name	Description	File Co
10021	1.0	My New App	this is just an example	1

Bundle Deployments

ID	Deployment Na	Bundle Versio	Descrip
10011	Deployment [:	1.0	Initial d

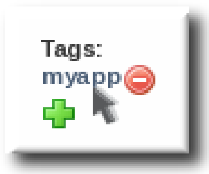
Bundle Destinations

Id	Name	Descripi	Group	Deploy [	Last Depl	Last Deplo	Last Deploym

Figure 5.7. Viewing Every Entry with a Specific Tag

5.9.3. Deleting Tags

1. Open the bundles entry from which to remove the tag.
2. Hover over the name of the tag to remove under the **Tags** label.



3. Click the delete icon that appears by the name of the tag. This immediately removes the tag from the entry.

Managing JBoss Products with JBoss ON

JBoss Operations Network provides extra tools that help manage JBoss server instances. These management tools cover everything from manually configuring a JBoss inventory to applying JBoss patches.

6.1. Deploying Applications on JBoss AS Instances

A number of different types of services can be deployed on a JBoss AS instance by adding it as a child of that instance. This follows the same basic outline as manually adding resources described in *Section 2.2.1, “Create”*, but JBoss ON provides additional guidance and options in the UI to simplify the process for deploying EARs, WARs, data sources, connection factories, and JMS queues.

6.1.1. Deploying EAR and WAR

1. Search for the JBoss AS server to which to deploy the EAR or WAR.

2. On the details page for the selected JBoss AS server, open the **Inventory** tab.

3. In the **Create New** drop-down menu, select the item for **- Web Application (WAR)** or **- Enterprise Application (EAR)**, as appropriate.

4. In the resource form, enter the information for the application to be deployed. Aside from the obvious settings, like the resource name, the WAR or EAR resource entry requires the following information:

   - The package name that the EAR or WAR file will be deployed *as*. In other words, the target entry name.

   - The package architecture.

     NOTE

     For EAR and WAR deployment, the package architecture should always be **`noarch`**.

   - The full path to the file to be deployed, in the **File** field.

   - Whether the file should be unzipped when it is deployed.

   - The path to the directory to which to deploy the EAR or WAR package. The destination directory is relative to the JBoss AS server installation directory; this cannot contain an absolute path or go up a parent directory.

6.1.2. Deploying Data Sources

1. Search for the JBoss AS server to which to deploy the data source.

2. On the details page for the selected JBoss AS server, open the **Inventory** tab.

3. In the **Create New** drop-down menu, select the item for **- Data Sources**.

4. Select a template for the data source. There are three data sources templates to populate a data source with common information:

   - The default template is used with SQL databases like PostgreSQL or MySQL
   - The Oracle Local TX is used for Oracle databases with local transactions.
   - The Oracle XA template is used for Oracle databases with XA transactions.

5. Along with the obvious settings, like the resource name, enter the information for the specific child resource to be deployed:

   - The type of data source to create, either No TX Data Sources, Local TX Data Sources or XA Data Sources
   - A unique JNDI name for the DataSource wrapper to use to bind under
   - The fully qualified name of the JDBC driver or DataSource class, such as **`org.postgresql.Driver`**
   - The JDBC driver connection URL string, such as **`jdbc:postgresql://127.0.0.1:5432/foo`**
   - The username and password to use to connect to the data source
   - The minimum and maximum connection pool sizes for this data source

   Additional settings are available under the **Advanced Settings** area. These are the same as the advanced settings available when the JBoss ON server is installed.

6.1.3. Deploying Connection Factories

1. Search for the JBoss AS server to which to deploy the connection factory.

2. On the details page for the selected JBoss AS server, open the **Inventory** tab.

3. In the **Create New** drop-down menu, select the item for **- Connection Factory**.

4. Along with the obvious settings, like the resource name, enter the information for the specific child resource to be deployed:

   - The type of connection factory to create, either tx-connection-factory (transaction) or no-tx-connection-factory (no transaction)
   - A unique JNDI name for the DataSource wrapper to use to bind under
   - The username and password to use to connect to the data source
   - The minimum and maximum connection pool sizes for this data source

   Additional settings are available under the **Advanced Settings** area. These are the same as the advanced settings available when the JBoss ON server is installed.

6.1.4. JMS Queues and Topics

JMS Queues and JMS Topics are child resources of a JBossMQ service or JBossMessaging service resource, which itself is a child of a JBoss AS server.

1. Search for the JBoss messaging service to which to deploy the JMS queue or topic.
2. On the details page for the selected JBoss messaging service, open the **Inventory** tab.
3. In the **Create New** drop-down menu, select the item for **- JMQ JMS Topic** or **- JMQ JMS Queue**, as appropriate.
4. Aside from the obvious settings, like the resource name, the JMS Queue or JMS Topic entry requires two additional parameters:

    - The name of the queue or topic to use as the JMX object name
    - A unique JNDI name for the DataSource wrapper to use to bind under

    Additional settings are available under the **Advanced Settings** area. These are the same as the advanced settings available when the JBoss ON server is installed.

6.2. Updating and Deleting Applications on JBoss Servers

EARs and WARs can both be updated on JBoss server instances simply by uploading the updated packages.

1. Browse to the EAR or WAR resource in the JBoss ON UI.
2. In the EAR or WAR resouce details page, open the **Content** tab, and click the **New** subtab.
3. Click the **UPLOAD NEW PACKAGE** button.
4. Click the **UPLOAD FILE** button.
5. In the pop-up window, click the **Add** button, and browse the local filesystem to the updated WAR or EAR file to be uploaded.
6. Click the **UPLOAD** button to load the file and dismiss the window.
7. In the main form, select the repository where the WAR or EAR file package should be stored. If one exists, select an existing repository or a subscribed repository for the resource. Otherwise, create a new repository.
8. Confirm the details for the new package, then click **CONTINUE**.

When the package is successfully uploaded, the UI redirects to the history page on the **Content** tab.

To undeploy an EAR or WAR, simply undelete the EAR or WAR child resource, as described in *Section 2.2.2, “Delete”*.

6.3. Applying JBoss Patches

Content management can be used to apply cumulative patches to JBoss AS, JBoss EAP, JBoss SOA-P, and JBoss EWP instances. By default, the JBoss ON server comes pre-configured with the JBoss Patch Content Source and JBoss Patch Repository.

A *content source* connects a JBoss ON server to all the JBoss patches. A *repository* maps a content source to the resources in the inventory that the patches are applied to. For JBoss patches, the default content provider connects the JBoss ON server to the cumulative patches provided by the JBoss Customer Service Portal. The default repository associated with the content provider is where the metadata about the patches and the patches themselves are stored within JBoss ON.

The patch process updates existing JAR and class files with upgraded JAR and class files that are contained in the patch package. Other changes that need to be completed manually (all the "Not Performed" steps, for example) are also listed. If the configuration does not include one of the JAR files to be patched, then that step is skipped.

The patching process does not care what the server configuration profile is called or which base configuration it is derived from.

6.3.1. Supported JBoss Products for Patch Updates

* JBoss AS 3.2.3.GA * JBoss AS 3.2.6.GA * JBoss AS 3.2.7.GA * JBoss AS 3.2.8.SP1 * JBoss AS 4.0.0.GA * JBoss AS 4.0.1.SP1 * JBoss AS 4.0.2.GA * JBoss AS 4.0.3.SP1 * JBoss AS 4.0.4.GA * JBoss AS 4.0.5.GA * JBoss EAP 4.2.0.GA * JBoss EAP 4.3.0.GA Since all versions of JBoss AS have reached their end-of-life for support, future CPs will only released for JBoss EAP, JBoss SOA-P, and JBoss EWP.

6.3.2. Summary of the Patch Process

The patch process is executed by the JON Agent that is running on the same machine as the JBoss instance being patched. It updates existing jar/class files within the JBoss installation directory with upgraded jar/class files that are contained in the patch zip. The patching process does not explicitly care about what the server configuration profile is called or which base configuration it is derived from.

The first step of the patch installation will be to stop the application server instance being patched if it is currently running, so make sure to perform patch installations during off hours and/or scheduled maintenance periods.

For some patches, there are additional changes that need to be completed manually (e.g. updating an XML configuration file). These changes are listed as special steps on the post patch installation summary; they are labeled "Not Performed." If your configuration does not include one of the jars to be patched, then that step will be skipped. Here are examples of the different types of manual steps that may be listed:

1. 1. The file to be patched is not present in your configuration.

2. 2. Files that need to be removed manually if present in your configuration:

3. 3. Configuration file (e.g. XML file or Java properties file) patches that need to be applied manually.

4. 4. Manually upgrade Seam (if you are using that).

5. 5. Start your JBAS instance once all other manual steps have been performed.

6.3.3. Enabling the Default JBoss Patch Content Source

1. Go to **Administration** > **View All Content Sources**.

2. Click the **JBoss Patch Content Source** repo.

3. Click the **Edit** button to modify the content source.

4. Fill in the required connection information:

   * The Customer Support Portal username and password
   * The URL for the content source (*https://support.redhat.com/jbossnetwork/restricted/feed/software.html?product=all&downloadType=&flavor=rss&version=&jonVersion=2.4*)
   * The activation state (**Yes**)

   Use the **Test Connection** button to verify that the connection details are valid.

   Most of the default settings, such as the schedule, can be kept.

   IMPORTANT

   Keep the **Lazy Load** checkbox activated, or all patches defined in the RSS feed, 1 GB of data, is preemptively downloaded by the JBoss ON server.

5. Click **Save**.

6. Optionally, use **Synchronize** button to force the content source to pull down the latest RSS Feed and synchronize it with the local data. The history of synchronization attempts is listed in the **Synchronization Results section**.

6.3.4. Subscribing a Specific Resource to the Default JBoss Patch Repository

1. # Go to the JBoss AS Server Resource that is to be a subscriber to the patches repo. Click its to Content > Subscriptions subtab. You should see the 'JBoss Patches' repo listed as available for subscription.

2. # Select the checkbox to the left of the 'JBoss Patches' repo, and then click the SUBSCRIBE button. The repo should then be listed under Current Resource Subscriptions, rather than Available Repositories.

6.3.5. Subscribing Multiple JBoss Resources to the Default JBoss Patch Repository

The repository is associated with a content source (the patches that can be applied) and resources are then subscribed to this repository (where the patches can be applied to).

1. Click the **Go To All Channels View** at the bottom of the Content Sources list.

   Alternately, go to **Administration** > **View All Channels**.

2. Click the **JBoss Patch Channel**.

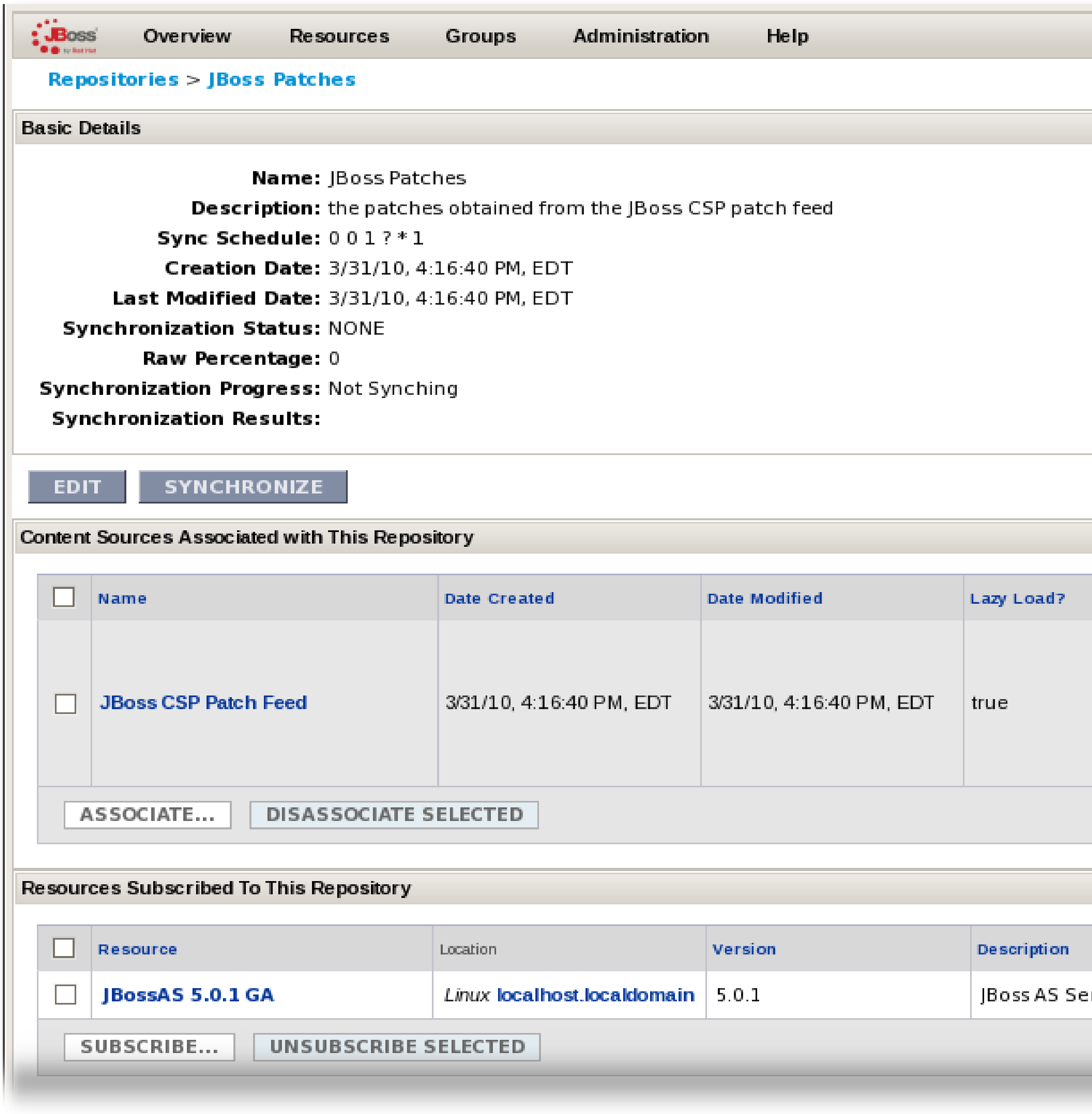
3. By default, the repository has already been associated with the "JBoss Patch Content". However, if you need to associate the repository to another content source, you can do this through the **ASSOCIATE...** button.

4. Click the **SUBSCRIBE...** button to subscribe JBoss resources you would like to patch.

   • Change the Available Resources drop down to **Server**

   • Select all the JBossAS Server resources to subscribe to this repository.

   • A screen will then appear that displays the content source, resources, and available packages. The screen will look similar to the following:



Basic Details

Name: JBoss Patches
Description: the patches obtained from the JBoss CSP patch feed
Sync Schedule: 0 0 1 ? * 1
Creation Date: 3/31/10, 4:16:40 PM, EDT
Last Modified Date: 3/31/10, 4:16:40 PM, EDT
Synchronization Status: NONE
Raw Percentage: 0
Synchronization Progress: Not Synching
Synchronization Results:

Content Sources Associated with This Repository

	Name	Date Created	Date Modified	Lazy Load?
☐	JBoss CSP Patch Feed	3/31/10, 4:16:40 PM, EDT	3/31/10, 4:16:40 PM, EDT	true

Resources Subscribed To This Repository

	Resource	Location	Version	Description
☐	JBossAS 5.0.1 GA	*Linux* localhost.localdomain	5.0.1	JBoss AS Se

6.3.6. Applying a Patch

1. 1. Now that the JBoss AS Resource is subscribed to the patches repo, go to the Content > New tab to see a list of patches which are available for the Resource. For example, the following patches might be available for a JBoss EAP 4.3.0.GA_CP01 server Resource:

2. 2. Select the checkbox to the left of the CP you want to install, then click the DEPLOY SELECTED button. This will take you to a review page like the following.

3. 3. Review the information on the page and verify everything is correct; click the VIEW link in the Instructions column to review the exact steps that will be performed during the installation of the CP. 4. Optionally enter some notes in the Notes field (e.g. "upgrade to latest tested EAP CPs as part of Q3 FY10 maintenance").

4. 5. Click the CONTINUE button to proceed with installing the patch. This will kick off the patch installation as a background job and automatically take you to the Resource's Content > History subtab, where you can monitor the progress of the job.

5. 6. Once the patch installation is complete, the job will move from the "Currently Executing Requests" list to the "Completed Requests" list. Clicking the VIEW button will display the list of steps that were performed and whether they succeeded, failed, or were skipped because an earlier step failed.

Managing Operations

JBoss Operations Network provides a way to manage resources by scheduling and launching *operations*. Operations are basic management tasks. The available tasks differ for every different type of resource; see the *Resource Monitoring Reference* for the list of operations for each resource type.

This chapter covers the procedures to use operations, including creating new operations, canceling pending operations, applying operations to a group, and executing local resource scripts as operations.

7.1. Understanding Operations

Operations manage the state of your resources. The agent carries out some basic, specified tasks on the managed resources, such as restarting a server or running a script. Operations can be carried out on any resource in the inventory, and even on the JBoss ON agent themselves.

The types of operations that are available for each resource depends on the type of resource being managed. For example, a JBoss AS server has different available operations than a cron service.

Operations can take arguments (required or optional, depending on the operation), which are values or paramters that are passed with the operations command. Arguments for operations use the same underlying logic that's in the resource configuration (hence, operations are resource-specific). JBoss ON can validate arguments just as it validates resource configuration.

When an operation is complete, it returns a result message. All executed operations return a simple success or failure message, and complex operations can return complex result messages. No matter how complicated the arguments or the results for an operation are, JBoss ON can configure and display it in the UI.

Operations are run on a schedule. By default, the operation is executed as soon as it is configured, but it can be set to run at some future time. Scheduled operations can be executed once or on a recurring schedule, indefinitely or with a termination date.

The agent queues operations so that only one operation is executed on a resource at any time. Pending operations can be canceled, launched immediately, or rescheduled. An optional timeout setting prevents an operation from hanging indefinitely and blocking other operations from running.

7.2. Creating New Operations

1. Click the **Inventory** tab in the top menu.

2. Click the name of the resource on which to create the operation.

   All resources are listed in the main window by default. The **Inventory** navigation tree in the left panel has saved searches for resources by type (platforms, servers, or services). A specific resource can be searched for using the **Search Resources** field in the main area.

3. Click the **Operations** tab.

4. In the **NEW** area of the **Operations** tab, click the name of the operation to configure.

   The types of operations that are available vary, depending on the specific type of resource.

5. The new operation form opens. Fill in all of the required information about the operation, such as the parameters to use to perform the operation (this can include port numbers, file locations, or command arguments).

6. In the **Scheduler Component** area, set when to run the operation. The operation can run once immediately, or be set to recur on a defined schedule.

   To run it once, leave the **Immediately** radio button selected. To set a schedule, select the radio button by the calendar icon, and then set the set the start date, end date, and frequency.

7. Set other rules for the operations, like a timeout period and notes on the operation itself.

8. Click the **Schedule** to save the operation.

If the operation was run immediately, the results are available in the **HISTORY** tab. If it was scheduled on a later date or with a recurring schedule, then the operation will be listed in the **SCHEDULES** tab.

7.3. Viewing Operation History

1. Click the **Inventory** tab in the top menu.

2. Click the name of the resource on which to create the operation.

   All resources are listed in the main window by default. The **Inventory** navigation tree in the left panel has saved searches for resources by type (platforms, servers, or services). A specific resource can be searched for using the **Search Resources** field in the main area.

3. Click the **Operations** tab.

4. Click the **History** area.

5. In the **Completed Operations** area the status of the operation is reported with either **Failure** or **Success**.

   Click the name of the operation to view further details.

7.4. Canceling Pending Operations

1. Click the **Inventory** tab in the top menu.

2. Click the name of the resource on which to cancel a pending operation.

   All resources are listed in the main window by default. The **Inventory** navigation tree in the left panel has saved searches for resources by type (platforms, servers, or services). A specific resource can be searched for using the **Search Resources** field in the main area.

3. Click the **Operations** tab.

4. Click the **SCHEDULES** area.

5. In the **Resource Operation Schedules** area click the tick box next to the name of the operation to cancel.

6. Click **UNSCHEDULE**.

The operation is canceled immediately, and a highlighted confirmation message appears above the **Resource Operation Schedules** area confirming the operation has been canceled.

Note

Once the agent has started an operation it cannot be canceled. If the user attempts to cancel an operation currently processing the request will be ignored.

7.5. Ordering Group Operations

Group operations can be scheduled. This is useful when operations need to be performed in a particular order.

NOTE

This procedure assumes groups are already set up. For more details, see *Section 2.5, “Organizing Resources in Groups”*.

1. Click **Groups** from the top menu.
2. Click the type of group to be managed.
3. Click the name of the group to be managed.
4. Click the **Operations** tab.
5. In the **NEW** area of the **Operations** tab, click the name of the operation to configure.
6. Enter any required parameters into the **Operation Parameters** area as required.
7. In the **Resource Operation Order** area, set the operation to execute **At the same time for all resources** or **In order** by selecting the appropriate radio button.
8. In the **Scheduler Component** area, set when to run the operation. The operation can run once immediately, or be set to recur on a defined schedule.

   To run it once, leave the **Immediately** radio button selected. To set a schedule, select the radio button by the calendar icon, and then set the set the start date, end date, and frequency.
9. Set other rules for the operations, like a timeout period and notes on the operation itself.
10. Click **Schedule** to save the operation.

If the operation was run immediately, the results are available in the **HISTORY** tab. If it was scheduled on a later date or with a recurring schedule, then the operation will be listed in the **SCHEDULES** tab.

7.6. Running Scripts as Operations

JBoss ON auto-discovers resource scripts when the resource is discovered. Scripts can be managed just like any other resource to perform operations. There are three types of scripts that JBoss ON discovers, depending on the operating system:

- `.bat` for Windows
- `.sh` for Unix/Linux
- `.pl` scripts for Unix/Linux

NOTE

Scripts on Linux and Unix systems need to have the x-bit set to be detected.

Connection properties and environment variables can be added to a script. This is described in *Section 2.2.3, “Managing Scripts as Resources”*.

To execute a script as an operation:

1. Click the **Inventory** tab in the top menu.
2. Click the name of the resource on which to execute the script.

   All resources are listed in the main window by default. The **Inventory** navigation tree in the left panel has saved searches for resources by type (platforms, servers, or services). A specific resource can be searched for using the **Search Resources** field in the main area.
3. Click the **Operations** tab.
4. In the **NEW** area of the **Operations** tab, select **Execute script**.

   NOTE

   The **Execute script** option is only available for certain types of resources, and only if a script is available to execute.
5. The **Operation Parameters** form opens. Enter any required parameters.
6. In the **Scheduler Component** area, set when to run the script. The script can be run once immeadiately, or be set to recur on a defined schedule.

   To run once, leave the **Immediately** radio button selected. To set a schedule, select the radio button by the calendar icon, and then set the start date, end date and frequency.
7. Set other rules for the script, like a timeout period and notes on the script itself.
8. Click the **Schedule** to save the operation.

If the operation was run immediately, the results are available in the **HISTORY** tab. If it was scheduled on a later date or with a recurring schedule, then the operation will be listed in the **SCHEDULES** tab.

7.7. Setting an Operation Timeout Default

Only one operation can run on a resource at one time. An optional timeout setting prevents an operation from hanging indefinitely and blocking other operations from running. A global default timeout can be set in the JBoss ON server configuration to prevent operations from being blocked on a resource, even if a timeout period isn't set on a specific operation.

NOTE

This server setting is a fallback value. Operation plug-ins can define their own timeouts in the plug-in descriptor or individual operations can specify a timeout. Both of those settings override the server default.

1. Open the **`rhq-server.properties`** file.

   ```
   vim serverRoot/jon-server-2.4.0.Beta1/bin/rhq-server.properties
   ```

2. Change or add the value of the *`rhq.server.operation-timeout`* parameter to the amoung of time, in seconds, for the server to wait before an operation times out.

