



JBoss Operations Network 3.1

Setting up Monitoring Alerts and Operations

for monitoring resources and responding to incidents

Edition 3.1.2

JBoss Operations Network 3.1 Setting up Monitoring Alerts and Operations

for monitoring resources and responding to incidents

Edition 3.1.2

Ella Deon Lackey

dlackey@redhat.com

Legal Notice

Copyright © 2012 Red Hat, Inc..

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

The primary function of JBoss Operations Network is monitoring the status of your resources. The core of monitoring includes critical availability monitoring, collecting metrics on platform and server performance, and tracking events. JBoss ON also provides a way to define alerts and then notify administrators whenever a resource is performing poorly. This guide provides GUI-based procedures to view monitoring information, to track events, to define alerts and notifications, and to initiate operations.

Table of Contents

1. DOCUMENT INFORMATION	3
1.1. Giving Feedback	3
1.2. Document History	3
2. INTRODUCTION: MONITORING AND RESPONDING TO RESOURCE ACTIVITY	3
2.1. Monitoring and Types of Data	4
2.2. Alerts and Responses to Changing Conditions	5
2.3. Potential Impact on Server Performance	5
2.4. Differences with Monitoring Based on Different Resource Types	6
3. AVAILABILITY	6
3.1. Core "Up and Down" Monitoring	6
3.2. Viewing a Resource's Availability Charts	10
3.3. Detailed Discussion: Availability Duration and Performance	12
3.4. Detailed Discussion: "Not Up" Alert Conditions	14
3.5. Viewing Group Availability	15
3.6. Disabling Resources for Maintenance	16
3.7. Allowing Plug-ins to Disable and Enable Resources Automatically	18
3.8. Changing the Availability Check Interval	19
3.9. Changing the Agent's Availability Scan Period	20
4. METRICS AND MEASUREMENTS	20
4.1. Direct Information about Resources	20
4.2. Viewing Metrics and Baseline Charts	23
4.3. Viewing Live Values	26
4.4. Defining Baselines	28
4.5. Setting Collection Intervals for a Specific Resource	31
4.6. Enabling and Disabling Metrics for a Specific Resource	32
4.7. Changing Metrics Templates	33
4.8. Adding a PostgreSQL Query as a Metric	35
5. EVENTS	37
5.1. Events, Logs, and Resources	37
5.2. Event Date Formatting	38
5.3. Defining a New Event	38
5.4. Viewing Events	40
5.5. Detailed Discussion: Event Correlation	40
6. URL RESPONSE TIME MONITORING	41
6.1. Call-Time (or Response Time) Monitoring for URLs	41
6.2. Viewing Call Time Metrics	42
6.3. Extended Example: Website Performance	42
6.4. Configuring EJB Call-Time Metrics	44
6.5. Configuring Response Time Metrics for JBoss EAP 6/AS 7	46
6.6. Setting up Response Time Monitoring for EWS/Tomcat and JBoss EAP 5	53
7. RESOURCE TRAITS	63
7.1. Collection Interval	64
7.2. Viewing Traits	64
7.3. Extended Example: Alerting and Traits	64
8. RESOURCES WHICH REQUIRE SPECIAL CONFIGURATION FOR MONITORING	65
8.1. Configuring Tomcat/EWS Servers for Monitoring	65

8.2. Configuring the Apache SNMP Module	66
8.3. Metrics Collection Considerations with Apache and SNMP	69
9. REPORTS AND DATA	69
9.1. Dashboards and Portlets	69
9.2. Summary Timelines	73
9.3. Resource-Level Metrics Charts	74
9.4. Creating Custom Metrics Pages	75
9.5. Suspect Metrics Report	76
9.6. Platform Utilization Report	78
10. STORING MONITORING DATA	79
10.1. Changing Storage Lengths	79
10.2. Exporting Raw Data	80
11. PLANNING ALERTS	81
11.1. An Alerting Strategy in Four Questions	81
11.2. Basic Procedure for Setting Alerts for a Resource	83
11.3. Enabling and Disabling Alert Definitions	87
11.4. Group Alerting and Alert Templates	89
12. ALERT CONDITIONS	92
12.1. Reasons for Firing an Alert	92
12.2. Detailed Discussion: Ranges, AND, and OR Operators with Conditions	93
12.3. Detailed Discussion: Conditions Based on Log File Messages	94
12.4. Detailed Discussion: Dampening	95
12.5. Detailed Discussion: Automatically Disabling and Recovering Alerts	97
13. ALERT RESPONSES	98
13.1. Notifying Administrators and Responding to Alerts	99
13.2. Detailed Discussion: Initiating an Operation	99
13.3. Detailed Discussion: Initiating Resource Scripts	105
13.4. Detailed Discussion: Launching JBoss ON CLI Scripts from an Alert	106
13.5. Configuring SNMP for Notifications	111
14. VIEWING ALERT DATA	115
14.1. Viewing the Alert Definitions Report	115
14.2. Viewing Alerts	116
14.3. Acknowledging an Alert	119
14.4. Troubleshooting Alerts	120
15. OPERATIONS: AN INTRODUCTION	121
15.1. A Summary of Operation Benefits	121
15.2. About Scheduling Operations	122
16. MANAGING OPERATIONS: PROCEDURES	123
16.1. Scheduling Operations	123
16.2. Viewing the Operation History	125
16.3. Canceling Pending Operations	126
16.4. Ordering Group Operations	127
16.5. Running Scripts as Operations for JBoss Servers	129
16.6. Setting an Operation Timeout Default	130
16.7. Operation History Report	131
INDEX	132

1. DOCUMENT INFORMATION

This guide is part of the overall set of guides for users and administrators of JBoss ON. Our goal is clarity, completeness, and ease of use.

1.1. Giving Feedback

If there is any error in this *Admin: Setting up Monitoring, Alerts, and Operations* 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 **JBoss** products group.
2. Select **JBoss Operations Network** from the list.
3. Set the component to **Documentation**.
4. Set the version number to 3.1.2.
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.

1.2. Document History

Revision 3.1.2-2.400 Rebuild with publican 4.0.0	2013-10-31	Rüdiger Landmann
Revision 3.1.2-2 Removed references to installing mod_rt for Apache resources. Updated/restricted support for mod_snmp for Apache resources to Apache 2.2.	January 23, 2013	Ella Deon Lackey
Revision 3.1.1-1 Bug fixing for JBoss ON 3.1.1.	September 19, 2012	Ella Deon Lackey
Revision 3.1-0 Initial release of JBoss ON 3.1.	June 12, 2012	Ella Deon Lackey

2. INTRODUCTION: MONITORING AND RESPONDING TO RESOURCE ACTIVITY

One of the core functions of JBoss Operations Network is that it lets administrators stay aware of the state of their JBoss servers, platforms, and overall IT environment.

The current state of individual servers and applications provides critical information to IT staff about traffic and usage, equipment failures, and server performance. JBoss Operations Network can supply a clearer picture of these critical data by automatically *monitoring* resources in its inventory.

The most powerful aspect of management is the ability to know, accurately, where your resources are and to respond to that ever-changing situation reliably.

2.1. Monitoring and Types of Data

Monitoring gives insight into how a specific machine, application, or service is performing. JBoss ON collects different types of information from different native and external sources for its managed resources.

JBoss Operations Network is not a real-time monitor, and it is not an archive of data points. JBoss ON is not a profiler. What JBoss ON does is, in essence, filter and process raw data so that long-term trends, operating parameters, and performance histories — the purpose of monitoring — are clear and accessible from the data. JBoss ON uses *schedules* to define what information to gather and how frequently (anywhere from 30 seconds to hours). This prioritizes the performance information for a resource and makes important information more visible and coherent.

Although the precise information gathered is different depending on the resource type, there are a few broad categories of monitoring data. Each category obtains information from a different place and is useful to determine a different aspect of resource behavior.

Availability or "up and down" monitoring

This is both basic and critical. Availability is *status* information about the resource, whether it is running or stopped.

Numeric metrics

Metrics are the core performance data for a resource. Almost every software product exposes some sort of information about itself, some measurable facet that can be checked. This is usually This numeric information is collected by JBoss ON, on defined schedules.

Metric information is processed by the server. There are three states of the monitoring data used:

- *Raw data*, which are the readings collected on schedule by the agent and sent to the server
- *Aggregated data*, which is compressed data processed by the server into 1-hour, 6-hour, and 24-hour averages and used to calculate baselines and normal operating ranges for resources. These aggregated data are the information displayed in the monitoring graphs and returned in the CLI as metrics.
- *Live values*, which are ad hoc requests for the current value of a metric.

Metric values are rolling live-streams of the resource state; they are essentially snapshots that the agent takes of the readings on predefined schedules. Those data are then aggregated into means and averages to use to track resource performance.

Live values are immediate, aggregated, current readings of a metric value.

Metric information is especially important because it is collected and stored long-term. This allows for historical views on resource performance, as well as recent views.

Logfile messages (events)

While JBoss ON is not a log viewer, it can monitor specified logs and check for important log messages based on severity or strings within the log messages. This is *event* monitoring, and it allows JBoss ON to identify incidents for a resource and to send an alert notification and, if necessary, take corrective action based on dynamic information outside normal metrics.

Response time metrics

Certain types of resources (URLs for web servers or session beans) depend on responsiveness as a component of overall performance. Response time or call-time data tracks how quickly the URL or session bean responds to client requests and helps determine that the overall application is performant.

Descriptive strings (traits)

Most resources have some relatively static information that describe the resource itself, such as an instance name, build date, or version number. This information is a *trait*. As with other attributes for a resource, this can be monitored. Traits are useful to identify changes to the underlying application, like a version update.

2.2. Alerts and Responses to Changing Conditions

A critical part of monitoring is being aware of when undesirable events occur. Alerting works with other functions in JBoss ON management (monitoring data and configuration drift detection) to define *conditions* for triggering an alert.

When an alert condition is met, alerting in JBoss ON serves two important functions:

- **Alerts communicate** that there has been a problem, based on parameters defined by an administrator.
- **Alerts respond** to incidents automatically. Administrators can automatically initiate an operation, run a JBoss ON CLI script to change JBoss ON or resource configuration, redeploy content, or run a shell script, all in response to an alert condition.

Automatic, administrator-defined responses to alerts make it significantly easier for administrators to address infrastructure problems quickly, and can mitigate the effect of outages.

Alerts are based on metrics information, call-time data, availability, and events, all normal monitoring elements. Alerting can also be based on critical changes to a resource, defined in drift definitions that track configuration drift. Tracking configuration for resources along with monitoring data lets administrators remedy unplanned or undesirable system changes easily and consistently.

2.3. Potential Impact on Server Performance

Theoretically, there is no limit to the number of metrics that can be collected or the number of alerts that can be fired.

In reality, there are natural constraints within the IT environment that limit both monitoring and alert settings:

- Database performance, which is the primary factor in most environments
- Network bandwidth

There are no hard limits on JBoss ON's alerting and monitoring configuration since it depends on the number of resources, number of metrics, collection frequency, and the number of alerts.

As a rule of thumb, there are these performance thresholds:

- Up to 30,000 metrics can be collected per minute
- Up to 100,000 alerts can be fired per day (roughly 70 per minute)

Plan how to implement metrics collection and alerting. Prioritize resources and then the information required from those resources when enabling metrics schedules and setting collection frequencies. Then, based on those priorities, plan what alerts are required.

Clear monitoring and alerting strategies can help maintain performance while still gathering critical information.

2.4. Differences with Monitoring Based on Different Resource Types

Available metrics, events, traits, and other monitoring settings are defined for each resource type in its plug-in descriptor.

Obviously, software of completely different types have different possible monitoring configuration.

However, monitoring settings can be different between releases of the same software. Either different metrics are available or the same metric may have different configuration names. For example, JBoss EAP 4 and 5 have the same metrics, related to monitoring the EAP server JVM, threads, and transactions. Because of the different management structure in JBoss EAP 6, there are different metrics, related to management requests between the servers in the EAP 6 domain.

The [Resource Reference: Monitoring, Operation, and Configuration Options](#) has a complete references of available metrics for the official JBoss ON agent plug-ins. Check this guide to see what differences there are between release versions.

3. AVAILABILITY

One of the most basic elements for monitoring is knowing whether your server or application is running. *Availability* monitoring tells administrators that a certain process is running and minimally responsive.

3.1. Core "Up and Down" Monitoring

The first question with monitoring is *is the resource running?* A resource's availability is the first thing to check for overall performance, for determining service levels, and for maintaining infrastructure.

Availability (sometimes called *up or down monitoring*) determines whether a resource is *up* or whether it is in some other state.

Up means that the resource is running and that it responds to the agent within a prescribed time.

How availability is determined depends on the resource; it could be checking a process ID or a JVM or something else. Availability for a resource type is defined in its plug-in descriptor. Therefore, the plug-in container is the intermediary between the resource and the agent. The agent checks the plug-in container for resource availability; the container obtains it from the resource component.

Usually, an availability check takes a fraction of a second; for certain types of resources or in certain environments, it could take longer. There is a timeout period for availability scans, set to five (5) seconds by default. If a resource is running and responds to the availability scan within that five-second window,

the resource is up.

Because availability — or "up and down" — monitoring is so critical to IT administrators, availability states in JBoss ON are highly visible. Availability is displayed on resource details pages, in every list of resources, in groups, and in monitoring reports. The idea is that it should only take a glance to be able to determine whether your resource is up.

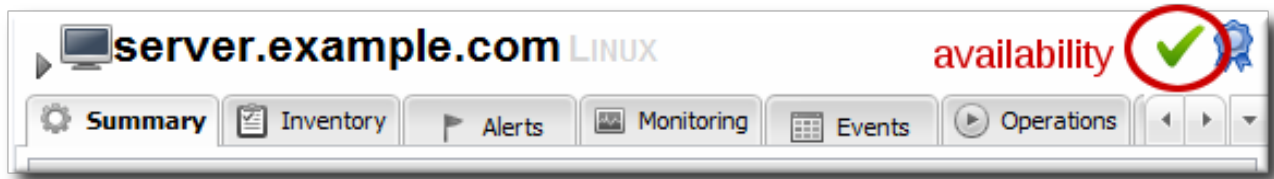


Figure 1. Resource Availability

Even though availability is not a true monitoring metric, the **Monitoring > Graphs** page even shows the percentage of time, within the display time period, that the resource has been in an up state. This is because availability (and concomitant uptime) impacts every other metric collected by the agent.

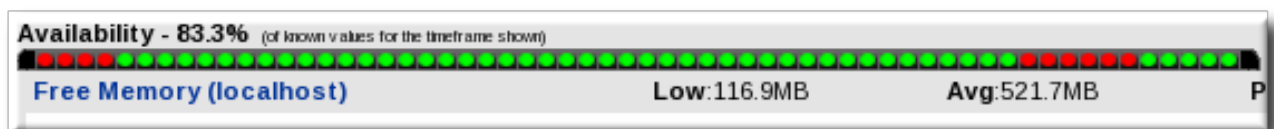


Figure 2. Availability Uptime Percentage



NOTE

Often, if a resource shows down availability even when it is running, it is a problem with the connection settings. The agent may not have information it requires, such as a username or new port number, that it requires to connect to the resource. Since the agent cannot connect to the resource, it assumes it is down.




3.1.1. Availability States

There is a gray area between *up* and *not up*. While a resource may not be up, it may be not up for different reasons. For instance, an agent could have been restarted, so no resource states are known. Or a resource may have been taken offline for maintenance, so no availability reports are being sent.

The different resource states are listed in [Table 1, "Availability States"](#).

Table 1. Availability States

State	Description	Icon
Available (UP)	The resource is running and responding to availability status checks.	✓
Down	The resource is not responding to availability checks.	!

State	Description	Icon
Unknown	The agent does not have a record of the resource's state. This could be because the resource has been newly added to the inventory and has not had its first availability check or because the agent is down.	
Disabled	The resource has been administratively marked as unavailable. The resource (in reality) could be running or stopped. Disabling a resource means that the server ignores the availability reports from the agent to prevent unnecessary alerts based on a (known) down or cycling state.	
Mixed (<i>For groups only.</i>) ^[a]	The resources in a group have different availability states.	

^[a] A similar warning sign can be displayed next to the resource availability at the top of the resource details page. That warning indicates that an error message or suspect metric has been returned for that resource, not that the resource's availability is in a warning state.

3.1.2. Collection Intervals and Agent Scan Periods

As alluded to, an availability reading is not the same as a metric collection. There are some superficial similarities, mainly in that they both are collected on schedules and that they both relate to resource performance.

Internally, availability and metrics are treated differently. Availability is called through different functions and reported separately, and, more important, availability reports are prioritized higher than other reports sent by the agent, including monitoring reports.

While availability reports are sent as first priority messages, resources themselves have different priorities for availability scans. Higher priority (more critical) resources are, by default, checked for availability more frequently:

- An agent heartbeat ping (analogous to the platform's availability) is sent to the server every minute.
- Server availability is checked every minute.
- Service availability is checked every 10 minutes.

The agent itself runs an availability scan at 30-second intervals. *Not every resource is checked with every scan.* When the agent scan runs, only those resources scheduled to be checked are checked. So, there are functionally two availability schedules working together in tandem, the agent scan interval and the resource collection schedule. For example, if a server is configured with a 60-second interval for

availability checks and the agent scan period is 30 seconds, the server is eligible to be checked every two scans. That means that the server is checked *roughly* every 60 seconds, but that is a best effort estimate; if the agent is under a heavy load or if there are a large number of resources, the agent may run its scans longer than every 30 seconds, so the *actual* interval between checks for a specific resource would be longer.

The agent only sends an availability report to the server if there is an availability state change for one of its managed resources.

If an agent goes down suddenly, it shows a down state within five minutes, the (default) agent quiet period. If the agent shuts down gracefully, the JBoss ON server recognizes the state change within about a minute. Once the server recognizes the agent is down, it begins backfilling the states of all of the resources in that agent's inventory ([Section 3.1.4, “Parent-Child States and Backfilling”](#)).

Down servers typically record a down state between one and two minutes after going down. This is not exactly real-time, but it is close enough for most infrastructure to be able to establish a reliable baseline of performance and even calculate service levels and uptime. A short window of 90 seconds can catch most resource cycling.

The default agent scan interval is 30 seconds, but, depending on a resource schedule, it could be over 10 minutes before some services are detected as down. If an administrator suspects that there has been a state change, it is possible to force an immediate availability scan for all resources for the agent through the interactive agent prompt:

```
> avail -- force
```

Using simply the **avail** command runs the check for the next scheduled resources, not all resources.

Additionally, resource plug-ins can be written so that any operation which could cause a state change (such as start, stop, and restart operations) automatically requests an availability check for the resource when the operation ends.

3.1.3. Long Scan Times and Async Availability Collection

Availability scans are performed by a resource plug-in itself, for its defined resource types, and then reported to the plug-in container..

Availability checks are typically very fast, fractions of a second, but there can be situations where an availability check takes longer. The plug-in container limits how long an availability check can run to five seconds, to prevent a rogue plug-in from delaying availability reporting for all other resources managed by the agent.

There can be instances where a certain plug-in or resource type consistently has scans longer than the five-second timeout period.

For custom plug-ins, plug-in writers can configure *asynchronous availability checking*. Basically, with async availability checks, the resource component creates its own, independent thread to run availability checks. Within that thread, the availability checks can take as long as they need to complete. The availability checks can also be run fairly frequently, every minute by default, to make sure that the availability state is current, even if the full check takes longer to complete.

The component caches and then reports the most recent availability result to the plug-in container. That stored last availability can be delivered very quickly, in the fractions of a second that the plug-in container expects.

Async availability checks are implemented through the **AvailabilityCollectorRunnable** class in the JBoss ON plug-in API. Details for this class are available in the [plug-in API](#) and [Writing Custom Plug-ins](#).



NOTE

It is also possible to address long availability check times by extending the scan timeout period in the agent configuration itself. For example, add a new timeout period to the **ADDITIONAL_JAVA_OPTIONS** parameters in the **rhq-agent-env.sh** file:

```
RHQ_AGENT_ADDITIONAL_JAVA_OPTS="-  
Drhq.agent.plugins.availability-scan.timeout=15000"
```

However, that timeout period applies to the *entire* plug-in container, not just one specific, slow-running plug-in. If there are several plug-ins that are running sluggish availability checks, then the availability report may take too long to complete, causing the agent to delay or even miss sending availability reports to the JBoss ON server.

Generally, it is preferable to configure async availability on a custom plug-in, rather than trying to reset the scan interval for all plug-ins.

3.1.4. Parent-Child States and Backfilling

Availability is assessed from the top of the resource tree downward. For example, if an application server is down, it is safe to assume that all of its dependent webapp children are also down.

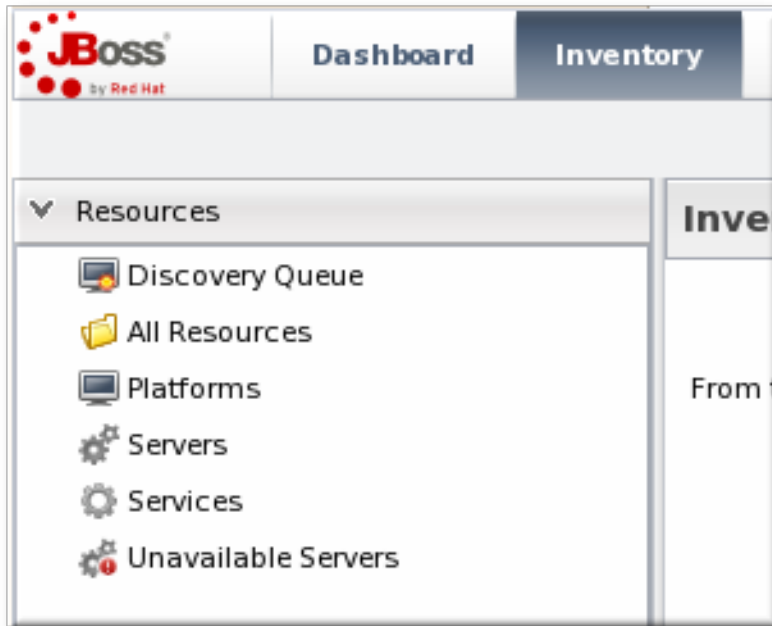
This is called *backfilling*. The parent's state is propagated to its children without running additional availability scans for each child. Backfilling can set children to down, unknown, or disabled states.

In some cases, backfilling even includes up states. Some dependent child resources (low priority services that only run if the parent is running) may not even have their own availability assessed independently by default. When a child's availability checking is disabled, the child presumptively uses its parent's state. If the parent is up, those children are assumed to be up.

There is one slight variation on backfilling — if a platform is marked as down. A platform being down is the same as the agent being down. It means that the agent has not reported to the server. There could be a number of reasons for that, apart from any servers or services actually being offline. In this case, the platform (functionally, the agent) is set to down, but its children are set to unknown.

3.2. Viewing a Resource's Availability Charts

1. Click the **Inventory** tab in the top menu.
2. Select the resource category, such as servers or services, in the **Resources** menu table on the left. Then browse or search for the resource.



3. Click the name of the resource in the list.
4. Open the resource's **Monitoring** tab.
5. Click the **Availability** subtab.

The **Availability** chart for a resource shows when, and for how long, a resource changes states. This includes timestamps of whenever the availability changes and total counts of how much time the resource spends in the up and down states.

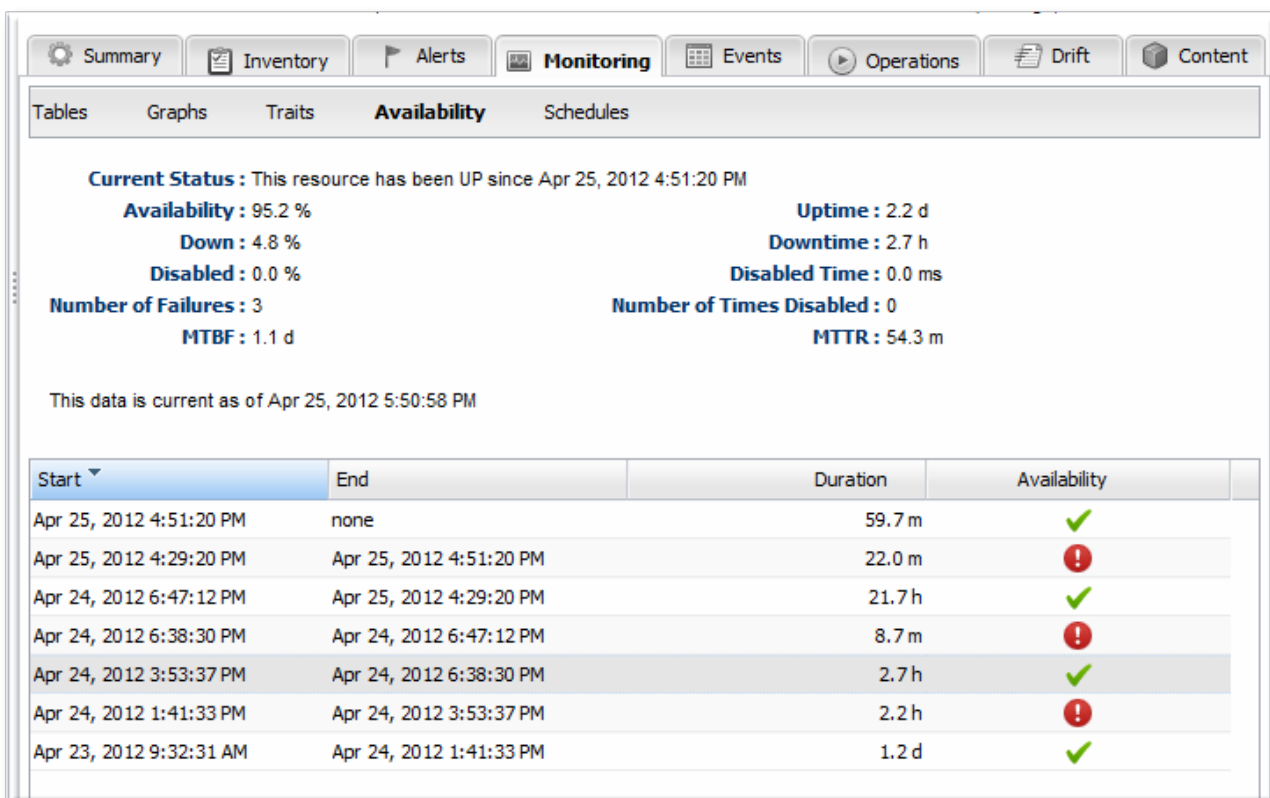


Figure 3. Availability Charts

3.3. Detailed Discussion: Availability Duration and Performance

Availability as a monitoring mechanism has two important facets: the immediate effect of when it changes and then the historic perspective on how changes in availability reflect resource performance.

An historic perspective introduces the idea of *availability duration*. How long was a resource in a particular state? How often does it change?



Figure 4. Availability Counts

The idea of availability duration is important to get an accurate picture of how a resource is performing. There are several ways that JBoss ON breaks out that information:

- Total time in up, down, and disabled states
- Percentage of time time in up, down, and disabled states
- The number of times the resource has been in a down or disabled state
- The mean time between failures (MTBF) and mean time to recovery (MTTR)



NOTE

Unknown states are not included in calculating the resource's overall availability history.

The last element is particularly important in assessing the resource's performance in light of its availability. The *mean time between failures* is the time between when a resource comes up and when it next goes down — it is the mean^[1] of all of its up periods. This gives an idea of how stable a system is. The *mean time to recovery* gives an idea of how long the resource stays down, which indicates its resilience or fault tolerance. A low MTBF and high MTTR indicate some potential maintenance problems or application instability on a resource.

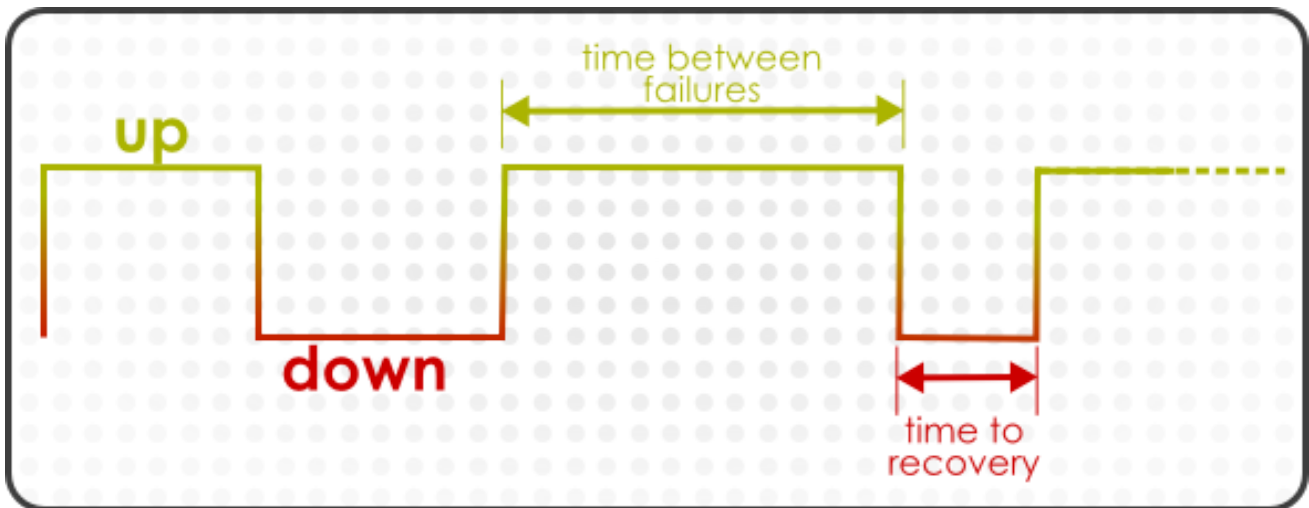


Figure 5. Up and Down Monitoring

From a monitoring perspective, the historic perspective is critical, particularly when planning equipment replacements and upgrades.

From an alerting perspective — from an immediate response perspective — only availability changes matter.

The first and most obvious alert condition issues an alert based solely on a state change.

However, resources can cycle or can have a few seconds or minutes where they are inaccessible but that doesn't affect the overall performance of the resource or of whatever function it performs. A resource hits a certain state and has to stay there for a certain amount of time before the state becomes important.

Add Condition

Condition Type: Availability Duration

Specify the availability state change and the duration that state must persist in order to trigger the condition. The duration is in minutes and should be long enough, several minutes, to give the agent time to detect another availability state change that may correct the problem.

Availability Duration: Stays Down

Duration: 5 minutes

OK Cancel

Figure 6. Availability Duration Alert

**NOTE**

An availability alert does not lend itself to dampening, because the state changes and then stays, such as an availability alert that fires when the resource changes to a down state. If a resource is cycling, it may go down and up several times, each time triggering a new alert, but it may all be related to the same performance issue on the resource.

Instead of dampening, a disable setting on the alert will fire the alert once, then disable that alert definition until it is acknowledged by an administrator, as described in [Section 12.5, “Detailed Discussion: Automatically Disabling and Recovering Alerts”](#). (In this case, do not set a corresponding recover setting; otherwise, if the resource is cycling, every UP reading would reset the alert and then the next DOWN report would fire another notification — essentially undoing the dampening effect of disabling the alert until acknowledgment.)

3.4. Detailed Discussion: “Not Up” Alert Conditions

There are four possible availability states for a resource:

- Up
- Down
- Unknown
- Disabled

These are summarized in [Section 3.1.1, “Availability States”](#).

Since one of the core monitoring factors for a resource is knowing its availability, alerts can be defined on any availability state change.

Generally, the condition can be set to send an alert on any explicit state. For example, a *goes down* condition alerts only when the availability state changes to DOWN. Any other state change is ignored.

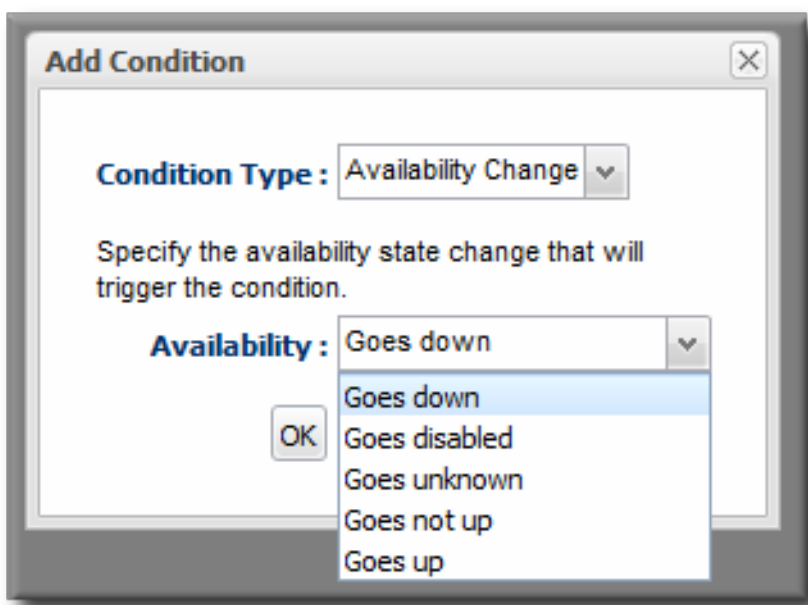


Figure 7. Availability Change Conditions

For critical platforms or resources, however, *any* change in availability other than UP may need to trigger an alert. Even known state changes like DISABLED.

The *goes not up* condition triggers an alert if there is a change to any availability state other than UP, so it is a logical OR combination of DOWN, UNKNOWN, and DISABLED conditions.



NOTE

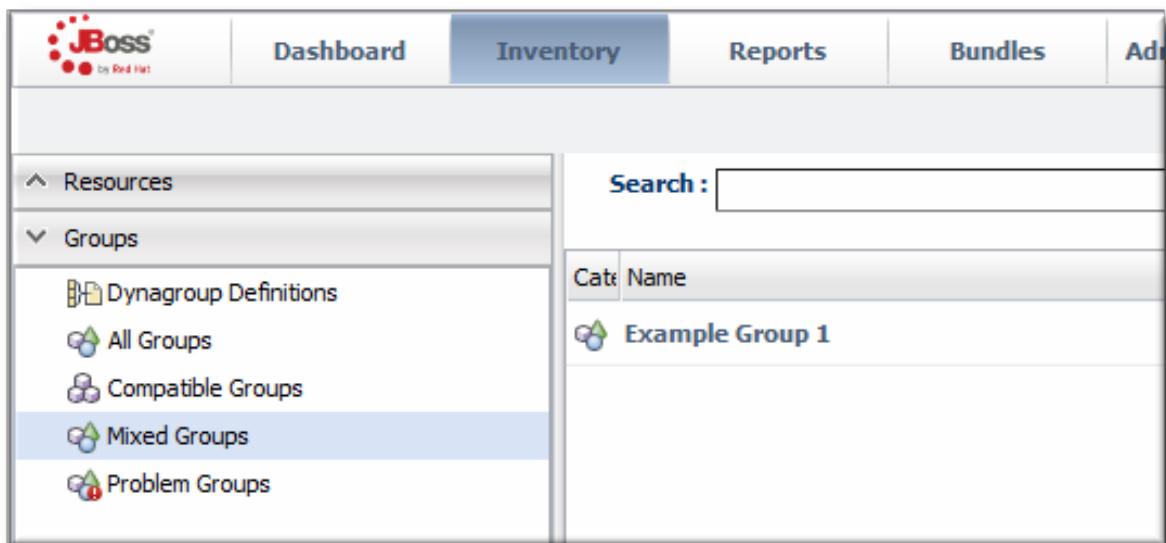
Availability change conditions are well suited to using *recovery alerts*. When a resource goes down (or not up) an alert can fire that informs the administrators and then enables (or recovers) a companion alert that will inform them when the resource is available again.