   ```
   rhq.server.operation-timeout=60
   ```

Monitoring Resources

JBoss Operations Network monitors resources by gathering data from each resource in inventory. The metrics can be represented in graphs, tables, and side by side comparisons of values across different resources. Monitoring resources enables alerts to be triggered when predefined metrics are reached, or certain event conditions are met.

By collecting metrics from the managed resources, JBoss Operations Network keeps administrators appraised of the overall state of the IT environment. By examining the metrics over different timeframes, it is possible to monitor to determine periods of peak demand, and effectively manage the resources.

Monitoring keeps the administrator in touch with the IT environment, by sending alerts when resources metrics fall outside of established baselines. Administrators can configure alerts to notify the appropriate parties of unusual or unexpected metrics. Monitoring can also be configured to run operations in response to alerts.

This chapter covers monitoring procedures, defining events, and how monitoring data is stored.

8.1. Monitoring Resources

8.1.1. Viewing Resources

1. Click the **Resource** tab in the **Top menu**.
2. Search for the resource to monitor, and click the name of the desired resource in the results.
3. Click the **MONITOR** tab.

From here it is possible to view the available metrics for the particular resource being monitored, change time frames for the metrics via the **Advanced Settings** link at the bottom of the screen, and access editable charts by clicking on the resource's name.

8.1.2. Recalculating Baseline Values

1. Click the **Resource** tab in the **Top menu**.
2. Search for the resource to monitor, and click the name of the desired resource in the results.
3. Click the **MONITOR** tab.
4. Click the name of the specific resource.
5. On the new screen scroll, down to the **Metric Baseline & Expected Range**, click **Change Value** next to the baseline figure.
6. Accept the new baseline value by clicking **Save Value**, or cancel and return to the previous baseline value.

Note

The same procedure can be carried out for the **High** and **Low** values.

8.1.3. Setting Collection Intervals

1. Click the **Resource** tab in the **Top menu**.
2. Search for the resource to monitor, and click the name of the desired resource in the results.
3. Click the **MONITOR** tab.
4. Click the **SCHEDULES** sub tab.
5. Select the metric or metrics being scheduled via the check box.
6. Enter the desired collection period in the **Collection Interval** field. Select **Hours**, **Minutes**, or **Seconds** from the drop down menu.
7. Click **SET** to confirm the change.

8.1.4. Enabling and Disabling Metrics

1. Click the **Resource** tab in the **Top menu**.
2. Search for the resource and click the name of the desired resource in the results.
3. Click the **MONITOR** tab.
4. Click the **SCHEDULES** sub tab.
5. Select the metric or metrics being enabled or disabled via the check box.
6. Select **ENABLE** or **DISABLE** from the bottom of the screen as required.

8.1.5. Changing Monitoring Defaults

By default, some metrics are disabled and must be mannually enabled.

1. Select **Administration** → **System Configuration** → **Templates** from the top navigation bar.
2. Select **Edit Metric Template** for the desired platform, server or service.
3. Select the required metric or metrics to be enabled.
4. Select **Update schedules for existing resources of marked type**.
5. Enter the desired time frame into the **Collection Interval for Selected:** field, and click the black arrow to confirm.

8.2. Events

Events are a type of measurement data, that, unlike metrics, are not collected at fixed intervals but occur at random points in time. JBoss ON is able to pick up those events, filter them by severity, display them in the user interface, and have alerts sent based on event conditions.

8.2.1. Defining Event Sources

1. From the **Dashboard** select the managed resource, either from the resource tree on the left, the **Resources** menu item on the top navigation, or from the **Favourite Resouces** area if the resources has been saved as a favourite.

2. Click the **INVENTORY** tab.

3. Select **CONNECTION** sub tab.

4. Click **EDIT** under the **Events Log**.

5. Click **Add New**.

6. Add events properties as required.

To capture new events ensure they are:

- Enabled.

- The log path is valid.

- The selected date format matches the format in the logs.

If you select the default date format when creating the event source, and that format does not match the format in the logs, nothing will be captured for the log events.

8.2.2. Viewing Events

1. From the **Dashboard** select the managed resource, either from the resource tree on the left, the **Resources** menu item on the top navigation, or from the **Favourite Resouces** area if the resources has been saved as a favourite.

2. Click the **EVENTS** tab.

3. Click the specific event for further details.

8.3. Configuring How Long Monitoring Data Are Stored

JBoss ON stores monitoring data using a tiered model. This reduces the amount of data that is stored and still provides the necessary granularity for useful historical metric data.

The following outlines the main steps JBoss ON uses to archive its metric data:

- Raw metrics get aggregated over a one-hour window producing minimum, average and maximum values.

- Those one hour values get aggregated across a six hour window.

- Finally, six hour values get aggregated across a 24 hour window.

Old data gets deleted according to the following schedule:

- Raw metric data which has been aggregated and is older than 7 days.
- One hour data which has been aggregated and is older than 14 days.
- Six hour data which has been aggregated and is older than 31 days.
- 24 hour data which is older than 365 days.

The following settings can be defined:

Name	Description	Default
Run Database Maintenance Every	(applicable when using a Postgres DB)	1 hour
Delete Response Time Data Older Than	Specifies the amount of time used when purging response times in days	31 days
Delete Alerts Older Than	Specifies the amount of time (in days) used when purging alerts	31 days
Delete Events Older Than	Specifies the amount of time (in days) used when purging events	31 days
Reindex Metric Data Tables Nightly	Yes or no	Yes

Configuring and Managing Alerts

Monitoring (*Section 8.1, “Monitoring Resources”*) is the first step in a larger workflow that has the sole intent of keeping administrators aware of what is happening in their network. The next two steps involve:

- Setting parameters for JBoss ON to trigger a warning (*alerts*)
- Notifying administrators when an alert is tripped (*notifications*)

This chapter details how to configure notifications and alerting templates as part of managing resources.

9.1. Brief Introduction to Alerts and Notifications

An *alert* is a configuration setting that lets an administrator know that something has happened to a resource. Conditions and notifications are configured together in an *alert definition* for a resource.

There are three major components to an alert definition:

- The information that identifies that specific alert definition (the name, priority, and whether it is active)
- The conditions that trigger the alert, which depends on the area of the resource being monitored
- The method and settings to use to send the alert

9.1.1. Alert Conditions

The *condition* is any situation, event, or level on a resource that crosses a certain threshold. Basically, a condition sets parameters on what is "normal" behavior or performance for a resource. Once it crosses that boundary, JBoss ON issues an alert.

An alert conditions answers four questions: *what*, *when*, *who*, and *where*. The *what* is the threshold or *condition* that triggers the alert (such as the free memory drops below a certain point). The *when* sets the frequency or timing for sending an alert using a defined *dampening* rule. And the *who* and *where* controls how administrators are *notified* of the alert.

A condition can bey any of five different metrics, listed in *Table 9.1, “Types of Alert Conditions”*. These alert conditions correspond directly to the monitoring metrics available for that type of resource. All of the possible metrics for each resource type are listed in the *Resource Monitoring Reference*.

Condition Type	Description
Metric	A specific monitoring area that is checked and the thresholds for that area which trigger a response. Metrics are usually numeric responses of some sort (e.g., percent CPU usage, number of requests, or a cache hit ratio).
Trait	A change in a value for a specific setting. Traits are usually string values.
Availability	A sudden change in whether the resource is available or unavailable.

Condition Type	Description
Operation	A specific action or task that is performed on the resource.
Severity	A certain type of error message, matching a given string, is recorded.

Table 9.1. Types of Alert Conditions

Along with setting the threshold, the condition sets *how* JBoss ON counts events for it to trigger alerts. For example, a condition may be set to alert if the CPU hits 80% usage. In real life, a server may bounce between 78% and 80% CPU over several minutes, it could hit 80% once for only a few seconds, or it could hit 80% and stay there.

The condition *dampening* setting tells JBoss ON how to interpret those monitoring data.

- JBoss ON could send an alert every time the condition is encountered. In that case, there would be multiple alerts issued if the CPU percentage bounced around, while only one alert would be sent if it hit it briefly or hit it and stayed there.
- JBoss ON could send an alert only if the condition was encountered a certain number of times consecutively or X number of times out of Y number of polls. In this case, only a recurring or sustained problem would trigger an alert. A momentary spike or trough wouldn't be enough to fire a notification.
- The other option is that a notification is sent only if the problem occurs within a set time period. This can be useful to track the frequency of recurring problems or to track how long a condition persisted.

9.1.2. Notification Methods

Once an incident occurs, there has to be a way to let a systems administrator know what is going on, so they can respond to an issue. This is done by configuring a *notification*.

JBoss ON has several different methods of sending a notification:

- Email
- SNMP traps
- Operations
- Users
- Roles

It is also possible to write custom alert methods, which are implemented as server-side plug-ins. Creating custom plug-ins is described in the *JBoss Operations Network Plug-ins Writing Guide*.

These alert methods can be configured individually for a specific alert definition.

TIP

You can "cluster" alert notifications.

Alert notifications can be broadcast through several different methods at the same time. For example, if a public website goes down, then a company may want notifications to be sent to their head web administrator and their company's external microblog feed at the same time.

9.1.3. Alert Operations

When a certain alert condition occurs, the JBoss ON agent can respond by initiating an operation on a resource. This is part of the alert definition configuration, but it's worth calling out because it is such a useful tool for managing responses to alerts. Whenever an alert is fired, the agent can perform some kind of action, like restarting a server. This can be done either on the resource which issued the alert or on another resource.

Remote operations can be exceedingly useful (and versatile). For example, a JBoss server may begin performing badly because its JVM is out of memory. The JVM is the resource which issues its alert, but the response by the agent is to restart the JBoss server.

Regular operations are either initiated immediately or run on defined schedules for a specific configured resource. Alert operations are even more flexible than regular operatoins for two reasons:

- Alert operations are fired responsively to address any alert or event.
- Alert operations can be initiated on *any* resource in the JBoss ON inventory, not only the resource which sent the alert. That means that an operation can be run for a different application on the same host server or even on an entirely different server.

NOTE

The operations performed in response to an alert are the same as the operations which can be scheduled to run on a resource. The operations available for an alert depend on the target resource on which the operation will run — not the resource where the alert is set.

The type of operation which is available to be run for an alert depends on the type of resource that is the target of the operation. (This may not be the same as the resource which has the alert configured.) There are two types of alert operations:

- Operations that are the same as regular operations.
- Scripts, such as JavaScript or Ruby, that can be run on any platform as an operation for script resources.

TIP

Alert operations senders can be used to run scripts on remote resources. For example, if a resource goes down, a diagnostic script can be run on its parent platform or another resource can be brought online and properly configured to take its place.

TIP

A single alert can initiate multiple operations. All alert operations, as with all alert notifications, are run in the order they are listed in the alert definition.

Alert operations can accept tokens to fill in certain values automatically. These have the following form:

```
<%space.param_name%>
```

The *space* gives the JBoss ON configuration area where the value is derived; this will commonly be either **`alert`** or **`resource`**. The *param_name* gives the entry value that is being supplied. For example, to point to the URL of the specific fired alert, the token would be **`<%alert.url%>`**, whiel to pull in the resource name, the token would be **`<%resource.name%>`**. The possible tokens are listed in *Table 9.2, “Available Alert Operation Tokens”*.

9.1.4. Alert Histories and Acknowledgements

Having a record of alert incidents can help improve performance, incident analysis, and other admin tasks.

Every time an alert is sent, JBoss ON makes a record of it. Each alert notification and the conditions that triggered it are stored in the *alert history* for the resource.

JBoss ON also enables users to acknowledge alerts. An administrator who takes or verifies an action after an alert can mark that alert as acknowledged to indicate that the issue is closed. The name of the user and the time of the acknowledgement are recorded with the alert details.

The alert history and acknowledgement history are both valuable for auditing and assessing infrastructure performance.

9.1.5. Group Alerting and Alert Templates

Most alerts can be defined consistently for multiple resources of the same type. JBoss ON has two ways to accomplish this:

- Alert templates
- Alerts on compatible groups

An alert template is a configuration setting for the JBoss ON server. An alert is configured for a specific resource type (even if no resource of that type exists in the inventory yet). Whenever a resource is added, any alert templates in the JBoss ON configuration are automatically applied to that resource.

Alert templates can be configured to allow local changes (for example, Resource A may have different baselines or expected behavior, so the alert conditions can be altered). Templates can also be strictly enforced, so that every resource of that type has exactly the same settings.

Alerts can be configured on compatible groups. As with alert templates, the compatible group's alert definitions trickle down to the rest of the group members. When a resource is added to a group, the alerts are automatically added to the resource. When the resource is removed from the group, the alert is automatically deleted. Group alerting works for both manual groups and dynamic groups. As with alert templates, group alerts can allow local changes or enfource the group alert settings.

9.2. Setting Alerts for a Resource

1. Select the **Resources** menu in the top navigation bar, and select the **Servers** menu item.
2. Click the resource in the list.
3. Click the **Alerts** tab.
4. In the **DEFINITIONS** subtab, click the **NEW DEFINITION** button to create the new alert.
5. Fill in the information about the new alert entry.
   - *Name*. Gives the name of the specific alert definition. This must be unique for the resource.
   - *Description*. Contains an optional description of the alert; this can be very useful if you want to trigger different kinds of alert responses at different conditions for the same resource.
   - *Priority*. Sets the priority or severity that is given to an alert triggered by this definition.
   - *Active*. Sets whether the alert definition is active. Alert definitions can be disabled to prevent unnecessary or spurious alerts if there is, for instance, a network outage or routine maintenance window for the resource.
6. There can be more than one condition configured for a single alert. Set whether the alert is sent if any of the conditions are met or only if all of the conditions are met.

   TIP

   There can be more than one condition set to trigger an alert. For example, you may only want to receive a notification for a server if its CPU goes above 80% *and* its available memory drops below 25MB. The **ALL** setting for the conditions restricts the alert notification to only when both criteria are met. Alternatively, you may want to know when either one occurs so that you can immediately change the load balancing configuration for the network. In that case, the **ANY** setting fires off a notification as soon as even one condition threshold is met.

7. Define the conditions that trigger the alert.

   a. Click the **Add a new condition** button.

   b. From the initial drop-down menu, select the type of condition. The categories of conditions are described in *Table 9.1, "Types of Alert Conditions"*, and the exact conditions available to be set for every resource are listed in the *Resource Monitoring Reference*.

   c. Set the values for the condition.

8. Set whether to send a recovery alert and whether the alert is disabled until the resource state is recovered.

9. Give the dampening (or frequency) rule on how often to send notifications for the same alert event.

   The frequency for sending alerts depends on the expected behavior of the resource. There has to be a balance between sending too many alerts and sending too few. There are several frequency settings:

   - *Every time the condition is true*. This sends the most alerts. Without a recovery filter, this alert fires every single time a condition occurs, even for momentary spikes. Since this is the safest setting, it's the default.

   - *Every X times the condition is true*. This sets a number of times that the condition has to occur before an alert is sent. This offsets any spikes. However, don't set this number too high or the alerts won't be sent when there really is a valid situation.

   - *Every X times the condition is true*. This sets a number of times that the condition has to occur before an alert is sent. This offsets any spikes. However, don't set this number too high or the alerts won't be sent when there really is a valid situation.

     This counts when the condition is true in continuous evaluations.

   - *Every X times the condition is true in Y evaluations*. This sets a number of times that the condition has to occur in a given number of monitoring evaluations cycles before an alert is sent.

   - *Every X times the condition is true in Y time period*. The other two similar dampening rules set a recurrence based on the JBoss ON monitoring cycles. This sets the alerting rule based on a specific time period.

10. Click **OK** to save the properties and conditions. This creates the alert definition. The next screen gives the option to add notifications and operations to the existing alert.

11. In the **Notifications** area, click **Add New**, and select the method to use to send the alert notification. There are two different ways to do this:

    - The *Sender* option selects a specific type of alert method (such as email or SNMP) and then allows you to fill in the details for that specific method.

    - The *Template* option selects a pre-configured group of senders — with already defined settings — to use to send notifications.

12. In the **Operations** area, click the **Edit** button to add some kind of automatic response to the alert. If the alert is fired for any condition, then the agent initiates the specified action on the specified resource. The default is not to take any action.

13. Click **OK** to save the alert definition.

9.3. Assigning an Operation to an Alert

To set an alert operation, select the **Operations** alert method when configuring notifications. Operations can perform tasks or run scripts on a target resource; this is detailed in *Section 9.1.3,* *“Alert Operations”* and correlates to using resource operations, as described in *Chapter 7, Managing Operations*.

9.3.1. Using Tokens with Alert Operations

Alert operations can use tokens to either send information or supply information about the event. For example, tokens can be used to supply resource information in a command-line script.

Figure 9.1. Using Tokens in a Command-Line Script

Alert operations can accept tokens to fill in certain values automatically. These tokens have the following form:

```
<%space.param_name%>
```

The *space* gives the JBoss ON configuration area where the value is derived; this will commonly be either **`alert`** or **`resource`**. The *param_name* gives the entry value that is being supplied. For example, to point to the URL of the specific fired alert, the token would be **`<%alert.url%>`**, whiel to pull in the resource name, the token would be **`<%resource.name%>`**.

JBoss ON has pre-defined token values that relate to the fired alert, the resource which issued the alert, the resource which is the target of the operation, and the operation that was initiated. These are listed in *Table 9.2, “Available Alert Operation Tokens”*. All of these potential token values are Java properties the belong to the operation's parent JBoss ON server.

The alert operations plug-in resolves the token value itself when the alert operation is processed to find the value. The realized value is sent to the script service, which ultimately plugs the value into the command-line argument or script which referenced the token.

Information about ...	Token	Description
Fired Alert	alert.willBeDisabled	Will the alert definition be disabled after firing?
Fired Alert	alert.id	The id of this particular alert
Fired Alert	alert.url	Url to the alert details page
Fired Alert	alert.name	Name from the defining alert definition
Fired Alert	alert.priority	Priority of this alert
Fired Alert	alert.description	Description of this alert
Fired Alert	alert.firedAt	Time the alert fired
Fired Alert	alert.conditions	A text representation of the conditions that led to this alert
Alerting Resource	resource.id	ID of the resource
Alerting Resource	resource.platformType	Type of the platform the resource is on
Alerting Resource	resource.platformName	Name of the platform the resource is on
Alerting Resource	resource.typeName	Resource type name
Alerting Resource	resource.name	Name of the resource

Information about ...	Token	Description
Alerting Resource	resource.platformId	ID of the platform the resource is on
Alerting Resource	resource.parentName	Name of the parent resource
Alerting Resource	resource.parentId	ID of the parent resource
Alerting Resource	resource.typeId	Resource type id
Target Resource	targetResource.parentId	ID of the target's parent resource
Target Resource	targetResource.platformName	Name of the platform the target resource is on
Target Resource	targetResource.platformId	ID of the platform the target resource is on
Target Resource	targetResource.parentName	Name of the target's parent resource
Target Resource	targetResource.typeId	Resource type of the target resource id
Target Resource	targetResource.platformType	Type of the platform the target resource is on
Target Resource	targetResource.name	Name of the target resource
Target Resource	targetResource.id	ID of the target resource
Target Resource	targetResource.typeName	Resource type name of the target resource
Operation	operation.id	ID of the operation fired
Operation	operation.name	Name of the operation fired

Table 9.2. Available Alert Operation Tokens

9.3.2. Setting Alert Operations

1. Configure the basic alert definition, as in *Section 9.2, “Setting Alerts for a Resource”*.

2. Click **OK** to get to the next screen.

3. The bottom of the page has the **Notifications** section. Click the **Edit** button to add a response.

4. Give the notification method a name, and select the **Resource Operations** method from the **Alert Senders** drop-down menu.

5. Select the new alert sender in the table. This opens the configuration for the operation.

6. First, set the resource that the operation will run on. The default is the resource that the alert is set for; it is also possible to set it on another specific resource or on the results of a search.

7. Select the operation type. The available operations and their configuration parameters depend on the type of resource selected as the target of the operation.

   The *Resource Monitoring Reference* lists the available operations per resource type. *Chapter 7, Managing Operations* has more information on setting operations in general.

8. Configure the parameters of the operation. The available settings depend on the type of operation selected.

9.4. Initiating Scripts from an Alert

To set an alert operation, select the **Operations** alert method when configuring notifications, and then . This is the same as using an operation to execute a script, as described in *Section 7.6, “Running Scripts as Operations”*.

NOTE

The script must be uploaded to the resource and added into the JBoss ON inventory before it can be used in an alert operation.

1. Upload the script to the resource on which is should run in response to the alert. If necessary, run manual discovery to detect and add the script. See *Section 2.1.2, “Manual Discovery”*.
2. Configure the basic alert definition, as in *Section 9.2, “Setting Alerts for a Resource”*.
3. Click **OK** to get to the next screen.
4. The bottom of the page has the **Notifications** section. Click the **Edit** button to add a response.
5. Give the notification method a name, and select the **Resource Operations** method from the **Alert Senders** drop-down menu.
6. Select the new alert sender in the table. This opens the configuration for the operation.
7. Select the script resource that will be run in response to the alert.

9.5. Enabling and Disabling Alert Definitions

When an alert definition is disabled, no alert notifications are triggered for that set of conditions. Disabling definitions is very useful when resources are being taken offline for a know reason (such as upgrades or maintenance) and any alerts triggered during that time would be wrong. Alert definitions can be re-enabled later just as easily.

1. Select the **Resources** menu in the top navigation bar, and select the **Servers** menu item.
2. Click the resource in the list.
3. Click the **Alerts** tab.
4. In the **DEFINITIONS** subtab, select the checkbox next to any of the definitions to enable or disable.
5. Click the **ENABLE** or **DISABLE** button.

9.6. Viewing Alert Definitions

The alert definitions for a specific resource are always available by viewing that resource in JBoss ON. It is also possible to view all of the alert definitions for a parent resource (including all its children) or some other subset of resource alert definitions by checking the **Subystem Views** snapshot for the server.

9.6.1. Viewing Alert Definitions for a Specific Resource

1. Select the **Resources** menu in the top navigation bar, and select the **Servers** menu item.
2. Click the resource in the list.
3. Click the **Alerts** tab, and make sure that the **Definitions** subtab is selected.

9.6.2. Searching for Alert Definitions

1. Select the **Overview** menu in the top navigation bar.
2. In the **Subsystem Views** menu, select **Alert Definitions**.
3. In the **Definitions** subtab, select the filters to use to search for the alert definitions. You can use wild cards (`*`) and other expressions in the filter to narrow the results.
   - *Resource*. The resource name or partial name.
   - *Parent*. The parent of any resources; this can be used to return alerts for all resources on a single parent or to return results for multiple servers with a filter such as **`*.example.com`**.
   - *Begin/end dates*. The date range for when the definition was created.
   - *Category*. The type of alert definition.
4. Click **GO**.

All of the matching alert definitions for the entire JBoss ON server are listed in the results table. The results table has four columns that provide the most useful ways to arrange the definitions:

- The resource (**Name**).
- The parent. Since resources are arranged hierarchically, sorting by the parent is very useful for finding all alert definitions for all services and applications that relate to a high-level resource like a server.
- The conditions on the alert.
- Whether it is active (enabled).

9.7. Using Alerting Templates and Group Alerts

Templates make configuration really easy to apply consistently and often. JBoss ON has two types of templates that can be configured for alerts:

- Alert templates
- Alert sender (notification) templates

Group alerts, like alert templates, apply equally to every member of a compatible group. Group alerts offer more control over which resources have the alert definition, however, since resources can be manually added to the group or selected based on a search filter. When a resource joins or leaves a group, its alert definitions are automatically updated.

9.7.1. Creating Alert Definition Templates

Alert templates are fully defined alert definitions — from conditions to notification methods — that are created for any of the managed resource types in JBoss ON. Servers or applications of the same type will probably have the same set of alert conditions that apply, such as free memroy or CPU usage. An alert definition template creates an alert based on the *general type of resource*. So, there can be alert templates for Windows, Linux, and Solaris servers, Tomcat and Apache servers, and services like sshd and cron. Every time a resource of that type is added, then the alert definition is automatically added to the resource with the predefined settings. Any alert assigned to a resource through a template can be edited locally for that resource, so these alert definitions are still flexible and customizable.

To create a alert definition template:

1. In the top navigation, open the **Administration** menu, and then the **System Configuration** menu.
2. Select the **Templates** menu item. This opens a long list of resource types, both for platforms and server types.
3. Locate the type of resource for which to create the template definition.
4. The possible alert conditions are based on *metrics* monitored by the agent. The metrics for the alerts can be defined globally.
    a. Click the **Edit Metric Template** button to define what metrics are collected.
    b. Select (or de-select) the check boxes by the available metrics. Most reasonable metrics for the resource type are probably already selected, but this can be edited.

    > **NOTE**
    > Changing these settings changes the available metrics for every alert created for that resource type, not just the ones in the definition template.

    c. Save the changes.
5. Click the **Edit Alert Templates** button to create a global alert definition. Set up the alert exactly the same way as setting an alert for a single resource (as in XREF).
6. Save the template.

The template definition is then applied to all current and new resources of that time.

9.7.2. Configuring Group Alerts

Group alerts can only be set on compatible groups.

1. In the **Groups** tab of the top menu, click the **Compatible Groups** link.
2. In the main window, select the group to add the alert to.

3. Click the **Alerts** tab for the group.

4. In the **Definitions** subtab, click the **NEW DEFINITION** button.

5. Configure the basic alert definition and notifications, as in *Section 9.2, “Setting Alerts for a Resource”*.

9.8. Viewing Alerts

The alert history can be reviewed for a resource, a group of resources, a parent, or the whole JBoss ON server.

9.8.1. Searching for Fired Alerts

1. Select the **Overview** menu in the top navigation bar.

2. In the **Subsystem Views** menu, select **Alerts**.

3. In the **History** subtab, select the filters to use to search for the alerts. You can use wild cards \(*) and other expressions in the filter to narrow results.

   - *Resource*. The resource name or partial name.

   - *Parent*. The parent of any resources; this can be used to return alerts for all resources on a single parent or to return results for multiple servers with a filter such as *.example.com.

   - *Begin/end dates*. The date range for when the alerts were triggered.

   - *Category*. The type of alert that was sent.

4. Click **GO**.

All of the matching alerts for all resources in JBoss ON are listed in the results table.

Several results elements are useful for analysis:

1. The resource (**Name**)

2. The parent

3. The definition which triggered the alert

4. The condition which triggered the alert

5. The value of the resource at the time the alert was sent

6. The date, which is very useful for correlating the alert notification to an external event

9.8.2. Listing Alerts for a Specific Resource

1. Select the **Resources** menu in the top navigation bar, and select the **Servers** menu item.

2. Click the resource in the list.

3. Click the **Alerts** tab, and make sure that the **History** subtab is selected.

4. To show all alerts, just click the **GO** button. It is also possible to filter the alerts by the definition which triggered them, the priority, or date.

9.8.3. Viewing Alert Details

1. Select the **Resources** menu in the top navigation bar, and select the **Servers** menu item.
2. Click the resource in the list.
3. Click the **Alerts** tab, and make sure that the **History** subtab is selected.
4. Set any filters, and click the **GO** button.
5. In the results list, click the timestamp or alert definition name for the fired alert.
6. The new page displays all the details for the alert, including which alert definition was triggered, the conditions that triggered, and any operations that were launched as a result.

9.9. Acknowledging an Alert

Acknowledging an alert is a way of identifying that the condition which triggered the alert has been addressed in some way. When an alert is acknowledged, the name of the user who acknowledged the alert is recorded. Recording the acknowledger's name allows the action to be audited later if necessary.

1. Select the **Resources** menu in the top navigation bar, and select the **Servers** menu item.
2. Click the resource in the list.
3. Click the **Alerts** tab, and make sure that the **History** subtab is selected.
4. Set any filters, and click the **GO** button.
5. Select the checkbox by the alert to acknowledge.
6. Click the **ACKNOWLEDGE SELECTED** button.

NOTE

It is also possible to acknowledge a single alert through the alert details page.

9.10. Configuring SNMP for Notifications

JBoss ON can send SNMP traps to other management stations and systems as part of alerting notificatons. The data transmitted contain details about the alert, such as the name of the alert that was triggered and the resource name.

The data to include in the traps, as with other SNMP notifications, are defined in the JBoss ON MIB file, in *serverRoot*`/etc/RHQ-mib.txt`. The default configuration for the MIB is shown in *Example 9.1, “Default Alert Object in JBoss ON MIB”*. The base OID for the JBoss ON alert is `1.3.6.1.4.1.18016.2.1` (`org.dod.internet.private.enterprise.jboss.rhq.alert`).

```
alertGroup OBJECT-GROUP
    OBJECTS {   alertName,
                alertResourceName,
                alertPlatformName,
                alertCondition,
                alertSeverity,
                alertUrl }
    STATUS  current
    DESCRIPTION "A collection of objects providing information about an alert"
```

Example 9.1. Default Alert Object in JBoss ON MIB

With the default MIB file, each trap sends the alert definition name, resource name, platform, alert conditions, severity, and a URL to the alert details page.

Before JBoss ON can send any SNMP notifications, SNMP traps have to be configured for the server.

1. Configure the basic alert conditions and information for the resource, as described in *Section 9.2, “Setting Alerts for a Resource”*. Click **OK** to go to the next page to configure notifications.

2. The bottom of the page has the **Notifications** section. Click the **Edit** button to add a response.

3. Give the notification method a name, and select the **SNMP** method from the **Alert Senders** drop-down menu.

4. Select the new alert sender in the table. This opens the configuration for the notification.

5. Fill in the information for the SNMP trap.

   - The hostname for the SNMP manager.

   - The port number for the SNMP manager. JBoss ON supports UDP, so this must be the UDP port.

   - The JBoss ON OID. This is **`1.3.6.1.4.1.18016.2.1`**.

Managing Users and Roles

In JBoss Operations Network, security is implemented through rules that are set on *users and roles*. Restrictions are set on what users and roles can access and what operations they can perform.

This chapter covers how to create users and roles, to set basic access restrictions, and to use LDAP for user authentication and role membership.

10.1. About Security in JBoss ON: Roles and Access Control

In JBoss Operations Network, security is achieved by establishing precise relationships between users, resources, and the tasks users can perform. Interactions between users and resources are ordered by including or excluding those users and resources (through groups) in defined *roles*, and then granting the role the ability to perform tasks.

When a user is allowed to perform a certain operation, that is called a *permission*. All permissions must be explicitly granted to explicit resources. If a user is not given permission to a specific resource group, then the user, by default, has no access to that group — even if the user has permission to perform a task. Likewise, if a user has access to a group but has no permissions assigned, then the user cannot perform any tasks.

Any permissions set in JBoss ON are given to a role, and the members of the role inherit those permissions. Allowing or restricting permissions is *access control*.

In JBoss ON, there are two areas of where access control is given:

- *Global permissions* apply to JBoss ON server configuration. This covers administrative tasks, like creating users, editing roles, creating groups, importing resources into the inventory, or changing JBoss ON server properties.
- *Resource-level permissions* apply to actions that a user can perform on resources in the JBoss ON inventory. These cover actions like creating alerts, configuring monitoring, and changing resource configuration.

All JBoss ON permissions are listed in *Table 10.1, “JBoss ON Access Control Definitions”*.

Access Control Type	Description
Global - Security	Equivalent to a superuser. **Security** permissions grant the user the rights to create and edit any entries in JBoss ON, including other users, roles, and resources, to change JBoss ON server settings, and to control inventory.

Access Control Type	Description
	WARNING The **Security** access control level is extremely powerful, so be cautious about which users are assigned it. Limit the number of superusers to as few as necessary.
Global - Inventory	Allows any operation to be performed on any JBoss ON resource, including importing new resources. This is a global permission.
Global - Settings	Allows a user to add or modify any settings in the JBoss ON server configuration itself. This includes operations like deploying plug-ins or using LDAP authentication. This is a global permission.
Resource - Modify	Allows a user to change the resource definition entry in JBoss ON. This does not grant rights to edit the resource configuration.
Resource - Delete	Allows the user to delete the resource from the inventory.
Resource - Create Child	Allows the user to manually assign a child resource to another resource.
Resource - Alerts	Allows the user to create alerts and notifications on a resource. Configuring new alert senders changes the server settings and is therefore a function of the global **Settings** permissions.
Resource - Measurements	Allows the user to configure monitoring settings for the resource.
Resource - Content	Allows the user to manage content providers and repositories that are available to resources.
Resource - Control	Allows a user to run operations (which are also called *control actions*) on a resource.
Resource - Configure	Allows users to change the configuration settings on the resource through JBoss ON. **NOTE** The user still must have adequate permissions on the resource to allow the configuration changes to be made.

Table 10.1. JBoss ON Access Control Definitions

10.2. Managing Users in JBoss ON

Users are part of the overal security planning for JBoss ON, even though they don't have access controls set on their accounts individually.

10.2.1. Creating a New User

1. Click the **Administration** tab in the top menu.
2. In the **Security** menu, select the **Users** item.
3. Click the **NEW** button at the bottom of the list of current users.
4. Fill in description of the new user. The **Enable Login** value must be set to **Yes** for the new user account to be active.
5. Click the **OK** button to save the new user and assign the new user to roles.
6. In the **Roles** area, select the checkboxes by the roles to assign to the user, and then click the arrow pointing to the **Add to Roles** box.
7. Click **OK** to save the role assignments.

10.2.2. Editing User Details

All users can edit their own account details, and users with administrative rights (who belong to a role which grants them rights over user entries) can edit other users' entries.

10.2.2.1. Editing Your Personal Entry

1. In the upper right of the UI window, click the **Preferences** link. This opens your personal user account information.
2. In the edit user form, change whatever details need to be changed, and save.

10.2.2.2. Editing Another User Entry

1. Click the **Administration** tab in the top menu.
2. In the **Security** menu, select the **Users** item.
3. Click the name of the user whose entry will be edited.
4. In the edit user form, change whatever details need to be changed, and save.

10.2.3. Changing Passwords

1. Click the **Administration** tab in the top menu.

TIP

Alternatively, to change your own password, click the **Preferences** link in the upper right part of the UI window.

2. In the **Security** menu, select the **Users** item.

3. Click the name of the user whose entry will be edited.

4. Fill in the **Password** and **Verify** fields with the new password.

5. Click **Save**.

10.2.4. Disabling User Accounts

User accounts can be temporarily disabled. This can be done for a security review or when there is some kind of breach, but users don't need to be deleted. The **Enable Login** property can prevent the user from logging into the JBoss ON UI and managing resources or making configuration changes.

1. Click the **Administration** tab in the top menu.

2. In the **Security** menu, select the **Users** item.

3. Click the name of the user whose entry will be edited.

4. In the edit user form, change the **Enable Login** radio button to **No**.

The user account can be re-enabled at any time by changing the **Enable Login** value back to **Yes**.

10.2.5. Changing Role Assignments for Users

Role assignments can be changed after a user is created. To add a role:

1. Click the **Administration** tab in the top menu.

2. In the **Security** menu, select the **Users** item.

3. Click the name of the user to edit.

4. In the **Roles Assigned To** area, click the **Add to list** button to add a new role to the user.

5. Select the checkboxes by the roles to assign to the user, and then click the arrow pointing to the **Add to Roles** box.

6. Click **OK** to save the role assignments.

To delete a role from a user:

1. Click the **Administration** tab in the top menu.

2. In the **Security** menu, select the **Users** item.

3. Click the name of the user to edit.

4. In the **Roles Assigned To** area, click the checkbox by the roles to remove.

5. Click the **Remove from list** button.

WARNING

JBoss ON will not prompt you to confirm the deletion. It immediately removes the role from the user.

10.2.6. Configuring LDAP User Authentication

Authentication is the process of verifying the identity of an entity who attempts to access a server. JBoss ON uses simple authentication, meaning it uses simple username-password pairs to verify identity.

10.2.6.1. About JBoss ON and LDAP Authentication

By default, JBoss ON stores authentication information in its internal database. JBoss ON can also use an external LDAP repository to store this user information. With LDAP authentication, the JBoss ON server sends all login requests to the LDAP directory to process.

First, the JBoss ON server searches the LDAP directory for a matching username, and then it attempts to log into (bind to) the LDAP server using the given username and password. If the bind attempt is successful, then the user is successfully authenticated to the JBoss ON server.

After the JBoss ON server is configured to use LDAP for authentication, all login attempts are authenticated against the LDAP server.

WARNING

When the JBoss ON server is reconfigured to use LDAP for authentication, the LDAP information isn't validated yet. Any errors with the LDAP authentication configuration won't show up until a user attempts to log into the UI.

TIP

The LDAP directory can't create JBoss ON users automatically. However, using LDAP for authentication allows new users to register themselves to JBoss ON. A new user can authenticate to JBoss ON as long as they have an LDAP account. At their first login attempt, they're redirected to a registration page which records the additional JBoss ON user information.

The JBoss ON server constructs the LDAP entry name to look for based on the JBoss ON username and information about the LDAP directory, like the parent distinguished name in the directory tree and the naming attribute used for user entries; from there, it dynamically cosntructs a search filter every time someone logs into JBoss ON. Custom attributes can be added to the LDAP schema, such as **`JONUser=true`**, which can make it easier and more precise to locate entries.

JBoss ON can use two major directory servers for authentication:

- Microsoft Active Directory 2003

- Red Hat Directory Server 8.1

The LDAP directory *only* verifies the login credentials. The LDAP server doesn't store any other JBoss ON user data, and it doesn't create, delete, or edit entries in JBoss ON. Likewise, JBoss ON doesn't create, delete, or edit entries in the LDAP directory. The only attributes in the LDAP database that relate to JBoss ON user accounts are the username and password. Other settings in the JBoss ON user entry are stored in the JBoss ON internal database (like the user's first name and surname, email address, and role assignments).

NOTE

The LDAP directory is used only to check the login credentials — it doesn't store any other information about the JBoss ON users, including role assignments, and it cannot create a JBoss ON user. The JBoss ON server also cannot *create* LDAP users, so the LDAP user has to be created separately.

Because the LDAP directory doesn't store other attributes related to JBoss ON, it can't store a user certificate. This means that JBoss ON cannot use an LDAP directory for certificate-based authentication.

10.2.6.2. Configuring LDAP Authentication

1. Click the **Administration** tab in the top navigation menu.
2. In the **System Configuration** menu, select the **Settings** item.
3. In the main window, scroll to the **LDAP Configuration Properties** area.
4. Check the **Use LDAP Authentication** checkbox so that JBoss ON will use the LDAP user directory as its identity store.
5. Configure the connection settings to connect to the specific LDAP directory.
   - Give the LDAP URL of the LDAP server. This has the format **`ldap://`***hostname[:port]*. For example:

     ```
     ldap://server.example.com:389
     ```

     By default, this connects to the localhost over port 389 (standard LDAP port) or 636 (secure LDAP port, if SSL is selected).
   - To use a secure connection, check the **Use SSL** checkbox. When using SSL, make sure that the LDAP directory is actually running over SSL, and make sure that the connection URL points to the appropriate SSL port and protocol:

     ```
     ldaps://server.example.com:636
     ```
   - Give the bind credentials to use to connect to the server. The username is the full LDAP distinguished name of the user as whom JBoss ON binds to the directory.

> **NOTE**
>
> The user *must* exist in the LDAP directory before configuring the LDAP settings in JBoss ON. Otherwise, login attempts to the JBoss ON server will fail.
>
> Also, make sure that the JBoss ON user has appropriate read and search access to the user and group subtrees in the LDAP directory.

6. Set the search parameters that JBoss ON uses when searching the LDAP directory for matching user entries.

   - The *search base* is the point in the directory tree where the server begins looking for entries. If this is used only for user authentication or if all JBoss ON-related entries are in the same subtree, then this can reference a specific subtree:

     ```
     ou=user,ou=JON,dc=example,dc=com
     ```

     If the users or groups are spread across the directory, then select the base DN:

     ```
     dc=example,dc=com
     ```

   - Optionally, set a search filter to use to search for a specific subset of entries. This can improve search performance and results, particularly when all JBoss ON-related entries share a common LDAP attribute, like a custom *`JonUser`* attribute. The filter can use wild cards (**`objectclass=*`**) or specific values (**`JonUser=true`**).

   - Set the LDAP naming attribute; this is the element on the farthest left of the full distinguished name. For example, in **`uid=jsmith,ou=people,dc=example,dc=com`**, the far left element is **`uid=jsmith`**, and the naming attribute is *`uid`*.

     The default naming attribute in Active Directory is *`cn`*. In Red Hat Directory Server, the default naming attribute is *`uid`*.

7. Save the LDAP settings.

> **NOTE**
>
> The **Group Filter** and **Member Property** fields aren't required for user authentication. They're used for configuring LDAP groups to be assigned to roles, as in *Section 10.3.4, “Associating LDAP User Groups to Roles in JBoss ON”*.

10.3. Managing Roles

JBoss ON handles access to both resources and JBoss ON configuration through *roles*. A role has certain permissions assigned to it, meaning things that members of the role are allowed to do.

Only two types of JBoss ON identities can belong to a role: users and groups.

Users are assigned to a role to be granted those permissions. Users can be added to a role individually or be added as a member of an LDAP group.

Resource groups are assigned to a role to provide a list of resources that those users can perform actions on. Another way of looking at it is that users can only manage resources that they are expressly given access to. Roles define that access.

NOTE

Be sure to create resource groups to assign to any custom roles you create. If no resource groups are assigned to a role, then none of the members of the role can see any resources. Creating groups is described in *Section 2.5.6.1, “Defining Groups”*.

One convenient feature of roles is that users can be automatically assigned to roles by assigning an LDAP group to the role (*Section 10.3.4, “Associating LDAP User Groups to Roles in JBoss ON”*). All of the LDAP users who belong to that group are automatically members of the role. (This is similar to the simplicity of using LDAP user to create JBoss ON users by enabling LDAP authentication, in *Section 10.2.6, “Configuring LDAP User Authentication”*.)

There are two roles already configured in JBoss ON by default:

- A superuser role provides complete access to everything in JBoss ON. This role cannot be modified or deleted. The user created when the JBoss ON server was first installed is automatically a member of this role.
- An *all resources* role exists that provides full permissions to every resource in JBoss ON (but not to JBoss ON administrative functions like creating users). This is a useful role for IT users, for example, who need to be able to change the configuration or set up alerts for resources managed by JBoss ON but who don't require access over JBoss ON server or agent settings.

10.3.1. Creating a New Role

NOTE

Be sure to create resource groups to assign to any custom roles you create. If no resource groups are assigned to a role, then none of the members of the role can see any resources. Creating groups is described in *Section 2.5.6.1, “Defining Groups”*.

1. Create any resources groups which will be associated with the role. Creating groups is described in *Section 2.5.6.1, “Defining Groups”*.

   By default, JBoss ON uses only *resource groups* to associate with a role, and these are required. However, optional user groups from an LDAP directory can also be assigned to a role, so that the group members are automatically treated as role members. *LDAP groups* must be configured in the server settings, as described in *Section 10.3.4, “Associating LDAP User Groups to Roles in JBoss ON”*.

2. In the top menu, click the **Administration** tab.

3. In the **Security** menu, select the **Roles** item.

4. The list of current roles comes up in the main task window. Click the **Add Role** button at the bottom of the list.

5. Give the role a descriptive name. This makes it easier to manage permissions..

6. Set permissions for the role. There are two types of permissions:

   * *Global permissions* grant permissions to areas of the JBoss ON server and configuration.

   * *Resource permissions* grant permissions for managing resources.

   The specific access permissions are described in *Table 10.1, “JBoss ON Access Control Definitions”*.

7. In the **Assigned Groups** area, the table on the left lists available groups, and the table on the right lists assigned groups.

   Select the groups in the table on the left to assign to the role. Click the right arrow (**=>**) to move the selected groups to the assigned table.

8. At the bottom, click the **Save** button.

9. Click the arrow in the upper right to close the create window.

10.3.2. Changing the Groups Assigned to Roles

1. In the top menu, click the **Administration** tab.

2. In the **Security** menu, select the **Roles** item.

3. The list of current roles comes up in the main task window. Click the name of the role to edit.

   The role's edit form opens beneath the list of roles.

4. In the **Assigned Groups** area, the table on the left lists available groups, and the table on the right lists assigned groups.

   Select the groups in either table to change the role assignments, and use the arrows to move the groups between the available and assigned tables.

5. At the bottom, click the **Save** button.

10.3.3. Changing the Access Controls for a Role

1. In the top menu, click the **Administration** tab.

2. In the **Security** menu, select the **Roles** item.

3. The list of current roles comes up in the main task window. Click the name of the role to edit.

   The role's edit form opens beneath the list of roles.

4. Edit the permissions for the role. All permissions can be added or removed; the specific access permissions are described in *Table 10.1, “JBoss ON Access Control Definitions”*.

5. At the bottom, click the **Save** button.

10.3.4. Associating LDAP User Groups to Roles in JBoss ON

Authentication is the process of verifying someone's identity. *Authorization* is the process of determining what access permissions that identity has. Users are authorized to perform tasks based on the permissions granted to their role assigments.

Section 10.2.6, “Configuring LDAP User Authentication” describes how an LDAP directory server can be used to authenticate users to JBoss ON. Any time a user attempts to log in, that request — with the username and password — is simply forwarded to the specified LDAP directory server to see if the credentials are correct.

Members of LDAP groups can be pulled in, automatically, as members of JBoss ON roles. The LDAP group is associated with a JBoss ON role and then the group members are *authorized* to do whatever the JBoss ON role is configured to allow. Any changes made in the LDAP group members are automatically reflected in JBoss ON, without having to edit the JBoss ON role.

JBoss ON supports two major directory servers for group authorization:

- Microsoft Active Directory 2003
- Red Hat Directory Server 8.1

10.3.4.1. About JBoss ON Roles and LDAP User Groups

Many LDAP directories already contain organizational groups with users who will need to access resources in JBoss ON. Configuring JBoss ON to connect to these directories allows JBoss ON to assign LDAP groups to roles and then pull in those member lists dynamically, so the roles are populated with pre-existing member lists. All of the LDAP users automatically inherit the permissions of that role.

In the role details page, these LDAP user groups are separated from the resource groups, so it's easy to distinguish which types of group are being added to the role.

JBoss ON determines what LDAP groups a user belongs to with a simple search. Whenever a user logs into JBoss ON and an LDAP connection is configured, JBoss ON maps that JBoss ON username to a user entry in the LDAP directory server. The specific LDAP distinguished name (DN) for the user is used as part of a search to find matching member attributes in LDAP group entries. That is, the LDAP server can check the member lists in group entries to see what groups the person with that DN belongs to.

For LDAP groups to be added to roles, three things are required:

- An LDAP directory server connection has to be configured.
- There has to be an LDAP attribute given to search for *group entries*.

  For Active Directory, this is generally the **`group`** object class. For Red Hat Directory Server, this is generally **`groupOfUniqueNames`**. Other standard object classes are available, and it is also possible to use a custom, even JBoss ON-specific, object class.
- There has to be an LDAP attribute given to identify *members* in the group.

  Common member attributes are *`member`* and *`uniqueMember`*.

JBoss ON constructs an LDAP search based on the group object class and member attribute in the server configuration, plus the DN of the user given when the user logs in.

```
(&(group_filter)(member_attribute=user_DN))
```

For example, this looks for the *member* attribute on an Active Directory group:

```
ldapsearch -h server.example.com -x -D "cn=Administrator,cn=Users,dc=example,dc=com"
 -W -b "dc=example,dc=com" -x '(&(objectclass=group)(member=CN=John
 Smith,CN=Users,DC=example,DC=com))'
```

Red Hat Directory Server uses the *uniqueMember* attribute on **groupOfUniqueNames** groups more commonly than *member* and *group*. For example:

```
/usr/lib64/mozldap6/ldapsearch -D "cn=directory manager" -w secret
 -p 389 -h server.example.com -b "ou=People,dc=example,dc=com" -s
 sub "(&(objectclass=groupOfUniqueNames)
(uniqueMember=uid=jsmith,ou=People,dc=example,dc=com))"
```

This search returns a list of all groups to which the user is a member. If any of these LDAP groups is assigned to a JBoss ON role, then that user is also automatically a member of that JBoss ON role.