See [Section 12.5, “Detailed Discussion: Automatically Disabling and Recovering Alerts”](#) for more information.

3.5. Viewing Group Availability

To view group availability:

1. Click the **Inventory** tab in the top menu.
2. Select the compatible or mixed groups item in the **Groups** menu on the left.



3. Click the name of the group.
4. Click the **Inventory** tab for the group.

Group availability is a composite of the states of its member resources. If all resources are in one state or another, the group as a whole is in that state. If the resources are in different states, then the group state is determined based on the mix of resource states.

group MIXED composite group availability

Summary Inventory Alerts Events

Members individual resource availability

Search:

Name	Ancestry	Description	Type	Version	Availability
1.2.3.4:80	server.ex...	Apache Web Server	Apache HTTP Server	2.2.15	!
Tomcat (9180)	server.ex...	Tomcat Web Application Server (server.example.com)	Tomcat Server	Unknown Version	!
server.example.com:2099 RHQ Server	server.ex...	JBoss Application Server 4 hosting the RHQ Server	JBossAS Server	AS 4.2.3.GA	✓
server.example.com	server.ex...	Configuration for Samba Server	Samba Server		✓

Figure 8. Group Availability

**NOTE**

Availability states are evaluated "top down." If a resource is down, disabled, or unknown, then all of its children are immediately assumed to be in that state, as well.

Table 2. Group Availability States

If the Resource States Are the Group State Is ...
Empty Group (Unknown)	Empty
All Red (Down)	Red (Down)
Some Down or Unknown	Yellow (Mixed)
Some Orange (Disabled)	Orange (Disabled)
All Green (Up)	Green (Up)

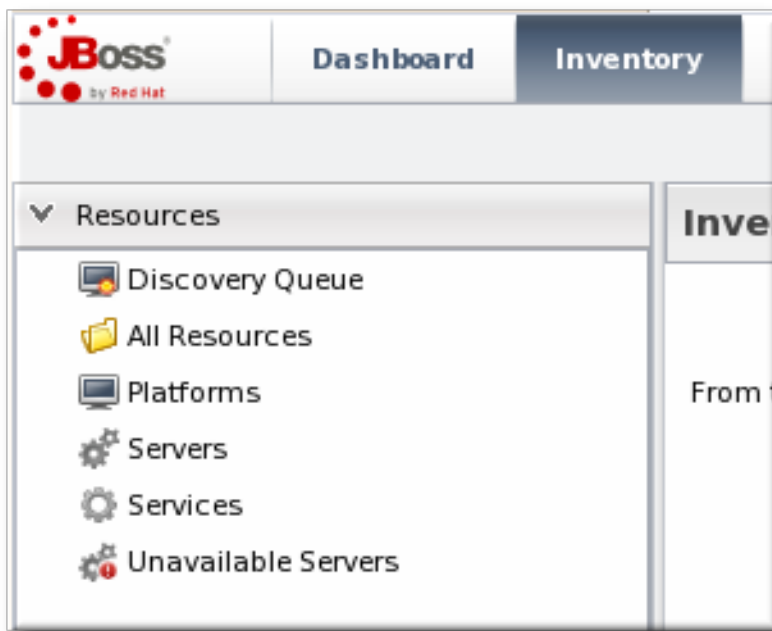
3.6. Disabling Resources for Maintenance

Disabling a resource essentially removes it from the JBoss ON server's view. There can be a lot of reasons why a resource will be taken offline — a machine could be moved to a new colocation facility, the platform may be upgraded, or there could be hardware changes. When an IT administrator knows that a resource will be unavailable, there is no reason to have an availability check which could trigger white noise of unnecessary reports. The resource can be *disabled*, which signals to the JBoss ON server that the resource availability is down (or cycling) and should be ignored.

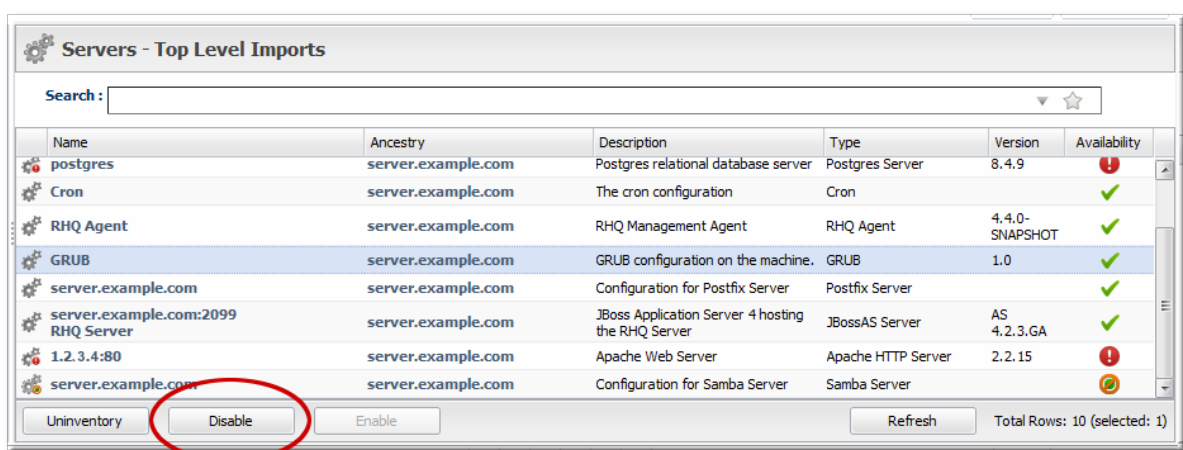
There are two things to remember when disabling a resource:

- If the agent is still up, then the resource availability is still reported. It is just ignored by the JBoss ON server, and is not included in any availability calculations.
- Disabling a parent resource automatically disables all of its children, too.

1. Click the **Inventory** tab in the top menu.
2. Select the resource category, such as servers or services, in the **Resources** menu table on the left. Then browse or search for the resource.



3. Select the resource in the list.
4. Click the **Disable** button at the bottom of the page.



5. When prompted, confirm that the resource should be disabled.

The disabled resource has an orange icon marking its state.

Name	Ancestry	Description	Type	Version	Availability
server.example.com		Linux Operati... System	Linux	Linux 2.6.32-262.el6.x...	✓
server.example.com	serve...	Config... for Samba Server	Samba Server		✓
1.2.3.4:80	serve...	Apache Web Server	Apache HTTP Server	2.2.15	!
server.example.com:2099	serve...	JBoss Applica... Server 4 hosting the RHQ Server	JBossAS Server	AS 4.2.3.GA	✓
server.example.com	serve...	Config... for Postfix Server	Postfix Server		disabled
GRUB	serve...	GRUB configur... on the machine.	GRUB	1.0	✓

Figure 9. Disabled Resource

**NOTE**

When the resource is re-enabled, it has an unknown state until the next scheduled availability scan.

3.7. Allowing Plug-ins to Disable and Enable Resources Automatically

Some child or dependent resources may consistently use a disabled state to indicate that the resource is inactive. For example, a managed server in a JBoss EAP 6 domain or a web context under `mod_cluster` may be offline because it is inactive, and this should be treated differently than being explicitly down. In this case, the parent resource can start or stop the dependent child automatically; when not started, the child is off, but not down.

The resource plug-in itself can automatically disable and enable dependent resources by using the `AvailabilityContext.disable()` and `AvailabilityContext.enable()` methods as part of its availability definition in its component JAR files.

**IMPORTANT**

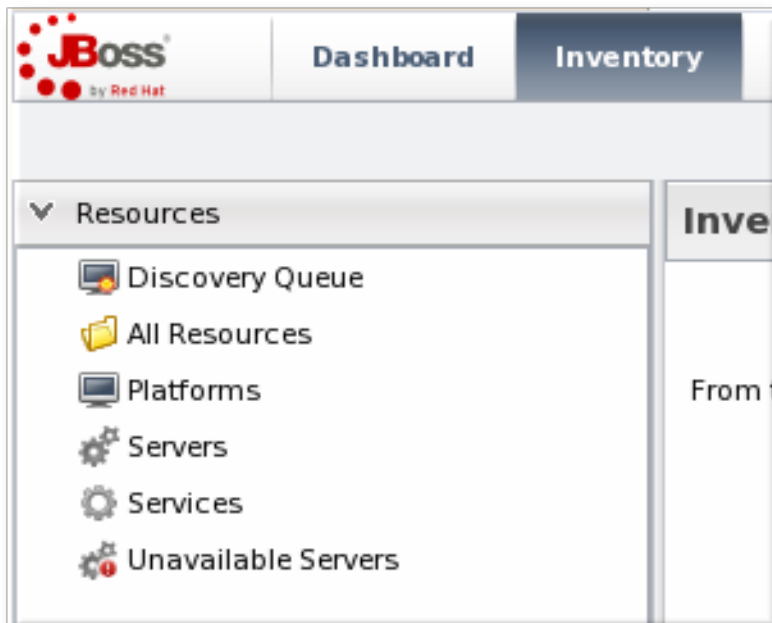
Be careful when allowing a resource plug-in to enable or disable a resource automatically. This potentially allows the plug-in to override whatever state the administrator has set.

For more information on writing resource plug-ins, see the *Development: Writing Custom Plug-ins*.

3.8. Changing the Availability Check Interval

While the availability check is not strictly a metric, it does have a collection schedule that can be edited with the other metric collection schedules.

1. Click the **Inventory** tab in the top menu.
2. Select the resource category, such as servers or services, in the **Resources** menu table on the left. Then browse or search for the resource.

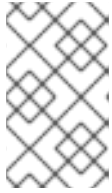


3. Click the **Monitoring** tab on the resource entry.
4. Click the **Schedules** subtab.
5. Select the availability metric, and enter the desired collection period in the **Collection Interval** field, with the appropriate time unit (seconds, minutes, or hours).

Metric ^	Description	Type	Enabled?	Collection Interval
Availability	The number of seconds between availability checks. The agent honors this setting as best as possible but the actual period can be longer based on agent activity.	availability	✓	1 minutes
Maximum request time	Max time for a request since last metric get	measurement	✓	2 minutes
Number of management requests	Number of requests sent to the controller	measurement	☐	2 minutes
Number of management requests per Minute	Number of requests sent to the controller	measurement	✓	2 minutes

Collection Interval : minutes

Total Rows: 11 (selected: 1)

**NOTE**

Availability schedules can be set on compatible groups or resource type templates. Setting it at the group or resource type level changes multiple resources simultaneously.

6. Click **Set**.

3.9. Changing the Agent's Availability Scan Period

Since availability is processed on the server, large environments with hundreds of agents and tens of thousands of resources can stress the server and hurt performance. In that case, the default scan interval may be too short, and setting a longer scan interval may improve JBoss ON server performance.

**NOTE**

When changing core agent or server settings, especially ones that impact JBoss ON performance, contact Red Hat Support Services for assistance.

1. Open the agent configuration file.

```
vim agentRoot/rhq-agent/conf/agent-configuration.xml
```

2. Uncomment the lines in the XML file, and set the new scan time (in seconds).

```
<entry key="rhq.agent.plugins.availability-scan.period-secs"  
value="60"/>
```

3. Restart the agent in the foreground of a terminal. Use the **--cleanconfig** option to force the agent to read the new configuration from the configuration file.

```
agentRoot/rhq-agent/bin/rhq-agent.sh --cleanconfig
```

4. METRICS AND MEASUREMENTS

Every operating system, application, and server has some mechanism for gaging its performance. A database has page hits and misses, servers have open connection counts, platforms have memory and CPU usage. These performance measurements can be monitored by JBoss Operations Network as *metrics*.

4.1. Direct Information about Resources

Metrics are a way of measuring a resource's performance or a way of measuring its load. The key word is measurement. A metric is some data point which software exposes, which is relevant to the operations or purpose of that software, that provides insight into the quantifiable behavior of that software.

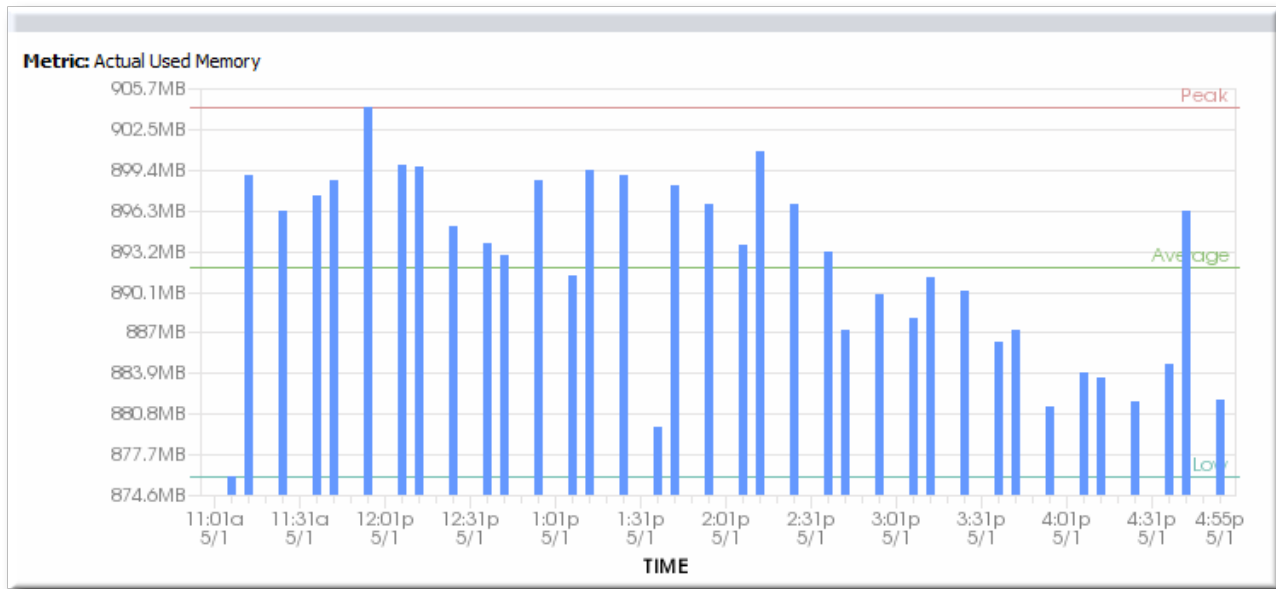


Figure 10. Metric Graph

Every type of resource has its own set of metrics, relevant to the resource type. Metrics are defined in the plug-in descriptor for that resource type. The plug-in descriptor lists the types of measurements which are possible and allowed for that resource; that's not necessarily the same thing as the metrics which are actually collected for a resource. Metrics themselves must be enabled (per resource or per metric template) and are then collected on schedule.

4.1.1. Baselines and Out-of-Bounds Metrics

After metrics have been collected for a reliable amount of time, JBoss ON automatically calculates a *baseline* for the metric. A baseline is the normal operating range for that metric on that resource.

Baselines are calculated values, not raw data points. Once every hour, a job is run that compresses these metric values into *one hour aggregates*. These aggregates contain the minimum, maximum, and average value of the measured data. Aggregates are also made for 6-hour and 24-hour windows.

Baseline metrics compare changes in actual data against a baseline value. Baselines allow effective trending analysis, SLAs management, and overall application health assessments as a form of fault management.

Baselines allow JBoss ON to identify metric values collected that fall outside (out-of-bounds) of the high and low baselines. Out-of-bounds metrics are reported as *problem metrics*.

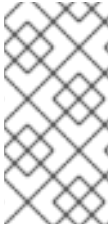


NOTE

When an alert is triggered in response to a metric value, the alerting event is tracked as a problem metric.

If there are no baselines present, because they have not yet been computed or because the metric is a trends-up or trends-down metric, no out-of-bounds factors will be calculated.

A baseline has a *bandwidth* that is the difference between its minimum and maximum values. The *difference* is the absolute amount that the problem metric is outside the baseline. To be able to compare out-of-bound values, an *out-of-bounds-factor* is computed by dividing the difference by the bandwidth. This creates a ratio to show comparatively how far out of the normal operation range the problem metric is.

**NOTE**

Calculating baselines can sometimes output non-intuitive results, as a band of (1,2) and an outlier value of 3 seems to be less than a band of (100, 200 MB) and an outlier value of 250 MB. The former is actually 100% outside the expected band, while the latter is only 50% outside.

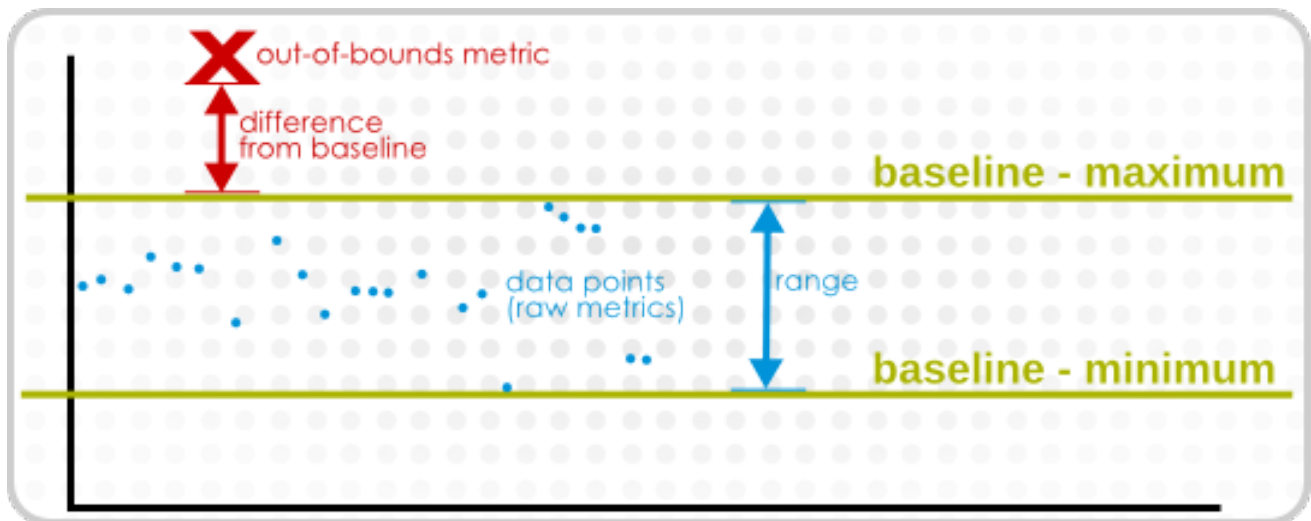


Figure 11. Out-of-Bound Factors

Out-of-bounds-factors are recalculated each hour during a calculation job. The job assesses the aggregate and determines if there is a more severe outlier than before. The chart always displays the most severe outlier.

When the baselines for a metric change, all recorded out-of-bounds values become invalid and are removed because the out-of-bounds measurement was computed against an old baseline.

4.1.2. Collection Schedules

The metric collection schedule is defined individually for each metric in the resource type's plug-in descriptor.

There is no rule on how frequently metrics are collected. Default intervals range between 10 minutes and 40 minutes for most metrics. While some metrics are commonly important (like free memory or CPU usage on platforms), the importance of many metrics depends on the general IT and production environments and the resource itself. Set reasonable intervals to collect important metrics with a frequency that adequately reflects the resource's real life performance.

The shortest configurable interval is 30 seconds, although an interval that short should be used sparingly because the volume of metrics reported could impact database performance.

4.1.3. Metric Schedules and Resource Type Templates

Unlike other types of monitoring data which are unique to an resource (availability, events, traits), metrics can be universal for all resources of that type.

Metric collection schedules define whether an allowed metric for a resource is actually enabled and what its collection interval is. A schedule is set at the resource-level, but administrator-defined default settings can be applied to all resources of a type by using *metrics collection templates*.

Templates are a server configuration setting. They define what metrics are active and what the collection

schedules are for all resources of a specific type. When templates are used, they supplant whatever default metrics settings are given in the plug-in descriptor. (A metric template only defines whether a metric is enabled and what its interval is — the plug-in descriptor alone defines what metrics are available for a resource type.)

These settings can be overridden at the resource-level, as necessary. Still, metrics collection templates provide a simple way to apply metrics settings consistently across resources and machines.

4.1.4. Raw Metrics, Displayed Metrics, and Storing Data

The live reading of metric information is *raw data*. This raw data is stored in the backend server, but it is not the information that is displayed in the web UI.

The information displayed in the web UI is *aggregated data*. The web UI has a limited display space, segmented into 60 x-axis segments. The JBoss ON server averages the raw data to create the data points for whatever the display time period is. For example, if the display range is 60 hours, each x-axis segment is 1-hour wide, and that data point is an average of all readings collected in that 1-hour segment. This aggregation is dynamic, depending on the monitoring window given in the chart views.

As [Section 4.1.1, “Baselines and Out-of-Bounds Metrics”](#) describes, the baseline calculations themselves are aggregates of the raw data, with 1-hour, 6-hour, and 24-hour windows to set minimum, maximum, and average baselines. Unlike the UI aggregates, these aggregated data are calculated and then stored as monitoring data in the server database.

Raw data are only stored for one week, by default, while aggregated values are stored for up to a year. The data storage times are configurable.

4.2. Viewing Metrics and Baseline Charts

The core of monitoring is the metric information that is collected for a resource. Each resource has different metrics (and these are listed in the *Resource Reference: Monitoring, Operation, and Configuration Options*). Three monitoring charts show the same information, but in different perspectives and different levels of detail:

- The resource-level Summary
- Graphs
- Tables

The **Summary** tab for resources, much like the Dashboard for the entire JBoss ON inventory, has portlets that show different resource information. Most resources have three portlets for measurements, events, and out-of-bound metrics. The **Measurements** portlet has small thumbnail charts that show the trend for the metric, along with the current reading.

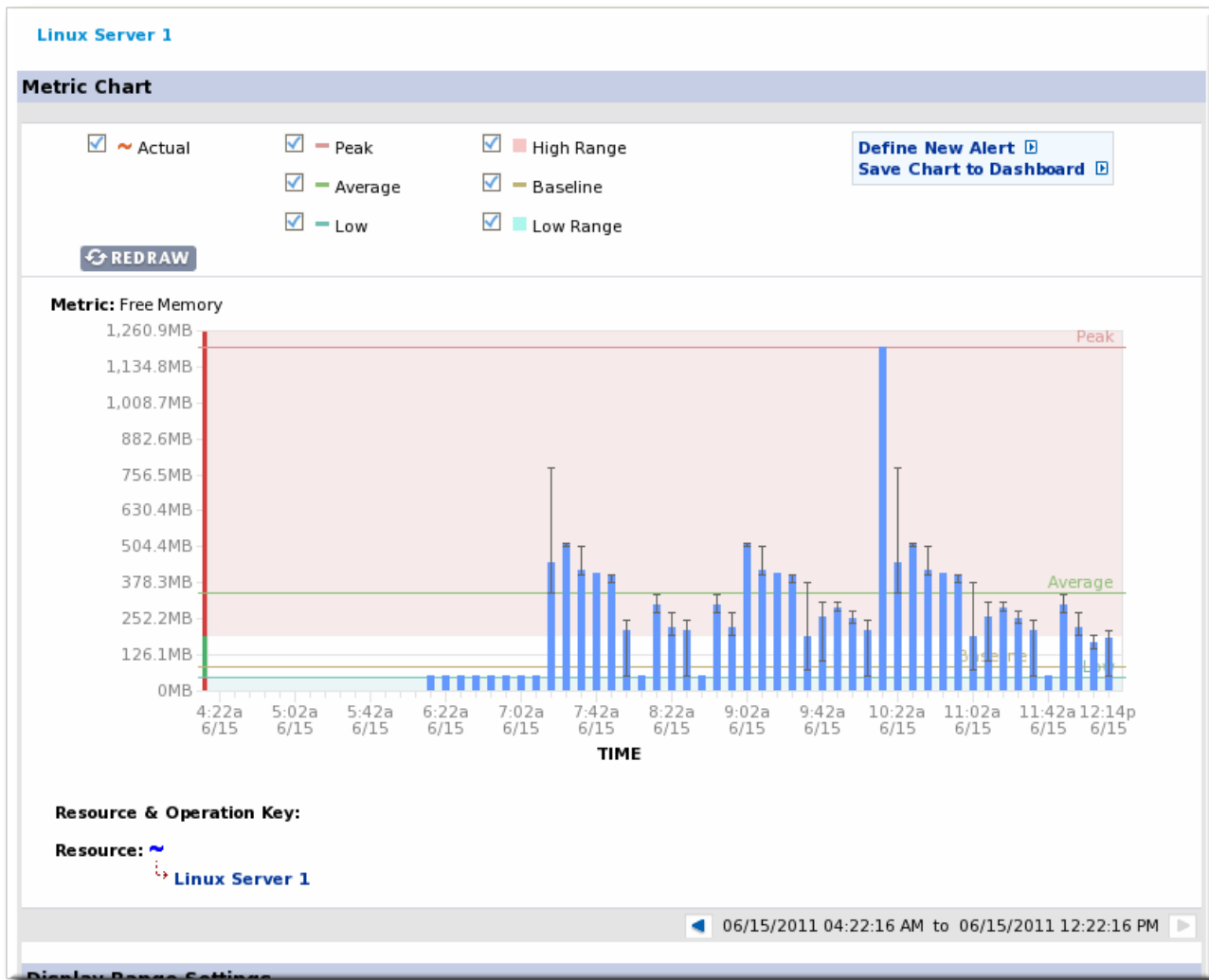
The screenshot displays the monitoring interface for a Linux server. The main title is "Linux Server 1" with a "Linux" tag. Below the title, there are navigation tabs for "Summary", "Inventory", "Alerts", "Monitoring", and "Events". The "Summary" tab is active, showing a "Resource: Measurements" panel with various metrics and their current values:

- Free Memory: 119.7MB
- Free Swap Space: 1.36GB
- System Load: 2.78%
- Total Memory: 1.93GB
- Total Swap Space: 2.5GB
- Used Memory: 1.81GB
- Used Swap Space: 1.14GB
- User Load: 10.4%

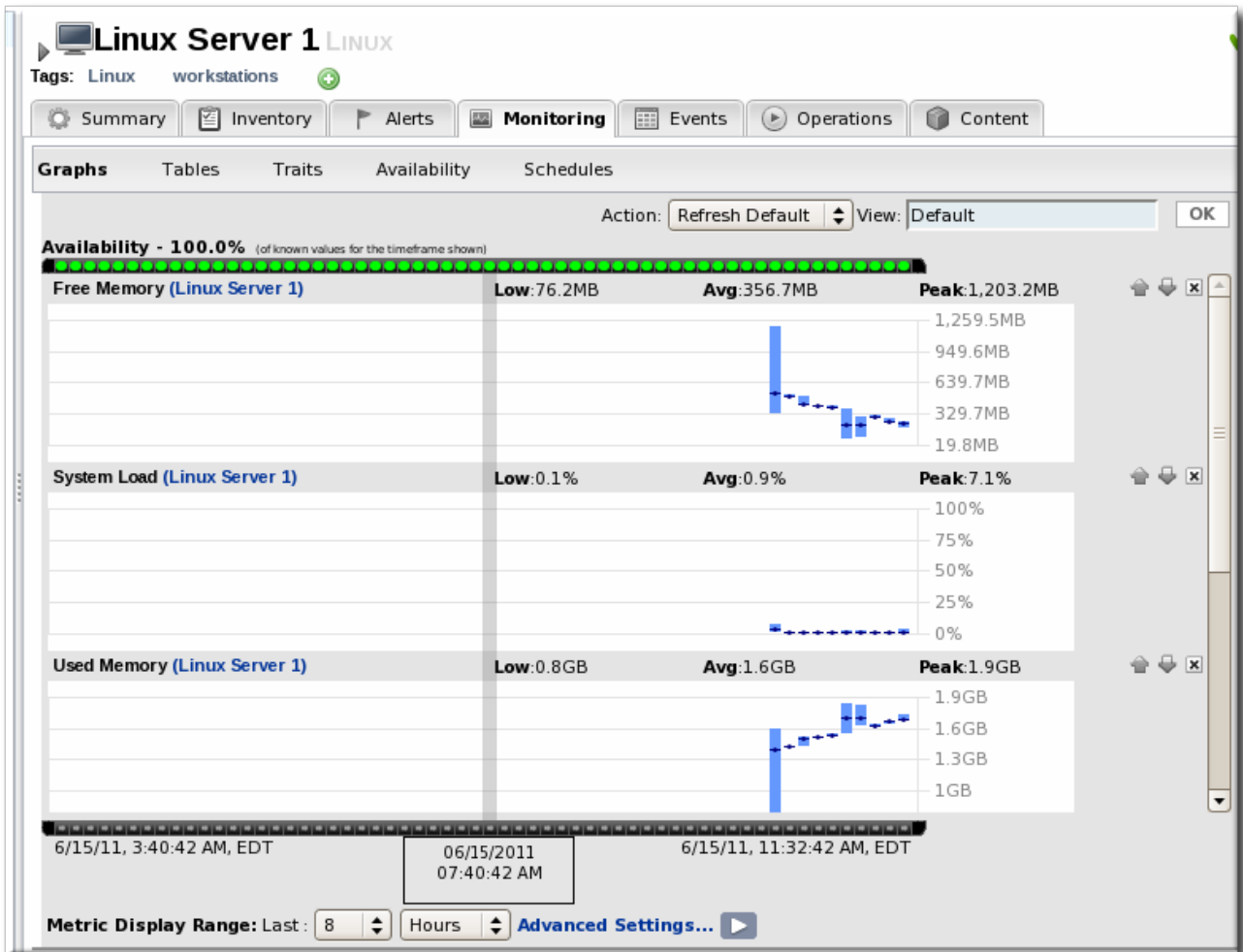
Below the measurements, there are three other panels:

- Resource: Event Counts:** No event counts based off display criteria.
- Resource: OOB Metrics:** Shows Out-of-Bounds (OOB) metrics for Used Memory, Free Swap Space, and Used Swap Space, all recorded at 10:00 AM -0400.
- Resource: Package History:** Lists installed packages, showing three instances of "publican-2.99-0.el6.t8.x86_64.rpm" installed at different times: 11 days, 19 hours ago, 11 days, 19 hours ago, and 11 days, 18 hours ago.

Clicking any of the metrics will open the baseline chart for that metric. As is described in [Section 4.1.1, “Baselines and Out-of-Bounds Metrics”](#), baselines calculate an average reading for a given period of time, with the high and low measurements in that period creating upper and lower bounds. Baselines, by default, are calculated every three days using the data from the previous seven days for the calculation. Baseline measurements are essential for establishing operating norms so that administrators can effectively set alerts for resources.



The **Graphs** area in the **Monitoring** tab shows all of the metrics on line graphs, giving the trend for the past eight hours, and the time span is dynamically configurable. This provides more granular detail than the summary or baselines charts, showing the readings for each collection period and the precise readings.



The **Tables** chart has the same information as the metrics graphs, only it is displayed in text, with columns for the high, low, and current readings. There is also a column which shows the number of active alerts for each metric.

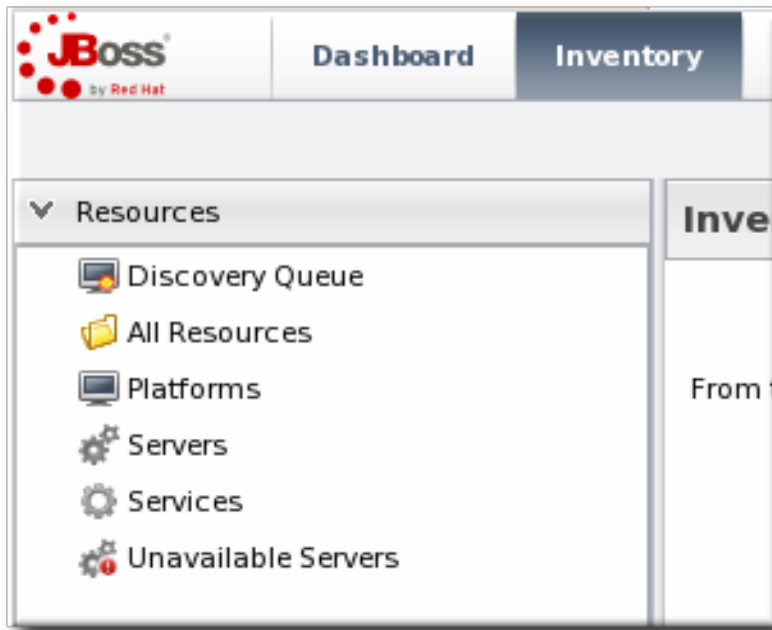
The screenshot shows the 'Tables' view of the monitoring interface. It displays a table with the following data:

Name	Alerts	Minimum	Maximum	Average	Last
Free Memory	0	48.9MB	1203.2MB	324.2MB	334.8MB
Free Swap Space	0	1.3091GB	1.3603GB	1.3541GB	1.3603GB
System Load	0	0.1%	9.4%	1.8%	1%
Total Memory	0	1.9255GB	1.9255GB	1.9255GB	1.9255GB
Total Swap Space	0	2.5GB	2.5GB	2.5GB	2.5GB
Used Memory	0	0.75GB	1.88GB	1.61GB	1.6GB
Used Swap Space	0	1.1387GB	1.1909GB	1.1455GB	1.1387GB
User Load	0	0.4%	35.8%	8.1%	2.5%

4.3. Viewing Live Values

The *live data* value is the current, one-minute average for the metric based on the last two metrics readings.

1. Click the **Inventory** tab in the top menu.
2. Select the resource category in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the resource name.
4. Open the **Monitoring** tab, and select the **Tables** subtab.
5. Select the metric (or metrics, using **Ctrl**) in the list.
6. Click the **Get Live Values** button at the bottom of the table.