TIP

Using custom LDAP group object classes can allow you to be very specific about which groups to use for JBoss ON roles.

10.3.4.2. Configuring LDAP User Groups

1. Click the **Administration** tab in the top navigation menu.
2. In the **System Configuration** menu, select the **Settings** item.
3. In the main window, scroll to the **LDAP Configuration Properties** area.
4. Configure the connection settings to connect to the specific LDAP directory.
   - Give the LDAP URL of the LDAP server. This has the format **ldap://***hostname[:port]*. For example:

     ```
     ldap://server.example.com:389
     ```

     By default, this connects to the localhost over port 389 (standard LDAP port) or 636 (secure LDAP port, if SSL is selected).

   - To use a secure connection, check the **Use SSL** checkbox. When using SSL, make sure that the LDAP directory is actually running over SSL, and make sure that the connection URL points to the appropriate SSL port and protocol:

     ```
     ldaps://server.example.com:636
     ```

   - Give the bind credentials to use to connect to the server. The username is the full LDAP distinguished name of the user as whom JBoss ON binds to the directory.

NOTE

The user *must* exist in the LDAP directory before configuring the LDAP settings in JBoss ON. Otherwise, login attempts to the JBoss ON server will fail.

Also, make sure that the JBoss ON user has appropriate read and search access to the user and group subtrees in the LDAP directory.

5. Set the search parameters that JBoss ON uses when searching the LDAP directory for matching user entries.

   - The *search base* is the point in the directory tree where the server begins looking for entries. If all JBoss ON-related entries are in the same subtree, then this can reference a specific subtree:

     ```
     ou=user,ou=JON,dc=example,dc=com
     ```

     If the users or groups are spread across the directory, then select the base DN:

     ```
     dc=example,dc=com
     ```

   - Optionally, set a search filter to use to search for a specific subset of entries. This can improve search performance and results, particularly when all JBoss ON-related entries share a common LDAP attribute, like a custom *`JonUser`* attribute. The filter can use wild cards (**`objectclass=*`**) or specific values (**`JonUser=true`**).

   - Set the LDAP naming attribute; this is the element on the farthest left of the full distinguished name. For example, in **`uid=jsmith,ou=people,dc=example,dc=com`**, the far left element is **`uid=jsmith`**, and the naming attribute is *`uid`*.

     The default naming attribute in Active Directory is *`cn`*. In Red Hat Directory Server, the default naming attribute is *`uid`*.

6. Set the parameters to use for the server to use to search for LDAP groups and their members.

   The search filter that JBoss ON constructs looks like this:

   ```
   (&(group_filter)(member_attribute=user_DN))
   ```

   - The **Group Filter** field sets how to search for the group entry. This is usually done by specifying the type of group to search for through its object class:

     ```
     (objectclass=groupOfUniqueNames)
     ```

   - The **Member Property** field gives the attribute that the specified group type uses to store member distringuished names. For example:

     ```
     uniqueMember
     ```

   The *user_DN* is dynamically supplied by JBoss ON when a user logs into the UI.

7. Save the LDAP settings.

General Management

This chapter covers the configuration, files, and options for the JBoss Operations Network server and agents.

11.1. JBoss ON File Locations

This section covers the common files and directories by JBoss ON servers and agents. A basic reference for these files can make managing and troubleshooting JBoss ON easier.

11.1.1. JBoss ON Server File Locations

All JBoss Operations Network servers are installed in a single, user-defined server root directory. In the documentation and examples, this is called *serverRoot*. The directory layout within that server root directory are the same for every server.

```
                              serverRoot
                                 |
                                jon
                                 |
      ----------------------------------------------------------
      |          |     |      |       |        |       |       |
alert-scripts/  bin/  etc/  EULA  jbossas/  LICENSE  logs/  plugins/
```

The directories and files that are most commonly used to managed JBoss ON servers are listed in *Table 11.1, "JBoss ON Server Directories and Files"*. The server root vaires for each installation and each platform, but the layout of the JBoss ON subdirectories is the same for every platform.

Configuration Area	Directory or File Location	Description
Configuration directory	*serverRoot*/jon/bin/	Contains the server start scripts, PID files, and configuration file.
Start scripts	*serverRoot*/jon/bin/rhq-server{.sh|.bat}	The script to start, stop, and check the status of the server.
Configuration file	*serverRoot*/jon/bin/rhq-server.properties	The configuration file for all server settings that are not stored in the JBoss ON database.
Password hash script	*serverRoot*/jon/bin/generate-db-password{.sh|.bat}	For a migrated server, it generates an encoded form of the database password to use in the **`rhq-server.properties`** file.
SNMP files	*serverRoot*/jon/etc/RHQ-mib.txt	THe JBoss ON MIB file to use for setting SNMP traps.
Log files	*serverRoot*/jon/logs/	The JBoss ON server log files are automatically created in this directory. The current log is named **`rhq-server-log4j.log`**. Older log files

Configuration Area	Directory or File Location	Description
		are named **`rhq-server-log4j.log`**.#, and the higher the number, the older the log file.
Custom plug-in deployment directory	*serverRoot*/jon/plugins/	The directory where custom plug-in files can be dropped for them to be automatically detected and polled by the JBoss ON server.
JBoss AS directory	*serverRoot*/jon/jbossas/	Contains all of the required JBoss AS client and server libraries.[1]
Server JAR files	*serverRoot*/jon/jbossas/default/deploy/rhq.ear/	Contains all of the JAR files used by JBoss ON servers, web interface, and clients.
Server-side plug-ins directory	*serverRoot*/jon/jbossas/default/deploy/rhq.ear/rhq-serverplugins/	Contains all of the JAR files for the default JBoss ON server-side plug-ins.
Agent plug-ins directory	*serverRoot*/jon/jbossas/default/deploy/rhq.ear/rhq-downloads/rhq-plugins/	Contains all of the JAR files for the default JBoss ON agent plug-ins.
Server-side plug-ins directory	*serverRoot*/jon/jbossas/default/deploy/rhq.ear/rhq-serverplugins/	Contains all of the JAR files for the default JBoss ON server-side plug-ins.
Agent package directory	*serverRoot*/jon/jbossas/default/deploy/rhq.ear/rhq-downloads/rhq-agent/	Contains the snapshot packages for the JBoss ON agent.
Web interface directory	*serverRoot*/jon/jbossas/default/work/jboss.web/localhost/	Contains the directories that hold the files for rendering the web interface.

Most of the libraries and files in this directory don't relate directly to JBoss ON.

Table 11.1. JBoss ON Server Directories and Files

11.1.2. JBoss ON Agent File Locations

Like the server, the JBoss ON agent is installed in a single, user-defined root directory. All of the agent files and directories are under the **`rhq-agent/`** directory in that root directory.

```
                 serverRoot
                     |
                rhq-agent/
                     |
     ------------------------------------
     |      |      |      |      |       |
   bin/   conf/  data/  lib/   logs/  plugins/
```

Configuration Area	Directory or File Location	Description
Start scripts	*serverRoot*/rhq-agent/bin/	Contains the agent start scripts.
Configuration file	*serverRoot*/rhq-agent/conf/agent-configuration.xml	The configuration file for basic agent settings.
Library files	*serverRoot*/rhq-agent/lib/	Contains the libraries used by the agent to monitor resources.
Start scripts	*serverRoot*/rhq-agent/logs/	The JBoss ON agent log files are automatically created in this directory. The current log is named **`agent.log`**. Older log files are named **`agent.log`**.#, and the higher the number, the older the log file.
Plug-ins directory	*serverRoot*/rhq-agent/plugins/	Contains the plug-ins used by the agent for managing resoiurces (like editing resource configuration).

Table 11.2. JBoss ON Agent Directories and Files

11.2. Default Server and Agent Ports

As with other servers and services, JBoss ON servers and agents communicate with each other by connecting over system ports. JBoss ON uses ports for three types of connections:

- Server to database communication

  The server has to be able to connect to its database. The database port number depends on both the type of database and the specific configuration for the database.

- Server to agent communication

  The server connects to an agent over a single port configured for the agent. This port is used for both standard and SSL communications between the server and agent.

- Agent to server communication

  An agent can talk to multiple JBoss ON servers, even if they use the same port (since each server is on a different host.) The agent will use either a standard port or an SSL port to connect to the JBoss ON server, depending on the connection (transport) method that is configured. The agent will only attempt to use a single port.

NOTE

Servers do not talk to one another directly, so there are no ports for server-to-server links.

The default port numbers for JBoss ON connections are listed in *Table 11.3, “Default JBoss ON Ports”*. The port numbers can be changed for any of the JBoss ON services or different values can be used at installation.

Connection Type	Port Number
Server to agent	16163
Agent to server (standard)	7080
Agent to server (secure)	7443
Server to database	5432 (default PostgreSQL port)

Table 11.3. Default JBoss ON Ports

11.3. Starting the JBoss ON Server

The JBoss ON server is actually a customized JBossAS server, included in the JBoss ON installation, so starting the JBoss ON server means starting that JBoss instance.

The JBoss ON server can be started manually or can be configured to start and run as a system service.

11.3.1. Starting the JBoss ON Server (Basic)

The JBoss ON server process is started by running a scriptin the *serverRoot*`/bin/` directory. There is an `.sh` script for Linux and Unix systems and a `.bat` script for Windows systems.

The simplest way to start the server is simply to run the script with the **`start`** command. This starts the server process and then exits from the script.

```
serverRoot/bin/rhq-server.{sh|bat} start
Trying to start the RHQ Server...
RHQ Server (pid 27547) is starting
```

The **`rhq-server.{sh|bat}`** script looks for specific environment variables during its execution, especially related to the JVM to use with the JBoss AS server instance. A complete list of environment variables is given in the script itself; defaults based on the installation information are used, so most environment variables don't need to be reset.

NOTE

The RHQ_SERVER_JAVA_HOME environment variable must be set on Red Hat Enterprise Linux systems for the server to start. This can be set to a general value like **`/usr/`**.

The server can also be started in console mode, which prints detailed information about the server process to the terminal and leaves the script open as long as the server is running.

```
serverRoot/bin/rhq-server.{sh|bat} console

Starting RHQ Server in console...
=========================================================================

  JBoss Bootstrap Environment

  JBOSS_HOME: serverRoot/jon-server-2.4.0.Beta1/jbossas

  JAVA: /usr/bin/java
  JAVA_OPTS: -Dprogram.name=run.sh -Dapp.name=rhq-server -Xms1024M -Xmx1024M -XX:PermSize=256M
 -XX:MaxPermSize=256M -Djava.net.preferIPv4Stack=true -Djboss.server.log.dir=serverRoot/
jon-server-2.4.0.Beta1/logs -Djava.awt.headless=true -Djboss.platform.mbeanserver -
Dsun.lang.ClassLoader.allowArraySyntax=true -Djava.util.logging.config.file=serverRoot/
jon-server-2.4.0.Beta1/jbossas/server/default/conf/logging.properties  -
Djava.net.preferIPv4Stack=true

  CLASSPATH: serverRoot/jon-server-2.4.0.Beta1/jbossas/bin/run.jar

=========================================================================

15:51:45,955 INFO  [Server] Starting JBoss (MX MicroKernel)...
15:51:45,956 INFO  [Server] Release ID: JBoss [Trinity] 4.2.3.GA (build: SVNTag=JBoss_4_2_3_GA
 date=200807181417)
15:51:45,957 INFO  [Server] Home Dir: serverRoot/jon-server-2.4.0.Beta1/jbossas
15:51:45,957 INFO  [Server] Home URL: file:serverRoot/jon-server-2.4.0.Beta1/jbossas/
15:51:45,957 INFO  [Server] Patch URL: null
15:51:45,958 INFO  [Server] Server Name: default
15:51:45,958 INFO  [Server] Server Home Dir: serverRoot/jon-server-2.4.0.Beta1/jbossas/server/
default
15:51:45,958 INFO  [Server] Server Home URL: file:serverRoot/jon-server-2.4.0.Beta1/jbossas/
server/default/
15:51:45,958 INFO  [Server] Server Log Dir: serverRoot/jon-server-2.4.0.Beta1/logs
15:51:45,958 INFO  [Server] Server Temp Dir: serverRoot/jon-server-2.4.0.Beta1/jbossas/server/
default/tmp
15:51:45,958 INFO  [Server] Root Deployment Filename: jboss-service.xml
15:51:46,183 INFO  [ServerInfo] Java version: 1.6.0_15,Sun Microsystems Inc.
15:51:46,183 INFO  [ServerInfo] Java VM: Java HotSpot(TM) Server VM 14.1-b02,Sun Microsystems
 Inc.
15:51:46,184 INFO  [ServerInfo] OS-System: Linux 2.6.18-164.15.1.el5,i386
15:51:46,377 INFO  [Server] Core system initialized
....
```

11.3.2. Running the JBoss ON Server as a Service

The JBoss ON server can be configured to run as a service, managed with systems tools, on both Red Hat Enterprise Linux and Windows.

11.3.2.1. Configuring the JBoss ON Server as a Service on Red Hat Enterprise Linux

The **rhq-server.sh** script can be managed by the **init** process so that the server starts automatically when the system boots. This also allows the server process to be managed by services like **service** and **chkconfig**.

1. Copy the **rhq-server.sh** script into the **/etc/init.d/** directory.

   ```
   cp serverRoot/bin/rhq-server.sh /etc/init.d/
   ```

2. Edit the **/etc/init.d/rhq-server.sh** script to set the RHQ_SERVER_HOME variable to the JBoss ON server install directory and the RHQ_SERVER_JAVA_HOME variable to the appropriate directory for the JVM. For example:

   ```
   RHQ_SERVER_HOME=serverRoot/jon-server-2.4.0.Beta1/
   ```

   ```
   RHQ_SERVER_JAVA_HOME=/usr/
   ```

3. Edit the **`/etc/init.d/rhq-server.sh`** script, and add the following lines to the top of the file, directly under `#!/bin/sh`.

   ```
   #!/bin/sh
   #chkconfig: 2345 95 20
   #description: JBoss Operations Network Server
   #processname: run.sh
   ```

   The last two numbers in the **`#chkconfig: 2345 95 20`** line specify the start and stop priority, respectively, for the JBoss ON server.

4. Add the service to the **`chkconfig`** service management command, and verify that it was added properly.

   ```
   chkconfig --add rhq-server.sh
   chkconfig rhq-server.sh --list
   ```

5. Set the **`rhq-server.sh`** service to run at run level 5.

   ```
   chkconfig --level 5 rhq-server.sh on
   ```

Once the **`init`** scripts and **`chkconfig`** files are updated, then the JBoss ON server can be started and stopped using the **`service`**command. The status of the process can also be checked.

```
service rhq-server.sh {start|stop|status}
```

11.3.2.2. Configuring JBoss ON as a Windows Service

The **`rhq-server.bat`** script has an installation option that installs the script as a Windows service. Once installed, the JBoss ON server can be started, stopped, and managed through Windows tools (Add and Remove Programs and Services) or through the **`rhq-server.bat`** script.

1. Set the environment variable to run the Windows service as.

   Every Windows service has to run as some system user. There are two environment variables in the **`rhq-server.bat`** script that set the user to use:

   • `RHQ_SERVER_RUN_AS` sets any Windows user to be the JBoss ON server user. The username given here must be in the standard Windows format, *DOMAIN\user*, such as **`EXAMPLEDOMAIN\jsmith`**.

   • `RHQ_SERVER_RUN_AS_ME` sets the server to run as whoever the current user is. This overrides the `RHQ_SERVER_RUN_AS`, if both as set.

   If neither environment variable is set, then the JBoss ON server runs as the system account.

2. Run the **`rhq-server.bat`** script with the **`install`** option to set up the service. This prompts for the password of whatever user account is used for the JBoss ON service.

   ```
   serverRoot\bin\rhq-server.bat install
   ```

After the service is set up, the JBoss ON server can be started or stopped using Windows adminsitrative tools or by using any of the options in *Table 11.4, “rhq-server.bat Options”* with the script.

Option	Description
start	Starts the server service.
stop	Stops the server service.
status	Prints the current status (running or stopped) of the service.
remove	Removes, or uninstalls, the JBoss ON server service.

Table 11.4. rhq-server.bat Options

The JBoss ON server Windows service can be modified by changing or adding parameters in the service wrapper configuration file, *serverRoot*`\bin\wrapper\rhq-server-wrapper.conf`. *Table 11.5, “Common Wrapper Properties”* lists some of the wrapper properties that are most commonly edited.

NOTE

Before editing the wrapper file, check out the list of properties in the Java Service Wrapper documentation at *http://wrapper.tanukisoftware.org/doc/english/properties.html*.

Parameter	Description
wrapper.app.parameter.#	Passes command-line options to the server (the JBoss AS container). Each individual option and its value must be given its own wrapper configuration property and must be placed in numerical order. **IMPORTANT** Do not change any of the five default properties, **`wrapper.app.parameter.1`**. The number for new properties must begin at 5.
wrapper.java.additional.#	Passes additional options to the virtual machine, such as **`-Xmx`** or **`-D`**. Increments the parameters upward numerically.

Parameter	Description

	IMPORTANT
	Do not edit the **`wrapper.java.additiona.1`** property unless you want to point to your own log configuration file. Any other properties can be added, removed, or modified.
	
	For example:
	
	`wrapper.java.additional.5=-XX:`
	`+DisableExplicitGC`

wrapper.ntservice.starttype	Sets the start type, either automatically when the system boots (**`AUTO_START`**) or manually (**`DEMAND_START`**).

Table 11.5. Common Wrapper Properties

Alternatively, the wrapper service can be configured by creating a wrapper include file, in the *serverRoot*`\bin\wrapper\rhq-server-wrapper.inc`. An include file augments the service wrapper configuration file and is the recommended way to add more Java VM.

11.3.3. Starting the JBoss ON Server in Debug Mode

Debug mode records debugging messages caused or encountered by the start scripts to the server logs. There are two ways to enable debug mode for the server start script:

- Modify the standard **`jboss-log4j.xml`** file located in the *serverRoot*`/jbossas/server/default/conf` directory.

  WARNING

  Do not modify anything else in the **`jbossas`** directory. This could adversely affect the performance of the JBoss ON server.

- Set the environment variable `RHQ_SERVER_DEBUG` to any value. To disable *debug* mode, unset the environment variable.

11.4. Starting the JBoss ON Agent

The agent is started and runs using a script in the agent's **`bin/`** directory. Unlike the server start script, which starts the server process and then exits the script, the agent script remains open, with a prompt to accept further input if necessary. (Usually, the script can simply be started and left to run in the boackground.)

```
/opt/rhq-agent/bin/rhq-agent.sh
```

```
RHQ 2.4.0-SNAPSHOT [cda7569] (Tue Apr 13 13:39:16 EDT 2010)
>
```

Most of the time, the JBoss ON agetn can run without any additional options or settings. All of the available options for the **`rhq-agent.sh`** script are listed in *Table 11.6, "Options for the rhq-agent.sh Script"*.

TIP

If there are any errors when starting the JBoss ON agent, run the agent start script with the **`--cleanconfig`** to wipe the previous agent configuration and start fresh.

Short Argument	Long Argument	Description
-a	--advanced	Runs the agent script in setup mode, rather than basic start mode.
-c	--config=*filename*	Specifies an agent configuration preferences file on filesystem or classpath.
-d	--daemon	Runs the agent in daemon mode, which means it will not read additional commands from stdin.
-D*name*[=*value*]		Overrides an agent configuration preference and sets a system property.
-e	--console=*type*	Specifies the implementation to use when reading console input. The three available values are jline, sigar, and java.
-h	--help	Opens the help message.
-i	--input=*filename*	Specifies a script file to use for input.
-l	--cleanconfig	Clears out any existing configuration and data files so the agent starts with a totally clean slate.
-n	--nostart	Runs the agent script without starting the agent process.
-o	--output=*filename*	Specifies a file to write all output from the scritp, excluding log messages (which are always written to the agent logs).
-p	--pref=*preferences_name*	Specifies the agent preferences name used to identify what configuration to use.

Short Argument	Long Argument	Description
-s	--setup	Forces the agent to ask setup questions.
-t	--nonative	Forces the agent to disable the native system.
-u	--purgedata	Purges persistent inventory and other data files.
--		Stops the agent from processing options.

Table 11.6. Options for the rhq-agent.sh Script

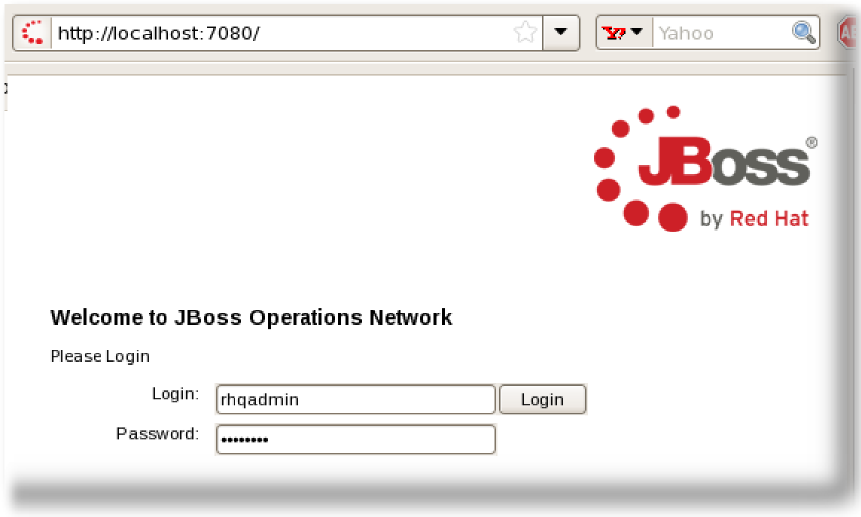
11.5. Logging into the JBoss ON Web UI

Aside from some minor configuration in its **`rhq-server.properties`** file, JBoss ON is completely administered through its web interface.

By default, the JBoss ON server listens over port 7080. (A different port can be configured when the server is installed, and the port number can be changed in the server configuration.) To connect to the server, then, simply open a standard HTTP page with a URL in the format *hostname:port*. For example:

```
http://server.example.com:7080
```

Then, log in using any valid username/password combination. The default administrative user has the name and password **`rhqadmin`**.


http://localhost:7080/

Welcome to JBoss Operations Network

Please Login

Login: rhqadmin

Password: ••••••••

Figure 11.1. Logging into JBoss ON

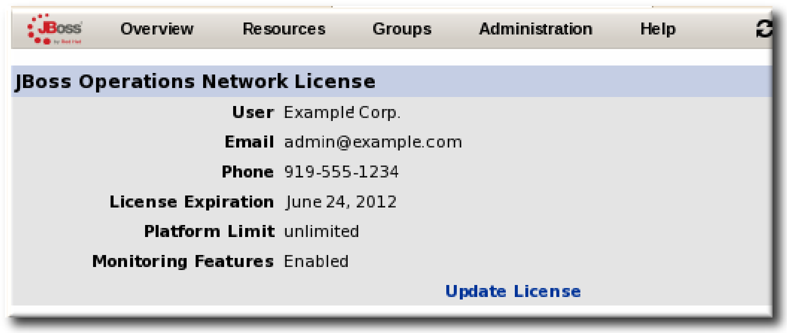
NOTE

By default, JBoss ON only runs over standard HTTP. SSL must be specially configured to open a secure connection over HTTPS. For HTTPS, JBoss ON uses the connection port 7443.

11.6. Updating the JBoss ON License

The license for using JBoss ON was uploaded the first time a user attempted to log into the web UI after the JBoss ON server was installed. The license has an expiration date, so it has to be updated at the end of your subscription period.

The details for the current JBoss ON license are listed with the system configuration. The details page has all of the information contained in the license, including the contact information for the primary JBoss ON owner, the expiration date, the maximum number of platforms (server boxes) that can be added to the inventory, and whether monitoring is enabled under the license.



JBoss Operations Network License

User	Example Corp.
Email	admin@example.com
Phone	919-555-1234
License Expiration	June 24, 2012
Platform Limit	unlimited
Monitoring Features	Enabled

Update License

Figure 11.2. JBoss ON License Details

To update the license:

1. Download the new license from the *Customer Support Portal*[1].

2. Log into the JBoss ON UI.

```
https://server.example.com:7080/
```

3. Click the **Administration** tab in the top menu.

4. In the **System Configuration** menu, select the **License** link.

5. Under the details of the current license, click the **Update license** link.

6. Browse to the saved license file, and then click **OK** to upload the new license.

11.7. Managing Databases Associated with JBoss ON

There are several basic tasks that can be done to manage the Oracle or PostgreSQL databases that are used by the JBoss ON server.

11.7.1. Running SQL Commands from JBoss ON

SQL commands can be run through the JBoss ON web UI on any database that the JBoss ON server is using for its data.

NOTE

The database management page is not accessible through the normal JBoss ON GUI. Administrators must manually navigate to the admin area of the JBoss ON UI.

NOTE

Whatever JBoss ON user you are logged in as must have adequate user rights *on the database* to execute the SQL commands.

1. Open the administrative page, with the location **`admin/test/sql.jsp`**. For example:

   ```
   http://server.example.com:7080/admin/test/sql.jsp
   ```

2. Enter the SQL commands, as appropriate for the JBoss ON Oracle or PostgreSQL database instance. If there are multiple commands, make sure the **Continue if statements fail?** checkbox is selected. That way, even if one command fails, the other commands will be submitted. Otherwise, the series will be terminated at the first failure.

3. Click the **Execute SQL** button.

11.7.2. Changing Database Passwords

The JBoss ON server connects to its database instance as a database user. It does this automatically, using the credentials given in its **`rhq-server.properties`** file. The database password is encoded automatically by the installer before it is stored in the **`rhq-server.properties`** file, to provide some extra protection against unautohrized access to the database password.

It's possible that the password for that database user account is changed. This change always occurs at the database, not in JBoss ON, so the password in the **`rhq-server.properties`** file has to be manually encoded and updated for JBoss ON to continue to function.

1. Change the password for the JBoss ON user (**`rhqadmin`** by default) in the database.

2. Use the **`generate-db-password.sh`** script to hash the password.

   ```
   serverRoot/bin/generate-db-password.sh mypassword
   Encoded password: 1d31b70b3650168f79edee9e04977e34
   ```

   JBoss ON stores its database password in an encoded form in the **`rhq-server-properties`** file. Therefore, the new database has to be properly encoded before it's added to the **`rhq-server-properties`** file so that the server reads it correctly.

3. Edit the *`rhq.server.database.password`* value in the **`rhq-server.properties`** file so that it has the new encoded password value.

   ```
   vim serverRoot/bin/rhq-server.properties

   rhq.server.database.password=1d31b70b3650168f79edee9e04977e34
   ```

11.8. Configuring Servers

The JBoss ON configuration is edited in one of two areas, depending on the configuration setting:

- In the JBoss ON GUI

NOTE

Settings that can be edited in the JBoss ON UI *must* be edited in the JBoss ON UI.

- In the **`rhq-server.properties`** configuration file

Additional configuration is stored in the database backend used by the JBoss ON server.

11.8.1. Changing the JBoss ON Server URL

The server URL is the URL used to identify and connect to the server in two ways:

- Connecting to the GUI
- Details on alerts, contained in email notifications of alerts

The server URL does not need to be changed unless the JBoss ON connects to the Internet through a proxy server.

1. Click the **Administration** tab in the top menu.
2. In the **System Configuration** menu, select the **Settings** item.
3. Scroll to the **JON General Configuration Properties** section in the main work area.
4. Change the hostname or IP address in the **GUI Console URL** field.
5. Scroll to the bottom of the page, and click **OK**.

11.8.2. Configuring Agent Update Settings

The JBoss ON server configuration defines two settings for how the server sends updates to the JBoss ON agents. One setting sets how long the server waits to hear from the agent before it considers that the agent is down. The second enables automatic updates for agent configuration.

1. Click the **Administration** tab in the top menu.
2. In the **System Configuration** menu, select the **Settings** item.
3. Scroll to the **JON General Configuration Properties** section in the main work area.
4. Change the agent settings:
   - **Agent Max Quiet Time Allowed** sets the longest interval that the server waits for agent availability reports before marking the agent as down.
   - **Enable Agent Auto-Updates** sets whether the agent is allowed to check, itself, for updates to configuration or new agent JAR files.
5. Scroll to the bottom of the page, and click **OK**.

11.8.3. Configuring High Availability

High availability with JBoss ON servers means that all JBoss ON servers which use the same, central database interact together in a cloud. This allows seamless failover between servers when a server has to be taken offline for maintenance, and it provides a natural method for load balancing agent and resource operations.

Agents are assigned, or *partitioned*, among servers in the high availability cloud. Agents aren't strictly assigned to a server to be managed. Agents are handled by servers first based on a preference, or *affinity*, for a server because they belong to to the same affinity group. After affinity, or if an affinity group isn't configured, then agents are managed by the server in a round-robin style for availability.

11.8.3.1. Putting Servers in Maintenance Mode

Putting a JBoss ON server in maintenance mode temporarily removes it from the high availability cloud so it no longer processes agent operations. This is useful when the server is offline for upgrades or because of a service interruption.

1. Click the **Administration** tab in the top menu.
2. In the **High Availability** menu, select the **Server** item.
3. Select the check box next to the name of the JBoss ON server to put into maintenance mode.
4. Click the **SET MAINTENANCE** button.

The JBoss ON server can be added back to the high availability cloud by clicking **NORMAL** button. Agents migrate back incrementally.

11.8.3.2. Removing Servers from the High Availability Cloud

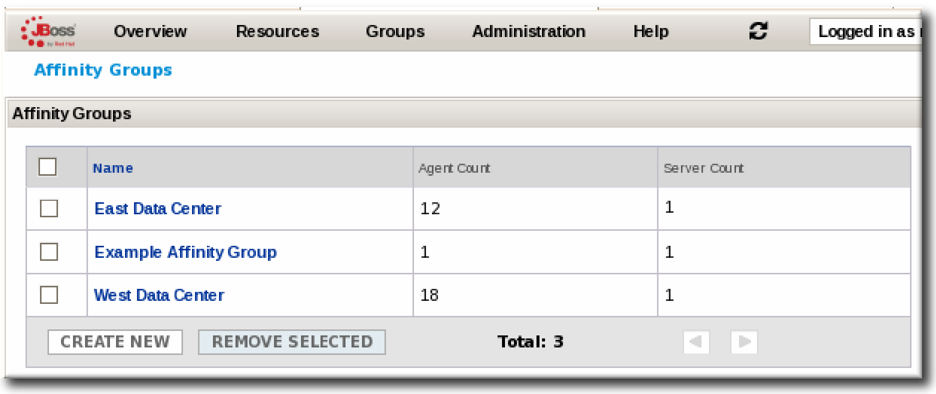
A JBoss ON server that is in maintenance mode can be permanently removed from the high availability cloud.

1. Click the **Administration** tab in the top menu.
2. In the **High Availability** menu, select the **Servers** item.
3. Select the check box next to the name of the JBoss ON server to remove from the cloud. This server must already be in maintenance mode.
4. Click the **REMOVE SELECTED** button.

11.8.3.3. Creating Affinity Groups

An *affinity group* sets a preference for which JBoss ON servers manage which JBoss ON agents. An affinity group only sets preference, not an absolute requirement. All agents are still managed within the JBoss ON server cloud, so any JBoss ON server can, theoretically, manage any JBoss ON agent based on the current load and performance.

The affinity groups page shows the number of agents and servers assigned to each affinity group.



Affinity Groups

☐	Name	Agent Count	Server Count
☐	East Data Center	12	1
☐	Example Affinity Group	1	1
☐	West Data Center	18	1

CREATE NEW | REMOVE SELECTED | Total: 3

Figure 11.3. Listing Affinity Groups

NOTE

An agent and a server can only belong to a single affinity group.

To create a new affinity group:

NOTE

To edit an affinity group, click its name, then manage it the same as creating a new affinity group.

1. In the top navigation menu, click the **Administration** link.
2. Open the **High Availability** menu, and select the **Affinity Groups** menu item.
3. Click the **CREATE NEW** button.
4. Enter a name for the new affinity group, and click **OK**.
5. In the new affinity group's details page, click the **EDIT GROUP AGENTS** button.
6. In the lower section, **Agents not part of an affinity group**, click the checkboxes by the agent names to add to the group, and click **ADD TO GROUP**.
7. Click the **Return to Affinity Group Link**.
8. In the new affinity group's details page, click the **EDIT GROUP SERVERS** button.
9. The lower section lists servers which do not currently belong to the affinity group. Click the checkboxes by the server names to add to the group, and click **ADD TO GROUP**.
10. Once both servers and agents have been added to the affinity group, the group is fully configured.

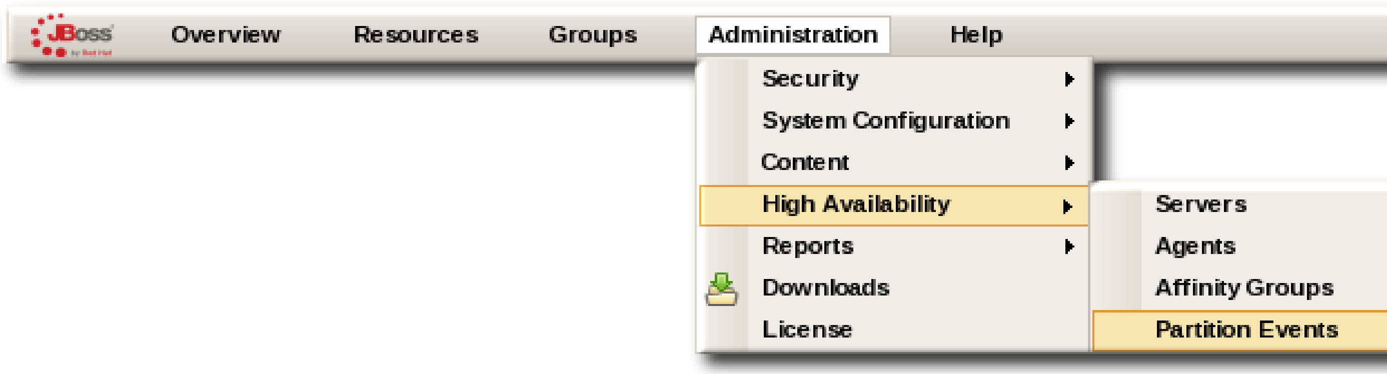
11.8.3.4. Managing Partition Events

When an agent is assigned to be managed by a server, that is a *partition*. *Partition events* are almost like log messages that occur whenever a change in the partition configuration occurs.

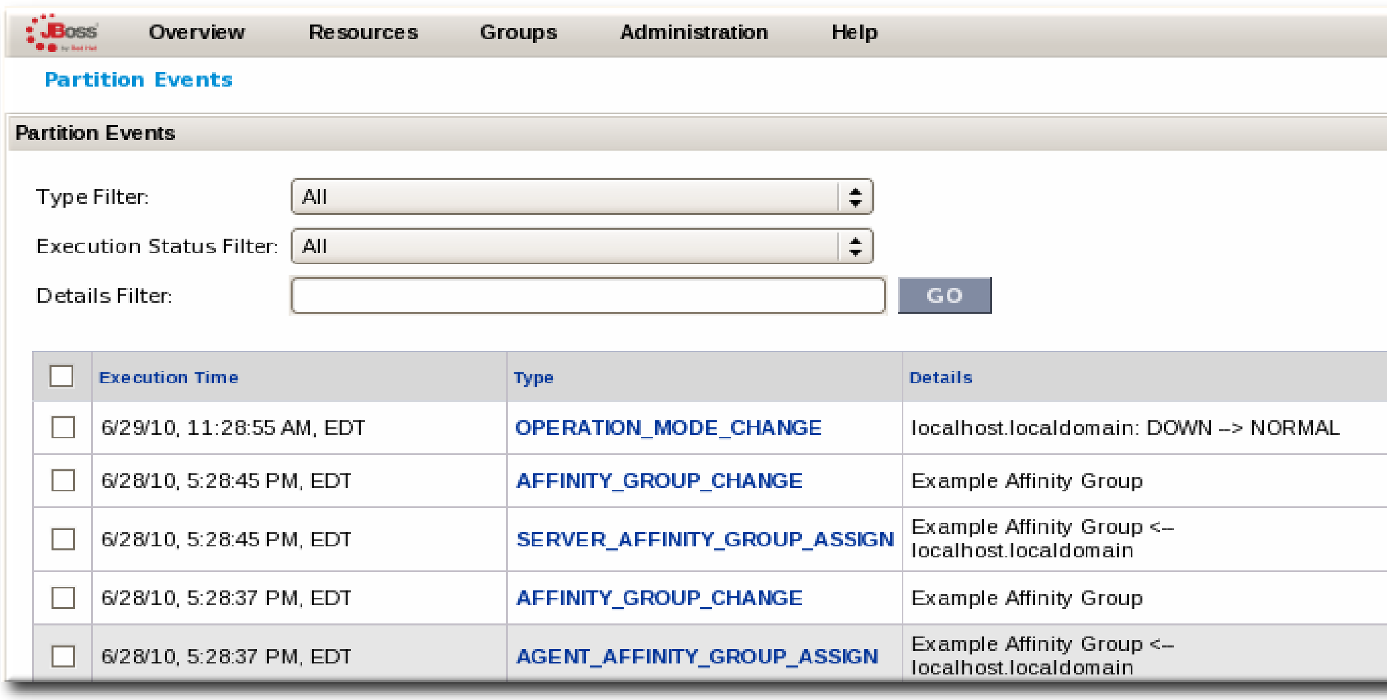
11.8.3.4.1. Viewing Partition Events

The partition events log is accessed in the high availability configuration.

1. In the top navigation menu, click the **Administration** link.
2. Open the **High Availability** menu.
3. Select the **Partition Events** menu item.



4. The partition events page lists all of the events that have been recorded. (*Table 11.8, "Partition Events Entries"* describes the different fields.) Click the type name of any partition event opens up that record with more information about how the partition assignments were affected.



Type Filter: All
Execution Status Filter: All
Details Filter:

	Execution Time	Type	Details
☐	6/29/10, 11:28:55 AM, EDT	OPERATION_MODE_CHANGE	localhost.localdomain: DOWN --> NORMAL
☐	6/28/10, 5:28:45 PM, EDT	AFFINITY_GROUP_CHANGE	Example Affinity Group
☐	6/28/10, 5:28:45 PM, EDT	SERVER_AFFINITY_GROUP_ASSIGN	Example Affinity Group <-- localhost.localdomain
☐	6/28/10, 5:28:37 PM, EDT	AFFINITY_GROUP_CHANGE	Example Affinity Group
☐	6/28/10, 5:28:37 PM, EDT	AGENT_AFFINITY_GROUP_ASSIGN	Example Affinity Group <-- localhost.localdomain

The partition events log can be filtered to display entries which match the type, status, or details in the event record.

There are basically four categories of partition events that are recorded.

- Affinity group changes

- Agent events
- Server events
- Partition changes

All of the recorded events are listed in *Table 11.7, “Types of Partition Events”*.

Partition Event	Description
Affinity Group Changes	
AFFINITY_GROUP_CHANGE	Registers a change in the agent or server assignments for an affinity group or that a group was added.
AFFINITY_GROUP_DELETE	Registers an affinity group was deleted from the JBoss ON configuration.
AGENT_AFFINITY_GROUP_ASSIGN	Registers that an agent was added to an affinity group.
AGENT_AFFINITY_GROUP_REMOVE	Registers that an agent was removed from an affinity group.
SERVER_AFFINITY_GROUP_ASSIGN	Registers that a server was added to an affinity group.
SERVER_AFFINITY_GROUP_REMOVE	Registers that a server was removed from an affinity group.
Agent Events	
AGENT_CONNECT	Shows that a JBoss ON agent was started.
AGENT_SHUTDOWN	Shows that a JBoss ON agent was stopped.
AGENT_LEAVE	Shows that a JBoss ON agent was permanently removed from the JBoss ON configuration.
AGENT_REGISTRATION	Shows that a new JBoss ON agent was added to the JBoss ON configuration.
Server Events	
SERVER_DELETION	Shows that a JBoss ON server was permanently removed from the JBoss ON configuration.
SERVER_COMPUTE_POWER_CHANGE	
OPERATION_MODE_CHANGE	Shows that a server went stopped, was started, or was newly installed. The type also shows how the mode transitioned (such as **`server.example.com: DOWN --> NORMAL`**).
Partition Changes	
SYSTEM_INITIATED_PARTITION	Shows that JBoss ON initiated a repartition of the servers.
ADMIN_INITIATED_PARTITION	Shows that a JBoss ON user initiated a repartition of the servers.

Table 11.7. Types of Partition Events

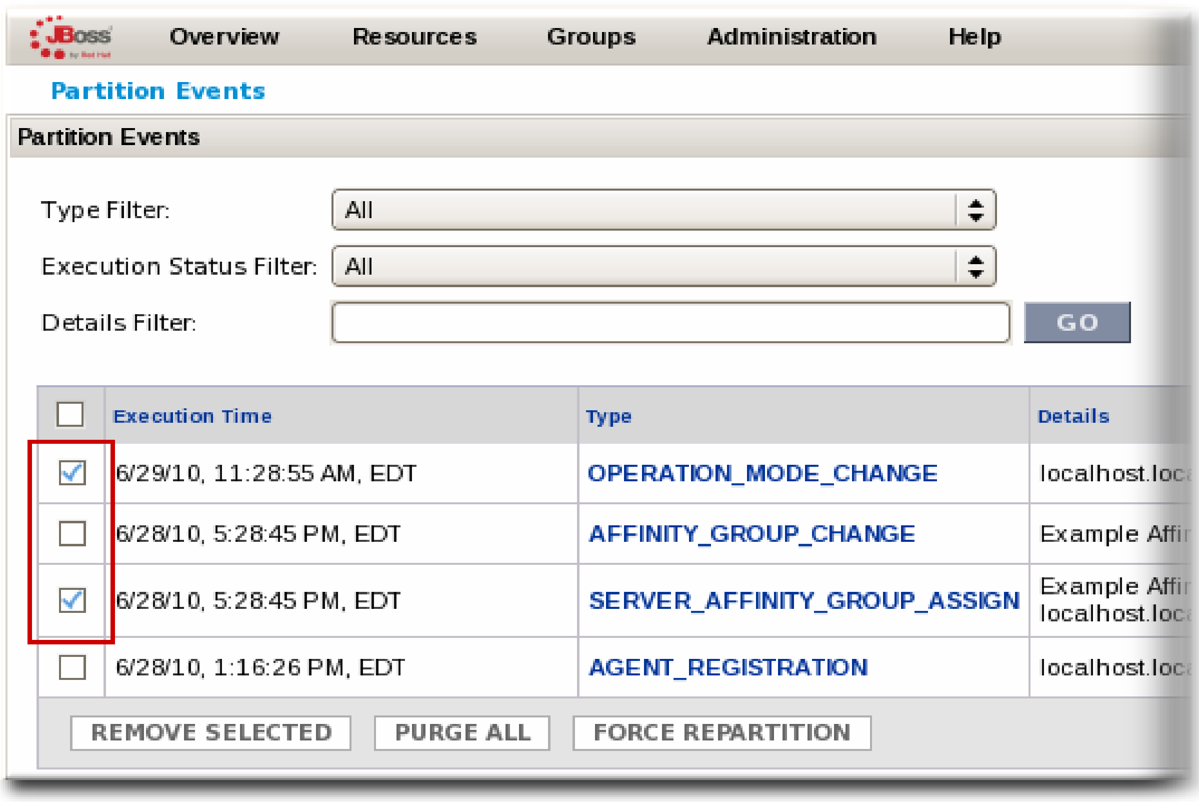
Field	Description
Execution Time	The time of the partition event.
Type	Shows the partition event type. This indicates what happened in the system affecting agent partition.
Details	Gives detailed information about the partition event; the type of information given varies based on the partition event type.
Initiated By	Shows the name of the user who initiated the action generating the partition event. Events initiated by the JBoss ON server itself show they were initiated by **`admin`**.
Execution Status	Shows low for status descriptions. There are four different status settings: • *Audit* shows that a change was made, but not an event that affects the partition topology. • *Immediate* shows that a partition change was made at the time of the event. • *Requested* shows that a partition change was requested and deferred until the next execution of the JBoss ON server cloud job (usually once a minute). Repartition requests usually have a requested status. • *Completed* shows that a change haas been completed.

Table 11.8. Partition Events Entries

11.8.3.4.2. Removing Partition Events

There are two ways to remove partition event records:

- By selecting individual records and click **REMOVE SELECTED**
- By clicking the **PURGE ALL** to remove all events



Type Filter: All

Execution Status Filter: All

Details Filter:

☐	Execution Time	Type	Details
✓	6/29/10, 11:28:55 AM, EDT	OPERATION_MODE_CHANGE	localhost.loc
☐	6/28/10, 5:28:45 PM, EDT	AFFINITY_GROUP_CHANGE	Example Affi
✓	6/28/10, 5:28:45 PM, EDT	SERVER_AFFINITY_GROUP_ASSIGN	Example Affi localhost.loc
☐	6/28/10, 1:16:26 PM, EDT	AGENT_REGISTRATION	localhost.loc

Figure 11.4. Removing Partition Events

11.8.4. Editing JBoss ON Server Configuration in rhq-server.properties

JBoss ON server configuration properties are stored either in the **`rhq-server.properties`** configuration file in the ***`serverRoot`*****`/jon-server-2.4.0.Beta1/bin`** directory or in the JBoss ON database. The configuration file contains most of the basic information about the JBoss ON server, such as the TCP/IP ports it listens on and its hostname or IP address.

The JBoss ON server configuration is loaded from the **`rhq-server.properties`** file when the server starts. The initial configuration is defined by the installer when the JBoss ON program is set up.

NOTE

Because the configuration properties are loaded from **`rhq-server.properties`** when the JBoss ON server starts up, you have to restart the JBoss ON server after making any chanegs to that configuration file so the new settings are loaded.