Name	Alerts	Minimum	Maximum	Average	Last
Actual Free Memory	0	90.7 MB	119 MB	103.1 MB	113.1 MB
Actual Used Memory	0	876 MB	904.2 MB	891.9 MB	881.9 MB
Free Memory	0	61.9 MB	75.6 MB	67.8 MB	70.3 MB
Free Swap Space	0	313.1 MB	430.1 MB	349.7 MB	318.8 MB
System Load	0	0 %	3 %	0.2 %	0.5 %
Used Memory	0	919.4 MB	933.1 MB	927.2 MB	924.7 MB
Used Swap Space	0	577.8711 MB	718.125 MB	660.0942 MB	718.125 MB
User Load	0	0 %	29 %	1.3 %	2.9 %

Time Range - Previous : 8 hours

Get Live Value Chart Selected Metrics Refresh Total Rows: 8 (selected: 1)

7. The server displays the current (not aggregated) reading of the selected metrics.

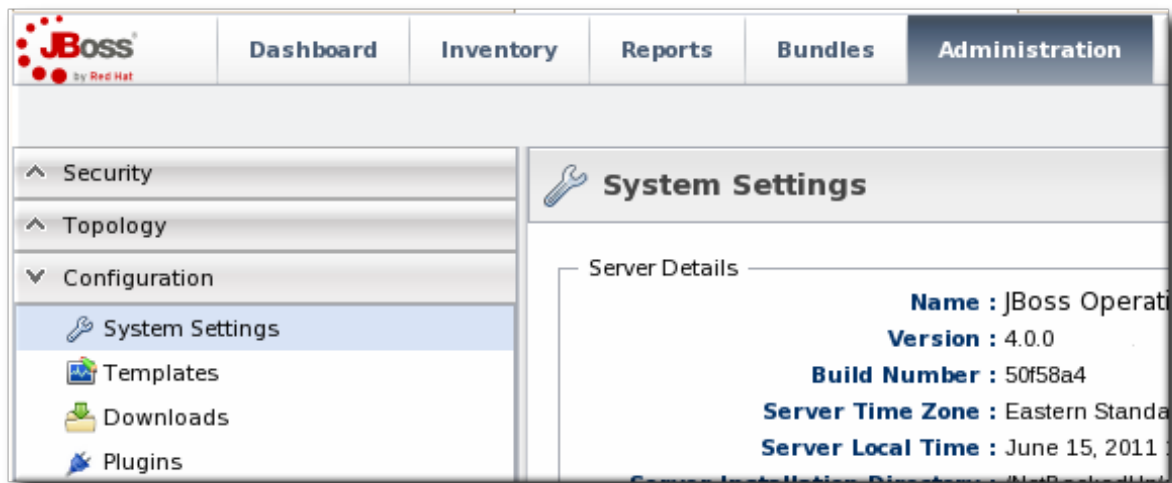
Metric	Value
Free Memory	67.8 MB

4.4. Defining Baselines

4.4.1. Setting Baseline Calculation Properties

The monitoring baselines have two configuration properties that define *how* the automatic metric baselines are calculated. These properties don't set the value; they set the window of time used for the baseline averages.

1. In the **System Configuration** menu, select the **Settings** item.



2. Scroll to the **Automatic Baseline Configuration Properties** section.
3. Change the settings to define the window used for calculation.

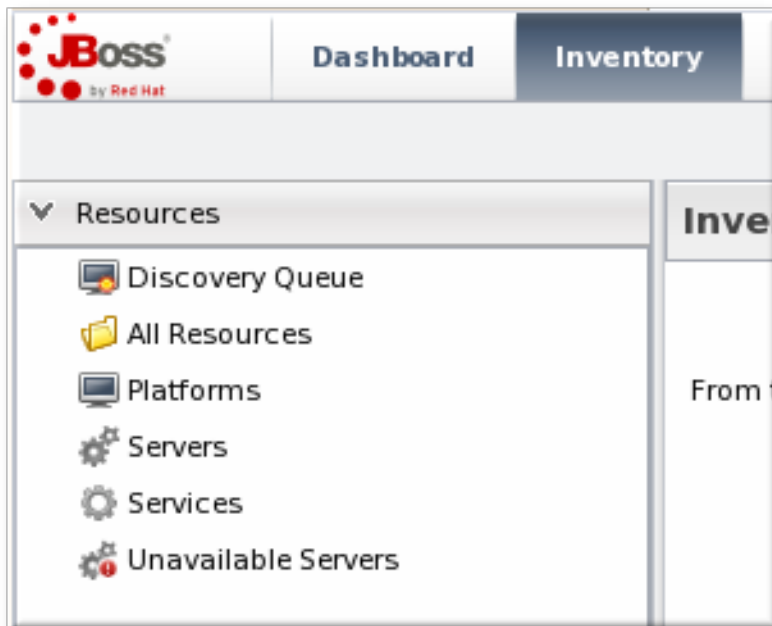
Property	Unset?	Value	Description
Baseline Dataset		7	The amount of past measurement data that is used to determine a baseline. This is specified in days.
Baseline Calculation Frequency		3	The frequency which the auto-calculation of baselines will be performed. If 0, baseline auto-calculation is disabled. This is specified in days.

- o *Baseline Frequency* sets the interval, in days, for how often baselines are recalculated. The default is three days.
- o *Baseline Dataset* sets the time interval, in days, used to calculate the baseline. The default is seven days.

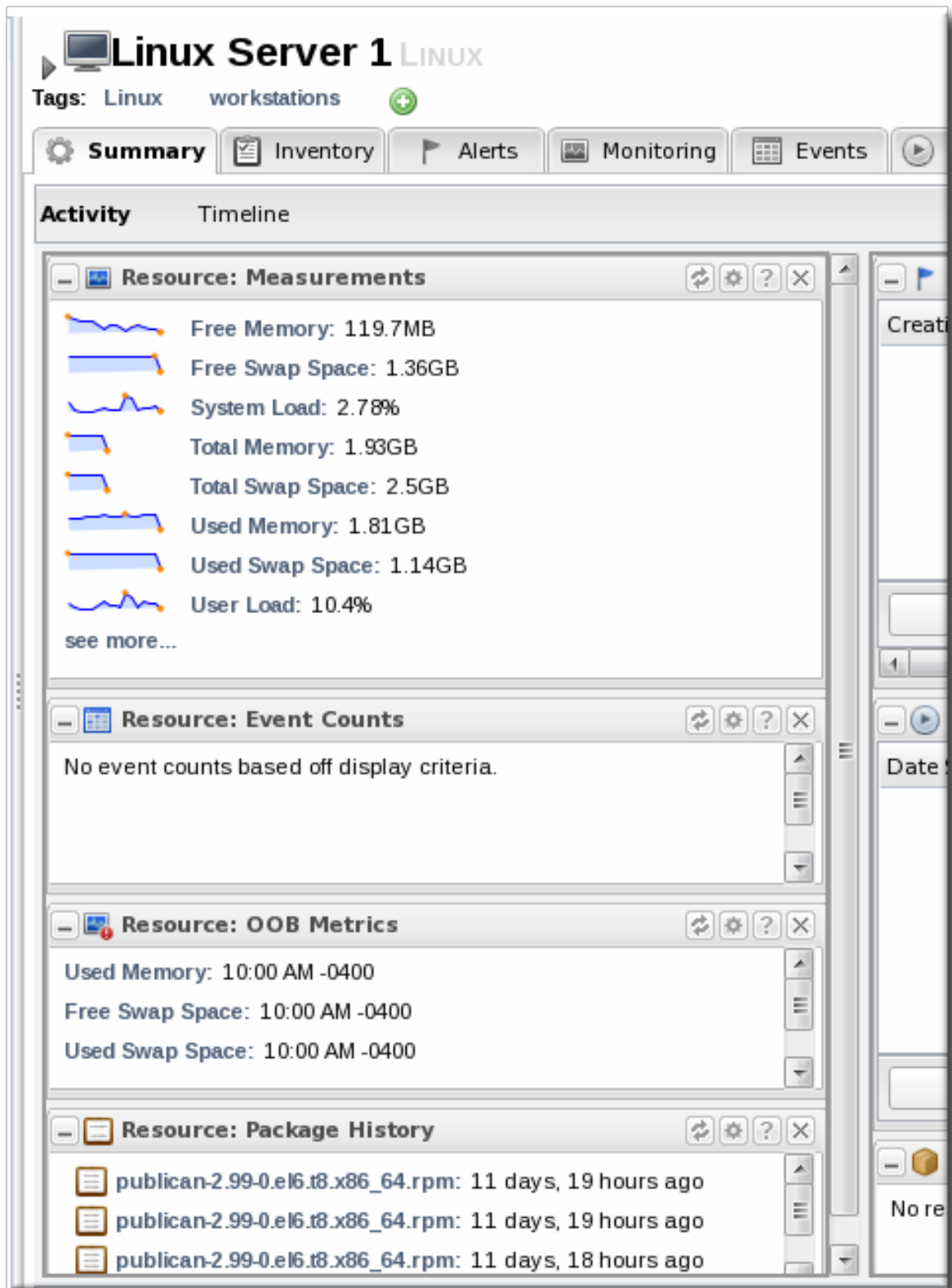
4.4.2. Recalculating Baseline Values

Baselines measure the average operating value of metrics. JBoss ON also collects the highest and lowest recorded readings to set a normal operating range. Comparing live metrics with pre-calculated baselines makes it possible to detect when resources are running outside of expected ranges. JBoss ON automatically calculates baselines; however, they can be recalculated for specific time periods or simply if the load has changed and new baselines are required.

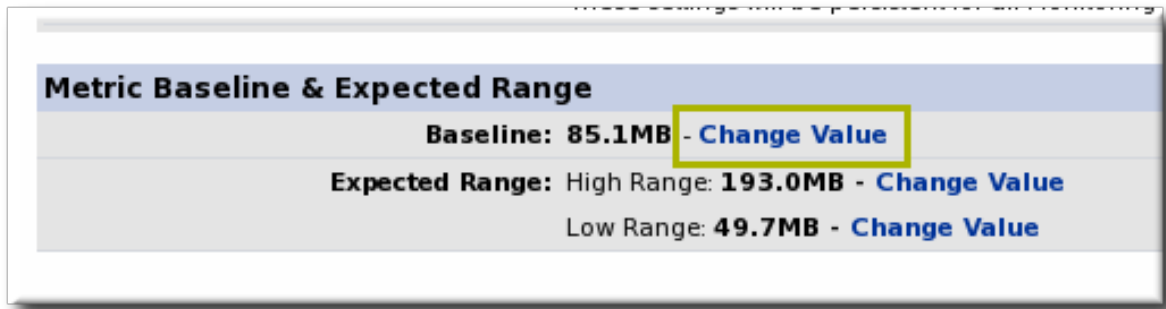
1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



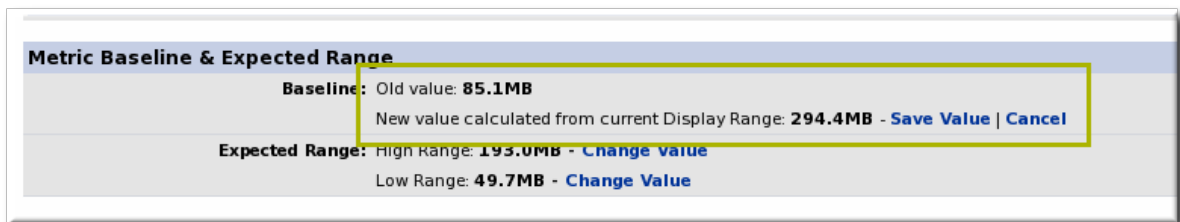
3. In the summary tab, click the name of the metric to recalculate.



4. Scroll to the bottom of the baseline chart, to the **Metric Baseline & Expected Range** area.
5. Click the **Change Value** link next to the baseline figure.



6. A new baseline is calculated using the baseline dataset property from the configuration properties, starting from the current time. Accept the new baseline value by clicking **Save Value**.



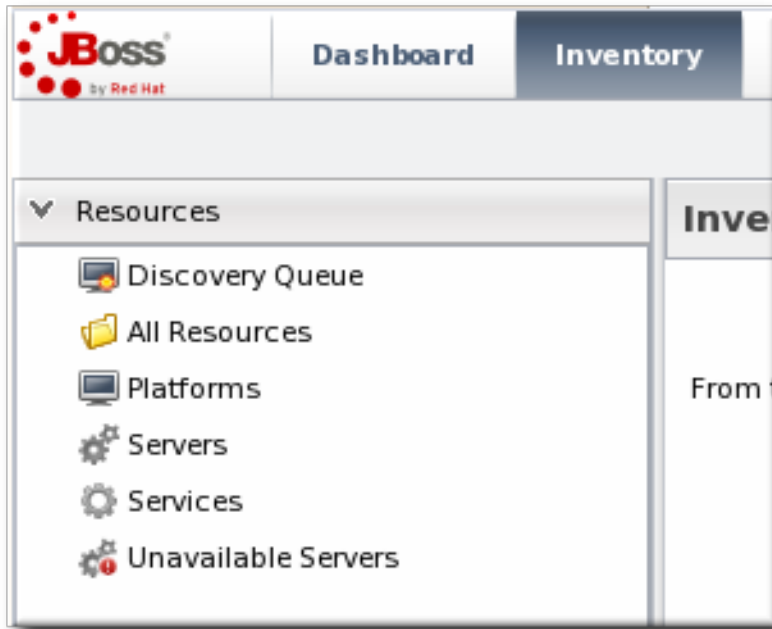
4.5. Setting Collection Intervals for a Specific Resource

Metrics are collected at the intervals specified by the collection schedule. Because not all metrics are mission critical or even likely to change, JBoss ON has different collection schedules for different metrics, with critical metrics collected more frequently.

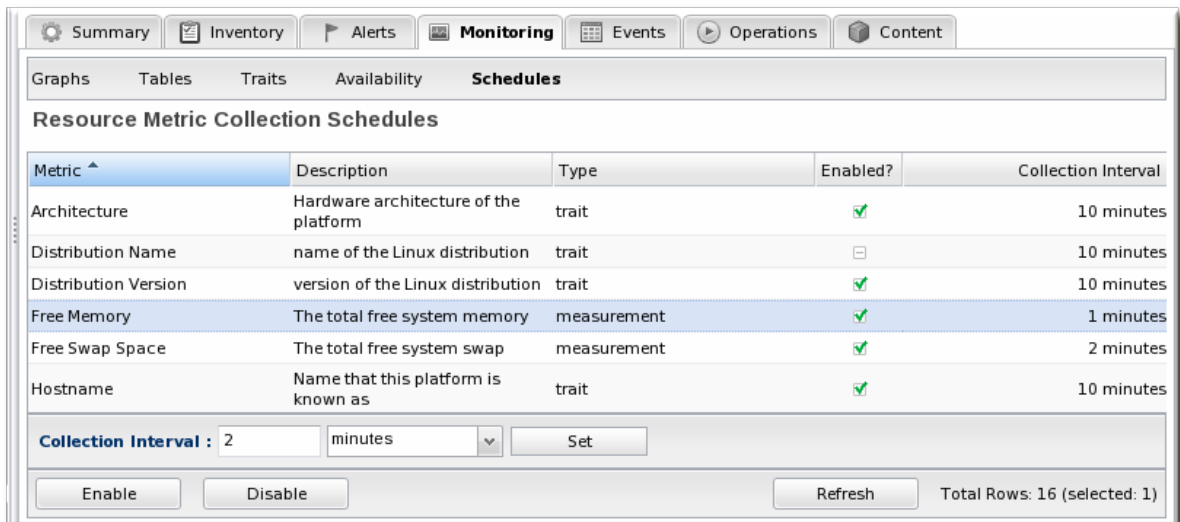
For most environments, setting a daily collection schedule (once every 24 hours) is sufficient.

To change the collection interval for a specific metric:

1. Click the **Inventory** tab in the top menu.
2. Select the resource category, such as servers or services, in the **Resources** menu table on the left. Then browse or search for the resource.



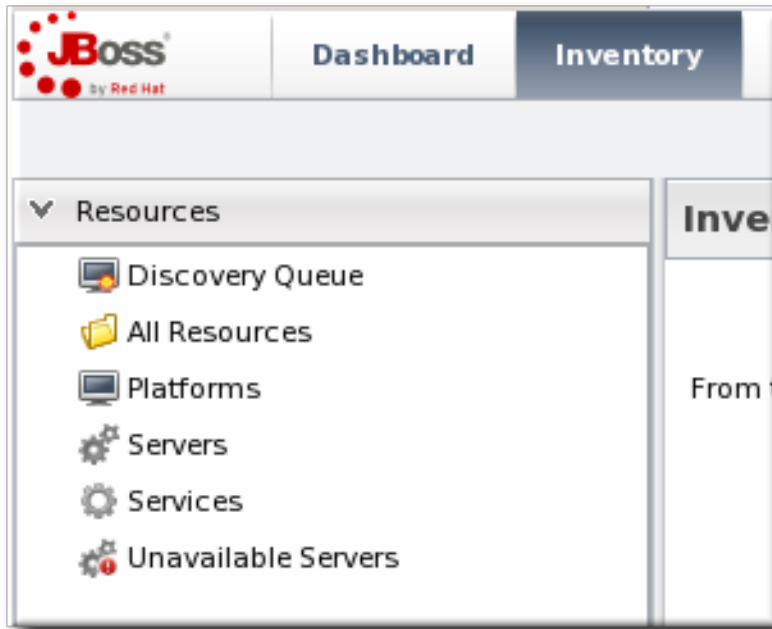
3. Click the **Monitoring** tab on the resource entry.
4. Click the **Schedules** subtab.
5. Select the metric for which to change the monitoring frequency. Multiple metrics can be selected, if they will all be changed to the same frequency.



6. Enter the desired collection period in the **Collection Interval** field, with the appropriate time unit (seconds, minutes, or hours).
7. Click **Set**.

4.6. Enabling and Disabling Metrics for a Specific Resource

1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the **Monitoring** tab on the resource entry.
4. Click the **Schedules** sub tab.
5. Select the metrics to enable or disable.

 A screenshot of the 'Linux Server 1' monitoring interface. The page title is 'Linux Server 1 LINUX'. Below the title, there are tags 'Linux' and 'workstations'. A navigation bar contains tabs for 'Summary', 'Inventory', 'Alerts', 'Monitoring', 'Events', 'Operations', and 'Content'. Under the 'Monitoring' tab, there are sub-tabs for 'Graphs', 'Tables', 'Traits', 'Availability', and 'Schedules'. The 'Schedules' sub-tab is active, showing a table titled 'Resource Metric Collection Schedules'. The table has columns for 'Metric', 'Description', 'Type', and 'Enabled?'. Below the table, there is a 'Collection Interval' field set to 'minutes' and a 'Set' button. At the bottom, there are 'Enable' and 'Disable' buttons, a 'Refresh' button, and a status indicator 'Total Rows: 16 (selected: 2)'.

Metric ^	Description	Type	Enabled?
Architecture	Hardware architecture of the platform	trait	<input checked="" type="checkbox"/>
Distribution Name	name of the Linux distribution	trait	<input type="checkbox"/>
Distribution Version	version of the Linux distribution	trait	<input checked="" type="checkbox"/>
Free Memory	The total free system memory	measurement	<input checked="" type="checkbox"/>
Free Swap Space	The total free system swap	measurement	<input checked="" type="checkbox"/>
Hostname	Name that this platform is known as	trait	<input checked="" type="checkbox"/>
Idle	Idle percentage of all CPUs	measurement	<input type="checkbox"/>
OS Name	Name that the operating system is known as	trait	<input checked="" type="checkbox"/>
OS Version	Version of the operating system	trait	<input checked="" type="checkbox"/>
System Load	Percentage of all CPUs running in system mode	measurement	<input checked="" type="checkbox"/>

6. Click the **Enable** or **Disable** button.

4.7. Changing Metrics Templates

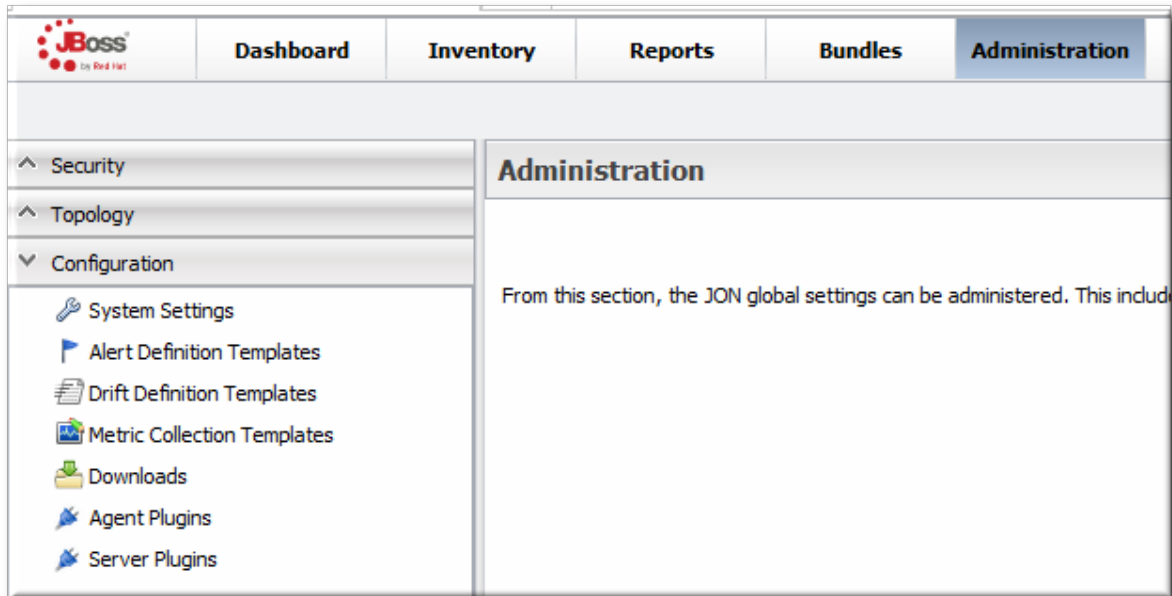
The metrics which are collected for a resource type are defined in the *monitoring template* for the resource type. Each resource type has some metrics disabled by default, and these must be manually enabled. Likewise, metrics which are enabled by default can be disabled.



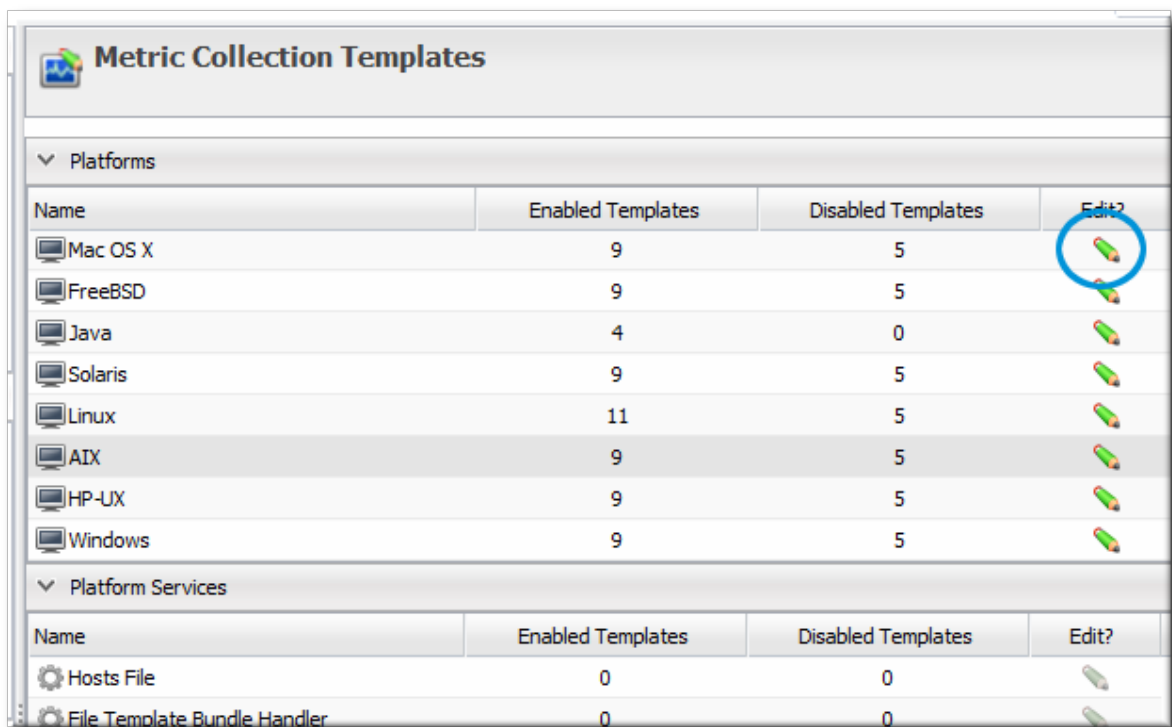
NOTE

Metric templates only apply to new resources of that resource type unless the checkbox is selected to apply them to existing resources as well as new resources.

1. In the top navigation, open the **Administration** menu, and then the **System Configuration** menu.



2. Select the **Metric Collection 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. Click the pencil icon to edit the metric collection schedule templates.

- Select the required metrics to enable or disable, and click the **Enable** or **Disable** button.
- To edit the frequency that a metric is collected, select the **Update schedules for existing resources of marked type** checkbox, and then enter the desired time frame into the **Collection Interval for Selected:** field.

Back to List

Template Metric Collection Schedules

Metric ^	Description	Type	Enabled?	Collection Interval
Cache size	Cache Size of this CPU	trait	<input checked="" type="checkbox"/>	60 minutes
CPU Model	Model of this CPU	trait	<input checked="" type="checkbox"/>	60 minutes
CPU Speed	Speed of this CPU in Mhz	trait	<input checked="" type="checkbox"/>	60 minutes
Idle	Idle percentage of this CPU	measurement	<input checked="" type="checkbox"/>	20 minutes
Idle Time	The total system CPU idle time	measurement	<input checked="" type="checkbox"/>	20 minutes
Idle Time per Minute	The total system CPU idle time	measurement	<input checked="" type="checkbox"/>	20 minutes
Nice Time	The total system CPU nice time	measurement	<input type="checkbox"/>	20 minutes
Nice Time per Minute	The total system CPU nice time	measurement	<input type="checkbox"/>	20 minutes
System Load	Percentage of this CPU running in system mode	measurement	<input checked="" type="checkbox"/>	10 minutes
System Time	The total system CPU kernel time	measurement	<input type="checkbox"/>	20 minutes
System Time per Minute	The total system CPU kernel time	measurement	<input type="checkbox"/>	20 minutes
User Load	Percentage of this CPU running in user mode	measurement	<input checked="" type="checkbox"/>	10 minutes
User Time	The total system CPU user time	measurement	<input type="checkbox"/>	20 minutes
User Time per Minute	The total system CPU user time	measurement	<input type="checkbox"/>	20 minutes

Collection Interval : 15 minutes Update Existing Schedules

Total Rows: 18 (selected: 1)

- Click the **Set** button.

4.8. Adding a PostgreSQL Query as a Metric

A SQL query can be added to a PostgreSQL database as a child resource. That entry becomes a custom metric for that PostgreSQL database.

A query metric **must** have two columns that allow the JBoss ON agent to collect data for the query:

- metricColumn
- count(id)

The query has to return a single row with those two columns. The first column signals that it is a collected metric, and the second gives the count for the metric.

For example, to track logged-in users:

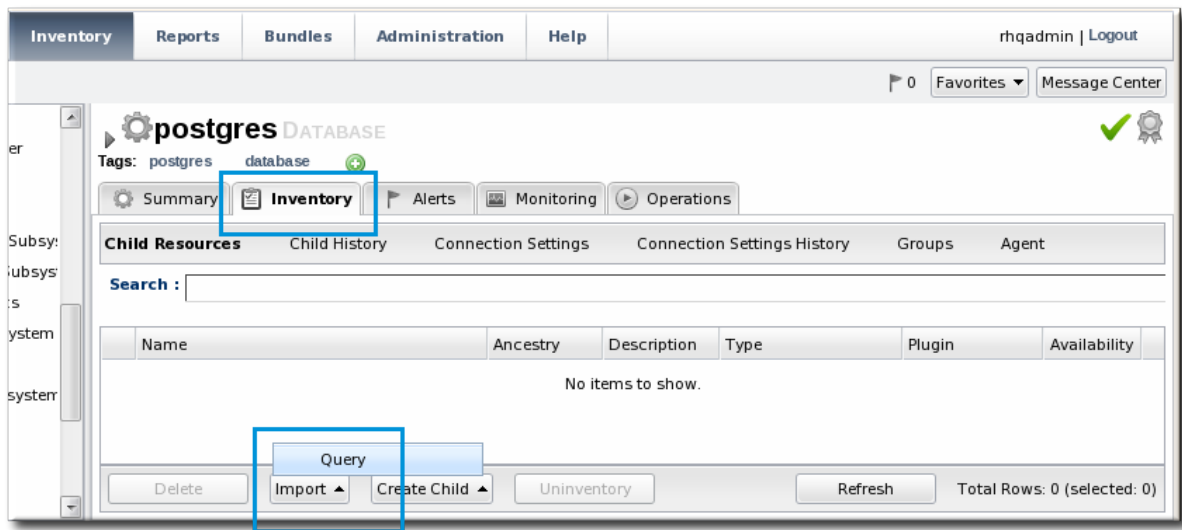
```
SELECT 'metricColumn', count(id) FROM my_application_user WHERE
is_logged_in = true
```

The **SELECT** statement defines the metric for the JBoss ON agent. The rest of the query collects the data from the database. Simple as that.

To add a metric based on a query:

- Click the **Inventory** tab in the top menu.
- Search for the PostgreSQL resource.

3. Click the **Inventory** tab for the PostgreSQL database.
4. Click the **Import** button in the bottom of the **Inventory** tab, and select **Query**.



5. Fill in the properties for the query metric. Three fields are particularly important:
 - o The **Table** gives which table within the database contains the data; this is whatever is in the **FROM** statement in the query.
 - o The **Metric Query** contains the full query to run. The **SELECT** statement must be `'metricColumn', count(id)` to format the query properly for the JBoss ON agent to interpret it as a metric.


```
SELECT 'metricColumn', count(id) FROM my_application_user WHERE is_logged_in = true
```
 - o The **Name** field is not important in configuring the metric, but it is important identifying the metric later.

Resource Import Wizard

Import Resource of Type [Query] Step 1 of 1

Deployment Options

Property	Unset?	Value	Description
Table		my_application_user	The table to discover
Name		Total Logged In Users	
Description		Number of users currently logged in	
Metric Query		SELECT 'metricColumn', count(id) FROM	The query that will gather metric data

Timeout : seconds

Once the query is created, then the agent begins collecting the counts for the data.

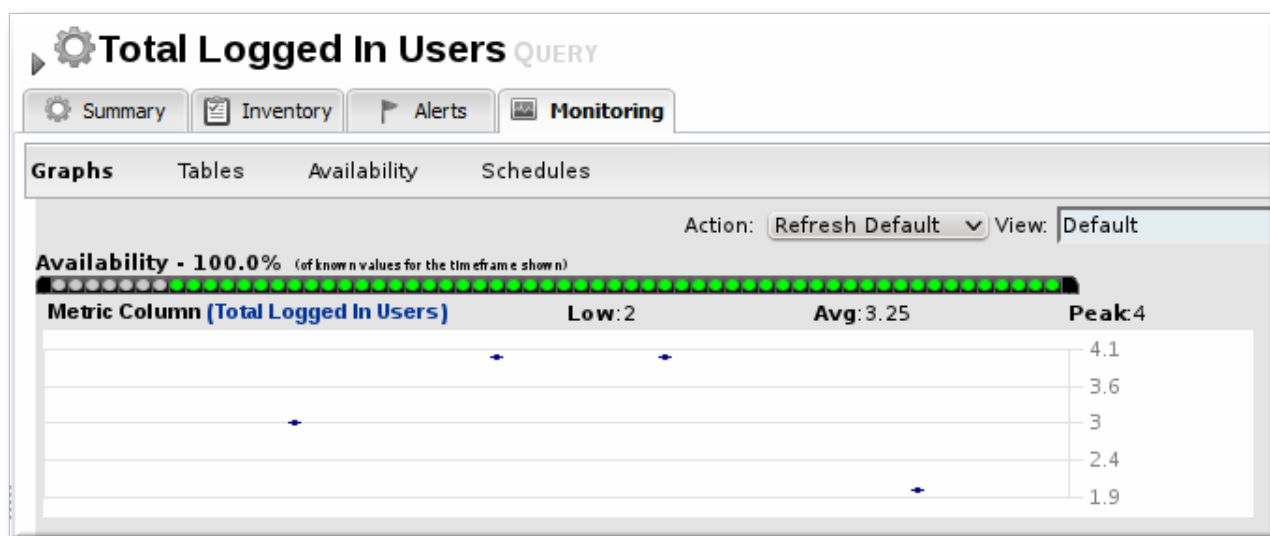


Figure 12. Query: Total Logged-in User Count

5. EVENTS

Metric data are collected according to a schedule. However, some actions occur on a resource sporadically, such as sudden system shutdowns. These are *events*. Since event data can be generated randomly, events are sent to agents immediately when they are detected.

5.1. Events, Logs, and Resources

Operating and some server types keep their own log files and register a steady stream of information about incidents for that resource, from simple debug information to critical errors. Metrics are collected on a (more or less) set schedule. Events are entirely random, because they are based on real actions as they occur. Events, then, give a different perspective on resource performance.

Events monitoring tracks those streaming log file messages. In a sense, JBoss ON event monitoring is a filtered log viewer. When events are enabled, all of the log messages flow through JBoss ON's event viewer. However, the types of messages that JBoss ON records and displays can be limited in the event configuration so that only certain types of messages or messages matching certain strings are included in the events view.

Three resource types record and display events:

- Windows (Windows event logs)
- Apache server (log files)
- JBoss AS server (log files)



NOTE

Events must be enabled before they become active.

Default events are taken from standard log files. Custom resource types can identify event sources in logs or in asynchronous messaging systems such as a JMX notification or a JMS messaging system.

5.2. Event Date Formatting

JBoss ON expects log messages to follow the log4j log pattern, over all.

```
date severity [class] message
```

The *date* format is configurable when event monitoring is enabled. For custom logs, a custom pattern can be added. If no date format is given, then the three standard formats used by log4j are tried.

```
YYYY-mm-dd HH:mm:ss,SSS
HH:mm:ss,SSS
dd MM yyyy HH:mm:ss,SSS
```

The severity must be next. It can be plain, surrounded by brackets, or surrounded by parentheses.

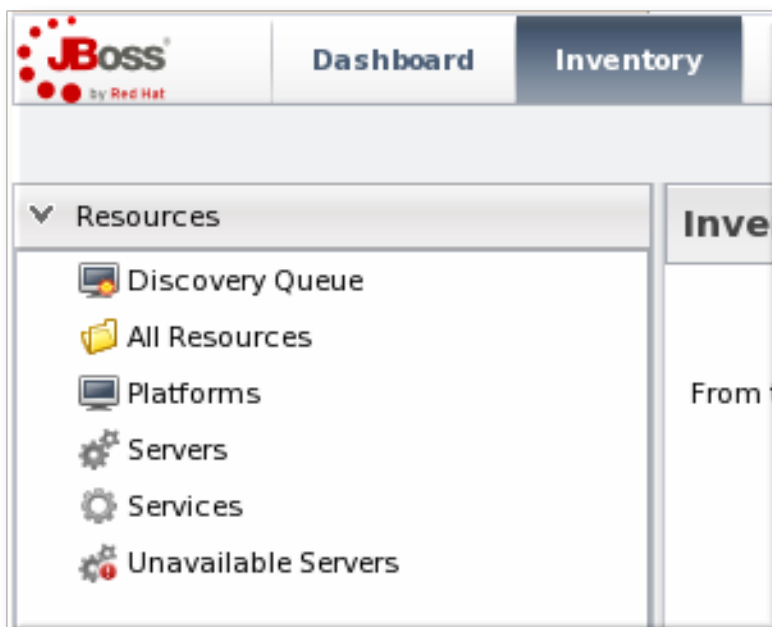
```
date SEVERITY [org.foo.bar] my message
date [SEVERITY] [org.foo.bar] my message
date ( SEVERITY ) [org.foo.bar] my message
```

After the severity, there are no real constraints on the format of the log entry. If classes or other identifiers are passed, they are properly displayed.

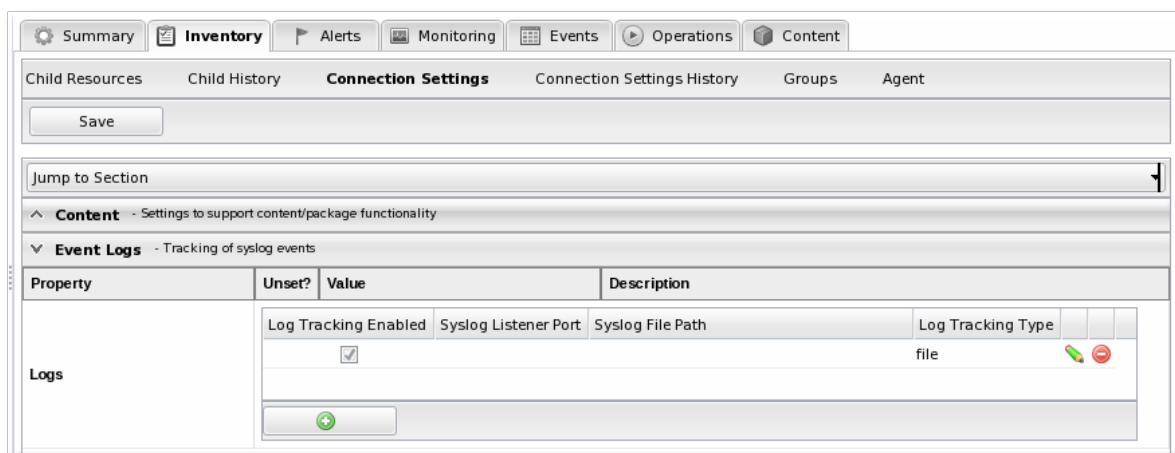
5.3. Defining a New Event

Events are only recognized by the monitoring service if events logging is properly enabled for the specific service being logged. This requires creating a log event for the log or system service, specifying a log path on the resource, and setting a date format which matches the format for the log.

1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the **Inventory** tab on the resource entry.
4. Select the **Connection Settings** subtab.
5. Click green plus icon under the **Events Log** section.



6. Set the path to the log file, enable the event entry, and set the date format. Other properties about the log file, such as whether it is a file or listener and a message parser, can also be configured.

Edit Row

Property	Unset?	Value	Description
Message Parser Regular Expression	<input checked="" type="checkbox"/>	<input type="text"/>	A regular expression used to parse the different pieces of the log messages. This regex must have 3 capturing groups - #1 is the timestamp, #2 is the severity and #3 is the message details text. If not defined, a best guess will be made to parse the log messages.
Log Tracking Enabled		<input type="radio"/> Yes <input type="radio"/> No	Enables the collection of syslog events
Includes Pattern	<input checked="" type="checkbox"/>	<input type="text"/>	A regular expression that is matched against a log message's detail text to determine if an Event should be fired for that log message. If not specified, no filtering of log messages will be done based on their detail.
Syslog Listener Port	<input checked="" type="checkbox"/>	<input type="text"/>	Port that the syslog listener will run on if using listener style. If not defined, the default will be 514.
Syslog File Path	<input checked="" type="checkbox"/>	<input type="text"/>	File path to watch for log events if using file style. If not defined, the default will be /var/log/messages
Syslog Listener Bind Address	<input checked="" type="checkbox"/>	<input type="text"/>	Address that the syslog listener will bind to if using listener style
Minimum Severity	<input checked="" type="checkbox"/>	<input type="radio"/> Information <input type="radio"/> Warning <input type="radio"/> Error	The minimum severity of log messages that will be collected. If not specified, there is no minimum severity (i.e. all will be collected).
Log Tracking Type		<input type="radio"/> file <input type="radio"/> listener	Defines if the log messages are to be found by polling a file or listening to a TCP socket
Date/Time Format	<input checked="" type="checkbox"/>	<input type="text"/>	A regular expression that indicates how the message's date/timestamp is formatted. See the Javadoc on java.text.SimpleDateFormat for the syntax of this regex. If not defined, the date will not be parsed and the time the message was seen will be used.

OK Cancel

5.4. Viewing Events

1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.
3. Click the **Events** tab on the resource entry. Events can be filtered by severity (debug, info, warn, error, and fatal).
4. Click the specific event for further details.

server.example.com LINUX

Summary Inventory Alerts Monitoring **Events** Operations Drift Content

History

Source Filter: Details Filter:

Severity Filter: Debug Info Warn Error Fatal

Timestamp	Severity	Details	Source Location
Mar 14, 2012 9:49:58 AM	Info	Mar 14 10:49:58 server01 dhclient[1712]: DHCPREQUEST on eth2 to 255.255.255 port 67 (xid=0x3b...	/var/log/messages
Mar 14, 2012 9:49:39 AM	Info	Mar 14 10:49:39 server01 dhclient[1712]: DHCPREQUEST on eth2 to 255.255.255 port 67 (xid=0x3b...	/var/log/messages
Mar 14, 2012 9:49:19 AM	Info	Mar 14 10:49:58 server01 dhclient[1712]: DHCPREQUEST on eth2 to 255.255.255 port 67 (xid=0x3b...	/var/log/messages
Mar 14, 2012 9:49:11 AM	Info	Mar 14 10:49:39 server01 dhclient[1712]: DHCPREQUEST on eth2 to 255.255.255 port 67 (xid=0x3b...	/var/log/messages
Mar 14, 2012 9:49:03 AM	Info	Mar 14 10:49:58 server01 dhclient[1712]: DHCPREQUEST on eth2 to 255.255.255 port 67 (xid=0x3b...	/var/log/messages
Mar 14, 2012 9:48:52 AM	Info	Mar 14 10:49:39 server01 dhclient[1712]: DHCPREQUEST on eth2 to 255.255.255 port 67 (xid=0x3b...	/var/log/messages
Mar 14, 2012 9:48:41 AM	Info	Mar 14 10:49:58 server01 dhclient[1712]: DHCPREQUEST on eth2 to 255.255.255 port 67 (xid=0x3b...	/var/log/messages
Mar 14, 2012 9:48:33 AM	Info	Mar 14 10:49:39 server01 dhclient[1712]: DHCPREQUEST on eth2 to 255.255.255 port 67 (xid=0x3b...	/var/log/messages

5.5. Detailed Discussion: Event Correlation

There are a lot of moving pieces in IT infrastructure, and one change can cause a cascade of results. One factor in being able to respond to events and manage infrastructure is the ability to correlate an event with its cause.

While there is not an exact way to pinpoint what change caused what event or alert, there is a way to help visualize how unrelated JBoss ON elements — alerts, configuration changes, event logs, drift detection, or inventory changes — might be related. Each resource has a **Timeline** area on its **Summary** tab. This collects all of the major operations that have occurred on that resource, either through JBoss ON or as detected by JBoss ON.

The thing to look for is clusters. For example, in this case, there was a configuration change on the JBoss server, and shortly after there was a critical alert and a simultaneous error in the JBoss server's logs. That is a reasonable indication that the configuration change caused a performance problem.

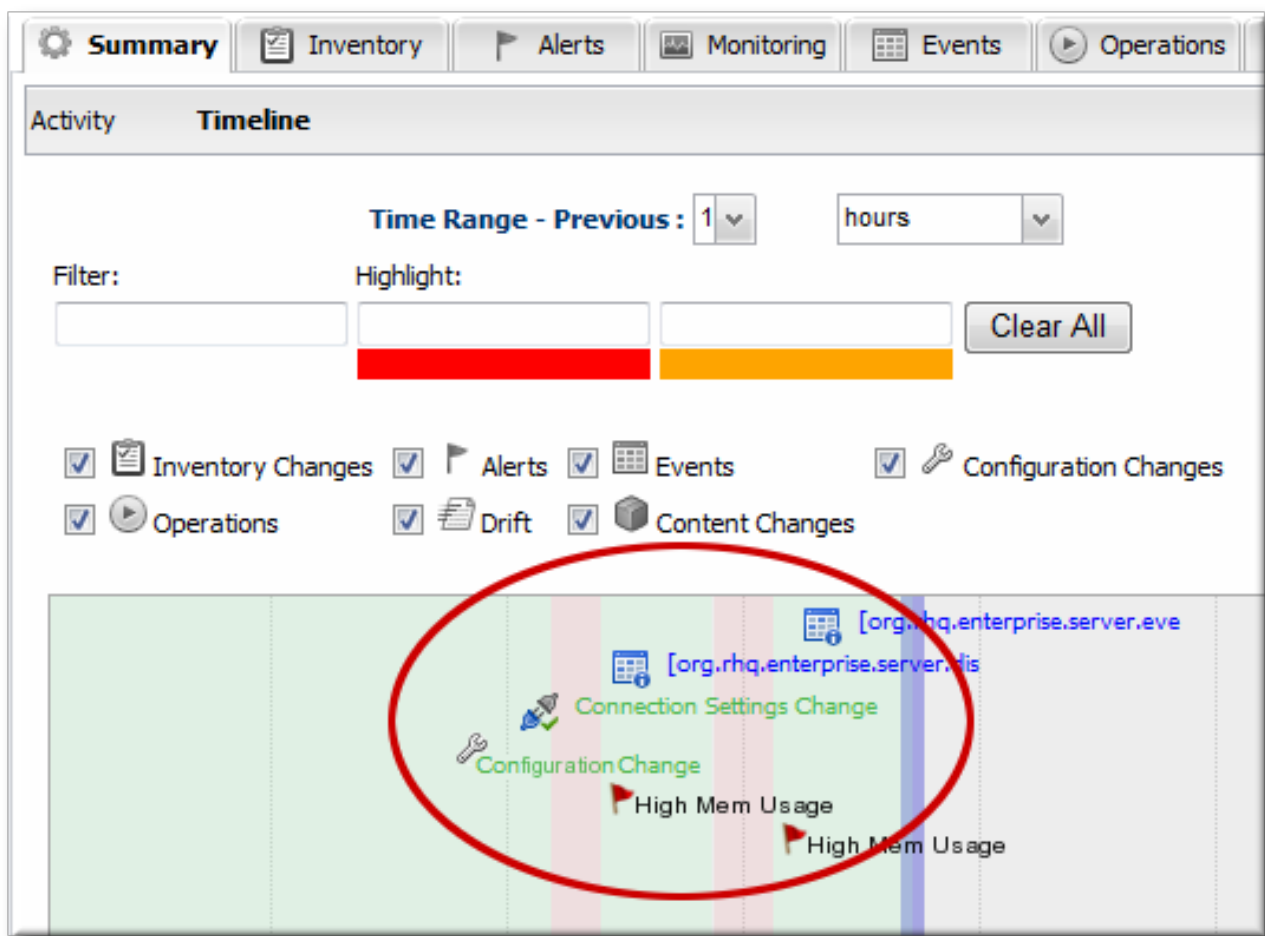


Figure 13. Resource Timeline Cluster

6. URL RESPONSE TIME MONITORING

Part of web application performance is determined by how responsive it is to requests. JBoss Operations Network supplies an extra monitoring setting called *response time filters* which measures the amount of time it takes for a URL to respond to a request.

6.1. Call-Time (or Response Time) Monitoring for URLs

Performance for web applications is more complex than simple availability. How quickly an application can respond to requests is as important as whether the application is running.

The time that it takes an application to respond to some kind of request is *call-time* or *response time* data.

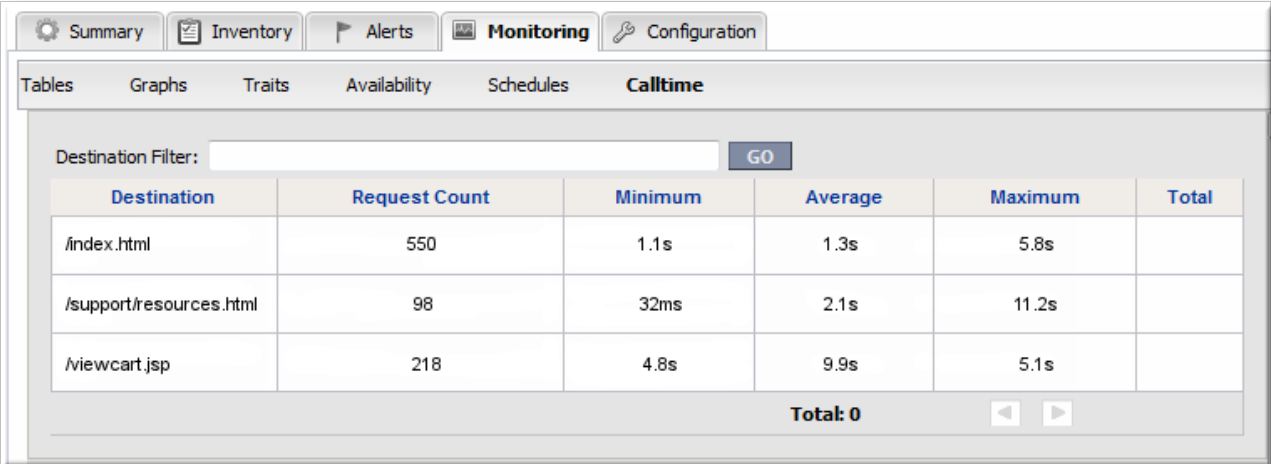
Two types of resources support call-time data by default:

- Session beans, for EJB method calls.
- Web servers (standalone or embedded in an application server), for URL responses. Web servers require an additional response time filter with configuration on what URL resources to measure for response times.

Response time monitoring is more based upon performance than it is on capturing a single data point. Response or call-time data are displayed as aggregates, showing maximum, minimum, and average response times per URL or per method.

6.2. Viewing Call Time Metrics

Both session bean resources and web server resources have an additional **Monitoring** subtab called **Calltime**. All of the call-time data or response time data ranges (minimum, maximum, and averages) are displayed for each URL resource or method. As new URLs or methods are accessed, they are dynamically added to the results table.



The screenshot shows a web application monitoring interface. At the top, there are tabs for Summary, Inventory, Alerts, Monitoring (selected), and Configuration. Below these, there are sub-tabs for Tables, Graphs, Traits, Availability, Schedules, and Calltime (selected). A 'Destination Filter' input field with a 'GO' button is present. The main content is a table with the following data:

Destination	Request Count	Minimum	Average	Maximum	Total
/index.html	550	1.1s	1.3s	5.8s	
/support/resources.html	98	32ms	2.1s	11.2s	
/viewcart.jsp	218	4.8s	9.9s	5.1s	
Total: 0					

Figure 14. URL Metrics for a Web Server

6.3. Extended Example: Website Performance

The Setup

A significant amount of Example Co.'s business, services, and support is tied to its website. Customers have to be able to access the site to purchase products, schedule training or consulting, and to receive most support and help. If the site is slow or if some resources are inaccessible, customers immediately have a negative experience.

The goal is not to monitor whether the web server is running, but whether the web application are responsive and performing as Example Co.'s customers expect.

What to Do

Tim the IT Guy identifies three different ways that he can capture web application performance information:

- Response times for individual URLs

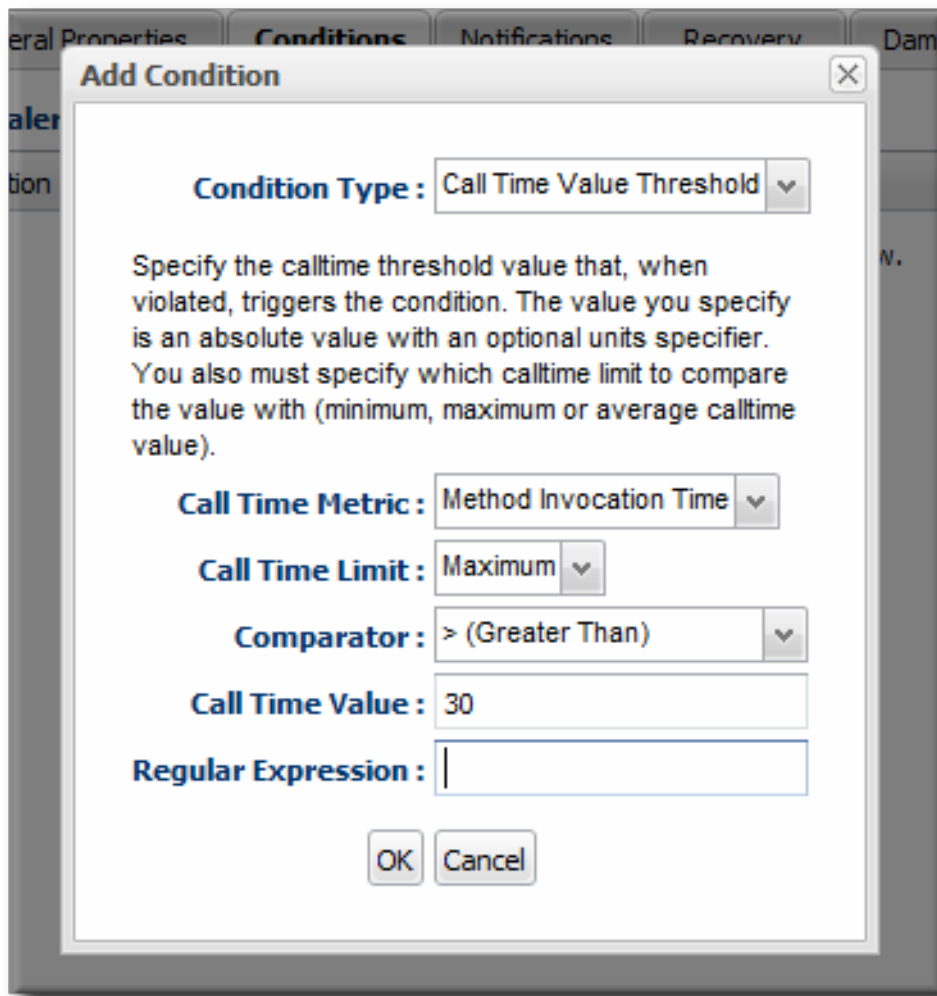
- Throughput information like total number of requests and responses
- Counts for critical HTTP response codes

Both monitoring and alerting can be configured based solely off response time and throughput metrics. However, bad website performance is indicative of an underlying problem with the web server or its associated database. Therefore, Tim not only wants to be informed when website performance is poor; he wants to correlate some performance metrics with underlying server and database performance and launch operations that can mitigate poor responsiveness.

Tim maps a few common scenarios which cause poor website or web server performance and plans simple, immediate operations that JBoss ON can perform until an IT staffer can analyze the problem. Tim attempts to narrow down potential causes to a performance. Alerts can be issued for a single condition or for a combination of conditions. In Tim's case, he creates a three different alerts based on different combinations of underlying causes for performance problems ([Section 11.2, "Basic Procedure for Setting Alerts for a Resource"](#)).

- If there are poor response times and a high number of HTTP error 500 responses, then the alert can be configured with an operation to restart the web server ([Section 13.2, "Detailed Discussion: Initiating an Operation"](#)).
- If there are poor response times and a high number of HTTP error 404 response (meaning that resources may not be delivered properly), then the alert is configured to restart the database.
- If there are poor response times and a high number of total requests per minute, then it may mean that there is simply too much load on the server. The alert can be configured to create another web server instance to help with load balancing; using a JBoss ON CLI script allows the JBoss ON server to create new resources as necessary and deploy bundles of the appropriate web apps ([Section 13.3, "Detailed Discussion: Initiating Resource Scripts"](#)).

The most critical factor is the response time, which is a factor in every alert. Each alert has one condition based on the call time data, specifically of the call-time data moves past a certain threshold.



Tim picks a reasonable threshold, about 15 seconds, for performance. If performance degrades so that the HTTP Response Time metric returns a value higher than 20 seconds to load pages, JBoss ON issues an alert.

Alternatively, he could alert on simple call-time changes. Call-time changes will trigger an alert for any change from the established baseline, meaning a new minimum, maximum, or average value. A change of any kind can alert in either a decrease in performance *or* an increase in performance. A threshold alert only alerts on a specific change.

Tim then adds the other condition, with an AND operator, to each alert he configures.

Also, most web app-related metrics are not enabled by default. Tim enables the Total Number of Requests per Minute, Total Number of Responses per Minute, Number of 404 Responses per Minute, and Number of 500 Responses per Minute metrics for each web server ([Section 4.7, “Changing Metrics Templates”](#)).

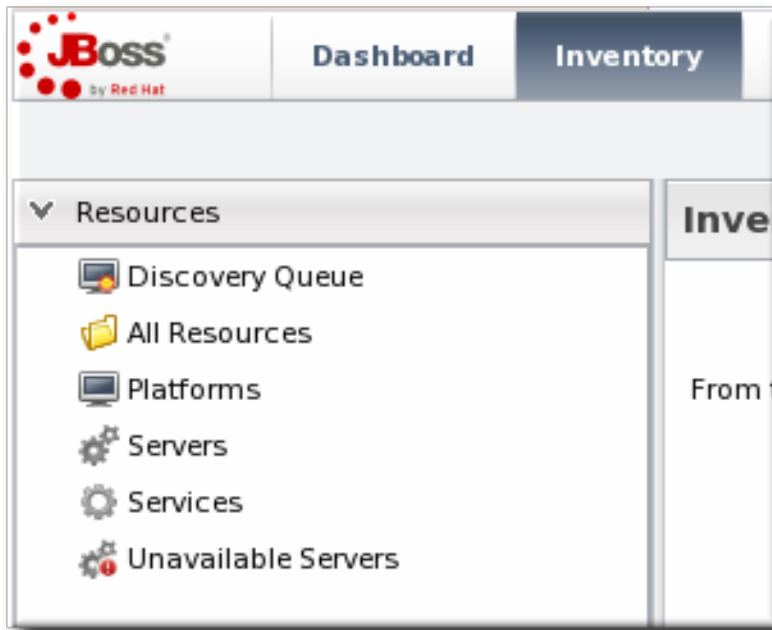
For every alert, Tim also configures an email notification along with the other responses, so that a member of the IT staff can evaluate any website performance problems and take additional actions if necessary.

6.4. Configuring EJB Call-Time Metrics

EJB method call-time measurements are not collected by default.

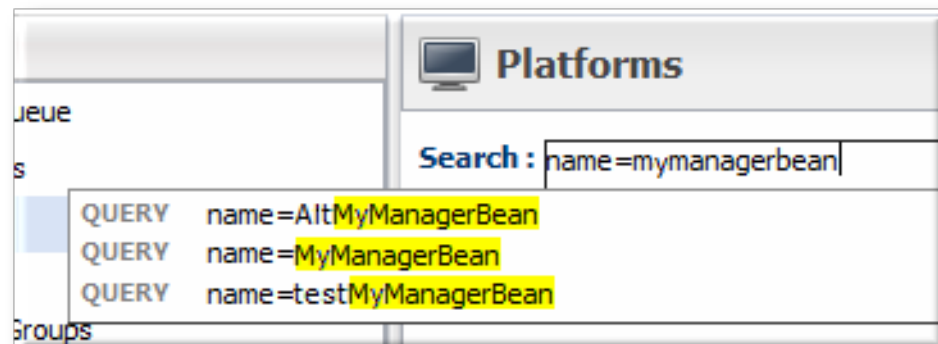
1. Click the **Inventory** tab in the top menu.

2. Select the **Services** menu table on the left, and then navigate to the EJB resource.



NOTE

It is probably easier to search for the session bean by name, if you know it.



3. Click the **Monitoring** tab on the EJB resource entry.
4. Click the **Schedules** subtab.
5. Select the **Method Invocation Time** metric. This metric is the *calltime* type.

Metric ^	Description	Type	Enabled?	Collection Interval ^
Available Count	The number of pooled instances of this EJB3 Session Bean in the method-ready state	measurement	<input type="checkbox"/>	40 minutes
Create Count	The number of instances of this EJB3 Session Bean that have been created since server start	measurement	<input type="checkbox"/>	20 minutes
Create Count per Minute	The number of instances of this EJB3 Session Bean that have been created since server start	measurement	<input checked="" type="checkbox"/>	20 minutes
Current Count	The total number of currently pooled instances of this EJB3 Session Bean	measurement	<input type="checkbox"/>	40 minutes
Max Size	The maximum number of instances that are allowed to be pooled	measurement	<input type="checkbox"/>	40 minutes
Method Invocation Time	The minimum, maximum, and average invocation times for each of the methods exposed by this EJB; NOTE: this metric is only available if JBoss EJB3 RC9 Patch 1 or later is being used (a capable version of EJB3 is included with JBossAS 4.2.0.GA or later)	calltime	<input checked="" type="checkbox"/>	10 minutes
Remove Count	The number of instances of this EJB3 Session Bean that have been	measurement	<input type="checkbox"/>	20 minutes

6. Click the **Enable** at the bottom of the list.

6.5. Configuring Response Time Metrics for JBoss EAP 6/AS 7

To collect response time metrics for a JBoss EAP 6/AS 7 server, first install the servlet JAR for the response time filter and configure the web server to use it. Then, enable the metrics collection for the web resource.

6.5.1. Installing the Response Time Filters

1. Make sure that you have created a management user to access the JBoss EAP 6 instance.

For more information, see the [JBoss AS 7.1 documentation](#).

2. Download the response time packages for JBoss from the JBoss ON UI. The response time filters are packaged as AS 7 modules. There are two modules to obtain:

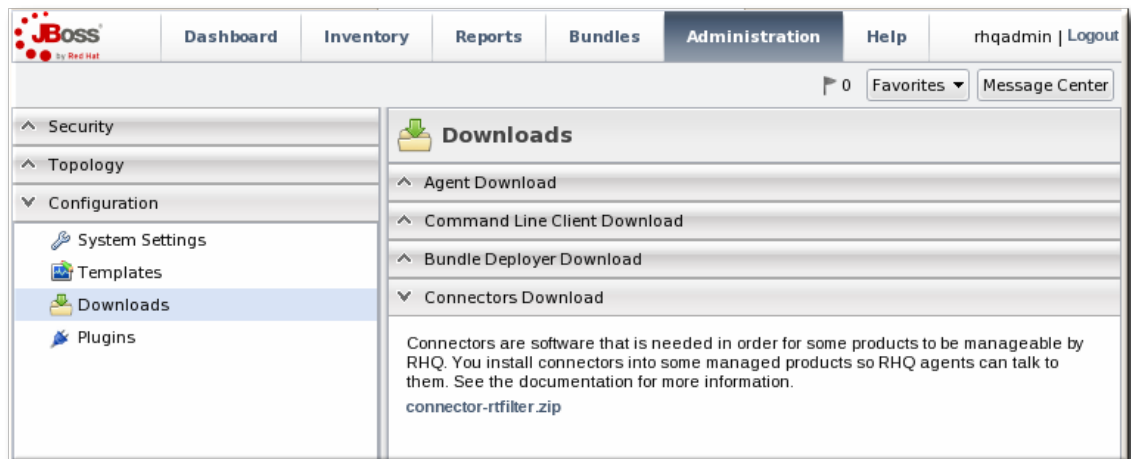
```
rhq-rtfilter-module.zip
rhq-rtfilter-subsystem-module.zip
```

NOTE

This can also be done from the command line using **wget**:

```
[root@server ~]# wget
http://server.example.com:7080/downloads/connectors/rhq-
rtfilter-module.zip
[root@server ~]# wget
http://server.example.com:7080/downloads/connectors/rhq-
rtfilter-subsystem-module.zip
```

1. Click the **Administration** tab in the top menu.
2. In the **Configuration** menu box on the left, select the **Downloads** item.



3. Click the **rhq-rtfilter-module.zip** and **rhq-rtfilter-subsystem-module.zip** links, and save the files to an accessible directory, like the **/tmp** directory.
3. Open the **modules/** directory for the JBoss EAP 6 instance. For example:

```
[root@server ~]# cd /opt/jboss-eap-6.0/modules/
```

4. Unzip the **rhq-rtfilter-module.zip** archive to install the response time filter JAR and the associated **module.xml** file.

```
[root@server modules]# unzip /tmp/rhq-rtfilter-module.zip
```

5. Open the configuration file for the server, **domain.xml** or **standalone.xml**.
6. Deploy the response time module globally by adding the module to the list of global modules in the **<subsystem>** element.

```
<subsystem xmlns="urn:jboss:domain:ee:1.0">
  <global-modules>
    <module name="org.rhq.helpers.rhq-rtfilter" slot="main"/>
  </global-modules>
</subsystem>
```

7. Save the file.
8. Unzip the **rhq-rtfilter-subsystem-module.zip** archive to install the subsystem response time filter JAR and the associated **module.xml** file.

```
[root@server modules]# unzip /tmp/rhq-rtfilter-subsystem-module.zip
```

This installs the filters as a subsystem for the application server or individual web apps.

9. After the filters have been installed, the JBoss EAP 6 server needs to be configured to use them.

The response time filter can be deployed globally, for all web applications hosted by the EAP/AS instance, or it can be configured for a specific web application.

To deploy the filter as a global subsystem:

1. Open the configuration file for the server, `domain.xml` or `standalone.xml`.
2. Add the an `<extensions>` element for the response time filter.

```
<extension module="org.rhq.helpers.rhq-rtfilter-subsystem"/>
```

3. Add a `<subsystem>` element beneath the `<profile` element.

All that is required for response time filtering to work is the default `<subsystem>` element, without any optional parameters. However, the parameters can be uncommented and set as necessary; the different ones are described in [Table 3, "Parameters Available for User-Defined <filter> Settings"](#).

The `<subsystem>` element should be added even if none of the optional parameters are set.

```
<subsystem xmlns="urn:rhq:rtfilter:1.0">
  <!-- Optional parameters.

  <init-param>
    <param-name>chopQueryString</param-name>
    <param-value>>true</param-value>
  </init-param>
  <init-param>
    <param-name>logDirectory</param-name>
    <param-value>/tmp</param-value>
  </init-param>
  <init-param>
    <param-name>logFilePrefix</param-name>
    <param-value>localhost_7080_</param-value>
  </init-param>
  <init-param>
    <param-name>dontLogRegEx</param-name>
    <param-value></param-value>
  </init-param>
  <init-param>
    <param-name>matchOnUriOnly</param-name>
    <param-value>>true</param-value>
  </init-param>
  <init-param>
    <param-name>timeBetweenFlushesInSec</param-name>
    <param-value>73</param-value>
  </init-param>
  <init-param>
    <param-name>flushAfterLines</param-name>
    <param-value>13</param-value>
  </init-param>
  <init-param>
    <param-name>maxLogFileSize</param-name>
    <param-value>5242880</param-value>
```

```

        </init-param>
    -->
</subsystem>

```

To configure the response time filters for an individual web application:

1. Open the web application's `web.xml` file.

```
[root@server ~]# vim WARHomeDir/WEB-INF/web.xml
```

2. Add the filter and, depending on the configuration, filter mapping elements to the file. This activates the response time filtering.

All that is required for response time filtering to work is the default `<filter>` element, without any optional parameters. However, the parameters can be uncommented and set as necessary; the different ones are described in [Table 3, "Parameters Available for User-Defined <filter> Settings"](#).

```

<filter>
    <filter-name>RhqRtFilter</filter-name>
    <filter-
class>org.rhq.helpers.rtfiler.filter.RtFilter</filter-class>

<!-- Optional parameters.

    <init-param>
        <param-name>chopQueryString</param-name>
        <param-value>>true</param-value>
    </init-param>
    <init-param>
        <param-name>logDirectory</param-name>
        <param-value>/tmp</param-value>
    </init-param>
    <init-param>
        <param-name>logFilePrefix</param-name>
        <param-value>localhost_7080_</param-value>
    </init-param>
    <init-param>
        <param-name>dontLogRegEx</param-name>
        <param-value></param-value>
    </init-param>
    <init-param>
        <param-name>matchOnUriOnly</param-name>
        <param-value>>true</param-value>
    </init-param>
    <init-param>
        <param-name>timeBetweenFlushesInSec</param-name>
        <param-value>73</param-value>
    </init-param>
    <init-param>
        <param-name>flushAfterLines</param-name>
        <param-value>13</param-value>
    </init-param>
    <init-param>
        <param-name>maxLogFileSize</param-name>

```

```

        <param-value>5242880</param-value>
    </init-param>
-->

</filter>

<!-- Use this only when also enabling the RhqRtFilter in the
filter
<filter-mapping>
    <filter-name>RhqRtFilter</filter-name>
    <url-pattern>*</url-pattern>
</filter-mapping>
-->

```

10. Restart the JBoss EAP/AS server to load the new `web.xml` settings.

Table 3. Parameters Available for User-Defined <filter> Settings

Parameter	Description
chopQueryString	Only the URI part of a query will be logged if this parameter is set to true. Otherwise the whole query line will be logged. Default is true.
logDirectory	The directory where the log files will be written to. Default setting is <code>{jboss.server.log.dir}/rt/</code> (usually <code>server/xxx/log/rt</code>). If this property is not defined, the fallback is <code>{java.io.tmpdir}/rt/</code> (<code>/tmp/</code> on UNIX®, and <code>~/Application Data/Local Settings/Temp</code> – check the TEMP environment variable) is used. If you specify this init parameter, no directory <code>rt/</code> will be created, but the directory you have provided will be taken literally.
logFilePrefix	A prefix that is put in front of the log file names. Default is the empty string.
dontLogRegEx	A regular expression that is applied to query strings. See <code>java.util.regex.Pattern</code> . If the parameter is not given or an empty string, no pattern is applied.
matchOnUriOnly	Should the <code>dontLogRegEx</code> be applied to the URI part of the query (true) or to the whole query string (false). Default is true.
timeBetweenFlushesInSec	Log lines are buffered by default. When the given number of seconds have passed and a new request is received, the buffered lines will be flushed to disk even if the number of lines to flush after (see next point) is not yet reached.. Default value is 60 seconds (1 Minute).