11.8.4.1. Editing Database Configuration

The JBoss ON server is always connected to a backend database to store most of its data, such as agents and resources in its inventory and plug-in configuration. The parameters for connecting with the database are defined in **`rhq-server.properties`**.

```
# Database
rhq.server.database.connection-url=jdbc:postgresql://127.0.0.1:5432/rhq
rhq.server.database.driver-class=org.postgresql.Driver
rhq.server.database.xa-datasource-class=org.postgresql.xa.PGXADataSource
rhq.server.database.user-name=rhqadmin
rhq.server.database.password=1eeb2f255e832171df8592078de921bc
rhq.server.database.type-mapping=PostgreSQL
rhq.server.database.server-name=127.0.0.1
rhq.server.database.port=5432
rhq.server.database.db-name=rhq
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Example 11.1. Default Configuration for a PostgreSQL Database

Parameter	Description
rhq.server.database.type-mapping	Gives the type or vendor of the database that is used by the JBoss ON server. This is either PostgreSQL or Oracle.
rhq.server.database.connection-url	The JDBC URL that the JBoss ON server uses when connecting to the database. An example is *jdbc:postgresql://localhost:5432/rhq* or *jdbc:oracle:oci:@localhost:1521:orcl*.
rhq.server.database.driver-class	The fully qualified class name of the JDBC driver that the JBoss ON server uses to communicate with the database. An example is *oracle.jdbc.driver.OracleDriver*.
rhq.server.database.xa-datasource-class	The fully qualified class name of the JDBC driver that the JBoss ON server uses to communicate with the database. Examples of this are *org.postgresql.xa.PGXADataSource* or *oracle.jdbc.xa.client.OracleXADatasource*.
rhq.server.database.user-name	The name of the user that the JBoss ON server uses when logging into the database
rhq.server.database.password	The password of the database user that is used by the JBoss ON server when logging into the database. This password is stored in a hash in the **`rhq-server.properties`** file. When the password is changed in the database, then the password must be manually hashed and copied into the **`rhq-server.properties`** file. This is described in *Section 11.7.2, “Changing Database Passwords”*.
rhq.server.database.server-name	The server name where the database is found. This must match the server in the connection URL. This is currently only used when connecting to PostgreSQL.
rhq.server.database.port	The port on which the database is listening. This must match the port in the connection URL. This is currently only used when connecting to PostgreSQL.
rhq.server.database.db-name	The name of the database. This must match the name found in the connection URL. This

Parameter	Description
	is currently only used when connecting to PostgreSQL.
rhq.server.quartz.driverDelegateClass	The Quartz driver used for connections between the server and the database. The value of this is set by the installer and depends on the type of database used to store the JBoss ON information. For PostgreSQL, this is **`org.quartz.impl.jdbcjobstore.PostgreSQLDelegat`** and for Oracle, this is **`org.quartz.impl.jdbcjobstore.oracle.OracleDele`**

Table 11.9. rhq-server.properties Parameters for Database Configuration

11.8.4.2. Configuring Communication Settings

JBoss ON servers are configured to communicate to agents by defining and identifying ways that the server and agent can connect, as well as how other clients (like users accessing the JBoss ON GUI) can connect to the server. These communication endpoints include identifying the server hostname or IP address, ports that the server listens on for different types of messages, and protocols used to access the server. All of the server connection parameters are described in *Table 11.10, “rhq-server.properties Parameters for Server Connections”*.

Connections, or transport methods, for the server are implemented through servlets (HTTP and HTTPS) or sockets (HTTPS). The **`servlet`** and **`sslservlet`** transports (HTTP and HTTPS, respectively) route traffic through the TOmcat server embedded in the JBoss ON server.

NOTE

Servlet-based transports leverage the Tomcat connector infrastructure to handle both agent and GUI requests. Using servlets, however, limits agents and GUI clients to use the same connection methods; for certificate-based SSL connections, servlets require users to authenticate to the GUI using a stored browser certificate. For SSL, then, it may be preferable to use socket connections, which allow different authentication methods for agent and GUI sessions.

See *Section 11.10.2, “Setting up Authentication Between Servers and Agents”* for setting up SSL sockets.

The general configuration settings set the port numbers that users can used to access the server.

```
# General Properties
rhq.server.startup.web.http.port=7080
rhq.server.startup.web.https.port=7443
```

Additional connection settings can be added to configure SSL connections for inbound connections to the server (messages from the agent to the server) and outbound connections (messages from the server to the agent). For example:

```
rhq.server.tomcat.security.client-auth-mode=want
rhq.server.tomcat.security.secure-socket-protocol=TLS
rhq.server.tomcat.security.algorithm=SunX509
rhq.server.tomcat.security.keystore.alias=RHQ
rhq.server.tomcat.security.keystore.file=conf/rhq.keystore
rhq.server.tomcat.security.keystore.password=RHQManagement
rhq.server.tomcat.security.keystore.type=JKS
rhq.server.tomcat.security.truststore.file=conf/rhq.truststore
rhq.server.tomcat.security.truststore.password=RHQManagement
rhq.server.tomcat.security.truststore.type=JKS
```

The third part of JBoss ON server communications provides more control over information the connection endpoints for JBoss ON servers and agents to use to talk with each other. These are *transport* parameters for the server. Both the JBoss ON agent and port use the same remoting framework, using invocation strings that are similar to URLs. These connection strings have the format:

```
protocol://server:port/transportParm1=value1&transportParam2=value2
```

IMPORTANT

For most communications, the JBoss ON server must use either servlet or sslservlet protocols. The only instance where socket can be used is for passing transport parameters. Otherwise, socket and sslsocket are not supported.

The transport configuration first sets up connectors (called *endpoints*) that the servers and agents jointly define and use for communications. This means that the same information must be in both the server and agent configuration files. Each aspect of the remoting URL is built using separate server configuration parameters.

The standard server configuration has four parts, for the transport method, server IP address, agent port, and any parameters to append to the URL. For example:

```
rhq.communications.connector.transport=servlet
rhq.communications.connector.bind-address=192.168.1.2
rhq.communications.connector.bind-port=16163
rhq.communications.connector.transport-params=/jboss-remoting-servlet-invoker/
ServerInvokerServlet
```

That standard configuration is merged to create this URL:

```
servlet://192.168.1.2:16163/jboss-remoting-servlet-invoker/ServerInvokerServlet
```

A corresponding entry, with the same endpoint definition, is also listed in the agent configuration so that it knows how to send communications to the server, such as sending registration and availability reports.

```
RHQ Server IP Address=192.168.1.2
RHQ Server Port=16163
RHQ Server Transport Protocol=servlet
RHQ Server Transport Parameters=/jboss-remoting-servlet-invoker/ServerInvokerServlet
```

A server with an IP address of **`192.168.0.10`** will connect to agents over the standard agent port of 16163. The server configuration has the following configuration to define the server address (*`rhq.communications.connector.bind-address`*), the agent communications port (**`rhq.communications.connector.bind-port`**), and the connection protocol (*`rhq.communications.connector.transport`*):

```
rhq.communications.connector.transport=servlet
rhq.communications.connector.bind-address=192.168.0.10
rhq.communications.connector.bind-port=16163
rhq.communications.connector.transport-params=enableTcpNoDelay=true&backlog=200
```

The connection URL, then, is:

```
servlet://192.169.0.10:16163/enableTcpNoDelay=true&backlog=200
```

The JBoss ON agent configuration will contain corresponding entries which match the server configuration:

```
RHQ Server IP Address=192.168.0.10
RHQ Server Port=16163
RHQ Server Transport Protocol=socket
RHQ Server Transport Parameters=enableTcpNoDelay=true&backlog=200
```

Example 11.2. Basic Server-Agent Transport Example

Transport parameters can pass relevant information about both incoming and outgoing messages (called server and client messages, respectively, because of how the JBoss ON server handles the messages). These transport parameters are strung together with ampersands (&), as with a standard web URL parametes.

Both server and client transport parameters are passed in the same URL; the JBoss ON server only processes whatever paramters are relevant for the current operation. In *Example 11.2, “Basic Server-Agent Transport Example”*, for instance, the configuration sets two transport parameters, **`enableTcpNoDelay`** (client) and **`backlog`** (server). When the JBoss ON server is receiving messages — when it function as a communications server — it uses the **`backlog`** parameter and ignore **`enableTcpNoDelay`** because **`enableTcpNoDelay`** is only for outgoing (client) messages.

Parameter	Description
General Connection Parameters	
jboss.bind.address[12]	Gives the IP address for the JBoss ON GUI console, among other services, to bind to. This is the host in the JBoss ON GUI URLs; e.g. the myhost in http://myhost:7080.
rhq.server.startup.web.http.port[12]	Gives the port that the JBoss ON GUI listens to for unsecured HTTP requests. This is the port number in the JBoss ON GUI URLs, such as the 7080 in *http://localhost:7080*. This is also the unsecure port used as the endpoint in high availability.
rhq.server.startup.web.https.port[12]	Gives the port that the JBoss ON GUI listens to for secured HTTPS requests. This is the port number in the JBoss ON GUI URLs, such as the 7443 in *https://localhost:7443*. This is also the secure port used as the endpoint in high availability.
rhq.server.startup.keystore.filename[2]	The JBoss ON GUI can accept browser requests over HTTPS. If you want to authenticate the JBoss ON GUI to remote browsers, you need

Parameter	Description
	to put an SSL certificate in a keystore file. This setting points to the location of the keystore file. Note that this file name must be a relative file path relative to the ***<JBoss ON server Install Dir>*`/jbossas/server/default/conf`** directory. The JBoss ON server ships with a self-signed certificate in a default keystore file.
rhq.server.startup.keystore.password[2]	The password of the keystore file. This is so the JBoss ON GUI can access the keystore file in order to be able to serve the certificate to browser clients.
rhq.server.startup.keystore.sslprotocol[2]	The protocol that browser clients should use to communicate with the JBoss ON GUI.
rhq.server.maintenance-mode-at-start	Sets whether to start the server in maintenance mode (true) or whether to start the server in whatever mode it was in when it shut down (false). The default is false.
• rhq.server.startup.webservice.port[12] • rhq.server.startup.namingservice.port[12] • rhq.server.startup.namingservice.rmiport[12] • rhq.server.startup.jrmpinvoker.rmiport[12] • rhq.server.startup.pooledinvoker.rmiport[12] • rhq.server.startup.ajp.port[12] • rhq.server.startup.unifiedinvoker.port[12] • rhq.server.startup.aspectdeployer.bind-port[12]	Ports used by internal services.
SSL Connection Parameters	
• rhq.communications.connector.security.secure-socket-protocol (agent to server) • rhq.server.client.security.secure-socket-protocol (server to agent)	The secure protocol that agents must use when communicating with this JBoss ON server.
• rhq.communications.connector.security.keystore.file (agent to server) • rhq.server.client.security.keystore.file (server to agent)	The keystore file that contains a certificate that authenticates the JBoss ON server to the agents.
• rhq.communications.connector.security.keystore.algorithm (agent to server) • rhq.server.client.security.keystore.algorithm (server to agent)	

Parameter	Description
• rhq.communications.connector.security.keystore.type (agent to server) • rhq.server.client.security.keystore.type (server to agent)	The file format of the keystore.
• rhq.communications.connector.security.keystore.password (agent to server) • rhq.server.client.security.keystore.password (server to agent)	The password that is used to gain access to the keystore file.
• rhq.communications.connector.security.keystore.key-password (agent to server) • rhq.server.client.security.keystore.key-password (server to agent)	The password that is used to gain access to the key inside the keystore.
• rhq.communications.connector.security.keystore.alias (agent to server) • rhq.server.client.security.keystore.alias (server to agent)	The alias that identifies the JBoss ON server's key within its keystore.
• rhq.communications.connector.security.truststore.file (agent to server) • rhq.server.client.security.truststore.file (server to agent)	The truststore file that contains certificates that this JBoss ON server trusts. If you need the JBoss ON server to authenticate JBoss ON agents, you must set this; otherwise it is not needed. This truststore contains certificates for all JBoss ON agents that need to communicate with this JBoss ON server. Refer to the *Incoming Client Authentication Mode*.
• rhq.communications.connector.security.truststore.algorithm (agent to server) • rhq.server.client.security.truststore.algorithm (server to agent)	
• rhq.communications.connector.security.truststore.type (agent to server) • rhq.server.client.security.truststore.type (server to agent)	The file format of the truststore file.
• rhq.communications.connector.security.truststore.password (agent to server) • rhq.server.client.security.truststore.password (server to agent)	The password that is used to gain access to the truststore file.
• rhq.communications.connector.security.client-auth-mode (agent to server) • rhq.server.client.security.server-auth-mode-enabled (server to agent)	Indicates if the JBoss ON server must authenticate the JBoss ON agents that are sending it messages. If the server is using secure connections, but does not have trusted certificates for all of the JBoss ON agents in a

Parameter	Description
	truststore, set this to none. The valid values are none, want, or need.
Transport Connection Parameters	
rhq.communications.connector.transport	Defines how the JBoss ON agents need to transport messages to the JBoss ON server. The allowed values are either servlet or sslservlet. The agent requests go through the JBoss ON server web application layer (i.e. the secure Tomcat Connector). With sslservlet, not only do agent requests route through the web application layer, but they are also secured through the secure Tomcat Connector. The keystore used for incoming agent message authentication is the same as that configured in *`rhq.communications.connector.security.keystore.file.`* **NOTE** This transport setting does *not* restrict agents from only going over that particular connction method. By default, the JBoss ON server always deploys the communications connector that allows for both servlet and sslservlet traffic. This setting tells the agent to decide what transport is used when it sends messages to the server. If the server has its transport set to servlet, but the agent is configured to talk to the server via sslservlet, the messages the agent sends will be via sslservlet.
rhq.communications.connector.bind-address	This is the address that is placed in the server's JBoss/Remoting locator URL. This defines the endpoint that the JBoss ON server will bind its connector to. It also represents the public endpoint address that all agents can use to connect to the server.
rhq.communications.connector.bind-port	Defines the endpoint that the JBoss ON server binds to, as well as the public address that all agents can use to connect to the server. This is hidden from view in the installer, although it still appears in the **`rhq-server.properties`** file. This value can be blank; the server sets this to

Parameter	Description
	either the HTTP or HTTPS port, depending on the transport configured for the server.
rhq.communications.connector.transport-params	Defines additional transport parameters the JBoss ON server will set on its connector that will accept incoming messages from the JBoss ON agents. All of the possible transport parameters are listed in *Table 11.11, "Transport Parameters"*.
rhq.communications.multicast-detector.enabled	If true, the JBoss ON server will attempt to auto-detect JBoss ON agents coming online and going offline using multicast detection. Your network must support multicast traffic for this to work.
rhq.communications.multicast-detector.bind-address	The address that the multicast detector directly binds to. This is not used, or needed, if you have not enabled multicast detection.
rhq.communications.multicast-detector.multicast-address	The address that the multicast detector will broadcast messages to. This is not used, or needed, if you have not enabled multicast detection.
rhq.communications.multicast-detector.port	The port that the multicast detector will broadcast messages to. This is not used, or needed, if you have not enabled multicast detection.

These settings configure specific IP addresses and ports for the JBoss ON server instance. If there are firewall issues the require different settings, then these parameters can be changed.

The JBoss ON server has to be restarted for any changes to this value to take effect.

Table 11.10. rhq-server.properties Parameters for Server Connections

Transport Parameter	Description	For Incoming Messages or for Outgoing Messages
serverBindAddress	The address on which the server socket binds to listen for requests. The default is an empty value which indicates the server socket should be bound to the host provided by the InvokerLocator URL (the host).	Incoming
serverBindPort	The port to listen for requests on.	Incoming
timeout	The socket timeout value. The default on the server side is 60000 (one minute). If the timeout parameter is set, its value will also be passed to the client-side (see below).	Incoming

Transport Parameter	Description	For Incoming Messages or for Outgoing Messages
backlog	The preferred number of unaccepted incoming connections allowed at a given time. The actual number may be greater than the specified backlog. When the queue is full, further connection requests are rejected. Must be a positive value greater than 0. If the value passed if equal or less than 0, then the default value will be assumed. The default value is 200.	Incoming
numAcceptThreads	The number of threads that exist for accepting client connections. The default is 1.	Incoming
maxPoolSize	The number of server threads for processing client requests. The default is 300.	Incoming
socket.check_connection	Indicates if the invoker should try to check the connection before re-using it by sending a single byte ping from the client to the server and then back from the server. This configuration needs to be set on both the client and server to work. The default value is false.	Incoming
clientConnectAddress	The IP address or hostname the client will use to connect to the server-side socket. This would be needed in the case that the client will be going through a router that forwards requests made externally to a different IP address or hostname internally. If no clientConnectAddress or serverBindAddress is specified, the local host's address is used.	Outgoing
clientConnectPort	The port the client will use to connect to the server-side socket. This would be needed in the case that the client will be going through a router that forwards requests made	Outgoing

Transport Parameter	Description	For Incoming Messages or for Outgoing Messages
	externally to a different port internally.	
timeout	The socket timeout value. The default on the client side is 1800000 (or 30 minutes).	Outgoing
enableTcpNoDelay	Indicates if the client socket should have TCP_NODELAY turned on or off. TCP_NODELAY is for a specific purpose; to disable the Nagle buffering algorithm. It should only be set for applications that send frequent small bursts of information without getting an immediate response. The default is false.	Outgoing
clientMaxPoolSize	The client-side maximum number of active socket connections. This basically equates to the maximum number of concurrent client calls that can be made from the socket client invoker. The default is 50.	Outgoing
numberOfRetries	The number of retries to get a socket from the pool. This basically equates to the number of seconds the client will wait to get a client socket connection from the pool before timing out. If the max retries is reached, a CannotConnectException will be thrown. The default is 30.	Outgoing
numberOfCallRetries	The number of retries for making the invocation. This is unrelated to numberOfRetries in that when this comes into play is after it has already received a client socket connection from the pool. However, it is possible that the socket connection timed out while waiting within the pool. Since a connection check is not done by default, the connection is thrown away and an attempt	Outgoing

Transport Parameter	Description	For Incoming Messages or for Outgoing Messages
	to get a new one will be made. This will happen for however many numberOfCallRetries is (which defaults to 3). However, when (numberOfCallsRetries - 2) is reached, the entire connection pool is flushed under the assumption that all connections in the pool have timed out and are invalid and will start over by creating a new connection. If this still fails, a MarshalException is thrown.	
socket.check_connection	Indicates if the invoker should try to check the connection before re-using it by sending a single byte ping from the client to the server and then back from the server. This configuration needs to be set on both client and server to work. This if false by default.	Outgoing

Table 11.11. Transport Parameters

11.8.4.3. Setting Concurrency Limits

JBoss ON can handle large numbers of agents, potentially hundreds. The JBoss ON server can possibly be flooded with messages if many agents attempt to communicate with the server simultaneously. This can happen if the JBoss ON server is restarted after being down for a period of time; when JBoss ON agents detect that the JBoss ON server has come back, they all immediately attempt to send it a backlog of messages.

The JBoss ON server can have a configurable limit on the number of concurrent messages that can be processed at one time, to mitigate any risk of flooding the server. Any messages that come in past that limit are dropped and the agent is asked to send them later.

All of the concurrency-related parameters are listed in *Table 11.12, “rhq-server.properties Parameters for Concurrency Limits”*.

Concurrency limits not only limit the number of agent connections, but also the number of connections to the GUI and other web connections to the server. There are two parameters that control the concurrency limits:

- A global limit on the total number of incoming messages to the server (`rhq.communications.global-concurrency-limit`)

  This is the total number of allowed connections, so this is the effective concurrency limit, even if other concurrency limits are set higher.

- A limit on the number of concurrent web connections allowed (*`rhq.server.startup.web.max-connections`*)

  The limit on web connections is the same for both non-secured HTTP requests and HTTPS requests, but the limit is additive so HTTP and HTTPS connections count against different pools. The *total* maximum connections allowed is actually twice whatever the *`rhq.server.startup.web.max-connections`* value is. For example, if the setting is 300, then 300 HTTP requests are allowed and 300 HTTPS requests are allowed, for total of 600 concurrent web connections.

- Limits on the number of downloads from agents (*`rhq.server.agent-downloads-limit`*) and from other clients (*`rhq.server.client-downloads-limit`*)

```
rhq.server.startup.web.max-connections=200
rhq.server.agent-downloads-limit=45
rhq.server.client-downloads-limit=5
rhq.communications.global-concurrency-limit=30
```

Example 11.3. Concurrency Limits

Parameter	Description
rhq.server.startup.web.max-connections	Sets a limit on the number of web connections that can be concurrently created, including both connections to the GUI and connections by agents. **NOTE** If agent requests are routed over web connections, make sure the *`rhq.communications.global-concurrency-limit`* value is slightly lower than the web connections limit. Otherwise, GUI users could be blocked from accessing the JBoss ON UI whenever there is a high agent load. The limit on web connections is the same for both HTTP and HTTPS (secure) requests, so the *total* max connections allowed is actually twice what this setting is. For example, if the max web connections is set to 300, then 300 HTTP requests will be allowed and 300 HTTPS requests will be allowed, for a total of 600 concurrent web connections.
rhq.communications.global-concurrency-limit	Sets the total number of agent messages that come into the server. This only affectd incoming agent messages, not GUI or browser requests.

Parameter	Description
	If this global concurrency limit is set to 300, no more than 300 total agent messages can be processed at any one time, regardless of what kinds of messages are coming in. Even if if other concurrency limits are set higher than this global limit, they are capped at this global limit since there can never ben more messages processed than the global limit. This value should be the same or higher than the number of allowed web connections so that web connections to the GUI aren't blocked when there is a high agent load.
rhq.server.concurrency-limit.inventory-report	Inventory reports are sent from the agent when the agent starts up, and periodically thereafter. Inventory reports can be large, depending on the number of resources on the agent machine.
rhq.server.concurrency-limit.availability-report	Availability reports are regularly sent from the agent, typically every 60 seconds. Availability reports are usually very small, but occur in large numbers due to the high frequency of their transmission.
rhq.server.concurrency-limit.inventory-sync	Inventory synchronizations occur when the agent needs to synchronize its inventory with that of the server. Agents typically synchronize at startup. Traffic that flows as part of inventory synchronizations is usually large, depending upon the number of resources managed by the agent.
rhq.server.concurrency-limit.content-report	Content reports are similar to inventory reports except they contain information about discovered content (i.e., installed packages of software). These reports can be large depending on the number of installed software the agent has discovered and is managing.
rhq.server.concurrency-limit.content-download	Content downloads occur when a resource on an agent needs to ask for the content of a package version, usually for the purpose of installing the package.
rhq.server.concurrency-limit.measurement-report	Measurement reports are periodically sent to the server whenever the agent completes measurement collections. The number and size of measurement reports can vary, depending on the number and frequency of measurements scheduled to be collected. The greater the number of schedule measurements the agent needs to collect, the more frequently

Parameter	Description
	measurement reports are sent, and the larger the reports will be.
rhq.server.concurrency-limit.measurement-schedule-request	Similar to inventory synchronization, measurement schedule requests are sent to the agent asking the server for an up-to-date set of measurement schedules that have to be collected.

Table 11.12. rhq-server.properties Parameters for Concurrency Limits

11.8.4.4. Configuring the SMTP Server for Email Notifications

Each JBoss ON server talks to a specific SMTP server. The SMTP server is defined in the **`rhq-server.properties`** file. The default configuration points to the local JBoss ON server hosts.

```
# Email
rhq.server.email.smtp-host=localhost
rhq.server.email.smtp-port=25
rhq.server.email.from-address=rhqadmin@localhost
```

These settings can be edited to use a different SMTP server or email account.

TIP

To confirm that the SMTP settings are correct and the server can send emails successfully, go to the test email page at **`http://`***`server`***`/admin/test/email.jsp`**.

Parameter	Description
rhq.server.email.smtp-host	Sets the hostname of the SMTP server used by the JBoss ON server.
rhq.server.email.smtp-port	Sets the port of the SMTP server used by the JBoss ON server.
rhq.server.email.from-address	Sets the address to use for the From header of all emails sent by the JBoss ON server.

Table 11.13. rhq-server.properties Parameters for SMTP

11.8.4.5. Installing a Server Silently

Some options in the **`rhq-server.properties`** file tell the installation process to load the server configuration from the file rather than the from the web-based installer.