Parameter	Description
flushAfterLines	Log lines are buffered by default. When the given number of lines have been buffered, they are flushed to disk. Default value is 10 lines.
maxLogFileSize	The maximum allowed size, in bytes, of the log files; if a log file exceeds this limit, the filter will truncate it; the default value is 5242880 (5 MB).
vHostMappingFile	<p>This properties file must exist on the Tomcat process classpath. For example, in the <code>./conf/vhost-mappings.properties</code>. The file contains mappings from the 'incoming' vhost (server name) to the vhost that should be used as the prefix in the response time log file name. If no mapping is present (no file or no entry response times are set), then the incoming vhost (server name) is used. For example:</p> <pre data-bbox="817 797 1406 927">pickeldi.users.acme.com=pickeldi pickeldi= %HOST%=</pre> <p>The first mapping states that if the incoming vhost is 'host1.users.acme.com', then the log file name should get a vhost of 'host1' as prefix, separated by a <code>_</code> from the context root portion. The second mapping states that if the 'incoming' vhost is 'host1', then no prefix, and no <code>_</code>, should be used. The third mapping uses a special left-hand-side token, '%HOST%'. This mapping states that if the 'incoming' vhost is a representation of localhost then no prefix, and no <code>_</code>, should be used.</p> <p>%HOST% will match the host name, or canonical host name or IP address, as returned by the implementation of <code>InetAddress.getLocalHost()</code>.</p> <p>The second and third mappings are examples of empty right hand side, but could just as well have provided a vhost.</p> <p>This is a one time replacement. There is no recursion in the form that the result of the first line would then be applied to the second one.</p>

6.5.2. Enabling the Call-Time Metric

Response time metrics are configured on a live application deployment. A deployment resource is a child of either a standalone EAP 6 server or of a server group.

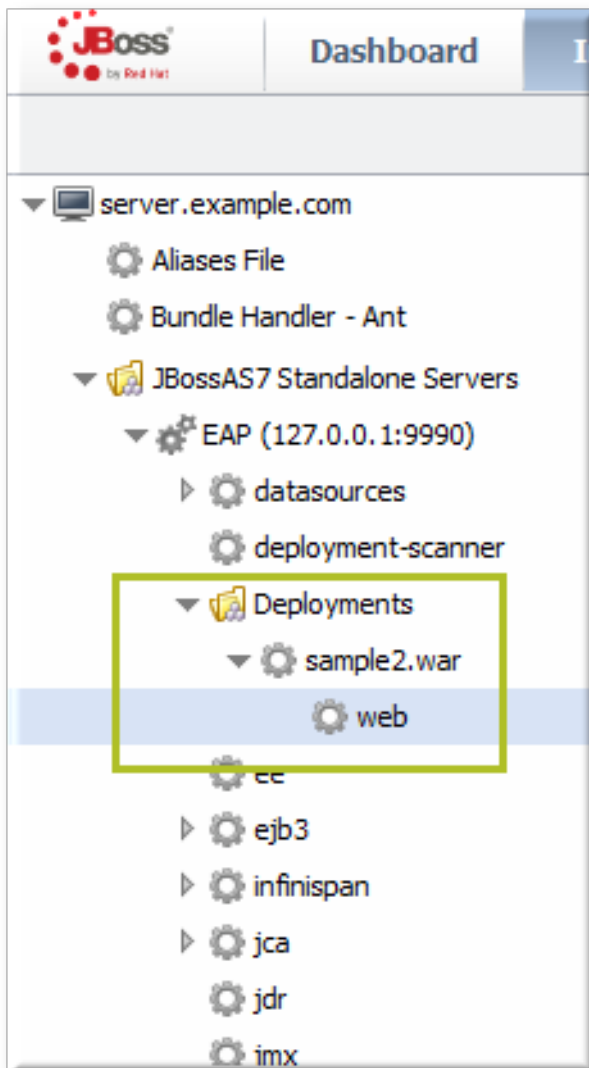


Figure 15. Web Runtime Resource

1. Click the **Inventory** tab in the top menu.
2. Click the **Servers - Top Level Imports** item, and select the JBoss EAP 6 resource.
3. Navigate to the deployment resource, and expand the application to the web subsystem.
4. Click the **Monitoring** tab on the web resource entry.
5. Click the **Schedules** subtab.
6. Select the **Response Time** metric. This metric is the *calltime* type.

Metric	Description	Type	Enabled?	Collection Interval
Availability	The number of seconds between availability checks. The agent honors this setting as best as possible but the actual period can be longer based on agent activity.	availability	<input checked="" type="checkbox"/>	10 minutes
Context Root	the context root of this webapp	trait	<input type="checkbox"/>	24 hours
Response Time	the minimum, maximum, and average response times for requests serviced by this webapp	calltime	<input checked="" type="checkbox"/>	10 minutes
Virtual Host	the virtual host this webapp is deployed to	trait	<input type="checkbox"/>	24 hours

Collection Interval: minutes

Total Rows: 4 (selected: 1)

7. Click the **Enable** at the bottom of the list.
8. Click the **Inventory** tab on the web entry.
9. Select the **Connection Settings** subtab.
10. Unset the check boxes for the response time configuration and fill in the appropriate values for the web application.

Connection Settings

Save

Jump to Section

General Properties

Property	Unset?	Value	Description
Path		deployment=sample2.war, subsystem=web	
Response Time			
Property	Unset?	Value	Description
Response Time Log File	<input checked="" type="checkbox"/>	<input type="text"/>	the full path to the log file containing response-time stats for this webapp
Response Time Uri Excludes	<input checked="" type="checkbox"/>	<input type="text"/>	a space-delimited list of regular expressions specifying URLs that should be excluded from response-time stats collection
Response Time Uri Transforms	<input checked="" type="checkbox"/>	<input type="text"/>	a space-delimited list of Perl-style substitution expressions that should be applied to all URLs for which response-time stats are collected (e.g. /?/dept/finance/?/dept/finance/?)

- o The response times log which is used by that specific web application. **The log file is a required setting for call-time data collection to work..**
- o Any files, elements, or pages to exclude from response time measurements. The response times log records times for all resources the web server serves, including support files like CSS files and icons or background images.
- o The same page can be accessed with different parameters passed along in the URL. The **Response Time Uri Transforms** field provides a regular expression that can be used to strip or substitute the passed parameters.

6.6. Setting up Response Time Monitoring for EWS/Tomcat and JBoss EAP 5

Response time monitoring is not available from application or web servers by default; a special module must be installed which allows JBoss ON to collect response time metrics.

After the module is installed, then the HTTP metrics can be enabled for the resource.

6.6.1. Parameters for User-Defined <filter>s

Administrators can set rules for how response time metrics are collected for application and web servers. This are response time *filters*.

For response time monitoring to work, JBoss and EWS/Tomcat all must have response time filters installed. Additional configuration options can be added to those default filters to customize the monitoring for the resource.

Table 4. Parameters for User-Defined <filter>s

Parameter	Description
chopQueryString	Only the URI part of a query will be logged if this parameter is set to true. Otherwise the whole query line will be logged. Default is true.
logDirectory	The directory where the log files will be written to. Default setting is <code>{jboss.server.log.dir}/rt/</code> (usually <code>server/xxx/log/rt</code>). If this property is not defined, the fallback is <code>{java.io.tmpdir}/rt/</code> (<code>/tmp/</code> on UNIX®, and <code>~/Application Data/Local Settings/Temp</code> – check the TEMP environment variable) is used. If you specify this init parameter, no directory <code>rt/</code> will be created, but the directory you have provided will be taken literally.
logFilePrefix	A prefix that is put in front of the log file names. Default is the empty string.
dontLogRegEx	A regular expression that is applied to query strings. See <code>java.util.regex.Pattern</code> . If the parameter is not given or an empty string, no pattern is applied.
matchOnUriOnly	Should the <code>dontLogRegEx</code> be applied to the URI part of the query (true) or to the whole query string (false). Default is true.
timeBetweenFlushesInSec	Log lines are buffered by default. When the given number of seconds have passed and a new request is received, the buffered lines will be flushed to disk even if the number of lines to flush after (see next point) is not yet reached.. Default value is 60 seconds (1 Minute).
flushAfterLines	Log lines are buffered by default. When the given number of lines have been buffered, they are flushed to disk. Default value is 10 lines.

Parameter	Description
maxLogFileSize	The maximum allowed size, in bytes, of the log files; if a log file exceeds this limit, the filter will truncate it; the default value is 5242880 (5 MB).
vHostMappingFile	<p>This properties file must exist in the Tomcat process classpath. For example, in the conf/vhost-mappings.properties file. The file contains mappings from the 'incoming' vhost (server name) to the vhost that should be used as the prefix in the response time log file name. If no mapping is present (no file or no entry response times are set), then the incoming vhost (server name) is used. For example:</p> <pre data-bbox="815 707 1406 842">pickeldi.users.acme.com=pickeldi pickeldi= %HOST%=</pre> <p>The first mapping states that if the incoming vhost is 'host1.users.acme.com', then the log file name should get a vhost of 'host1' as prefix, separated by a _ from the context root portion. The second mapping states that if the 'incoming' vhost is 'host1', then no prefix, and no _, should be used. The third mapping uses a special left-hand-side token, '%HOST%'. This mapping states that if the 'incoming' vhost is a representation of localhost then no prefix, and no _, should be used.</p> <p>%HOST% will match the host name, or canonical host name or IP address, as returned by the implementation of <code>InetAddress.getLocalHost()</code>.</p> <p>The second and third mappings are examples of empty right hand side, but could just as well have provided a vhost.</p> <p>This is a one time replacement. There is no recursion in the form that the result of the first line would then be applied to the second one.</p>

6.6.2. Configuring Response Time Metrics for JBoss EAP/AS 5

To collect response time metrics for a JBoss EAP/AS 5 server, first install the servlet JAR for the response time filter and configure the web server to use it.

1. Download the Response Time packages for JBoss from the JBoss ON UI.

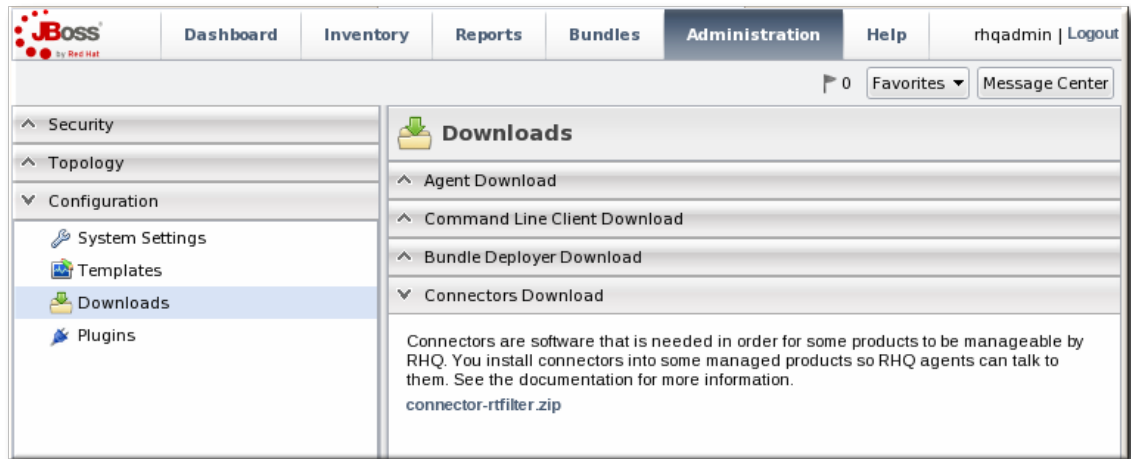


NOTE

This can also be done from the command line using **wget**:

```
[root@server ~]# wget
http://server.example.com:7080/downloads/connectors/connector-rtfilter.zip
```

1. Click the **Administration** tab in the top menu.
2. In the **Configuration** menu box on the left, select the **Downloads** item.



3. Click the **connector-rtfilter.zip** link, and save the file.
2. Unzip the connectors.
3. Copy the **rhq-rtfilter-version.jar** file into the **lib/** directory for the profile.

```
[root@server ~]# unzip connector-rtfilter.zip
```

```
[root@server ~]# cp connector-rtfilter/rhq-rtfilter-version.jar
JbossHomeDir/server/profileName/lib/
```

JBoss EAP/AS already includes the **commons-logging.jar** file, which is also required for response time filtering.

4. Then, configure the **web.xml** for the EAP/AS instance.

The response time filter can be deployed globally, for all web applications hosted by the EAP/AS instance or it can be configured for a specific web application.

To configure it globally, edit the global **web.xml** file:

```
[root@server ~]# vim
JbossHomeDir/server/configName/default/deploy/jbossweb.sar/
```

To configure it for a single web app, edit that one web app's **web.xml** file:

```
[root@server ~]# vim WARLocation/WEB-INF/web.xml
```

- 5. Add the filter and, depending on the configuration, filter mapping elements to the file. This activates the response time filtering.

All that is required for response time filtering to work is the default `<filter>` element, without any optional parameters. However, the parameters can be uncommented and set as necessary; the different ones are described in [Table 4, "Parameters for User-Defined <filter>s"](#).

```

<filter>
  <filter-name>RhqRtFilter</filter-name>
  <filter-
class>org.rhq.helpers.rtfiler.filter.RtFilter</filter-class>

  <!-- Optional parameters.

  <init-param>
    <param-name>chopQueryString</param-name>
    <param-value>>true</param-value>
  </init-param>
  <init-param>
    <param-name>logDirectory</param-name>
    <param-value>/tmp</param-value>
  </init-param>
  <init-param>
    <param-name>logFilePrefix</param-name>
    <param-value>localhost_7080_</param-value>
  </init-param>
  <init-param>
    <param-name>dontLogRegEx</param-name>
    <param-value></param-value>
  </init-param>
  <init-param>
    <param-name>matchOnUriOnly</param-name>
    <param-value>>true</param-value>
  </init-param>
  <init-param>
    <param-name>timeBetweenFlushesInSec</param-name>
    <param-value>73</param-value>
  </init-param>
  <init-param>
    <param-name>flushAfterLines</param-name>
    <param-value>13</param-value>
  </init-param>
  <init-param>
    <param-name>maxLogFileSize</param-name>
    <param-value>5242880</param-value>
  </init-param>
  -->
</filter>

<!-- Use this only when also enabling the RhqRtFilter in the
filter
  <filter-mapping>
    <filter-name>RhqRtFilter</filter-name>

```

```

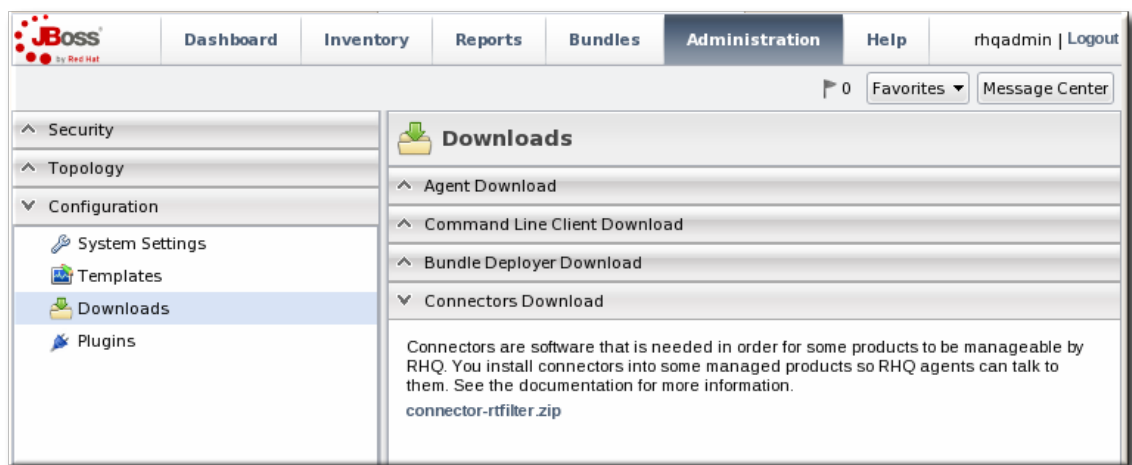
        <url-pattern>/*</url-pattern>
    </filter-mapping>
-->

```

- Restart the JBoss EAP/AS server to load the new `web.xml` settings.
- Enable the HTTP metrics, as described in [Section 6.6.4, “Configuring HTTP Response Time Metrics”](#), so that JBoss ON checks for the response time metrics on the application server.

6.6.3. Configuring Response Time Filters for Tomcat

- Download the Response Time packages for Tomcat from the JBoss ON UI.
 - Click the **Administration** tab in the top menu.
 - In the **Configuration** menu box on the left, select the **Downloads** item.



- Click the `connector-rtfilter.zip` link, and save the file.
- Unzip the Response Time connectors.

```
unzip connector-rtfilter.zip
```

The package contains two JAR files, `commons-logging-version.jar` and `rhq-rtfilter-version.jar`. Tomcat 5 servers use only the `commons-logging-version.jar` file, while Tomcat 6 servers require both files.

- Copy the appropriate JAR files into the Tomcat configuration directory. The directory location depends on the Tomcat or JBoss instance (for embedded Tomcat) being modified.

For example, on a standalone Tomcat 5.5:

```
cp commons-logging-version.jar /var/lib/tomcat5/server/lib/
```

On Tomcat 6:

```
cp rhq-rtfilter-version.jar /var/lib/tomcat6/lib/
cp commons-logging-version.jar /var/lib/tomcat6/lib/
```

For example, on an embedded Tomcat instance:

```

cp rhq-rtfilter-version.jar
JBoss_install_dir/server/default/deploy/jboss-web.deployer/
cp commons-logging-version.jar
JBoss_install_dir/server/default/deploy/jboss-web.deployer/

```

4. Open the **web.xml** file to add the filter definition. The exact location of the file depends on the server instance and whether it is a standalone or embedded server; several common locations are listed in [Table 5, “web.xml Configuration File Locations”](#).
5. Add either a **<filter>** or a **<filter-mapping>** entry to configuration the Response Time filter in the Tomcat server. Either a **<filter>** or a **<filter-mapping>** entry can be used, but *not* both.

The most basic filter definition references simply the Response Time filter name and class in the **<filter>** element. This loads the response time filter with all of the default settings.

```

<filter>
  <filter-name>RhqRtFilter </filter-name>
  <filter-class>org.rhq.helpers.rtfilter.filter.RtFilter
</filter-class>
</filter>

```

The filter definition can be expanded with user-defined configuration values by adding **<init-param>** elements. This loads the response time filter with all of the default settings.

```

<filter>
  <filter-name>RhqRtFilter </filter-name>
  <filter-class>org.rhq.helpers.rtfilter.filter.RtFilter
</filter-class>
  <init-param>
    <description>Name of vhost mapping file. This
properties file must be in the Tomcat process
classpath.</description>
    <param-name>vHostMappingFile</param-name>
    <param-value>vhost-mappings.properties</param-
value>
  </init-param>
  ...
</filter>

```

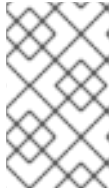
The available parameters are listed in [Table 4, “Parameters for User-Defined <filter>s”](#).

Alternatively, set a **<filter-map>** entry which gives the name of the response time filter and pattern to use to match the URL which will be monitored.

```

<filter-mapping>
  <filter-name>RhqRtFilter </filter-name>
  <url-pattern>/* </url-pattern>
</filter-mapping>

```

**NOTE**

Put the Response Time filter in front of any other configured filter so that the response time metrics will include all of the other response times, total, in the measurement.

6. Restart the Tomcat instance to load the new configuration.
7. Enable the HTTP metrics, as described in [Section 6.6.4, “Configuring HTTP Response Time Metrics”](#), so that JBoss ON checks for the response time metrics on the application server.

Table 5. web.xml Configuration File Locations

Tomcat Version	Embedded Server Type	File Location
Tomcat 6	Standalone Server	<i>/var/lib/tomcat6/webapps/project/WEB-INF/web.xml</i>
Tomcat 5	Standalone Server	<i>/var/lib/tomcat5/webapps/project/WEB-INF/web.xml</i>
Tomcat 6	EAP 5 EAP 5.0.0	<i>JBOSS_HOME/server/config/deployers/jbossweb.deployer/web.xml</i>
Tomcat 6	JBoss 4.2, JBoss EAP4	<i>JBOSS_HOME/server/config/deploy/jbossweb.deployer/conf/web.xml</i>
Tomcat 5.5	JBoss 4.0.2	<i>JBOSS_HOME/server/config/deploy/jbossweb-tomcat55.sar/conf/web.xml</i>
Tomcat 5.0	JBoss 3.2.6	<i>JBOSS_HOME/server/config/deploy/jbossweb-tomcat50.sar/conf/web.xml</i>
Tomcat 4.1	JBoss 3.2.3	<i>JBOSS_HOME/server/config/deploy/jbossweb-tomcat41.sar/web.xml</i>

6.6.4. Configuring HTTP Response Time Metrics

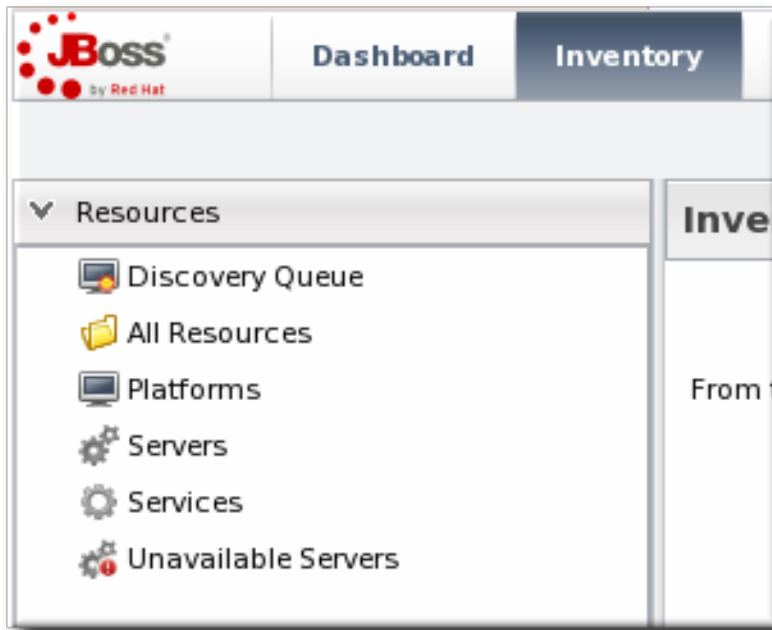
Configuring response time metrics is in some respects analogous to configuring events. The JBoss ON agent polls certain log files kept by the web server to identify the performance times for different resources served by the web server.

1. Install the response time filter for the web server. If necessary, set up the filter entry in the **web.xml** file.

See [Section 6.6, “Setting up Response Time Monitoring for EWS/Tomcat and JBoss EAP 5”](#).

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

3. Select the **Servers** menu table on the left, and then navigate to the web server



4. Click the **Connection Settings** tab on the web server resource entry, and scroll to the **Response Time** configuration section.
5. Configure the response time properties for the web server. The agent has to know what log file the web server uses to record response time data.

Optionally, the server can perform certain transformations on the collected data.

- The response times log records times for all resources the web server serves, including support files like CSS files and icons or background images. These resources can be excluded from the response time calculations in the **Response Time Url Excludes** field.
- The same page can be accessed with different parameters passed along in the URL. The **Response Time Url Transforms** field provides a regular expression that can be used to strip or substitute the passed parameters.

Main APACHE VIRTUAL HOST

Summary Inventory Alerts **Monitoring** Configuration

Tables Graphs Traits Availability **Schedules** Calltime

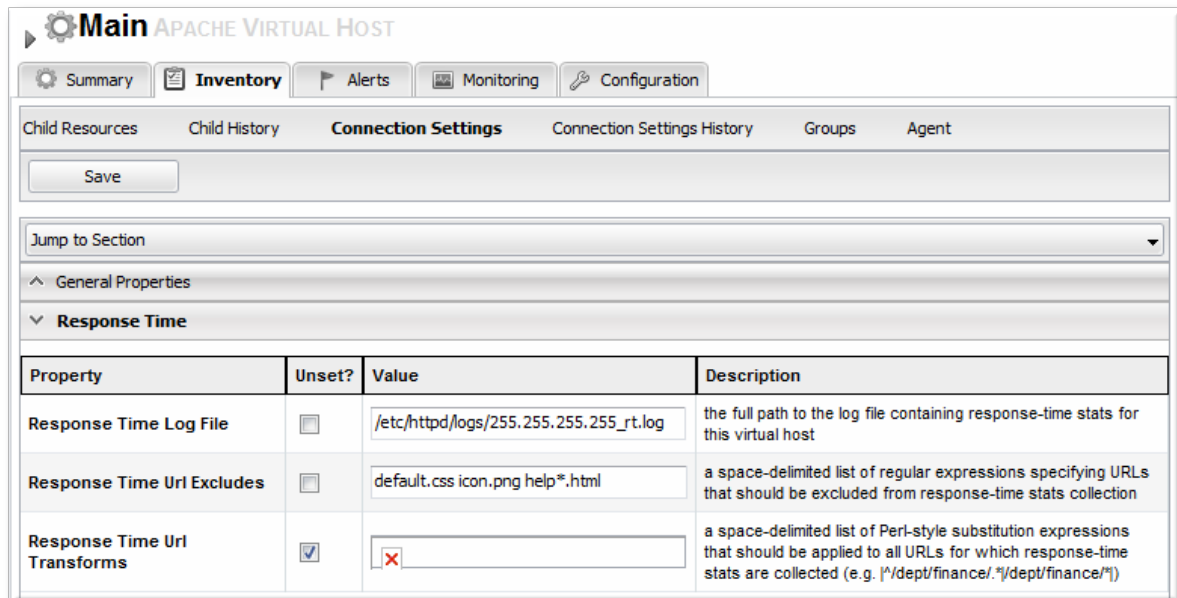
Resource Metric Collection Schedules

Metric ^	Description	Type	Enabled?	Collection Interval
per Minute	this service	measurement	<input type="checkbox"/>	10 minutes
Bytes Sent for 401 Responses	The number of bytes sent for 401 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Bytes Sent for 401 Responses per Minute	The number of bytes sent for 401 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Bytes Sent for 403 Responses	The number of bytes sent for 403 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Bytes Sent for 403 Responses per Minute	The number of bytes sent for 403 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Bytes Sent for 404 Responses	The number of bytes sent for 404 responses generated by this service	measurement	<input checked="" type="checkbox"/>	40 minutes
Bytes Sent for 404 Responses per Minute	The number of bytes sent for 404 responses generated by this service	measurement	<input checked="" type="checkbox"/>	40 minutes
Bytes Sent for 500 Responses	The number of bytes sent for 500 responses generated by this service	measurement	<input checked="" type="checkbox"/>	40 minutes
Bytes Sent for 500 Responses per Minute	The number of bytes sent for 500 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Host	the host name or IP address of this virtual host	trait	<input checked="" type="checkbox"/>	24 hours
HTTP Response Time	The minimum, maximum, and average response times for HTTP requests serviced by this virtual host	calltime	<input checked="" type="checkbox"/>	10 minutes
Number of 200 Responses	The number of 200 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Number of 200 Responses per Minute	The number of 200 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Number of 301 Responses	The number of 301 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Number of 301 Responses per Minute	The number of 301 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Number of 302 Responses	The number of 302 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes
Number of 302 Responses per Minute	The number of 302 responses generated by this service	measurement	<input type="checkbox"/>	40 minutes

Collection Interval: minutes

Total Rows: 57 (selected)

6. Click the **Save** button.
7. Click the **Monitoring** tab on the web server resource entry.
8. Click the **Schedules** subtab.
9. Select the **HTTP Response Time** metric. This metric is the *calltime* type.



10. Click the **Enable** at the bottom of the list.

7. RESOURCE TRAITS

One category of information that is collected about a resource is its *traits*. Traits are descriptive information, usually information that does not change very frequently.

For example, traits for a platform include its operating system name and version, its distribution, its architecture, and its hostname. Most resources have similar identifying information, such as a version number or vendor information.

Traits are in-between information. They are detectable and are detected by the JBoss ON agent's monitoring processes. But they are also generalized descriptive information. Trait information is even shown on the resource's details page.

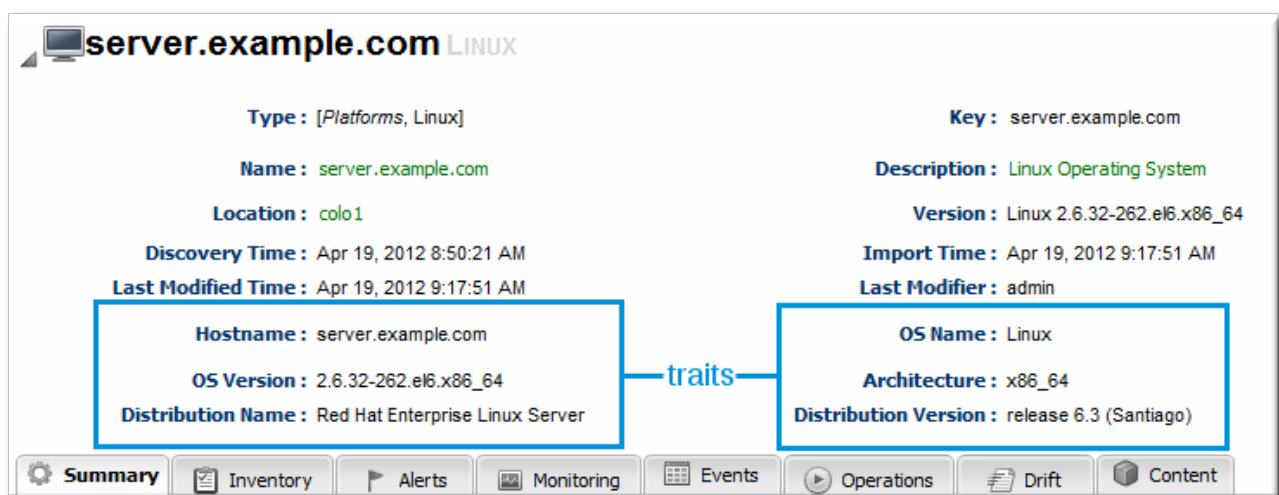


Figure 16. Resource Details

The traits that are collected are defined in the resource plug-in itself, so this information is viewable but not configurable through the UI. The list of traits for each resource type is covered in the *Resource Reference: Monitoring, Operation, and Configuration Options*.

7.1. Collection Interval

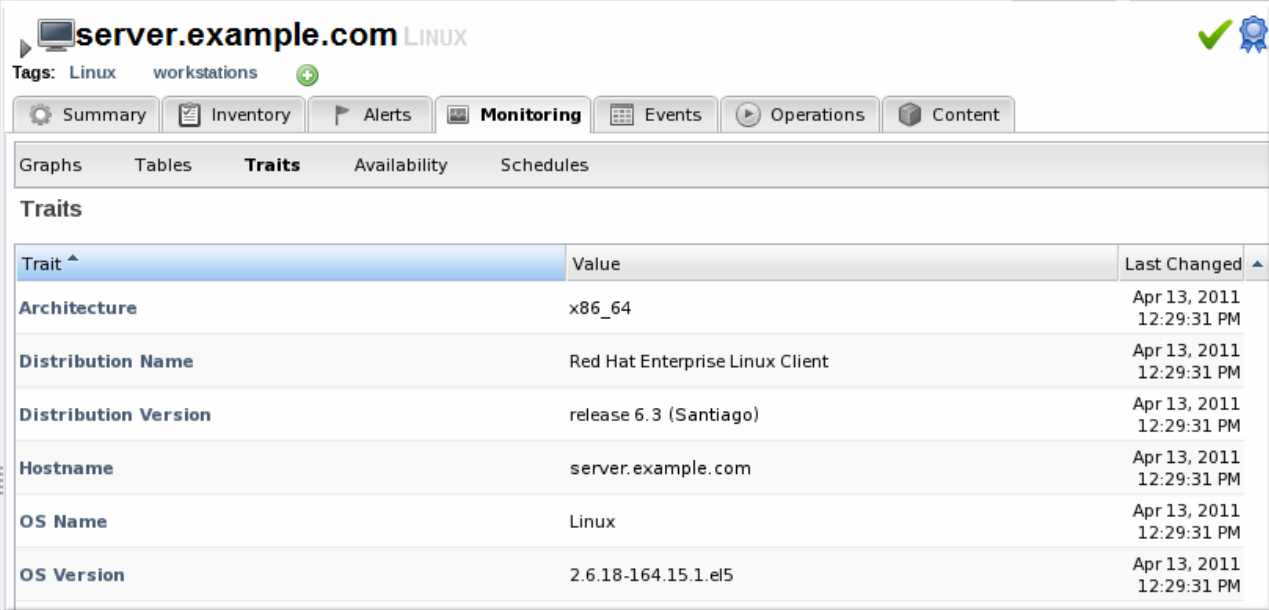
By default for most resource types, traits are checked every 24 hours. Because traits change infrequently, they do not need to be collected very often.

The trait collection interval is configured the same as metrics collection intervals. To change the collection interval for a trait, see [Section 4.5, “Setting Collection Intervals for a Specific Resource”](#).

7.2. Viewing Traits

The **Traits** subtab in the **Monitoring** tab for a resource displays three pieces of information:

- The trait name. The traits which are monitored for a resource are defined with other monitoring settings in the resource type's plug-in descriptor.
- The trait value.
- The time of the last collection where a change in trait information was detected.



Trait ^	Value	Last Changed ^
Architecture	x86_64	Apr 13, 2011 12:29:31 PM
Distribution Name	Red Hat Enterprise Linux Client	Apr 13, 2011 12:29:31 PM
Distribution Version	release 6.3 (Santiago)	Apr 13, 2011 12:29:31 PM
Hostname	server.example.com	Apr 13, 2011 12:29:31 PM
OS Name	Linux	Apr 13, 2011 12:29:31 PM
OS Version	2.6.18-164.15.1.el5	Apr 13, 2011 12:29:31 PM

Figure 17. Trait Charts

7.3. Extended Example: Alerting and Traits

The Setup

Trait information tends to be static. While traits can, and do, change, they do so infrequently. Also, traits convey descriptive information about a resource, not state data or dynamic measurements, so traits are not critical for IT administrators to track closely.

However, trait information can be valuable in letting administrators know when some configuration or package on the resource has changed because one trait that most resources collect is version information. If a version number changes, then some update has occurred on the underlying resource.

What to Do

For example, Tim the IT Guy has automatic updates configured for his Red Hat Enterprise Linux development and QA servers. Because his production environment has controlled application and system updates, there are no automatic updates for those servers.

Tim wants to be informed of version changes, but they are not necessarily a big deal. JBoss ON can issue an alert when a trait changes, with the **Trait Value Change** condition. (Cf. [Section 11.2, “Basic Procedure for Setting Alerts for a Resource”](#).)

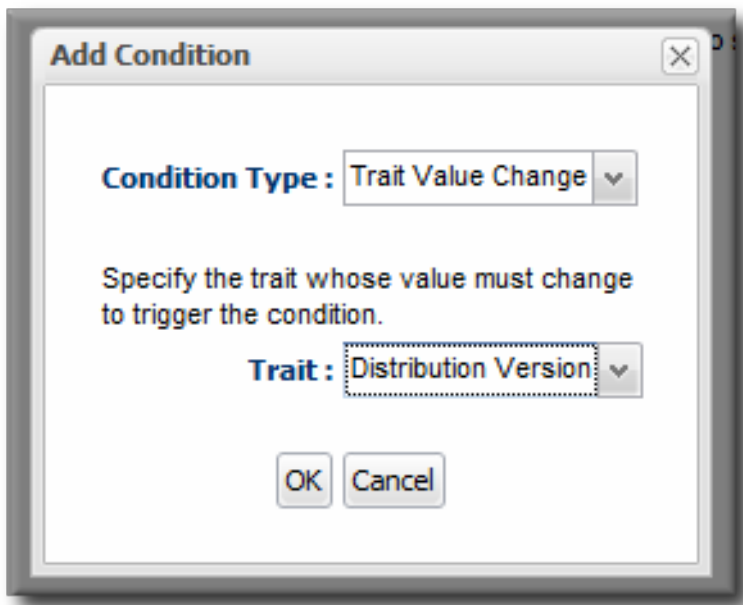


Figure 18. Trait Alert Condition

The alert for the development and QA systems is simple:

- He sets two conditions, using an OR operator. The alert triggers when the distribution version changes *or* when the operating system version changes. This catches both minor and major updates to the operating system or kernel.
- It is set to low priority so it is informative but not critical.
- Tim decides that the alert notification is sent to his JBoss ON user, so he sees notifications when he logs in. He could also configure an email notification for high-priority resources.

For Tim's production resources, he sets the alert priority to high and uses an email notification to multiple IT administrators so that they are quickly aware of any change to the production systems.

8. RESOURCES WHICH REQUIRE SPECIAL CONFIGURATION FOR MONITORING

Web servers (Tomcat and Apache resources) require additional configuration for JBoss ON to be able to manage and monitor those resources.

8.1. Configuring Tomcat/EWS Servers for Monitoring

For instructions on setting up Tomcat or Red Hat JBoss Web Server (JWS) for monitoring with JBoss Operations Network, see the JBoss Web Server Installation Guide chapter on [Monitoring Red Hat JBoss Web Server with JBoss ON](#)



NOTE

For more detail on configuring Tomcat, see the Tomcat documentation.

8.2. Configuring the Apache SNMP Module

To discover an Apache server's virtual hosts and collect metrics for them, the SNMP module must be configured on that Apache server.

Apache 2.2 is supported on Red Hat Enterprise Linux and Windows platforms.

IMPORTANT

To use the Response Time module, the Apache server needs to have been compiled with shared object support. For Red Hat Enterprise Linux systems and EWS servers, this is enabled by default.

To verify that the Apache server was compiled with shared object support, use the **apachectl -l** command to list the compiled modules and look for the **mod_so.c** module:

```
[root@server ~]# apachectl -l
Compiled in modules:
  core.c
  prefork.c
  http_core.c
  mod_so.c
```

Use the **--enable-module=so** option:

```
$ ./configure --enable-module=so
$ make install
```

1. Download the Apache binaries from the JBoss ON UI.

1. Log into the JBoss ON UI.

```
https://server.example.com:7080
```

2. Click the **Administration** tab in the top menu.

3. In the **Configuration** menu box on the left, select the **Downloads** item.



4. Scroll to **Connector Downloads**, and click the **connector-apache.zip** link to download the Apache connectors.
2. Unzip the Apache connectors in a directory that is accessible to the JBoss ON agent.

```
unzip connector-apache.zip
```

3. Each Apache version and platform has its own package that contains the Apache-SNMP connectors. Extract the Apache connectors in a directory that is accessible to the JBoss ON agent. Binaries are available for Red Hat Enterprise Linux 32-bit and 64-bit and Windows 32-bit.

For example, on Red Hat Enterprise Linux 32-bit:

```
[jsmith@server ~]$ cd apacheModuleRoot/apache-snmplib/binaries/
[jsmith@server binaries]$ tar xjvf snmp_module-x86-linux-
apache#.tar.bz2
```

is the Apache server version number.



NOTE

Apache connectors can be compiled for other platforms, like Solaris, from the source files in **apacheRoot/apache-snmplib/binaries/sources**. For example:

```
[jsmith@server ~]$ cd
JON_AGENT_INSTALL_DIR/product_connectors/apache-
snmp/sources
[jsmith@server sources]$ ./build_apache_snmplib.sh
APACHE_VERSION APACHE_2.x_INSTALL_DIR/bin/apxs
```

To compile the Apache-SNMP connector, **apxs**, **perl**, **make**, and **automake** must all be installed and in user **PATH**.

4. Install the module. For example:

```
[root@server ~]# cd apacheModuleRoot/apache-
snmp/binaries/snmp_module_#

[root@server snmp_module]# cp module/*
apache_install_directory/modules

[root@server snmp_module]# cp conf/* apache_install_directory/conf

[root@server snmp_module]# mkdir apache_install_directory/var
```

On Windows:

```
> xcopy /e JON_AGENT_INSTALL_DIR\product_connectors\apache-
snmp\binaries\x86
```

5. Open the **httpd.conf** file for editing. For example:

-

```
[root@server ~]# vim apache_install_directory/conf/httpd.conf
```

6. Enable the module by adding these lines to the **httpd.conf** file.

```
LoadModule snmpcommon_module modules/libsnmpcommon.so
LoadModule snmpagt_module modules/libsnmpmonagt.so

SNMPConf    conf
SNMPVar     var
```

For Windows:

```
LoadModule snmpcommon_module modules/snmpcommon.so
LoadModule snmpagt_module modules/snmpmonagt.so

SNMPConf    conf
SNMPVar     var
```

7. Make sure the main Apache configuration section, as well as each **<VirtualHost>** configuration block, contains a **ServerName** directive with a port. The SNMP module uses this directive to uniquely identify the main server and each virtual host, so each **ServerName** directive must contain a unique value. For example:

```
ServerName main.example.com:80
...

<VirtualHost vhost1.example.com:80>
ServerName vhost1.example.com:80
...
</VirtualHost>
```

8. If there is more than one Apache instance on the same machine, it is possible to use different SNMP files for each instance.
 1. Each Apache instance has its own **httpd.conf** file. Set the **SNMPConf** directory in each file to its own SNMP configuration directory. For example, for instance1:

```
vim instance1-httpd.conf

SNMPConf /opt/apache-instance1/conf
```

Then, for instance2:

```
vim instance2-httpd.conf

SNMPConf /opt/apache-instance2/conf
```

Each **snmpd.conf** file should be in the specified directory.

2. Edit the **agentaddress** property in *apache_install_directory/conf/snmpd.conf* so that each instance has a different value agent address and port, so there is no conflict between instances.

See the [snmpd.conf](#) documentation for a description of this property's syntax.

- Restart the Apache server. For example:

```
apachectl -k restart
```

- Verify that the SNMP module was properly installed. If the module is loaded, then there will be lines referencing the SNMP module in the errors log:

```
grep SNMP apache_installation_dir/logs/error_log

[Wed Mar 19 09:54:34 2008] [notice] Apache/2.0.63 (Unix)
CovalentSNMP/2.3.0 configured -- resuming normal operations
[Wed Mar 19 09:54:35 2008] [notice] SNMP: CovalentSNMP/2.3.0 started
(user '1000' - SNMP address '1610' - pid '26738')
```

8.3. Metrics Collection Considerations with Apache and SNMP

Three metrics show values of zero when monitoring an Apache instance with the SNMP module:

- Bytes received for GET requests per minute
- Bytes received for POST requests per minute
- Total number of bytes received per minute

This is because of how SNMP interprets information from the request body. First, SNMP provides various length values for the request body and a GET request does not have a body, so GET responses are not calculated and, therefore, have a value of zero. Second, Apache does not calculate a request body size if there is request chunking.

9. REPORTS AND DATA

Monitoring information in JBoss ON is easy to find. Resources and groups both have dashboards which contain snapshot views of the most recent metrics values and a series of graphs and tables which break down the different metrics values over a given time window.

Monitoring information is available in several areas:

- Dashboards with metrics portlets for individual resources, compatible groups, and the main dashboard
- Timelines, which aggregate all collected data, events, configuration, operations, and other changes for a resource
- Resource-level charts and tables for metrics
- A Suspect Metrics report

9.1. Dashboards and Portlets

The fastest place to view monitoring and alerting information is through one of the JBoss ON dashboards. The dashboards collect almost all monitoring, event, alert, and operations data into a single location.

Each data set is collected in a separate box or *portlets* displayed in the dashboard. These portlets can be edited, added, and removed from the dashboards; this is covered in the *Admin: Initial Setup for the Resource Inventory, Groups, and Users*.

9.1.1. Resource-Level Dashboards

The **Summary > Activity** tab for an individual resource (or compatible group) shows a snapshot of all recent actions on the resource, such as new packages and content, inventory changes, events, operations, and alerts. There is also a portlet that displays the most recent detected value of the primary metrics for the resource.

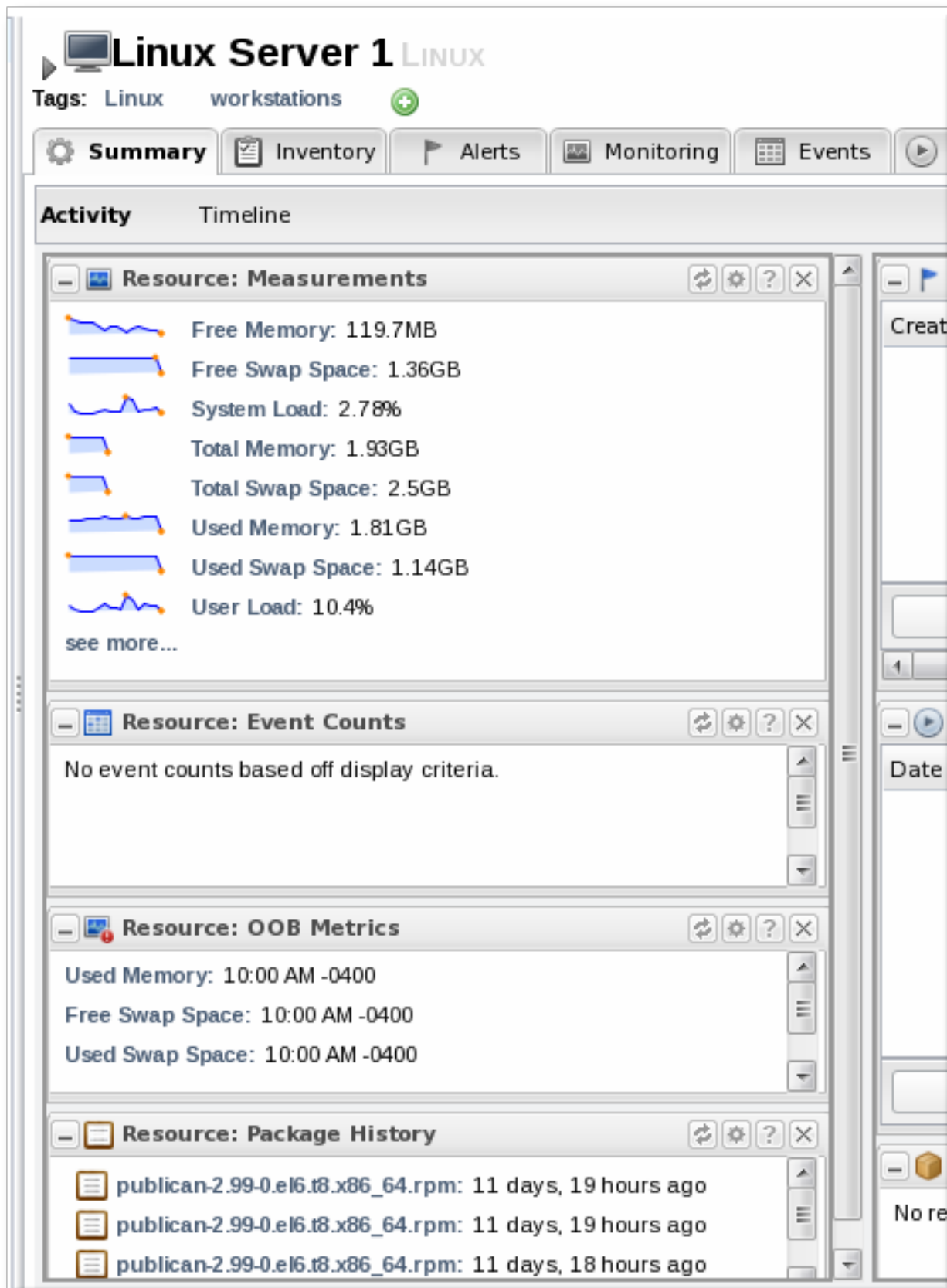


Figure 19. Resource Summary Tab

Click on any metric name in the **Resource: Measurements** portlet opens the metric graph. Clicking the **see more...** link opens the metrics charts in the **Monitoring** tab.

9.1.2. Main Dashboard

The **Dashboard** main page has a global view of all resources in the inventory. By default, this page shows only alerting data and unavailable resources. However, the **Dashboard** can be customized to show different portlets of monitoring data. Additionally, the main page can have multiple dashboards, so a dashboard can be created to look at different metrics for the same resource, the same metrics for different resources, or a combination of relevant metrics for a group of related resources — whatever you design.

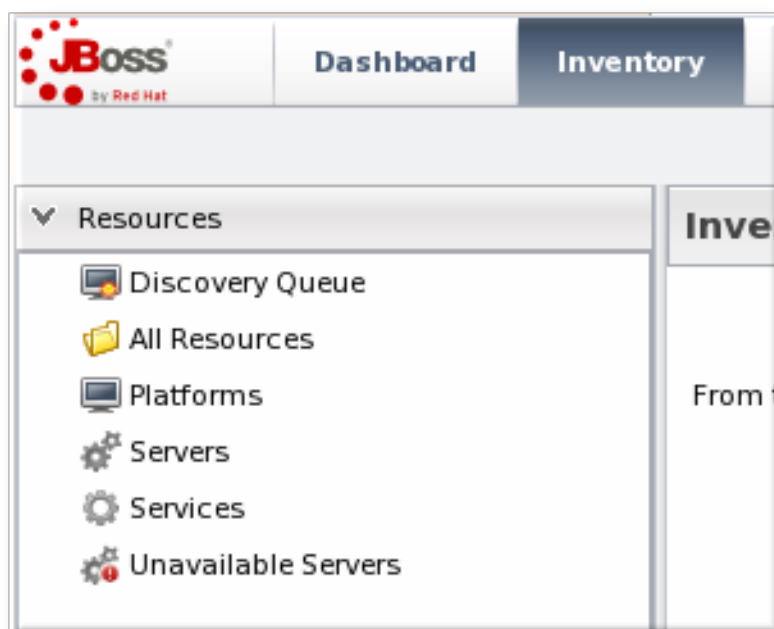
The main dashboard has several types of portlets specifically for monitoring data:

- Platform Utilization, which shows free memory, CPU usage, and other metrics related to platform performance.
- Alerted or Unavailable Resources, which shows a list of the most recent five resources which have issued an alert or been reported as down
- A graph for a specific metric for a compatible group
- A graph for a specific metric for a resource

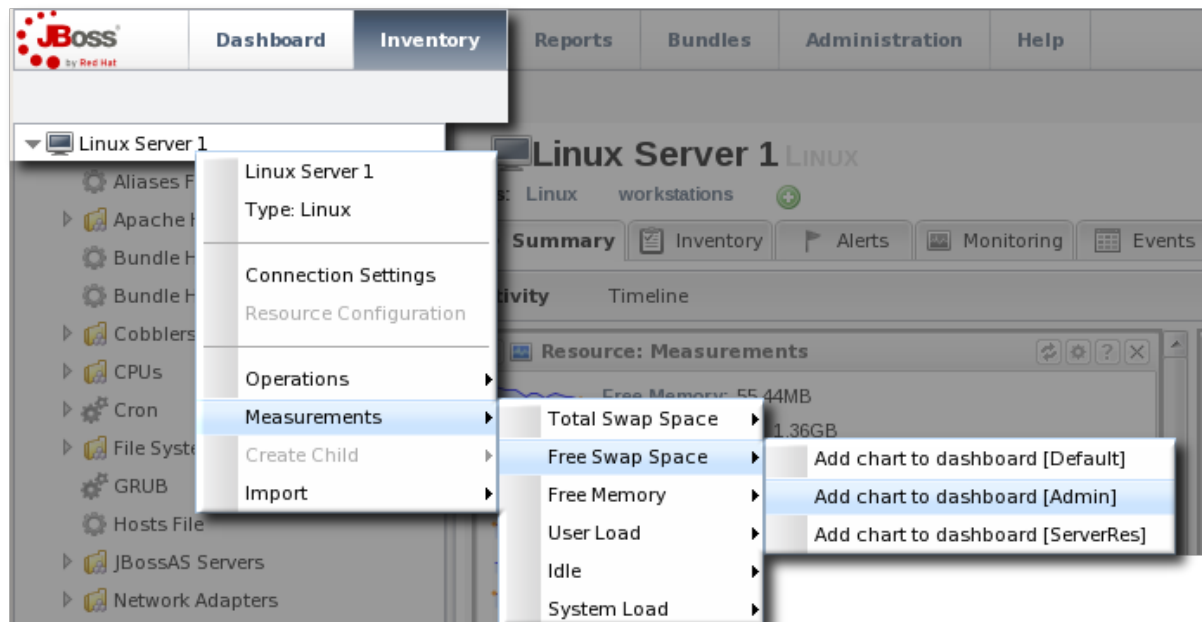
9.1.3. Adding Monitoring Metrics to the Main Dashboard

Charts for a specific metric for a resource can be added to the Dashboard. This makes it easier to see the current state of important readings for common or critical resources immediately, without having to configure alerts or check resource entries.

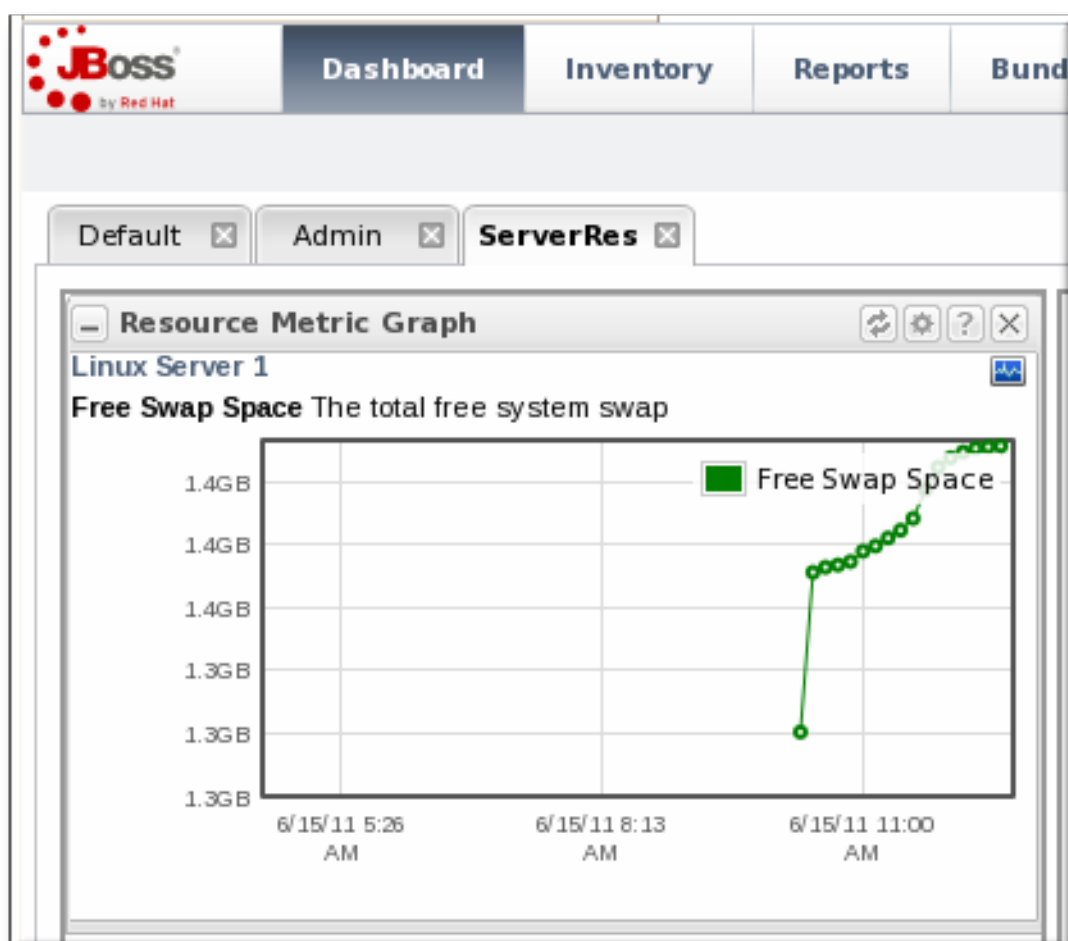
1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. In the resource hierarchy on the left, right-click the resource name.
4. Scroll down to the **Measurements** menu item, select the metric from the list, and then select the dashboard to add the chart to.



A chart for that specific metric on that specific resource is automatically added to the Dashboard that was selected.



9.2. Summary Timelines

The **Timeline** subtab in the **Summary** tab shows a line chart of all of the activity for the resource (with the exception of metrics collection, which is all under the **Monitoring** tab and charts). The **Timeline** aggregates all configuration changes, inventory changes, drift, events, content and bundle changes,

operations, and alerts. Clicking any given point opens up the details for that specific action.

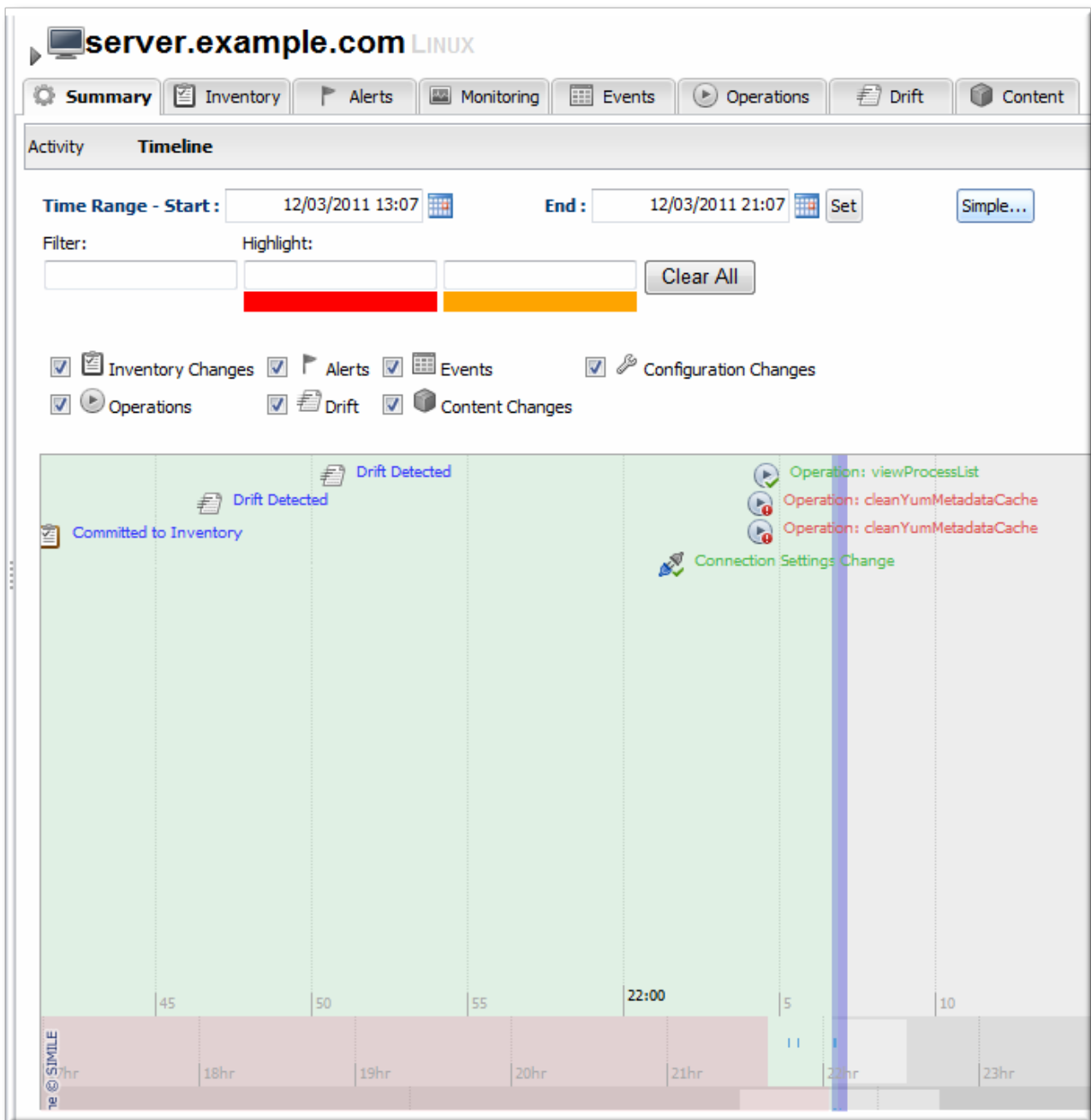


Figure 20. Summary Timeline

Because all information is on a single timeline, it becomes much easier to correlate incidents and events and to get a better understanding of the overall activity on that resource.

9.3. Resource-Level Metrics Charts

The **Monitoring** tab for a resource (or for a compatible group) has a series of different subtabs, each marking a different type of monitoring data. Each data group has its own monitoring charts.

- [Section 3.2, “Viewing a Resource's Availability Charts”](#)
- [Section 4.2, “Viewing Metrics and Baseline Charts”](#) for graphs and tables of the same metrics information
- [Section 5.4, “Viewing Events”](#)

- [Section 7.2, “Viewing Traits”](#)

Most of the monitoring data is on the **Graphs** tab, which has a series of line graphs displaying the collected data for every metric for a resource. These same data are repeated on the **Tables** tab, only laid out in a text-based table rather than visually.

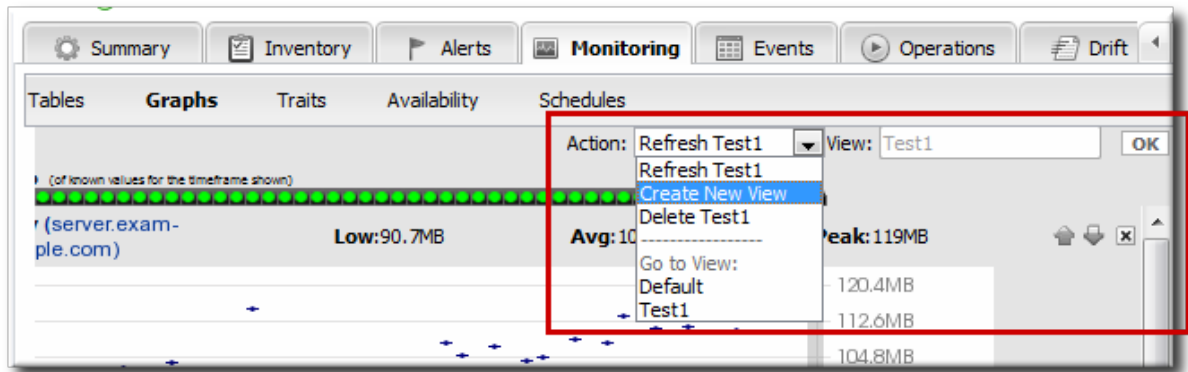


Figure 21. Metrics Chart

9.4. Creating Custom Metrics Pages

Much like creating a custom dashboard, it is possible to create a custom monitoring page for resources, so that the specific metrics displayed and the display order are different.

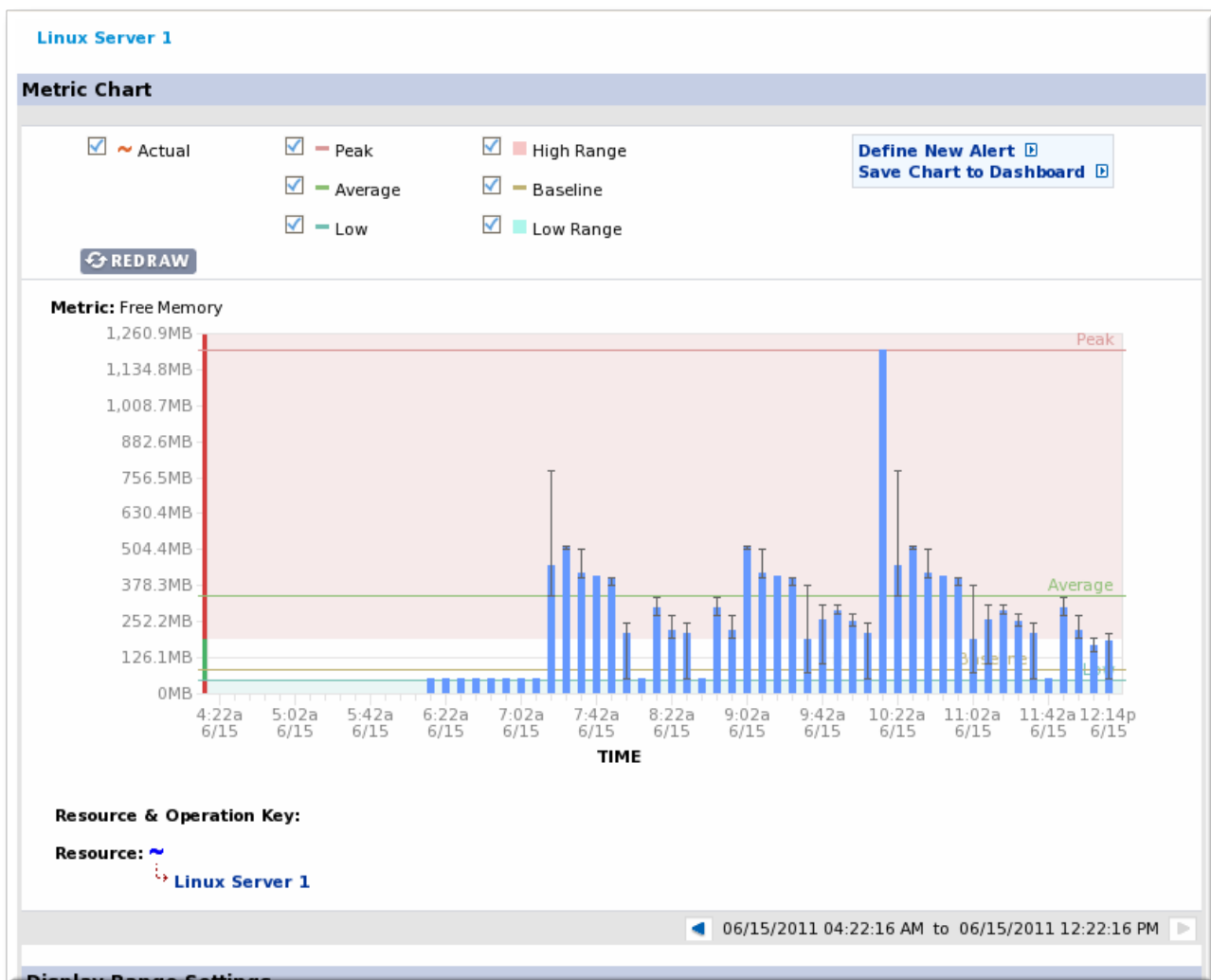
1. Open the **Monitoring > Charts** page.
2. In the **Action** drop-down menu, select **Create New View**.



3. Enter a name in the **View** field, and click **OK**.
4. To switch to a view, open the **Action** drop-down menu, and select the view name.
5. Use the arrows by the different metrics charts to change the order the metrics are displayed. To remove a metric from the view, click the X box.

9.5. Suspect Metrics Report

As described in [Section 4.1.1, "Baselines and Out-of-Bounds Metrics"](#), once metrics have been collected a few times, JBoss ON begins calculating a normal operating range for that specific resource and that specific metric. This creates a range based on the lowest and highest values.



If a metric data point comes in that is outside that normal range, higher or lower, that is a *suspect metric*. It could be a fault of the metric collection or it could indicate a resource problem.

Each individual resource has a portlet on its **Summary** tab which lists suspect, or out-of-bounds, metrics.

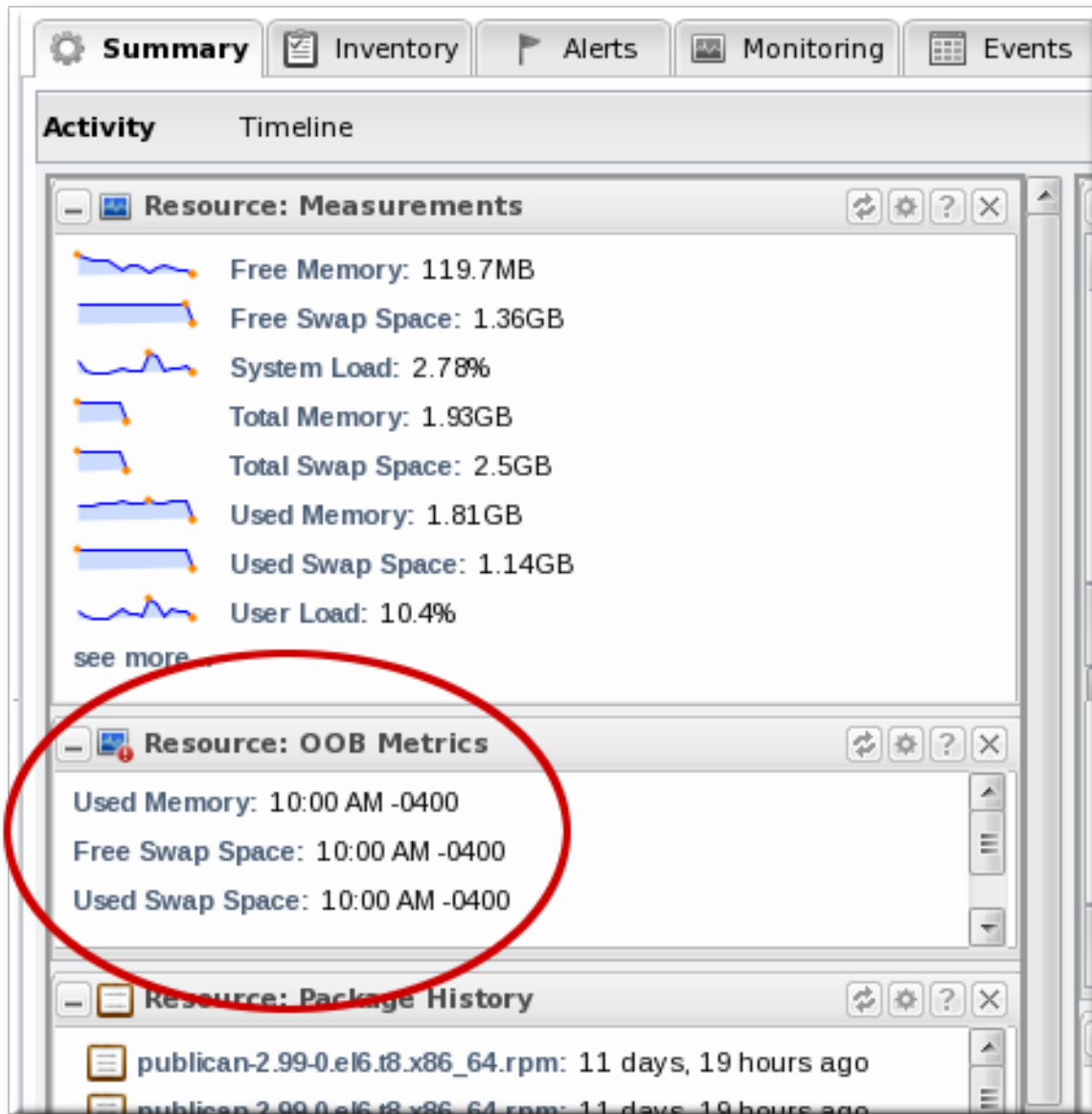


Figure 22. Out of Bounds Portlet

All resources, across the inventory, which have a suspect metric are listed in the **Suspect Metrics** report with the metric, its normal range, its suspect reading, and the factor or percentage of how far outside the metric is from normal readings.