```
# Auto-Install Pre-Configuration Settings
rhq.autoinstall.enabled=false
rhq.autoinstall.database=auto
rhq.autoinstall.public-endpoint-address=
```

These settings are only used for installation.

Parameter	Description
rhq.autoinstall.enabled	Tells the installation process whether to load the configuration from the **`rhq-server.properties`** file (**`true`**) or from the web-based installer (**`false`**).
rhq.autoinstall.database	Tells the install process how to load or add database schema. There are three options: • **`auto`** creates a new schema for new installation or upgrades existing schema without overwriting the data. • **`overwrite`** overwites the database and creates a new, empty schema. • **`skip`** skips the entire database process so no database is created or updated.
rhq.autoinstall.public-endpoint-address	Sets the IP address or hostname to use for the server. If no value is given, then the server detects and sets its own value when it starts.

Table 11.14. rhq-server.properties Parameters for Silent Installation

11.9. Configuring Agents

The agent's configuration is initially read from **`agent-configuration.xml`** and overlaid with the values entered at the setup prompts at start up. After the agent is initially configured, it will persist that configuration and never look at **`agent-configuration.xml`**, unless you clear the configuration. For more details, read the comments at the top of the **`agent-configuration.xml`** file. See the *Installation Guide*[2] for more information.

11.9.1. Setting Agent Heap Size

RHQ_AGENT_JAVA_OPTS="-Xms64m -Xmx128m -Djava.net.preferIPv4Stack=true"

RHQ_AGENT_JAVA_OPTS - Java VM command line options to be # passed into the agent's VM. If this is not defined # this script will pass in a default set of options. # If this is set, it completely overrides the # agent's defaults. If you only want to add options # to the agent's defaults, then you will want to # use RHQ_AGENT_ADDITIONAL_JAVA_OPTS instead.

11.9.2. Viewing the Persisted Configuration

There are several ways in which you can view the agent's persisted configuration:

1. If the agent is in your JBoss ON inventory, simply go to your agent's **Configuration** tab to view its live configuration. This is the same configuration that is persisted.

2. If the agent is currently running in non-daemon mode (i.e. you have the agent prompt on your console), you can use the **`getconfig`** or **`config`** prompt commands to view the live configuration. Type **`help getconfig`** or **`help config`** for more information.

[2] http://www.redhat.com/docs/en-US/JBoss_ON/2.3/html/Installation_Guide/index.html

3. If the agent is in your JBoss ON inventory, you can execute the "Execute Prompt Command" operation and invoke the **`getconfig`** prompt command to view one or more preferences.

4. Because the agent configuration is stored in the standard *Java Preferences API*[3] backing store, you can use any tool that can examine Java Preferences. One such tool is the *Java Preferences Tool*[4]. This is a GUI tool that can give you a file system-like view into your Java Preferences. The agent preferences are stored in the "User" preferences node under the node name "rhq-agent". Depending on the **`-p`** option that is passed to the agent when it is started, the actual configuration settings are found under a sub-node under `rhq-agent`. The default preferences node is called `default` so typically your agent's persisted configuration is found in the user preferences under "rhq-agent/default".

WARNING

Do not attempt to change the values of the preferences using third-party tools without knowing what you are doing - you could render the agent useless if you change the wrong preference to the wrong value. Use this mechanism only to view your agent's configuration

11.9.3. Changing the Agent IP Address

The agent has a configuration preference named "rhq.communications.connector.bind-address" whose value is that of the IP address the agent binds to when it starts its server socket (the thing it listens to for incoming messages from the server).

If you change the agent's IP address (and invalidate the old agent IP address), you have to do a couple things:

1. You have to change the agent's configuration so that preference value is the same as the new IP address. You can do this by issuing a setconfig prompt command on the agent prompt: setconfig rhq.communications.connector.bind-address=<the new IP address>. (NOTE: do not change agent-configuration.xml and think the change will take effect - please read and understand the comments at the top of agent-configuration.xml before you change that configuration file). If your agent is running in the background as a daemon process, you'll have to shut it down via rhq-agent-wrapper.sh/bat stop and re-start it via "rhq-agent.sh".

2. Restart the agent once you change the IP address preference value.

Once the agent is restarted, it will use that new IP address.

11.9.4. Viewing the Server Failover Lists for Agents

JBoss ON agents are automatically included in high availability in order to assign them to servers for management. Agent-server preferences are assigned through affinity groups (*Section 11.8.3.3, "Creating Affinity Groups"*). The agent high availability settings show its affinity groups, the server currently managing it, and any servers available for failover.

The first server that an agent contacts is defined in its **`agent-configuration.xml`** file, and that is the server that the agent sends its initial registration request. After registration, the agent joins the high availability cloud, and it sends its updates — monitoring information, resource changes — to any server in the cloud. At registration, the agent gets its first affinity group assignment. If its primary server is different than its registration server, then the agent switchs communication over to the primary server.

The high availability server cloud helps define the relationships between servers and agents once the agent is running normally.

The group of servers that an agent sends updates to can be loosely restricted by defining an affinity group. The affinity group creates a list of servers that the agent prefers to access. This list is ordered; the first server entry is the primary server that the agent connects to. If that primary server is unavailable, then the agent cycles through the other servers in the list in order. This allows the agent to connect to defined servers in the high availability cloud gracefully and automatically, without interrupting JBoss ON performance.

If the agent cannot connect to any server in the failover list, then the agent temporarily stops communication and spools its messages. After a period of time, it will run through the failover list again, beginning with its primary server.

An agent always try to ensure that it is connected to its primary server. Once an hour, by default, it checks its connection to verify that the server it is using is its primary server. If it is not, then the agent tries to reconnect to its primary server.

The actual failover list for an agent is generated by the server and edited in the affinity group configuration for the server. Any changes to the affinity group, like new servers or agents, changed server priority, or new group assignments, are sent to the agent hourly when the agent polls the server for configuration changes.

To view the agent's failover list from the agent command prompt:

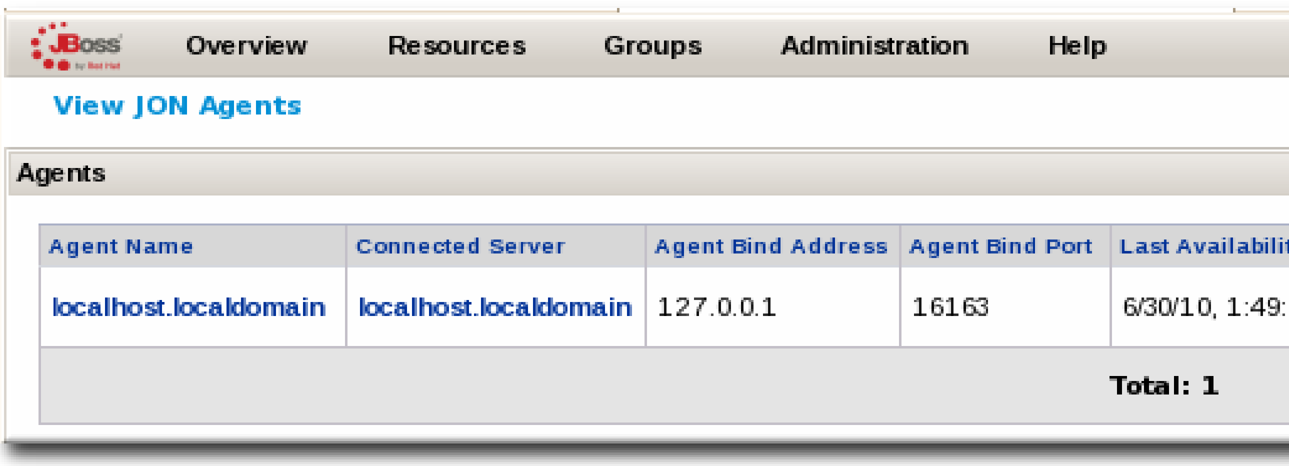
```
> failover --list
localhost.localdomain:7080/7443
server2.example.com:7080/7443
1.2.34.56:7080/7443
```

To view the failover list from the UI:

1. Click the **Administration** tab in the top menu.

2. In the **High Availability** menu, select the **Agents** item.

3. The agent high availability page shows information about the agents, including three things that are relevant for high availability:



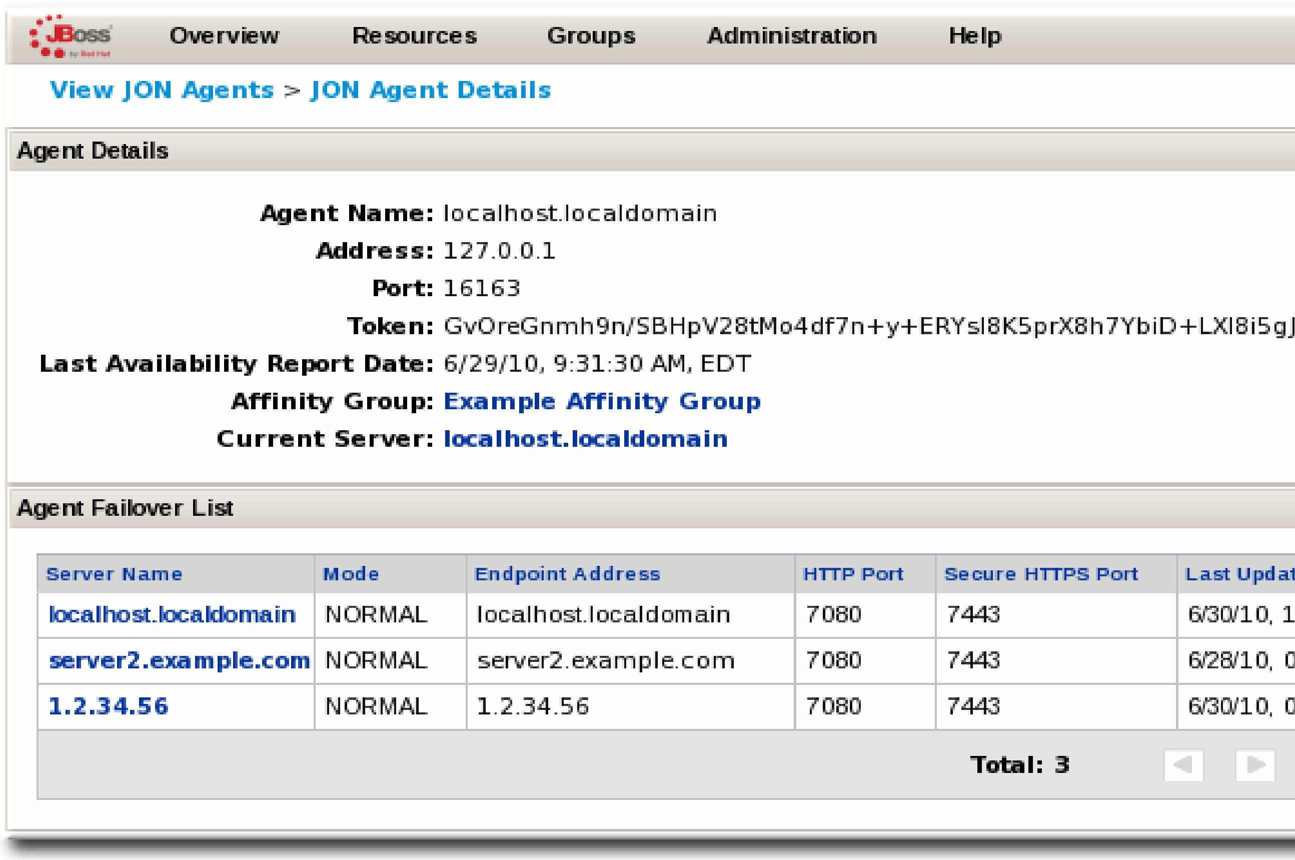
Agents

Agent Name	Connected Server	Agent Bind Address	Agent Bind Port	Last Availabili
localhost.localdomain	localhost.localdomain	127.0.0.1	16163	6/30/10, 1:49:

Total: 1

- The JBoss ON server that the agent is currently connected to (or the one it was most recently connected to).
- The time that the last agent availability report was sent to the server.
- The affinity group that the agent is assigned to.

4. Click the name of the agent. This opens the agent's server failover list. The first server listed is the primary server for the agent; all other servers are available in the high availability cloud. The connected server is usually also the primary server, unless the primary is offline.



Agent Details

Agent Name:	localhost.localdomain
Address:	127.0.0.1
Port:	16163
Token:	GvOreGnmh9n/SBHpV28tMo4df7n+y+ERYsl8K5prX8h7YbiD+LXl8i5gJ
Last Availability Report Date:	6/29/10, 9:31:30 AM, EDT
Affinity Group:	Example Affinity Group
Current Server:	localhost.localdomain

Agent Failover List

Server Name	Mode	Endpoint Address	HTTP Port	Secure HTTPS Port	Last Updat
localhost.localdomain	NORMAL	localhost.localdomain	7080	7443	6/30/10, 1
server2.example.com	NORMAL	server2.example.com	7080	7443	6/28/10, 0
1.2.34.56	NORMAL	1.2.34.56	7080	7443	6/30/10, 0

Total: 3

11.9.5. JBoss ON Agent Communications Services

When The JBoss ON agent starts for the first time, it enters *setup mode*. You can also manually enter this setup mode by using the **`--setup`** command line option or use the **`setup`** agent prompt command. Once in setup mode, you will be prompted to provide values for a series of preference settings. Some of these settings involve setting up the communications services of the agent. Other settings are only prompted if you enter "advanced" setup mode or "all" setup mode (these "advanced" and "all" settings are marked with a ***(a)*** below). To enter advanced setup mode and thus be able to set advanced settings, use the **`--advanced`** command line option with **`--setup`** or use the prompt command **`setup advanced`**. To enter the "all" setup mode (which allows you to set every preference available), you must use the prompt command **`setup all`**.

- **Agent Hostname or IP Address** - this is the address that the agent will bind to when listening for incoming messages. This usually is the same address that the agent's remote clients (aka JBoss ON servers) will use when trying to connect to the agent. However, in some network setups, this may not always be the case (e.g. a remote client going through a router that forwards requests to a different host). If you want the JBoss ON server to connect to this JBoss ON agent via a different address, you need to set up some special transport parameters to indicate this.

- **Agent Port** - this is the port that the agent will actually be listening to. This usually is the same port that the agent's remote clients (JBoss ON servers) will use when trying to send messages to the agent. However, in some network setups, this may not always be the case (e.g. a remote client going through a router that forwards requests to a different port). If you want the JBoss ON server to connect to this JBoss ON agent via a different port, you need to set up some special transport parameters to indicate this.

- **Agent Transport Protocol*(a)*** - this is the transport the agent expects incoming messages to adhere to. This is usually "socket" or "sslsocket" (for raw binary socket messages, either unsecured or secured). JBoss/Remoting has several different types of transports available. Please refer to the JBoss Remoting documentation if you wish to experiment with other types of transports.

- **Agent Transport Parameters*(a)*** - these are additional parameters to be used when the agent creates its communications services and when its remote clients (JBoss ON servers) needs to talk to the agent. See *Section 11.9.7, "Agent Communications Transport parameters"* for the different types of transport parameters you can set.

- **RHQ Server Hostname or IP Address** - this is the IP address for the endpoint of the primary JBoss ON server this agent will talk to. This JBoss ON server IP Address value is dictated by the way the JBoss ON server has configured *its* communications services (using similar settings as the ones being described for the agent). Please refer to your JBoss ON server's communications configuration to see what exact value this should be.

- **RHQ Server Port** - this is the port that the primary JBoss ON server is listening to. This JBoss ON server Port value is dictated by the way the JBoss ON server has configured *its* communications services (using similar settings such as these being described for the agent). Please refer to your JBoss ON server's communications configuration for what exact value this should be.

- **RHQ Server Transport Protocol*(a)*** - this is the transport the primary JBoss ON server will expect its incoming messages to flow over. This JBoss ON server Transport value is dictated by the way the JBoss ON server has configured *its* communications services (using similar settings such as these being described for the agent). Please refer to your JBoss ON server's communications configuration for what exact value this should be.

- **RHQ Server Transport Parameters*(a)*** - these are additional transport parameters that are to be used when the agent connects to the primary JBoss ON server. This JBoss ON server Transport Parameters value is dictated by the way the JBoss ON server has configured *its* communications services (using similar settings such as these being described for the agent). Please refer to your JBoss ON server's communications configuration for what exact value this should be. In particular, you need to know what additional transport parameters the JBoss ON server wants its clients to define. This is especially important if the JBoss ON agent needs to connect to a different host and/or port than what the JBoss ON server actually binds to.
- **Command Send Timeout*(a)*** - this is the amount of milliseconds the agent will wait before aborting a command (i.e. the amount of time in milliseconds that the JBoss ON server has in order to process commands and return its results). Please ensure that this value is the same as the timeout specified in the transport parameters of your JBoss ON server URI (if specified), since both timeouts will be enforced. If this Command Send Timeout value is less than or equal to 0, the agent will not timeout its messages (note that if a timeout is specified in the JBoss ON server URI transport parameters, that timeout *will* be enforced).
- **Command Send Retry Interval*(a)*** - This is the minimum amount of time, in milliseconds, the agent will wait before trying to resend a guaranteed command that previously failed.
- **Command Send Max Retries*(a)*** - If a guaranteed delivery message is sent, but the agent fails to connect to the server and deliver the message, it will

  always be retried. However, if the error was something other than a 'cannot connect' error, the command will only be retried this amount of times before the command is dropped (at which time it will be considered lost forever).
- **Maximum Commands To Concurrently Send*(a)*** - This is the maximum number of commands the agent can send to the server at any one time. If you defined *clientMaxPoolSize* in your JBoss ON server URI transport parameters, make sure its value is the same as this "Maximum Commands To Concurrently Send" value since you effectively cannot have one higher than the other.

11.9.6. Configuring the JBoss ON Agent

11.9.6.1. Persistent Configuration

The agent configuration, made up of what are called **`preferences`**, are persisted on a per-user basis in an operating system-specific way (i.e. on Windows, they are stored in the registry; on UNIX in a directory located under the user's home directory). You can define different configuration preferences by specifying the **`-p`** command line argument of the agent to the name of the preferences node you want to use. This allows you to have several sets of configuration preferences and switch between them by passing in different **`-p`** options.

When an agent starts, it will check the preferences to see if they can be upgraded. Your preferences will then be upgraded to the latest schema. This is particularly helpful when you upgrade the agent. When the new agent starts up, it carries over any previous configuration, adds the old and new preferences, while also deleting any obsolete preferences.

You can pre-configure your agent so the first time it starts up, it can immediately register with a server and begin operating without manual intervention.

11.9.6.2. Prompt Commands

The agent can process simple prompt commands, entered as either input from a file (see command line option **`--input`**) or as input from the keyboard (assuming **`--daemon`** is not specified as a command line argument). You can see the list of prompt commands that are accepted by the agent by entering **`help`** at the prompt. To get detailed help on a particular prompt command, enter **`help`** followed by the name of the prompt command you are interested in. The following are the current set of prompt commands that the agent understands:

Prompt Command	Description
avail	Provides availability of inventoried resources
config	Manages the agent configuration
debug	Provides features to help debug the agent
discovery	Asks a plug-in to run a server scan discovery
download	Downloads a file from the JBoss ON server
dumpspool	Shows the entries found in the command spool file
exit	Shuts down the agent's communications services and kills the agent
failover	Shows/updates the HA failover list
getconfig	Displays one, several or all agent configuration preferences
help	Shows help for a given command
identify	Asks to identify a remote server
inventory	Provides information about the current inventory of resources
log	Configures some settings for the log messages
metrics	Shows the agent metrics
native	Accesses native system information
pc	Starts and stops the plug-in container and all deployed plug-ins
ping	Pings the JBoss ON server
piql	Executes a PIQL query to search for running processes
plugins	Updates the agent plug-ins with the latest versions from the server
quit	An alias for exit.
register	Registers this agent with the JBoss ON server
sender	Controls the command sender to start or stop sending commands
setconfig	Sets an agent configuration preference
setup	Sets up the agent configuration by asking a series of questions

Prompt Command	Description
shutdown	Shuts down all communications services without killing the agent
start	Starts the agent comm services so it can accept remote requests
timer	Times how long it takes to execute another prompt command
update	Provides agent update functionality
version	Shows the agent version information

11.9.6.3. JBoss ON server auto-detection and polling

The agent can be configured to auto-detect its JBoss ON server. It can do this in two different ways:

1. **Multicast detection**: Using JBoss/Remoting's multicast detection technology, the agent can usually detect the JBoss ON server coming online or going offline within a matter of seconds (a time which is configurable). This requires your network to support multicast traffic; if it does not, then you cannot use this method of server auto-detection. The following configuration preferences affect auto-detection using the multicast detector:

   - **rhq.agent.server-auto-detection** must be set to `true` in order to enable this feature.
   - **rhq.communications.multicast-detector.enabled** must be set to `true` in order to enable this feature.
   - **rhq.communications.multicast-detector.default-time-delay** is the number of milliseconds that must pass without hearing from the JBoss ON server before the JBoss ON server is to be considered "offline". To quickly detect a JBoss ON server shutting down or starting up, please set this to a short time. To reduce the amount of network traffic, please set this value to a longer time. However, ensure that this value is longer than the server's heartbeat-time-delay, otherwise, unnecessary network traffic will result.
   - **rhq.communications.multicast-detector.heartbeat-time-delay** is the number of milliseconds that must pass between the agent's own heartbeat messages. This value must be shorter than the JBoss ON server's default-time-delay otherwise, unnecessary network traffic will result.

2. **Server polling**: This mechanism polls the JBoss ON server periodically to determine if it is online or offline. This method of auto-detection does not require multicast traffic but does require the agent to periodically connect to the JBoss ON server and send it a ping command. The following configuration preference affects this "manual" server detection via polling:

   - **rhq.agent.client.server-polling-interval-msecs** is set to the number of milliseconds that must pass before polling the server. To quickly detect the JBoss ON server going down or coming up, set this to a short time; to reduce the amount of network traffic, set this to a longer time. If this value is 0 or less, server polling is disabled.

Typically, one or both of these mechanisms are enabled. With the ability to auto-detect the JBoss ON server going offline, the agent will be given the opportunity to persist commands that are waiting to be sent and allows the agent to shutdown its attempts to send commands. When the JBoss ON server comes back online and is auto-detected, the agent can resume. If, however, both auto-detection features are disabled, then the agent, upon start up, will immediately assume the JBoss ON server is online and will allow commands to be sent. If, at some point, the JBoss ON server is down, the agent will continually attempt to send it commands - and receive "connection refused" errors. If the JBoss ON server is down for a long period of time, this will cause the agent log file to grow very large. This is one reason why it is best to have at least one auto-detection mechanism enabled.

11.9.6.4. Throttling

The agent has several configuration preferences that define its client-side commands sender - they limit how many resources it can use and how "fast" it can perform some functions (called *throttling*). These configuration preferences have two main purposes: 1) to help limit the amount of resources the agent is able to claim for itself and 2) to help avoid flooding the server with large amount of commands which could put too-heavy a load on the JBoss ON server and/or starve other agents from being able to communicate with the JBoss ON server. The following configuration preferences define the settings that enable the agent to throttle its outbound messages. Most of these settings should be configured with the other settings in mind. While these do work independently, their effects are usually determined not by their own value but by related values. For example, a queue-size should be set to a larger number if the command timeout is lengthened. This is because if commands are given more time to complete, then more commands will be in the queue waiting to be sent. But, if max-concurrent is raised, this would allow more commands to be dequeued at any one time, so an increase in the queue-size may not be needed. As you can see, all of those preferences set an independent parameter within the agent, but their effects on the agent's behavior as a whole is dependent on the other agent's preferences.

- **rhq.agent.client.queue-size** defines the maximum number of commands the agent can queue up for sending to the JBoss ON server. The larger the number, the more memory the agent will be able to use up. Setting this to 0 effectively sets the queue to be unbounded. Please take caution when setting this to 0; if the JBoss ON server is down for a long period of time, the agent may run out of memory if it attempts to queue up more commands than it has memory for.

- **rhq.agent.client.max-concurrent** is the number of messages the agent can send at any one time. The larger the number, to more messages the agent can dequeue (thus freeing up space in the queue for more messages to come in). However, the higher this number is, the more messages will get sent to the server at the same time and may require the agent to use more CPU cycles.

- **rhq.agent.client.command-timeout-msecs** defines the amount of time the agent will wait for the JBoss ON server to reply with a response from a command before that command will be aborted. The longer this time is, the less of a chance the agent will abort a command that otherwise would have succeeded (e.g. if the server just needs a lot of time to process the particular command). However, the longer this time is, the more messages have to be queued up and wait before being sent to the server.

- **rhq.agent.client.retry-interval-msecs** is the amount of time the agent will wait before attempting to retry a command. Only those commands that are flagged for guaranteed delivery will be retried. Non-guaranteed commands (aka volatile commands) will not be retried and thus this setting will have no effect.

- **rhq.agent.client.send-throttling** , if defined, enables send-throttling. When this is enabled, only a certain number of commands can be sent before the agent enters a *quiet period*. During the quiet period, no throttle-able commands are allowed to be sent to the server. The commands can resume after the quiet period ends. Send throttling only affects those commands configured as "throttle-able" - these are typically commands containing metric collection data (i.e. those commands that tend to be sent to the JBoss ON server very frequently and in large numbers). Any other commands are not affected by the send-throttle. Send throttling helps in preventing message storms on the

JBoss ON server, thus helping to avoid the server from getting flooding with incoming messages and preventing agent starvation (that is, not locking out other agents from being able to talk to the JBoss ON server). The send-throttling preference defines both the maximum number of commands that can be sent and the length of the quiet period. For example, a preference value of "50:10000" means that after 50 'throttleable' commands are sent, a quiet period will commence and last for 10000 milliseconds. After that time expires, 50 more commands can be sent before the next quiet period begins.

- **rhq.agent.client.queue-throttling** , if defined, enables queue throttling. This limits the amount of commands that can be dequeued in a given amount of time, called the *burst period*. If more commands are attempted to be dequeued during the burst period than allowed, those dequeue requests will be blocked until the next burst period begins. For example, if this is set to "50:10000", it means that at most 50 commands can be dequeued in any 10000 millisecond interval. If, during a burst period, a 51st command attempts to be dequeued, that dequeue request will block until the burst period finishes (at which time a new burst period begins and the dequeue request becomes the first of the next 50 allowed dequeue requests). The purpose of queue throttling is not so much to limit the amount of requests being sent to the server (although this does have that side-effect), it really is to prohibit the agent from spinning the CPU too much as it attempts to dequeue and send commands as fast as it can. If an agent is using too much CPU cycles, you can throttle the queue thus (hopefully) reducing the amount of CPU required for the agent to send its commands. Note that if you enable queue-throttling, you must take care in ensuring your queue-size is large enough (since you are limiting the amount of commands that can be dequeued in a specific amount of time, you need to make sure you have enough space in the queue to support the extra amount of commands that get queue up).

11.9.6.5. Guaranteed Delivery

Some commands that the agent sends to the JBoss ON server are not critical in the grand scheme of things. For example, if a ping request fails to make it to the JBoss ON server, we do not want to retry it nor do we want to persist the command to ensure it survives an agent shutdown. These commands are called *volatile commands*. Volatile commands are sent once - if they fail for whatever reason to be successfully processed by the JBoss ON server, the failure is logged and the agent drops the command and moves on to the next that it needs to send.

However, there are some commands that must make their way to the JBoss ON server and the agent must ensure the JBoss ON server processes them. The agent must guarantee that these commands are delivered - these are called *guaranteed commands*.

Important

While the agent will do its best to guarantee the delivery of guaranteed commands; this guarantee is not 100%. That is to say, there may be rare circumstances that arise that cause a guaranteed command to fail to get delivered (e.g. if the JVM crashes suddenly in the middle of an attempt to send a guaranteed command).

Guaranteed commands are retried every X milliseconds while the agent is alive and actively sending commands to the server (where X is the *rhq.agent.client.retry-interval-msecs* preference setting). Guaranteed commands also survive agent shutdowns. If an agent shuts down prior to being able to deliver a guaranteed command, that command is persisted to disk in what is called the *command spool file*. The next time the agent starts up, it will load up commands it has spooled to disk and immediately queue them for sending to the JBoss ON server.

There are a couple preferences that define the behavior of this command spool file:

- **rhq.agent.client.command-spool-file.params** defines the parameters for the spool file. The value's format is defined as "max-file-size:purge-percentage". The first number is the size, in bytes, of the maximum file size threshold. If the spool file grows larger than this, a "purge" will be triggered in order to shrink the file. The second number is the purge percentage which indicates how large the file is allowed to be after a purge. This is specified as a percentage of the first parameter - the max file size threshold. For example, if the max file size is 100000 (i.e. 100KB) and the purge percentage is 90, then when the spool file grows larger than 100KB, a purge will be triggered and the file will be compressed to no more than 90% of 100KB - which is 90KB. In effect, 10KB will be freed to allow room for new commands to be spooled. When this occurs, unused space is compressed first and if that does not free up enough space, the oldest commands in the spool file will be erased in order to make room for the newer commands.

- **rhq.agent.client.command-spool-file.compressed** is a true or false flag. If this flag is true, the commands stored in the spool file will be compressed. This can potentially save about 30%-40% in disk space (give or take), however, it slows down the persistence mechanism considerably. The performance hit will only appear when unusual conditions occur, such as shutting down while some guaranteed commands have not been sent yet or if the JBoss ON server is down. It will not affect the agent under normal conditions (while running with the JBoss ON server up and successfully communicating with the agent).

11.9.6.6. Transports

Both the JBoss ON agent and JBoss ON server use the same underlying communications services (based on JBoss/Remoting technology). One feature this enables is the ability for the communications layer to use different transports simply by changing configuration preferences. The following configuration preferences define the transports used by the agent:

- **rhq.agent.server.transport** defines the transport protocol that the agent will use to talk to the JBoss ON server

- **rhq.communications.connector.transport** defines the transport that the agent, itself, expects the JBoss ON server to use when the server wants to send messages to the agent.

The transports that are supported are those supported by JBoss/Remoting - which today includes: *socket* (raw and unencrypted socket based transport), *sslsocket* (encrypted and optionally authenticated SSL transport), *servlet* and *sslservlet*.

In additional to customizing the transport, you can also provide transport parameters that help define the behavior of the connection using the configured transport.

- **rhq.agent.server.transport-params** defines the transport parameters used when connecting to the JBoss ON server

- **rhq.communications.connector.transport-params** defines what transport parameters the JBoss ON server should use when sending messages to the agent

See *Section 11.9.7, “Agent Communications Transport parameters”* for more information on configuring these transport parameters.

11.9.7. Agent Communications Transport parameters

NOTE

Unlike JBoss ON servers, JBoss ON agents do not host a servlet container. This means that servlets cannot be used for server-to-agent communications; these connections use sockets. Only agent-to-server connections use servlets.

Both the JBoss ON server and JBoss ON agent use JBoss/Remoting as its underlying remoting framework. JBoss/Remoting defines remote endpoints (which identify how to connect to a JBoss ON server or JBoss ON agent) via *InvokerLocator* strings, which look like simple URLs. An example is **`socket://myhost.corp.com:16163/?transportParam1=value1&param2=value2`**. InvokerLocators consist of a transport protocol (**`socket:`**) as well as the host and port of the remote endpoint. Also as part of an InvokerLocator, are transport parameters which further customize the endpoint. They help define the behavior of the underlying communications connector (which is the thing that accepts incoming messages from remote clients). The JBoss ON server and JBoss ON agent can each define their own transport parameters via their *rhq.communications.connector.transport-params* preference setting (the agent's advanced setup mode will prompt you with **Agent Transport Parameters** when asking for this value).

Transport parameters are appended to the end of the InvokerLocator - in the same way a query string is appended to a URL. When you define your transport parameters, you must set them using the same syntax, specifically, multiple transport parameters are separated by ampersand characters. If you set transport parameters inside your XML configuration file, be sure you use the proper **&** string to represent the ampersand.

The **`backlog`** transport parameter is a server-side configuration and the **`enableTcpNoDelay`** transport parameter is a client-side configuration. When the JBoss ON server creates its server socket and begins listening for incoming messages from JBoss ON agents, it sets its backlog to 300. The JBoss ON server ignores the **`enableTcpNoDelay`** parameter because it is only useful for clients that want to talk to its server socket. When a JBoss ON agent sends a message to this JBoss ON server, the JBoss ON agent will ignore the **`backlog`** parameter because it is strictly a server-side setting but it will disable its TCP_NODELAY setting. As you can see, a single InvokerLocator can provide useful information for both ends of the communications channel (client and server).

Note

These parameters are actually features supplied by the underlying JBoss/Remoting infrastructure used by the JBoss ON communications layer. For more information on all available transport params, please refer to the *JBoss/Remoting configuration documentation*[5].

There are two main components that can be configured:

1. Multicast Detector
2. Connector

11.9.7.1. Multicast detector

The Multicast Detector is an optional service and is only needed if server auto-detection is enabled (rhq.agent.server-auto-detection=true). This service is responsible for listening for multicast heartbeat messages from remote JBoss ON servers running on the network. The Multicast Detector role is to detect if the JBoss ON server starting up or shutting down and notify the agent. The agent will immediately stop sending messages when notified that its JBoss ON server has shut down. It will begin spooling guaranteed commands to disk and continue queuing volatile commands in case the JBoss ON server starts. When the Multicast Detector sees the JBoss ON server is online, it will notify the agent which will then begin to start sending messages again.

You can configure the multicast address and port the detector will use. You can also configure the amount of time must pass without hearing from the JBoss ON server before that server is considered dead.

11.10. Configuring SSL Connections for Server-Agent Communication

By default, the JBoss ON server and JBoss ON agents talk to each other *in the clear*, meaning all communications traffic is unencrypted and no authentication is performed on either end.

Running servers in the clear, particularly since JBoss ON can perform configuration changes on some types of resources, can have security considerations for your netowrk. JBoss ON should only be run without encryption or authentication if JBoss ON is being tested or if all JBoss ON servers and agents are deployed on a fully secured network, with access limited by a firewall or VPN and restricted to trusted personnel.

JBoss ON uses SSL/TLS to secure connections between agents and servers in two separate ways:

- *Encryption* specially encodes the data sent between agents and servers during a session.
- *Authentication* uses SSL server and client certificates to verify the identity of an agent before it connects to a server, and vice versa.

NOTE

There is a basic authentication mechanism employed by the server in which it assigns security tokens to its agents which are used to identify and "authenticate" registered agents. This token mechanism should not, however, be considered a strong authentication scheme for the purposes of protecting your JBoss ON network from infiltration.

Setting up encryption is very simple; it only requires enabling the proper transport mechanism between servers and agents. This prevents an attacker from intercepting communications or data between a legitimate JBoss ON server and a legitimate JBoss ON agent, by sniffing data or setting up a man-in-the-middle attack.

Authentication adds another layer of protection by preventing an attacker from installing a "rogue" JBoss ON agent and letting it register itself on the JBoss ON system, so that the rogue agent has access to the network. Although setting up authentication is more complicated than using encryption alone, it is worth the effort to implement for the additional protection, especially if there are vulnerabilities in the network setup.

11.10.1. Setting up Encryption

All that need to be done to set up encryption is to enable the SSL transport connectors in the JBoss ON server and agent configuration files. There are two transport options for SSL, **`sslservlet`** and **`sslsocket`**.

The JBoss ON server has a default certificate that it can use for encryption and the agent can generate a self-signed certificate, so it's not necessary to generate or install additional SSL certificates for encryption alone.

1. First, enable SSL encryption on the JBoss ON server.

   a. Shut down the JBoss ON server.

   ```
   serverRoot/jon-server-2.4.0.Beta1/bin/rhq-server.sh stop
   ```

   b. Open the ***`serverRoot`*****`/jon-server-2.4.0.Beta1/bin/rhq-server.properties`** file for the sJBoss ON server.

   c. Edit the **`rhq.communications.connector.*`** settings to use SSL. To use the **`sslsocket`** transport method, which is recommended for authentication, update the **`rhq.communications.connector.transport`** method, set the port number to use for the socket, and remove the servlet specified in the transport parameters setting.

   ```
   rhq.communications.connector.transport=sslsocket
   rhq.communications.connector.bind-address=
   rhq.communications.connector.bind-port=55555
   rhq.communications.connector.transport-params=
   ```

   To use the **`sslservlet`** transport method, all that's necessary is to change the **`rhq.communications.connector.transport`** method.

   ```
   rhq.communications.connector.transport=sslservlet
   rhq.communications.connector.bind-address=
   rhq.communications.connector.bind-port=
   rhq.communications.connector.transport-params=/jboss-remoting-servlet-invoker/
   ServerInvokerServlet
   ```

   d. For setting encryption alone, make sure that certificate-based authentication is disabled:

   ```
   rhq.server.tomcat.security.client-auth-mode=false
   rhq.server.client.security.server-auth-mode-enabled=false
   ```

   e. Optionally, define the secure protocol to use. The default is TLS (which is usually fine), but you can set it to SSL.

   ```
   rhq.server.tomcat.security.secure-socket-protocol=TLS
   rhq.server.client.security.secure-socket-protocol=TLS
   ```

   f. Save the changes, and restart the JBoss ON server.

   ```
   serverRoot/jon-server-2.4.0.Beta1/bin/rhq-server.sh start
   ```

2. Then, enable SSL encryption in the agent.

   NOTE

   This shows how to edit the agent configuration by editing the agent configuration file. The agent configuration can also be edited by going through the advanced setup mode in the agent start script:

   ```
   agentRoot/rhq-agent/bin/rhq-agent.sh --cleanconfig --setup --advanced
   ```

   a. Open the agent configuration file:

   ```
   vim agentRoot/rhq-agent/conf/agent-configuration.xml
   ```

   b. Change the transport protocol to **`sslsocket`**.

   ```
   <entry key="rhq.communications.connector.transport"        value="sslsocket" />
   ```

   c. Set the server connection information so that it matches the configuration for the server. The bind address for the server is commented out by default, and the other parameters are set to the JBoss ON server defaults.

   ```
   <entry key="rhq.agent.server.transport"        value="sslsocket" />
   <entry key="rhq.agent.server.bind-port"        value="55555" />
   <entry key="rhq.agent.server.bind-address"     value="server.example.com" />
   <entry key="rhq.agent.server.transport-params" value="" />
   ```

   d. For setting encryption alone, make sure that certificate-based authentication is disabled. These parameters can be left commented out or can be explicitly set to turn off authentication.

   ```
   <entry key="rhq.communications.connector.security.client-auth-mode"       value="none"
    />
   <entry key="rhq.agent.client.security.server-auth-mode-enabled" value="false" />
   ```

   e. Optionally, define additional protocol settings for the agent. This is necessary if the server is configured to use transport protocols other than TLS.

   ```
   <entry key="rhq.communications.connector.security.secure-socket-protocol"
    value="TLS" />
   <entry key="rhq.agent.client.security.secure-socket-protocol"   value="TLS" />
   ```

   f. Exit the agent and restart it, using the **`--cleanconfig`** option to load the new configuration.

   ```
   agentRoot/rhq-agent/bin/rhq-agent.sh --cleanconfig
   ```

11.10.2. Setting up Authentication Between Servers and Agents

*Authentication*is the process of verifying something's identity. With certificate-based authentication, an entity has to obtain a certificate file from a trusted source and, when initiating an SSL connection, that certificate is used to identify that entity. This ensures that the only parties involved in an SSl connection are who they say they are.

To set up certificate-based authentication for JBoss ON, several steps need to be taken. Encryption has to be enabled, certificates have to be issued and stored for the JBoss ON server and agents, and the servers and agents have to be configured to reject messages from untrusted clients.

SSL authentication for JBoss ON is *bi-directional*. The agents are configured to authenticate to the server, and then the server is configured to authentication to the agents.

NOTE

It is possible to configure one-way authentication, where only the server or only the agents have to authenticate. The best security is with bi-directional authentication, which is the configuration given here.

There are two transport methods in JBoss ON that allow SSL connections, **`sslservlet`** and **`sslsocket`**.

The procedure below uses **`sslsocket`**, which allows the default given port to be used for GUI connections while a special port is used for server-agent SSL connections.

Using **`sslservlet`** leverages the embdedd Tomcat server, but this requires GUI users to authenticate to the server as well as enabling certificate-based authentication for agents. To allow GUI users to authenticate using their usernames and passwords, set up SSL more or less as outlined below (with some difference in the configuration file settings) and edit the JBoss ON server's Tomcat configuration file (***`serverRoot`*****`/jon-server-2.4.0.Beta1/jbossas/server/default/deploy/jboss-web.deployer/server.xml`** to uncomment the **`<Connector>`** section which says *Provides a secure but un-authenticated https connector for browsers to use.* and set the port for them to use.

1. Enable encryption, as in *Section 11.10.1, “Setting up Encryption”*, only make sure that client authentication is *not* disabled.

2. SSL socket connections will occur over a user-defined port. If necessary, open the firewall or VPN to allow access to that port.

3. Generate SSL certificates for each JBoss ON server and agent. For example:

   ```
   keytool -genkey -dname "CN=server1.example.com"  -keystore server1-keystore.dat -validity
    3650 -alias server1 -keyalg DSA -storetype JKS -keypass secret -storepass secret
   ```

   This creates a self-signed certificate with the following characteristics:

   - A common name (CN) value that is the same as the server hostname, **`server1.example.com`**. The **`-dname`** value must be the same as the hostname because during the initial steps of the SSL connection (the SSL handshake), the client will verify that the same identity which was issued the certificate is the same as the one presenting it. Meaning, it will match the hostname in the CN against the hostname of the server or agent presenting the certificate.

   - A keystore file called **`server1-keystore.dat`**

   - A validity period of 3650 days

- An alias of **server1**
- A key algorithm of DSA
- Stored in the JKS format in the keystore
- Key and storage passwords of **secret**

Your organization may have a method already for generating or obtaining certificates. This example uses **keytool**; other utilities, like **certutil**, can be used as well. The **keytool** documentation is available through the Oracle-Sun site at *http://java.sun.com/javase/6/docs/technotes/tools/windows/keytool.html*.

4. Put each self-signed certificate in a single truststore file.

   a. Export the self-signed certificate from each keystore:

   ```
   keytool -export -keystore server1-keystore.dat -alias server1 -storetype JKS -
   storepass secret -file server1-cert
   ```

   b. Import every certificate into a single truststore file:

   ```
   keytool -import -keystore truststore.dat -alias server1 -storetype JKS -file server1-
   cert -noprompt -keypass secret -storepass secret
   ```

   -alias is the name to give to the imported certificate in the truststore. For convenience, this is the same as the alias of the original keystore file.

   IMPORTANT

   Import *every* exported server and agent certificate into the same truststore file.

   c. Verify that all the certificates were successfully imported by using the **keytool** to list the certificates:

   ```
   keytool -list -keystore truststore.dat -storepass secret -storetype JKS