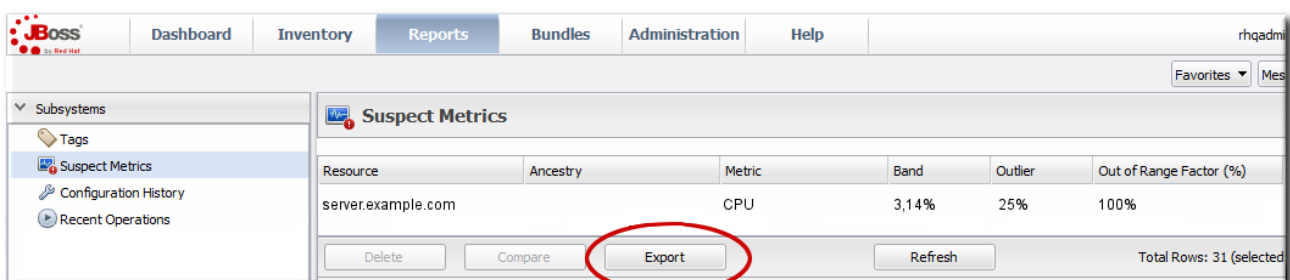


Figure 23. Suspect Metrics Reports

**NOTE**

Reports can be exported to CSV, which can be used for office systems or further data manipulation.

To export a report, simply click the **Export** button. The report will automatically be downloaded as **suspectMetrics.csv**.

9.6. Platform Utilization Report

For general infrastructure monitoring, the primary resource is the platform. The **Platform Utilization** report shows a very quick snapshot on the health of every platform in the inventory by showing its current system performance, in three metrics:

- Current CPU percentage
- The actual memory usage, based on the available physical memory, buffer, and cache
- Swap

Name	Version	CPU	Memory	Swap
server.example.com	Linux 2.6.32-220.el6.x86_64	25.6 %	4.7 %	0.0 %

Figure 24. Platform Utilization Report

**NOTE**

This report can also be added to the main **Dashboard** or a resource-level **Summary** dashboard as a portlet.

There are a couple of caveats. Only *available* platforms are listed. Other platforms in the inventory that are not in an available state are not listed. Also, the utilization is based on the most recent live data, not averages or historical values. It provides an immediate look at the platform resources.

**NOTE**

Reports can be exported to CSV, which can be used for office systems or further data manipulation.

To export a report, simply click the **Export** button. The report will automatically be downloaded as **platformUtilization.csv**.

10. STORING MONITORING DATA

JBoss ON monitoring information reveals both current measurements and historical trends and averages. JBoss ON stores data in a kind of cascade, where raw data are aggregated and compressed on a schedule. This preserves the trends of data without inflating the size of the monitoring data. Data are handled like this:

- Raw metrics are collected every few minutes and are aggregated in a rolling average in one-hour windows to produce minimum, average, and maximum values.
- One-hour values are combined and averaged in six-hour periods.
- Six-hour periods are combined and aggregated into 24-hour (1 day) windows.

10.1. Changing Storage Lengths

The raw measurements, six-hour periods, and 24-hour periods are preserved in the JBoss ON database for a configurable amount of time, ranging from one week for raw measurements to one year for 24-hour aggregates.

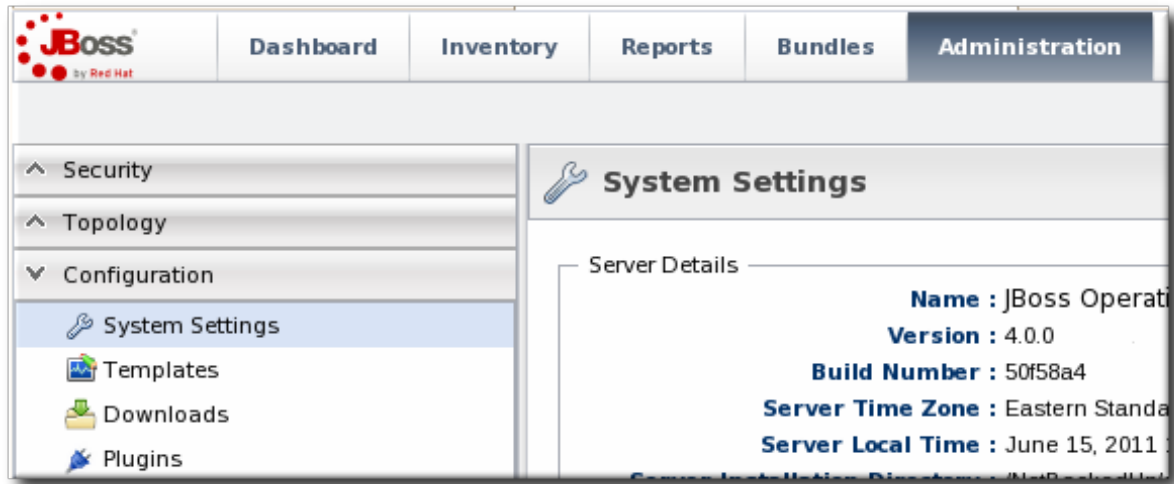
**NOTE**

The storage size of raw metrics can grow very quickly. Be careful when adjusting the storage time of raw metrics to account for the size of your database and the number and frequency of all metrics collected across your inventory.

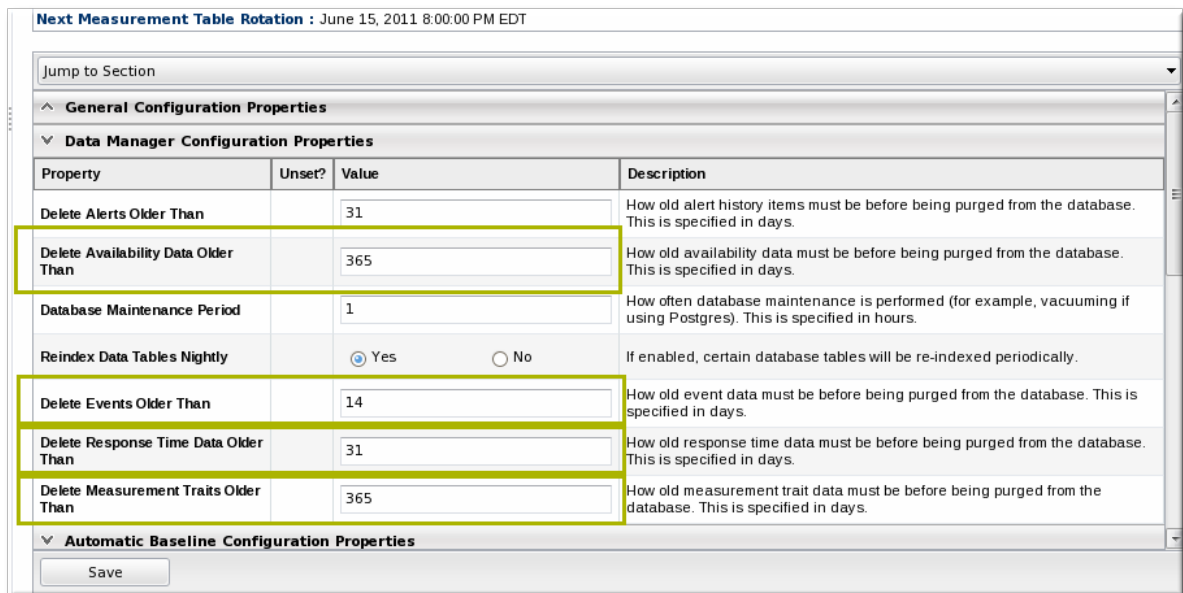
If you need to store raw data for long periods, consider exporting the raw data from the database and archiving it separately.

To change the amount of time that monitoring data are stored:

1. In the **System Configuration** menu, select the **Settings** item.



2. Scroll to the **Data Manager Configuration Properties** section.
3. Change the storage times for the different types of monitoring data.



There are four settings that relate directly to storing monitoring data:

- Response time data for web servers and EJB resources. This is kept for one month (31 days) by default.
- Events information, meaning all of the log files generated by the agent for the resource. The default storage time for event logs is two weeks.
- All measurement data, both metrics and traits. The default time is one year (365 days).
- Availability information. The default time is one year (365 days).

10.2. Exporting Raw Data

Raw monitoring data is, by default, purged from the database every week. To save the raw data, export it using the CLI.

The `MeasurementDataManager` class has a method to find the metric values for a specific resource within a certain time range:

```
findDataForResource(resourceId,
[metricId], startTime, endTime, numberOfRecords)
```

For example, for a resource with the ID 10003 and a metric ID of 10473:

```
exporter.file = '/export/metrics/metrics.csv'
exporter.format = 'csv'
var start = new Date() - 8 * 3600 * 1000;
var end = new Date()
var data = MeasurementDataManager.findDataForResource(10003,
[10473], start, end, 60)
exporter.write(data.get(0))
```

11. PLANNING ALERTS

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 ([Section 11.2, “Basic Procedure for Setting Alerts for a Resource”](#))
- The conditions that trigger the alert, which depends on the area of the resource being monitored ([Section 12, “Alert Conditions”](#))
- The method and settings to use to send the alert ([Section 13, “Alert Responses”](#))

11.1. An Alerting Strategy in Four Questions

Alerting can be very basic, an immediate notification of something going on with your IT infrastructure. Alerting can also define complex scenarios and detailed responses, allowing administrators to respond automatically and proactively to problems in resources.

Both approaches are equally valid and equally useful, depending on your environment, management processes, and plans. Putting a little effort into planning your alerts — simple and complex — can make your overall management much easier.

11.1.1. What's the Condition?

This is probably the single most important thing to determine. What event do you want to know about?

An alert definition can have one condition or multiple conditions, and those conditions can trigger an alert if only one is true or only if all are true. One of the easiest things to do is picture the scenario that needs to be reported and work backward to determine what condition or set of conditions best represents that scenario.

Alerts can be very general (“there was a change”) or very specific. General conditions are more informational. Very specific conditions can allow very specific responses — meaning that detailed and useful responses can be automated until an administrator can review and intervene.

There is no limit on the number of alert definitions that can be configured, apart from reasonable performance constraints. It may make sense to have one alert that sends a notification for a simple metric value change, while a series of other alerts look at, say, the change for that metric in relation to a

different metric, and then runs a certain JBoss ON CLI script depending on what other factors are involved.

11.1.2. What's the Frequency?

A condition can occur and recur, across multiple monitoring scans. How many times do you need to be informed of the same condition?

There are a lot of factors to consider: what kind of response will be taken, what kind of audit trail you need to keep, how common a condition is likely to be, and how long a condition may exist.

Some factors in the alert definition such as dampening rules and disable/recover alert settings throttle how many notifications are sent.

One thing to remember is that an alert notification is every response that the JBoss ON server takes when an alert is triggered. It can be sending an email, posting a message in the UI, running a CLI script, or initiating an operation.

For common, relatively low-priority alerts, probably a single notification is sufficient, so dampening and disable rules can be set to prevent spamming administrators with emails or flooding the recent alerts portlet.

For infrequent issues, critical failures, or high priority resources, the frequency of the alert notification depends on what actions you need to take. Informational alerts may be sent for as long as the condition persists, while an alert with an aggressive response like running a CLI script may only be run once and then disabled to prevent from repeatedly running the same script.

11.1.3. What's the Response to Take?

Responding to an alert has two phases: the immediate response and then long-term mitigation.

When an alert for a certain condition or event on your resource is fired, what is the very first thing that should happen? Should there be an email to an administrator? Should the JBoss ON server take some action to manage the resource? JBoss ON supports SNMP traps, so should an SNMP notification be sent to an external auditing/monitoring application?

Like conditions, there is no limit to the number of notifications that can be sent when an alert occurs. Multiple responses may make sense in some cases, where JBoss ON should email an administrator and restart a resource can simultaneously.

Operations and CLI script allow administrators to automate responses, and particularly for business-critical systems, this allows JBoss ON to take an immediate action before an administrator may even be aware there is a problem.

Once an immediate action has been taken (either by JBoss ON or by an administrator), then the administrator can acknowledge the alert, essentially dismissing it. That final step can be an important procedure, a way of stating that an administrator has reviewed the alert, identified what caused it, and found a solution.

Having a way of reviewing and signing off on events — not just as a JBoss ON task, but in real life — establishes a reliable procedure for handling IT problems and improving overall maintenance strategies.

11.1.4. How Many Resources Does This Affect?

Alert definitions can be set for an individual resource, for a group of compatible resources, or for an entire resource type.

Being able to apply the same definition to multiple resources is invaluable because of how easy it is to manage complex definitions.



NOTE

Alert templates for a resource type automatically apply to all existing and new resources of that type.

Be cognizant, though, of how different resources of the same type need to be alerted. For example, production servers require a very high level of reporting, monitoring metrics, and, possibly, automated response. QA or development servers may only require general alert information with little scripting or advanced responses. The number, frequency, and type of metrics collected on a production system may vary from those collected on QE or development systems, and that has an impact on what conditions can be used for alert definitions.

Grouping is an asset. QE, development, staging, and production resources can be separated into different groups, with the appropriate levels of monitoring and alerting set on each.

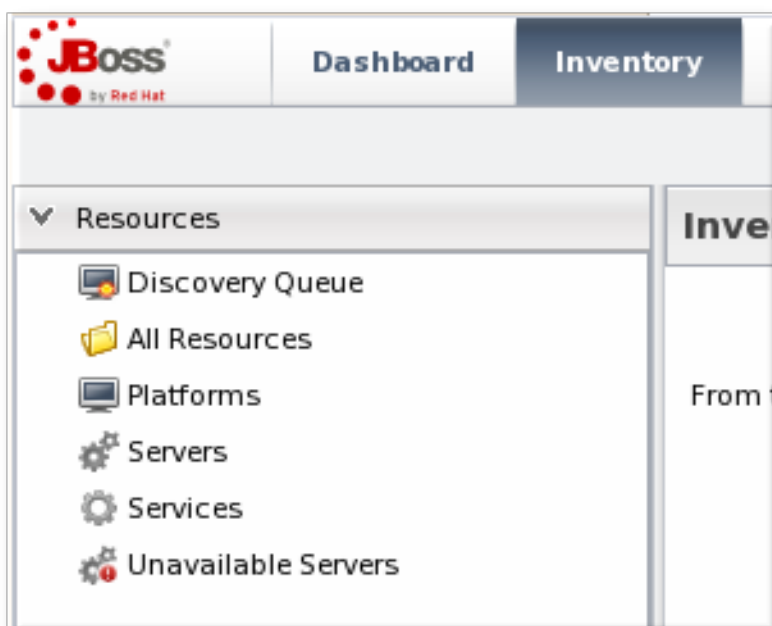
11.2. Basic Procedure for Setting Alerts for a Resource



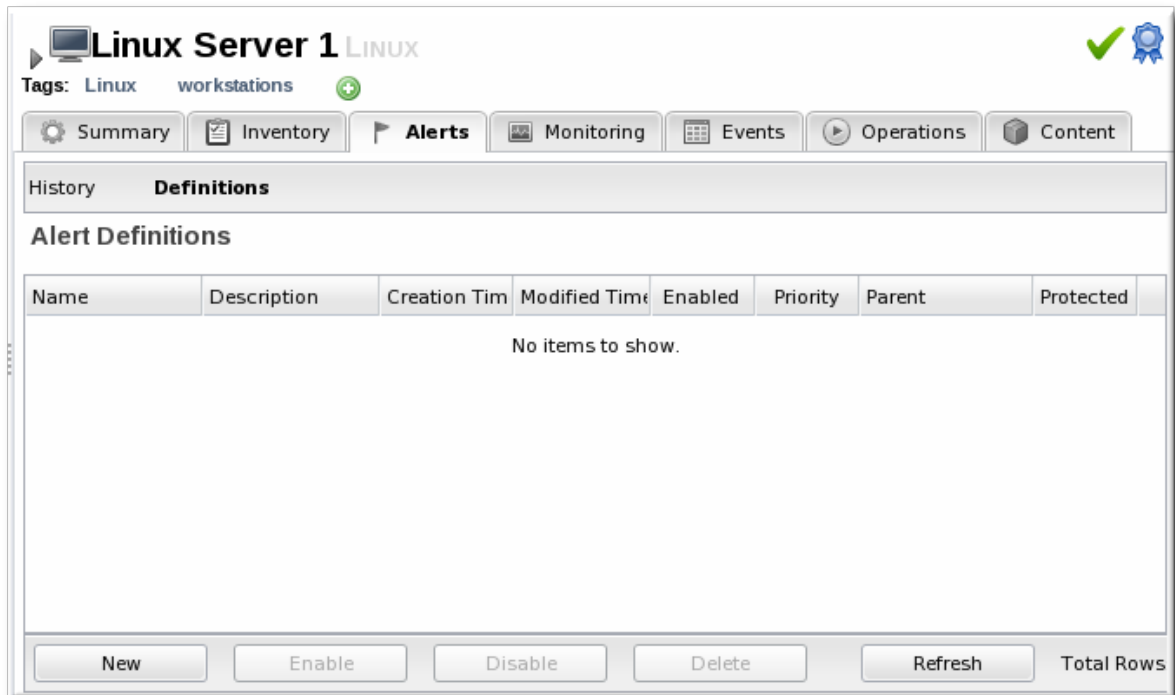
NOTE

It is not possible to edit an alert condition or an alert notification after they are set. To change the conditions or notifications for an alert definition, delete the condition or notification and create a new one.

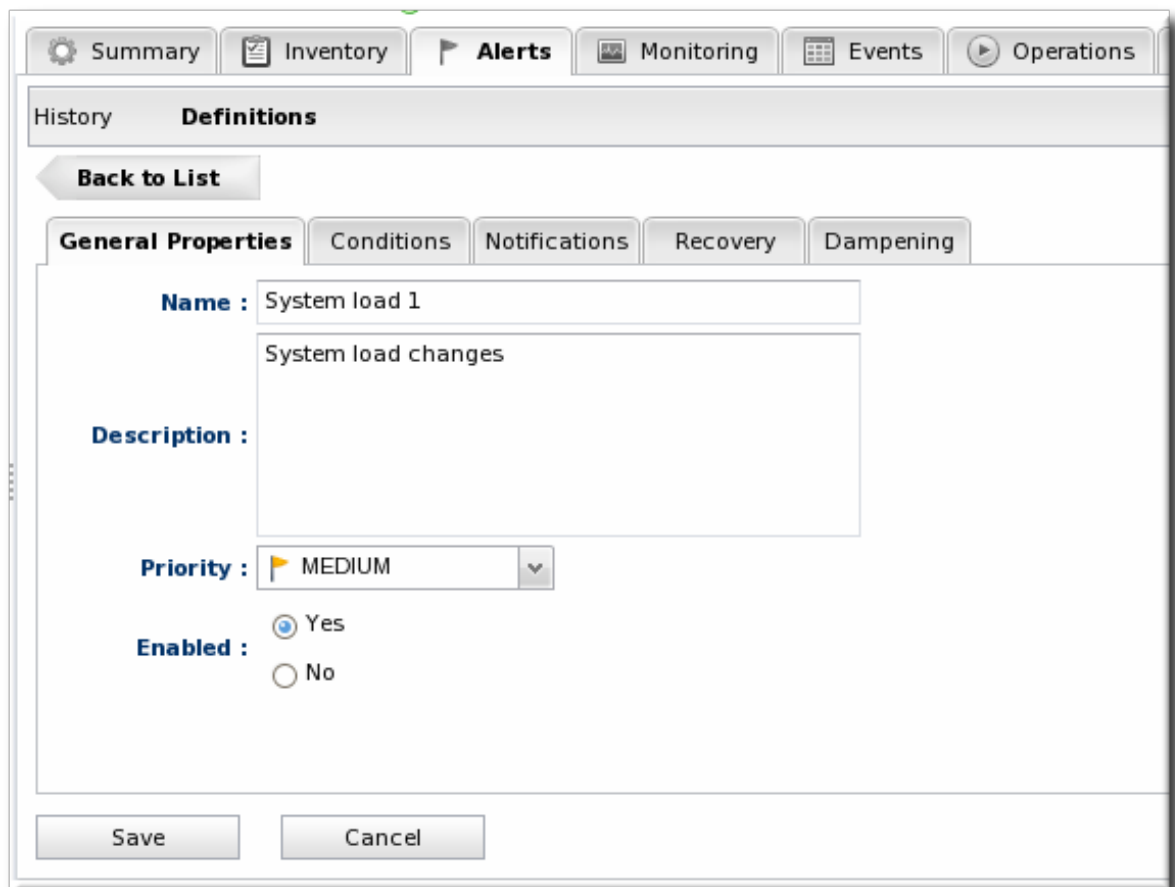
1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the resource name in the list.
4. Click the **Alerts** tab for the resource.

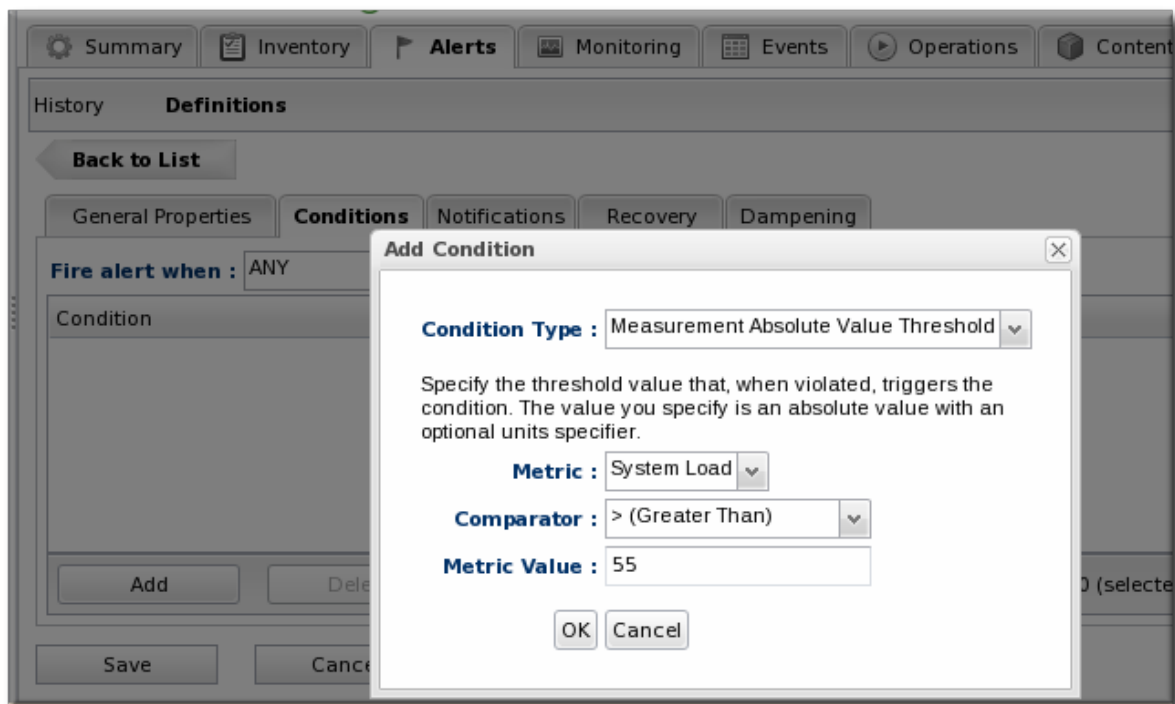


5. In the **Definitions** subtab, click the **New** button to create the new alert.
6. In the **General Properties** tab, give the basic information about the alert.



- o *Name*. Gives the name of the specific alert definition. This must be unique for the resource.
- o *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.

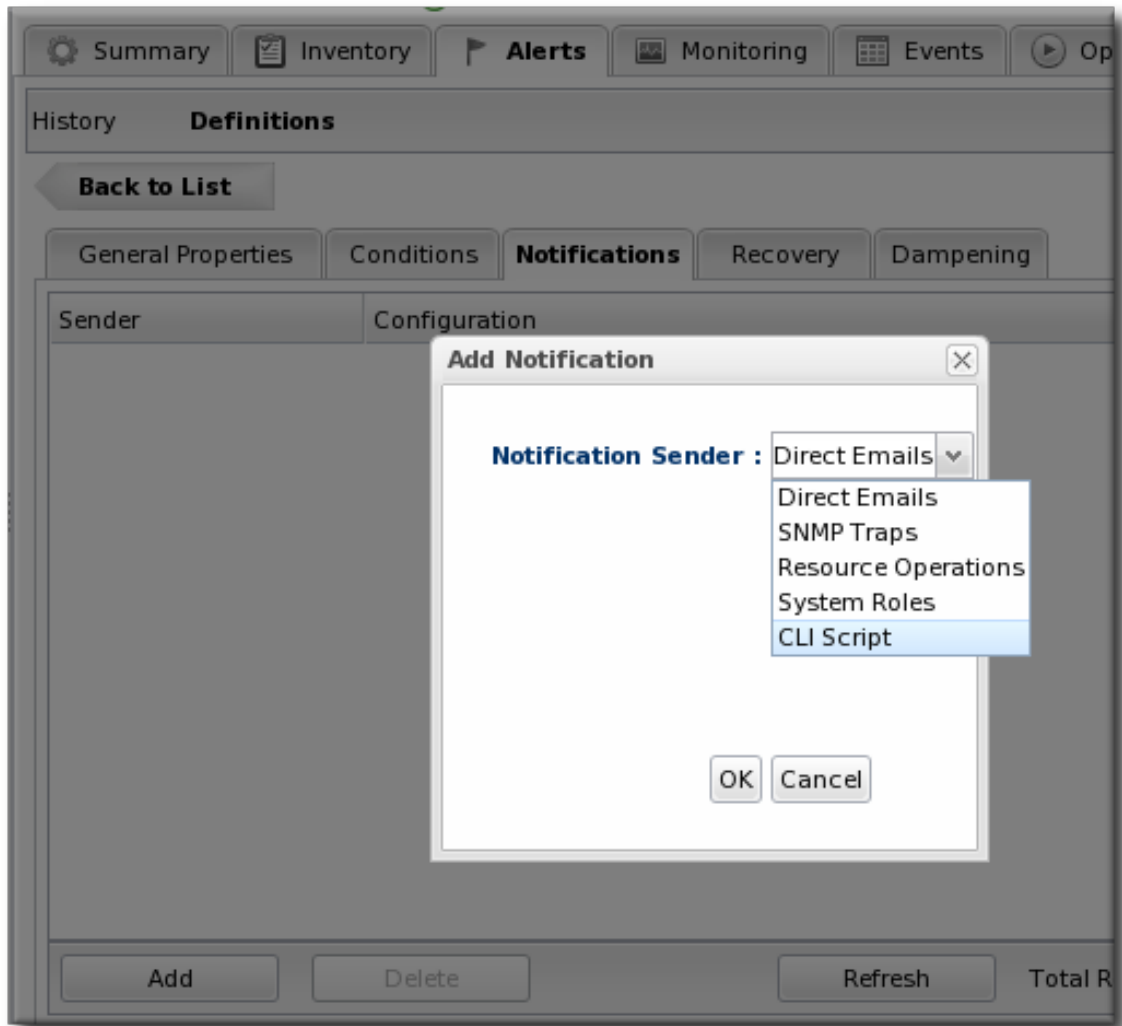
- *Priority*. Sets the priority or severity that is given to an alert triggered by this definition.
 - *Enabled*. Sets whether the alert definition is active. Alert definitions can be disabled to prevent unnecessary or spurious alerts if there is, for instance, a network outage or routine maintenance window for the resource.
7. In the **Conditions** tab, set the metric or issue that triggers the alert. Click the **Add** button to bring up the conditions form.



NOTE

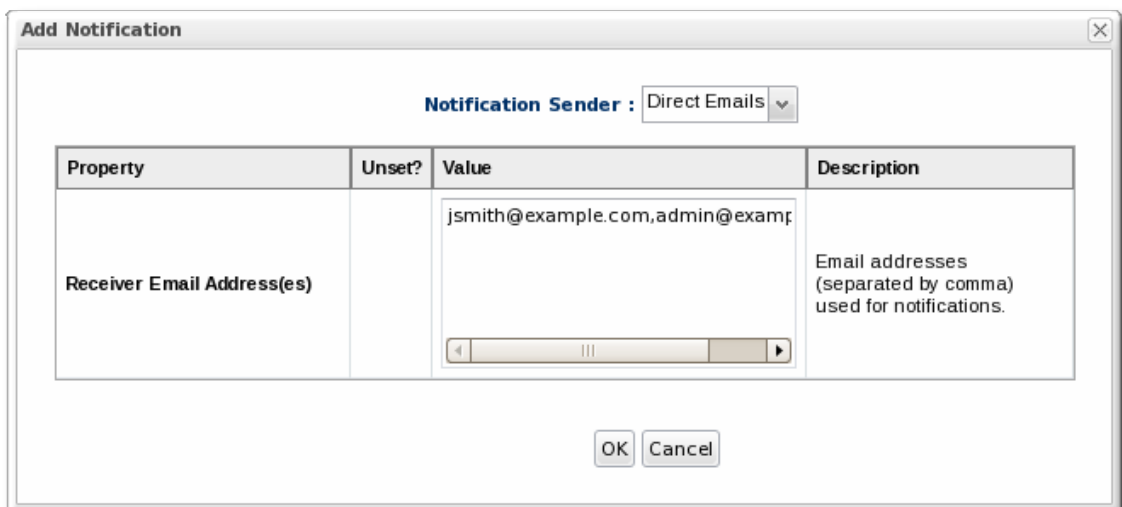
There can be more than one condition set to trigger an alert. For example, you may only want to receive a notification for a server if its CPU goes above 80% *and* its available memory drops below 25MB. The **ALL** setting for the conditions restricts the alert notification to only when both criteria 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.

1. Click the **Add a new condition** button.
 2. From the initial drop-down menu, select the type of condition. The categories of conditions are described in [Table 6, “Types of Alert Conditions”](#), and the exact conditions available to be set for every resource are listed in the *Resource Monitoring Reference*.
 3. Set the values for the condition.
8. In the **Notifications** tab, click **Add** to set a notification for the alert.
1. Select the method to use to send the alert notification in the *Sender* option.



The *Sender* option first sets the specific type of alert method (such as email or SNMP) and then opens the appropriate form to fill in the details for that specific method.

2. Fill in the required information for the alert sender method. The method may require contact information, SNMP settings, operations, or scripts, depending on what is selected.



9. In the **Recovery** tab, set whether to disable an alert until the resource state is recovered. Optionally, select another alert to enable (or recover) when this alert fires.

The screenshot shows the 'Alerts' configuration window with the 'Recovery' tab selected. The 'Recover Alert' dropdown is set to '- None -'. The 'Disable When Fired' options are 'Yes' (unselected) and 'No' (selected).

A recover alert takes a disabled alert and re-enables it. This is used for two alerts which show changing states, like a pair of alerts to show when availability goes down and then back up.

10. In the **Dampening** tab, give the dampening (or frequency) rule on how often to send notifications for the same alert event.

The screenshot shows the 'Alerts' configuration window with the 'Dampening' tab selected. The 'Dampening' dropdown is set to 'Last N Evaluations'. The 'Occurrences' spinner is set to 3, and the 'Evaluations' spinner is set to 5.

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:

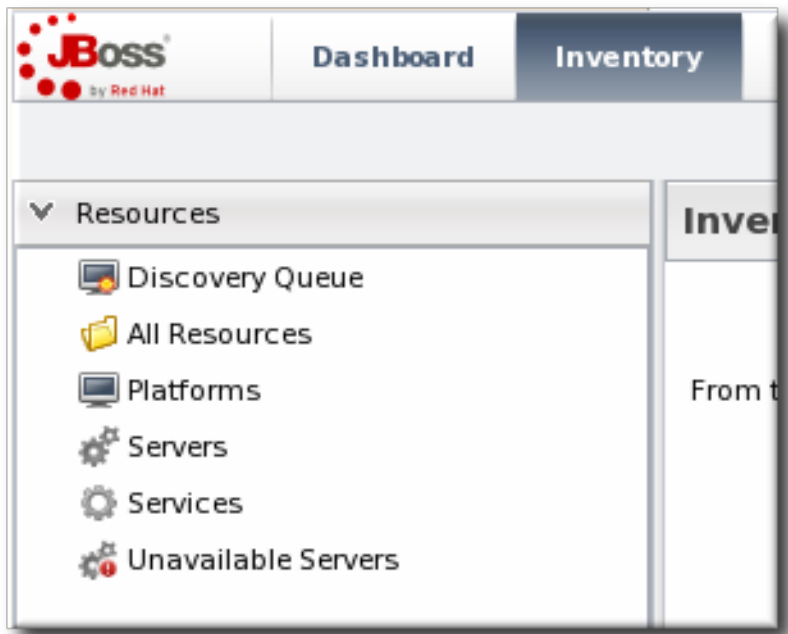
- *Consecutive*. Sends an alert if the condition occurs a certain number of times in a row for metric calculations. For example, if this is set to three, then the condition must be detected in three consecutive metric collection periods for the alert to be fired. If this is set to one, then it sends an alert every time the condition occurs.
- *Last N 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.
- *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.