   Keystore type: JKS
   Keystore provider: SUN

   Your keystore contains 2 entries

   server2, Feb 25, 2010, trustedCertEntry,
   Certificate fingerprint (MD5): 24:D9:8A:50:BA:1B:26:08:DC:44:A8:2A:9E:8A:43:D9
   server, Feb 25, 2010, trustedCertEntry,
   Certificate fingerprint (MD5): 91:F8:78:15:21:E8:0C:73:EC:B6:3B:1D:5A:EC:2B:01
   ```

5. Distribute both the keystore and the truststore files to all the JBoss ON and server and agent machines. Be sure to distribute the keystores only to the machines which match the hostname in the CN of the certificate; putting the keystore on the wrong machine will cause SSL connections to fail.

a. For the server, copy the keystore into the ***serverRoot*/jon-server-2.4.0.Beta1/jbossas/server/default/conf/** directory of the JBoss AS server embedded in the JBoss Operations Network server. Make sure this file is named **keystore.dat**.

b. For the server, copy the truststore into the ***serverRoot*/jon-server-2.4.0.Beta1/jbossas/server/default/conf/** directory of the embedded JBoss AS server. Make sure this file is named **truststore.dat**.

c. For the agent, copy the keystore into the ***agentRoot*/rhq-agent/conf** directory. Any certificate file in the ***agentRoot*/rhq-agent/conf** directory is retained even after an automatic update.

d. For the agent, copy the truststore into the ***agentRoot*/rhq-agent/conf** directory.

6. Shut down the JBoss ON server.

```
serverRoot/jon-server-2.4.0.Beta1/bin/rhq-server.sh stop
```

7. Open the **rhq-server.properties** file for the sJBoss ON server.

```
vim serverRoot/jon-server-2.4.0.Beta1/bin/rhq-server.properties
```

8. Enable client authentication by setting the **rhq.communications.connector.security.client-auth-mode** parameter to **need** and the **rhq.server.client.security.server-auth-mode-enabled** parameter to **true**.

Set the information about the keystore and truststore files.

All of the configuration for incoming messages (agent-to-server communications) is set in **rhq.communications.connector.security.*** parameters. The configuration for outgoing messages is set in **rhq.server.client.security.*** parameters.

```
# Server-side SSL Security Configuration (for incoming messages from agents)
# These are used when secure transports other than sslservlet are used
rhq.communications.connector.security.secure-socket-protocol=TLS
rhq.communications.connector.security.keystore.file=${jboss.server.home.dir}/conf/
keystore.dat
rhq.communications.connector.security.keystore.algorithm=SunX509
rhq.communications.connector.security.keystore.type=JKS
rhq.communications.connector.security.keystore.password=secret
rhq.communications.connector.security.keystore.key-password=secret
rhq.communications.connector.security.keystore.alias=server1
rhq.communications.connector.security.truststore.file=${jboss.server.home.dir}/conf/
truststore.dat
rhq.communications.connector.security.truststore.algorithm=SunX509
rhq.communications.connector.security.truststore.type=JKS
rhq.communications.connector.security.truststore.password=secret
rhq.communications.connector.security.client-auth-mode=need

...

# Client-side SSL Security Configuration (for outgoing messages to agents)
rhq.server.client.security.secure-socket-protocol=TLS
rhq.server.client.security.keystore.file=${jboss.server.home.dir}/conf/keystore.dat
rhq.server.client.security.keystore.algorithm=SunX509
rhq.server.client.security.keystore.type=JKS
rhq.server.client.security.keystore.password=secret
rhq.server.client.security.keystore.key-password=secret
rhq.server.client.security.keystore.alias=myhost
rhq.server.client.security.truststore.file=${jboss.server.home.dir}/conf/truststore.dat
rhq.server.client.security.truststore.algorithm=SunX509
rhq.server.client.security.truststore.type=JKS
rhq.server.client.security.truststore.password=secret
rhq.server.client.security.server-auth-mode-enabled=true
```

9. Save the file and restart the server.

   ```
   serverRoot/jon-server-2.4.0.Beta1/bin/rhq-server.sh start
   ```

10. Uncomment the lines related to secure connections. These parameters begin with **`rhq.communications.connector.security.*`** and **`rhq.agent.client.security.*`** for agent-to-server communications and server-to-agent connections, respectively.

    Fill in the appropriate values.

    ```
    <entry key="rhq.communications.connector.security.secure-socket-protocol" value="TLS" />
    <entry key="rhq.communications.connector.security.keystore.file"          value="conf/
    keystore.dat" />
    <entry key="rhq.communications.connector.security.keystore.algorithm"
     value="SunX509" />
    <entry key="rhq.communications.connector.security.keystore.type"          value="JKS" />
    <entry key="rhq.communications.connector.security.keystore.password"      value="rhqpwd" /
    >
    <entry key="rhq.communications.connector.security.keystore.key-password"  value="rhqpwd" /
    >
    <entry key="rhq.communications.connector.security.keystore.alias"         value="rhq" />
    <entry key="rhq.communications.connector.security.truststore.file"        value="conf/
    truststore.dat" />
    <entry key="rhq.communications.connector.security.truststore.algorithm"
     value="SunX509" />
    <entry key="rhq.communications.connector.security.truststore.type"        value="JKS" />
    <entry key="rhq.communications.connector.security.truststore.password"    value="" />
    <entry key="rhq.communications.connector.security.client-auth-mode"       value="none" />

    <entry key="rhq.agent.client.security.secure-socket-protocol"   value="TLS" />
    <entry key="rhq.agent.client.security.keystore.file"            value="conf/
    keystore.dat" />
    <entry key="rhq.agent.client.security.keystore.algorithm"       value="SunX509" />
    <entry key="rhq.agent.client.security.keystore.type"            value="JKS" />
    <entry key="rhq.agent.client.security.keystore.password"        value="rhqpwd" />
    <entry key="rhq.agent.client.security.keystore.key-password"    value="rhqpwd" />
    <entry key="rhq.agent.client.security.keystore.alias"           value="rhq" />
    <entry key="rhq.agent.client.security.truststore.file"          value="conf/
    truststore.dat" />
    <entry key="rhq.agent.client.security.truststore.algorithm"     value="SunX509" />
    <entry key="rhq.agent.client.security.truststore.type"          value="JKS" />
    <entry key="rhq.agent.client.security.truststore.password"      value="" />
    <entry key="rhq.agent.client.security.server-auth-mode-enabled" value="false" />
    ```

NOTE

This shows how to edit the agent configuration by editing the agent configuration file. The agent configuration can also be edited by going through the advanced setup mode in the agent start script:

```
agentRoot/rhq-agent/bin/rhq-agent.sh --cleanconfig --setup --advanced
```

Index