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

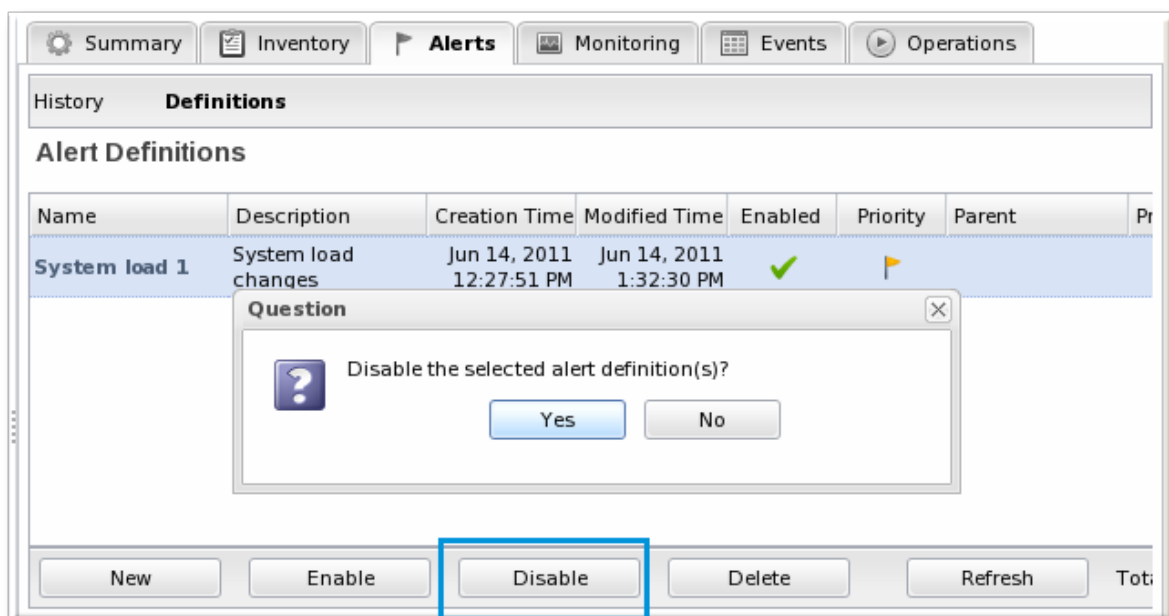
11.3. Enabling and Disabling Alert Definitions

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 known 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. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the **Alerts** tab.
4. In the **Definitions** subtab, select any of the definitions to enable or disable.
5. Click the **Enable** or **Disable** button.



6. Confirm the action.

11.4. 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.



NOTE

Alert templates for a resource type automatically apply to **all** existing and new resources of that type. This can be deselected when the template is edited, so that changes only apply to new resources.

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 enforce the group alert settings.

Templates make configuration really easy to apply consistently and often, and JBoss ON allows templates to be set for alerts based on their general resource type.

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.

11.4.1. 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 memory 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.

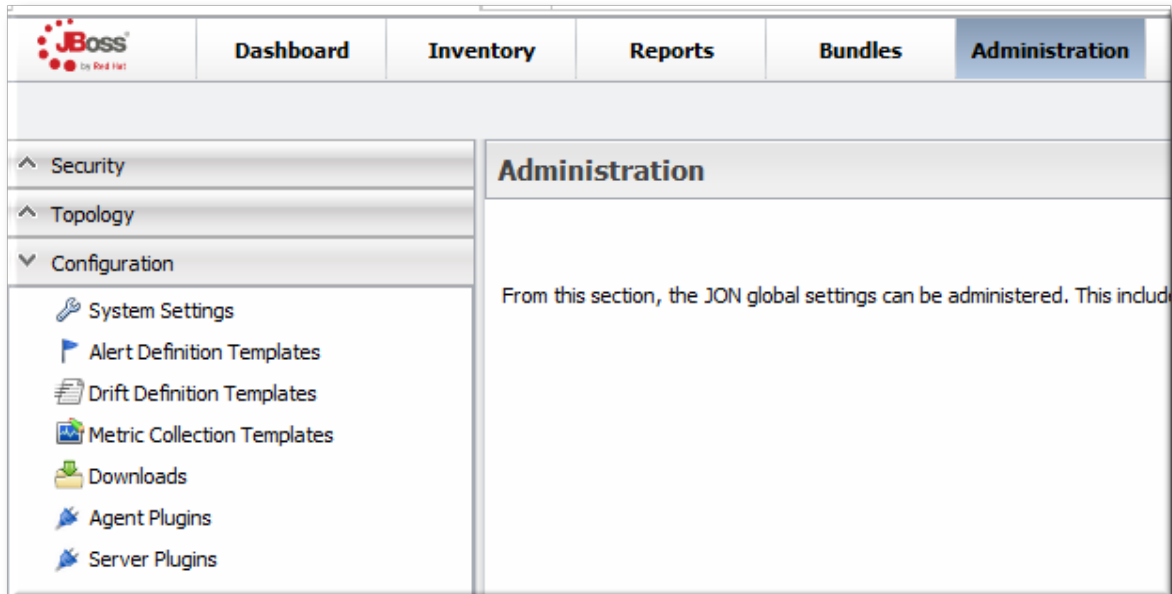


NOTE

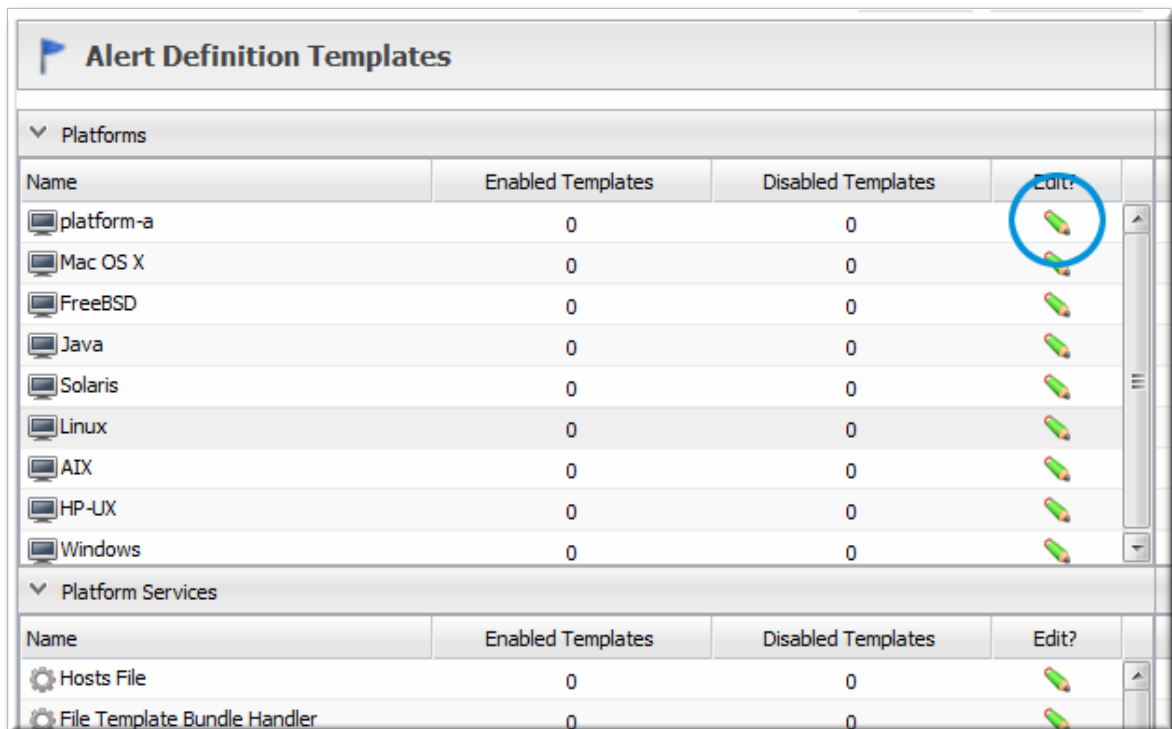
Alert templates for a resource type automatically apply to **all** existing and new resources of that type. This can be deselected when the template is edited, so that changes only apply to new resources.

To create an alert definition template:

1. In the top navigation, open the **Administration** menu, and then the **System Configuration** menu.



2. Select the **Alert 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. Click the **New** 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 [Section 11.2, "Basic Procedure for Setting Alerts for a Resource"](#)).

Name	Description	Creation Time	Modified Time	Enabled	Priority
Example Alert 1	For memory conditions.	Dec 1, 2011 3:19:49 PM	Dec 1, 2011 3:19:49 PM	✓	1

Buttons: New, Enable, Disable, Delete, Refresh

Total Rows: 1 (selected: 0)

5. Save the template.

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

11.4.2. Configuring Group Alerts

Group alerts can only be set on compatible groups.

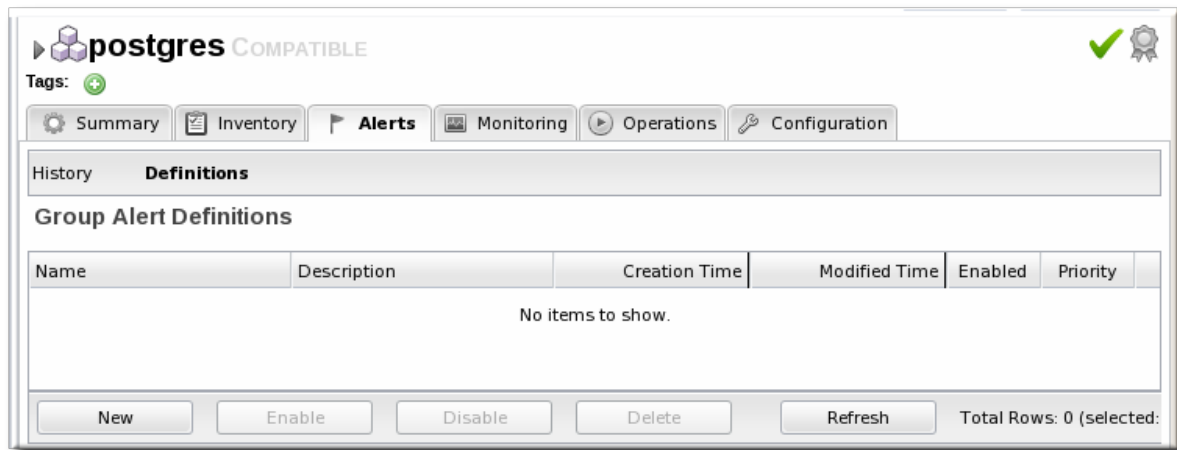
1. In the **Inventory** tab in the top menu, select the **Compatible Groups** item in the **Groups** menu on the left.

Name	Description	Type	Plugin	Children	Descendants
postgres		Postgres Server	Postgres	✓ 1	✓ 1

Buttons: Delete, New, Refresh

Total Rows: 1 (selected: 0)

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** button.

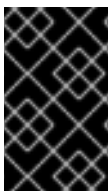


5. Configure the basic alert definition and notifications, as in [Section 11.2, “Basic Procedure for Setting Alerts for a Resource”](#).

12. ALERT CONDITIONS

Monitoring ([Section 4, “Metrics and Measurements”](#)) is closely associated to alerting, in a larger work flow of keeping administrators aware of what is happening in their network. Alerting is based on *conditions*, the signal that an alert should be issued.

Conditions can be pretty straightforward — if A happens, do B — or they can be complex, with multiple conditions or combinations of readings required before an alert is initiated.



IMPORTANT

Alert conditions cannot be edited after they are added to an alert definition. To change an alert condition, delete the original condition and create a new one with the desired settings.

12.1. Reasons for Firing an Alert

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. This can be a metric value that has changed to an undesirable level, an event, or a recurring metric reading.

Alerts through alert definitions against are defined for individual resources or for compatible groups of resources. An alert definition specifies the conditions that trigger the alert and the type and settings of any notification that should be triggered.

When an alert is registered, the alert identifies the alert definition which was triggered (which identifies the alert condition) and the metric or event value which precipitated the 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 single condition can be enough to issue an alert, or an alert definition can require that an alert is issued only if multiple conditions are met simultaneously. This provides very granular control over when an alert is issued, which makes alerting information more valuable and relevant.

A condition can be based on any detectable monitoring or system metric, listed in [Table 6, “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*.

Table 6. Types of Alert Conditions

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.
Operation	A specific action or task that is performed on the resource.
Events	A certain type of error message, matching a given string, is recorded. Events are filtered from system or application log files, and the types of events recognized in JBoss ON depend on the event configuration for the resource.
Drift	A resource has changed from a predefined configuration.

12.2. Detailed Discussion: Ranges, AND, and OR Operators with Conditions

Alerting is based on monitoring information. It is an extension that allows an administrator to receive a notification or define an action to take if a certain event or metrics value occurs.

The monitoring point that triggers an alert is the alert condition. At its most simplistic, an alert condition is a single event or reading. If X occurs, then that triggers an alert.

In real life, X may not be enough to warrant an alert or to adequately describe the state of a resource. Different conditions may require the same response or a situation may only be critical if multiple conditions are true. Alerting is very flexible because it allows multiple conditions to be defined with established relationships between those conditions.

The next level of complexity is to send an alert if either X *or* Y is true. In the alert definition, this is the *ANY* option, which is a logical OR. The alert definition checks for any of those conditions, but those conditions are still unrelated to each other.

The last level of complexity is when the conditions have to relate to each other for an alert to be issued. This is the *ALL* option, which is a logical AND. Both X and Y must occur for the alert to be issued. In this case, when one condition occurs, the server puts a lock on that definition and begins waiting for the

second condition to occur. When the second condition occurs, then the alert is issued.

An AND operator is very effective on different metrics, but because the conditions do not have to occur simultaneously, using a simple AND operator does not make sense for the *same* metric.

For example, Tim the IT Guy only wants an alert to be issued when the user load is between 40% to 60%, indicating slightly increased loads on his platform. Attempting to use an AND operator returns strange values when the load spikes over 70% (which trips the above 40% condition) and then falls back to 15% (which triggers the below 60% condition).

In this case, Tim uses a *range* condition. A range requires two values from the same metric that are within the given boundaries. A range can be inside values (40-60%) or it can be an outside range (below 40% and above 60%).

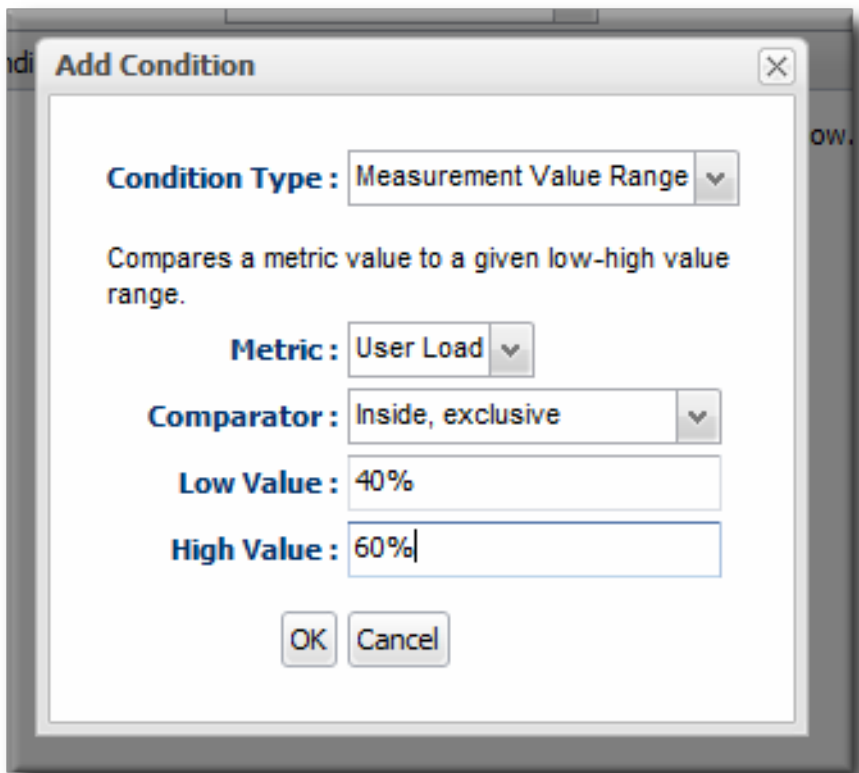


Figure 25. Alert Condition Range

12.3. Detailed Discussion: Conditions Based on Log File Messages

Events (Section 5, "Events") are filtered log messages. Certain resource in JBoss ON maintain their own error logs, like platforms and JBoss EAP servers. JBoss ON can scan these error logs to detect events of certain severity or events matching certain patterns. This allows JBoss ON administrators to provide an easy way to identify and view important error messages.

Because JBoss ON can detect log events, JBoss ON can alert on log events. An event-based condition requires the severity of the log file message and, optionally, a pattern to use to match specific messages.

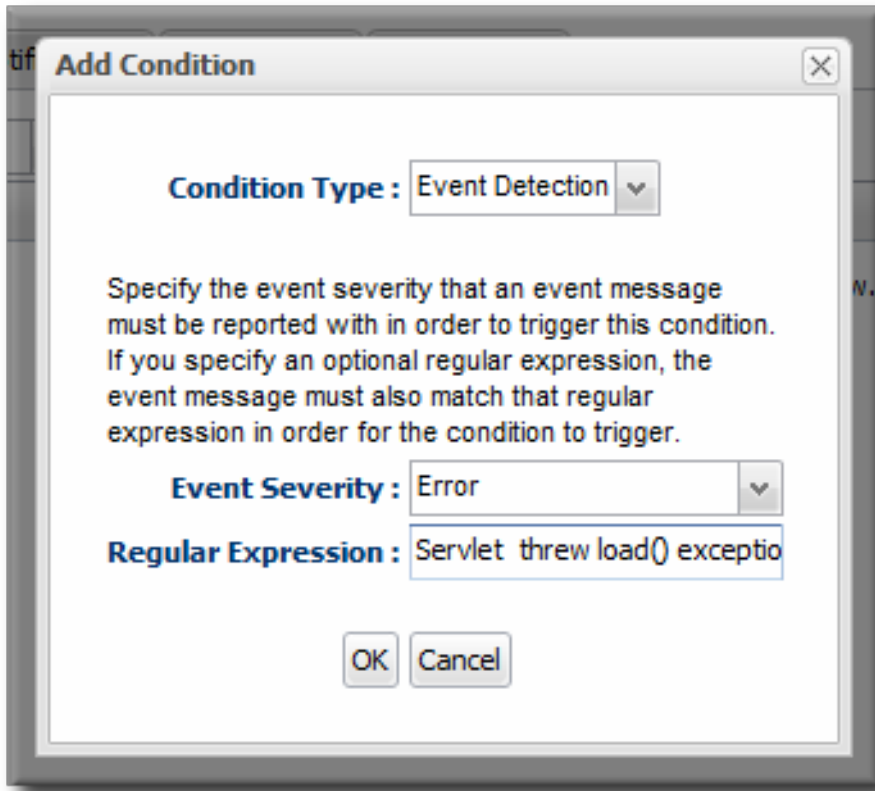


Figure 26. Log File Conditions

Setting only a severity alerts on any event with that severity. That is useful for severe or fatal errors, which are relatively infrequent and need immediate attention.



NOTE

For general error message, use a pattern to filter for a specific error type. Then, use a resource operation or CLI script to take a specific action to address that specific error, like restarting a resource or starting a new web app.

12.4. Detailed Discussion: Dampening

Dampening does not define an alert condition, but it tells the JBoss ON server how to handle recurring conditions.

An alert can be issued every single time a condition is met, or an alert can be issued and then disabled until an administrator acknowledges it. *Dampening* a condition is useful to prevent multiple alerts and notifications from being sent for a single ongoing set of circumstances.

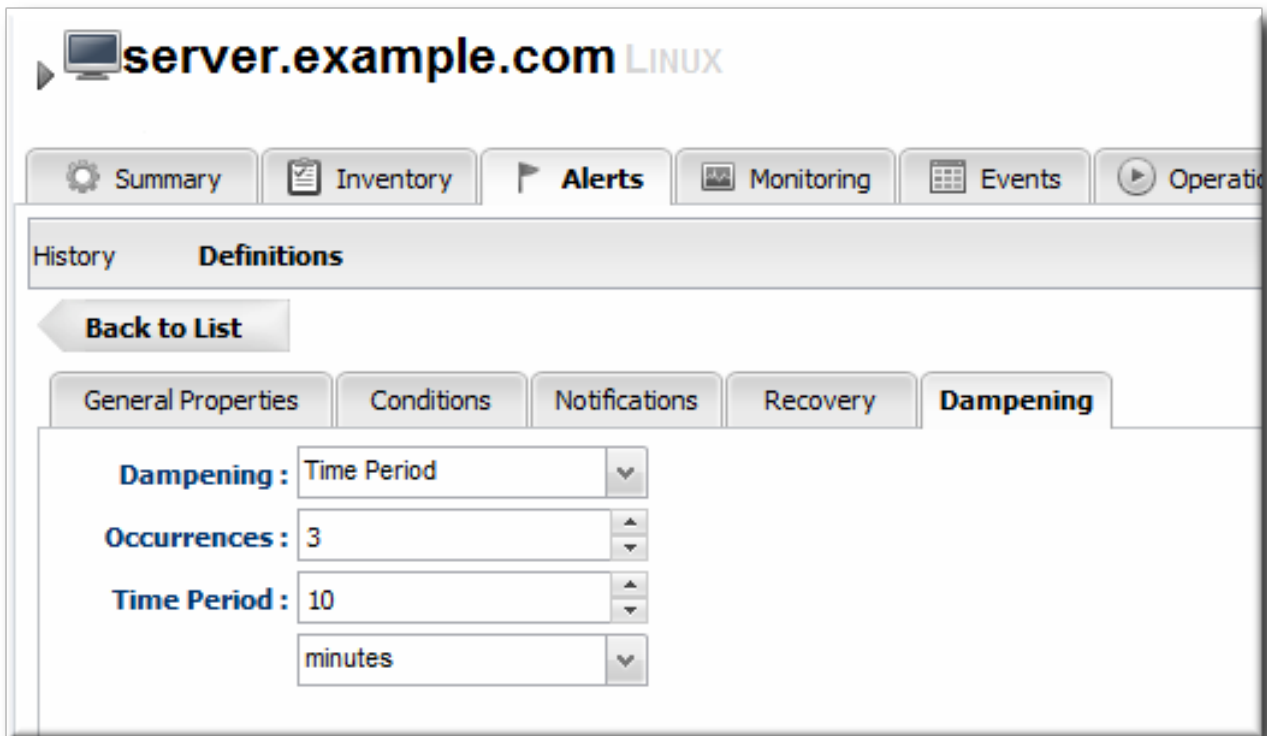


Figure 27. Dampening Filter

For example, Tim the IT Guy sets an alert to let him know if his platform CPU usage spikes above 80%. Every time the monitoring scan runs, the metric is again determined to be true, and is treated as a new and discrete instance. If the metric schedule is set for 10 minutes and it takes him an hour to respond to the alert, he could have six or seven alert notifications before he has a chance to respond. If the only alert response is an email, this is an annoyance, but not a problem.

If, however, Tim the IT Guy has an alert response to create a new JBoss server if his EAP connection count goes above a certain point, the same response could be taken six or seven times, when it actually should be done once and then the condition will naturally resolve itself.

Dampening is another set of instructions on how to evaluate the condition before triggering another alert. It 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.

A condition may need to occur several times over a short period of time for it to be a problem, but once is not a problem. For example, 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 may only be relevant if the CPU hits 80% and stays, and the other readings can be ignored.

- 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.



NOTE

Dampening is only relevant for metrics which are variable and compared to some kind of baseline. Monitoring metrics with thresholds and value changes are dynamic, so each reading really is a new condition, even if it matches the previous reading. Dampening, then, controls those multiple similar readings.

Conditions which relate directly to a status change do not compare themselves against a baseline — they only compare against a previous state. For example, if the system configuration changes from the drift template or if the availability changes, that is a one-time change. After that, the resource has a new status and future changes would be compared against the new status — so, in a sense, it is a different condition.

Dampening does not apply conceptually for drift and availability changes.

12.5. Detailed Discussion: Automatically Disabling and Recovering Alerts

There are a couple of different ways to limit how often an alert notification is sent for the same observed condition. One method is a dampening rule. An alternative is to disable an alert the first time it is fired, and then only re-enabling the alert when an administrator does it manually *or* if the condition resets itself. This second option — disabling and then resetting itself — recovers the alert.

A *recover alert* is actually a pair of alerts which work in tandem to disable and enable relevant alerts as conditions change.

A couple of workflows are common with recover alerts:

- A pair of alerts work as mutual toggle switches. When one alert is active, the other is disabled. When Alert A is fired, it can be set to recover a specified Alert B — so Alert B essentially takes its place.
- Alerts work as a kind of cascade. If Alert A is fired, that enables Alert B, which then enables Alert C. In some situations, any one given condition may not be a problem, but it becomes a problem if they occur sequentially in a short amount of time.

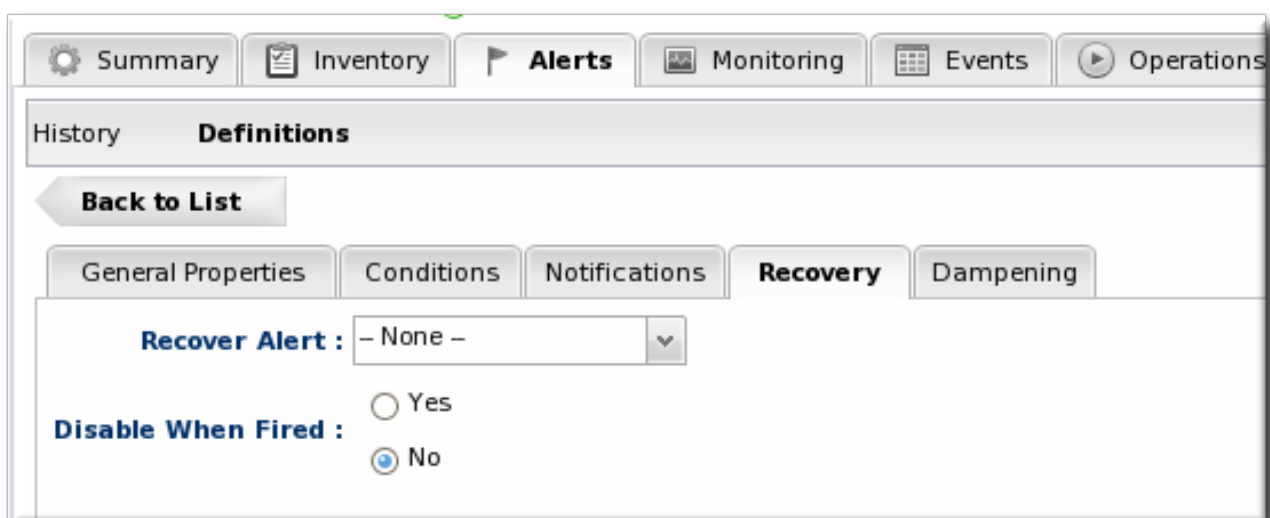


Figure 28. Disable and Recover Alerts

For example, Alert A triggers an alert when a resource availability goes down. When Alert A fires, it disables itself and recovers (or enables) Alert B. Alert B fires an alert when the resource's availability goes up. When Alert B fires, it likewise disables itself and recovers Alert A.

Recover alerts inform an administrator first of when an issue occurs and then second when it is resolved. In the availability examples, the first alert lets the administrator know that a resource is offline, while the second alert lets the administrator know that the resource is back online.

The Setup: Toggle Recover Alerts for Availability

Tim the IT Guy has several servers that he uses for email routing and other business operations, and then he has a couple of machines that he holds in reserve as backups.

He has **mail-server-a.example.com** has his primary mail server, and he only wants to bring **mail-server-b.example.com** online if **mail-server-a** goes offline, and then he wants it to go back in reserve when **mail-server-a** comes back.

The Plan

Tim creates a set of alert definitions to help handle the transition between his mail servers.

- The first alert definition fires when the **mail-server-a** platform changes availability state to **goes down**.

The notification does a couple of things:

- Deploy a bundle with the latest mail server configuration to another platform, **mail-server-b**.
- Execute a command-line script on **mail-server-b** to start the mail service.
- Email Tim the IT Guy to let him know that **mail-server-a** is unavailable.

For recovery, the alert does two things:

- Disable the current alert. It only needs to fire once, to get the backup server online.
 - Recover (or enable) Alert B, so that JBoss ON waits for **mail-server-a** to come back up.
- The second alert definition, Alert B, is only in effect *while mail-server-a* is offline. This alert fires as soon as **mail-server-b** changes availability state to **goes up**.
 - This alert definition basically waits around as long as **mail-server-a** is down. When **mail-server-a** is back online, Alert B's notification is to execute a command-line script on **mail-server-b** to stop the mail service.
 - Alert B also sends a notification email to Tim the IT Guy to let him know that **mail-server-a** is available again.

For recovery, the alert does two things:

- Disable the current alert. Like with Alert A, Alert B only needs to fire once, to shut off the backup as soon as the primary server is back.
- Recover (or enable) Alert A, so the JBoss ON waits again for **mail-server-a** to go down.

13. ALERT RESPONSES

Alert conditions define *when* an alert is supposed to be triggered. An alert notification defines *how* that alert notification is communicated. An alert notification can be a way of informing administrators and users of the condition, but it can also be a way of having JBoss ON itself address an issue.

A single alert can have multiple responses, to make managing potential problems easier.



IMPORTANT

Alert notifications cannot be edited after they are added to an alert definition. To change an alert notification, delete the original notification and create a new one with the desired settings.

13.1. Notifying Administrators and Responding to Alerts

Every alert is recorded and viewable in the JBoss ON GUI. Alerts have an optional configuration, though, of sending an external notification whenever the alert is issued.

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*. Alert notifications fall into two categories: a way of communicating the alert and an action to take in response to an alert.

There are three methods, by default, of communicating that an alert has been fired:

- Email, to one or multiple addresses
- SNMP traps
- Messages to JBoss ON users

There are also a few default methods of having JBoss ON respond to an alert:

- Running a resource operation (on the alerting resource or any other resource in inventory)
- Running a resource script (specific type of resource operation)
- JBoss ON CLI scripts



NOTE

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*.

Alert notifications can be *clustered*. That just means that the same alert 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 to have the web server restarted at the same time.



NOTE

A single alert can initiate multiple notifications. Alert notifications are run in the order they are listed in the alert definition.

13.2. Detailed Discussion: Initiating an Operation

A parallel response to an alert is to launch an *operation*. Resource operations (which, like metrics, are defined in the resource type agent plug-in) are launched, like a notification, in response to a triggered alert. Alert operations can be run on the resource that issued the alert or on any other resource in the

inventory, which allows immediate and automatic responses to alert conditions. For instance, 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.

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 operations 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.

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.

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.

13.2.1. 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.

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%>`, while 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 7, “Available Alert Operation Tokens”](#). All of these potential token values are Java properties that belong to the operation's parent JBoss ON server.

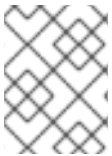
The alert operations plug-in resolves the token value itself when the alert operation is processed to find the value. The realized value is sent to the script service, which ultimately plugs the value into the command-line argument or script which referenced the token.

Table 7. Available Alert Operation Tokens

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
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

Information about ...	Token	Description
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

13.2.2. Setting Alert Operations



NOTE

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.

The **Notifications** tab has a **Resource Operations** method.

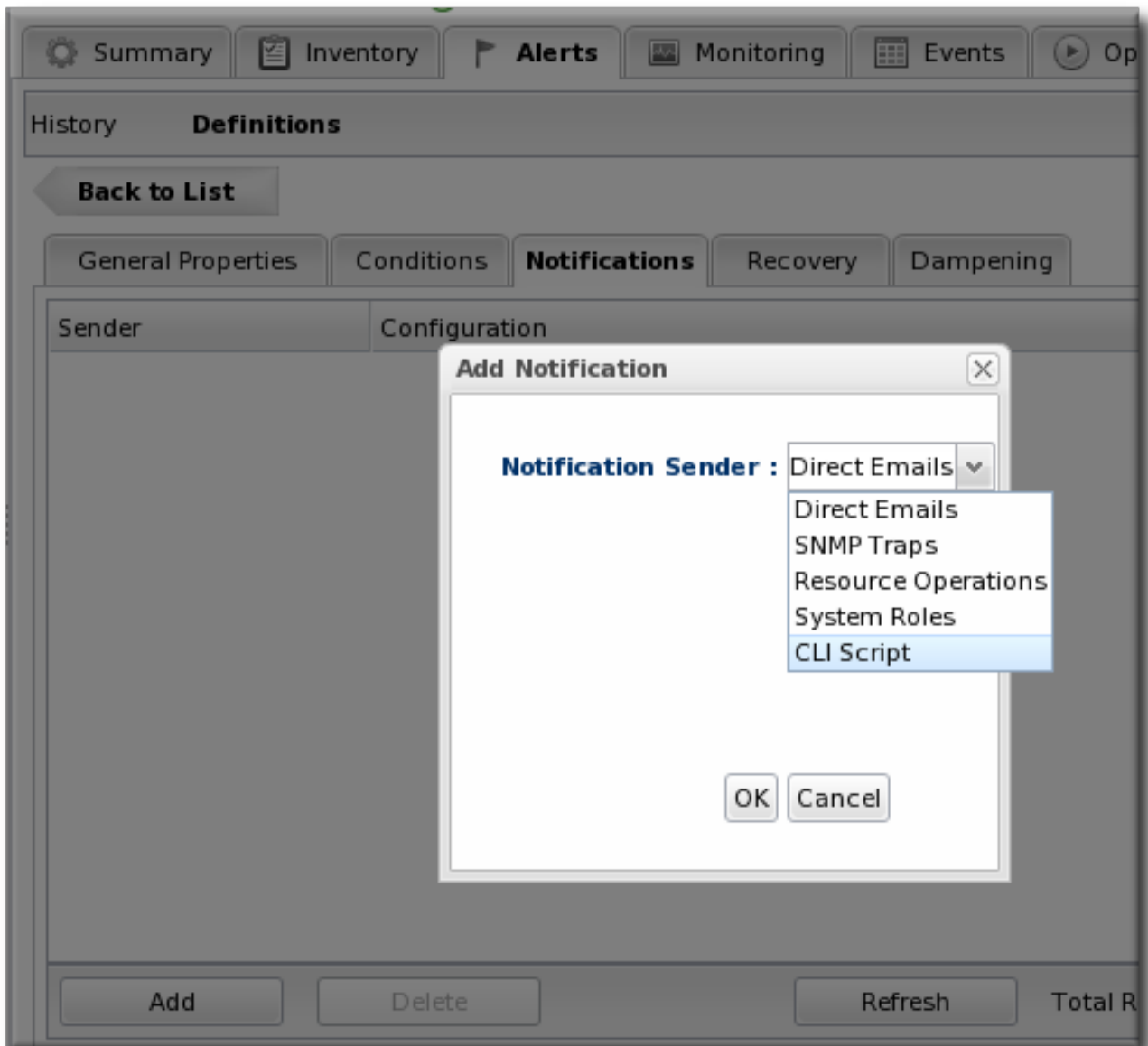


Figure 29. Senders

There are two parts to the operation. First is selecting which resource the operation will run on. The resource type determines what operations are available.

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.

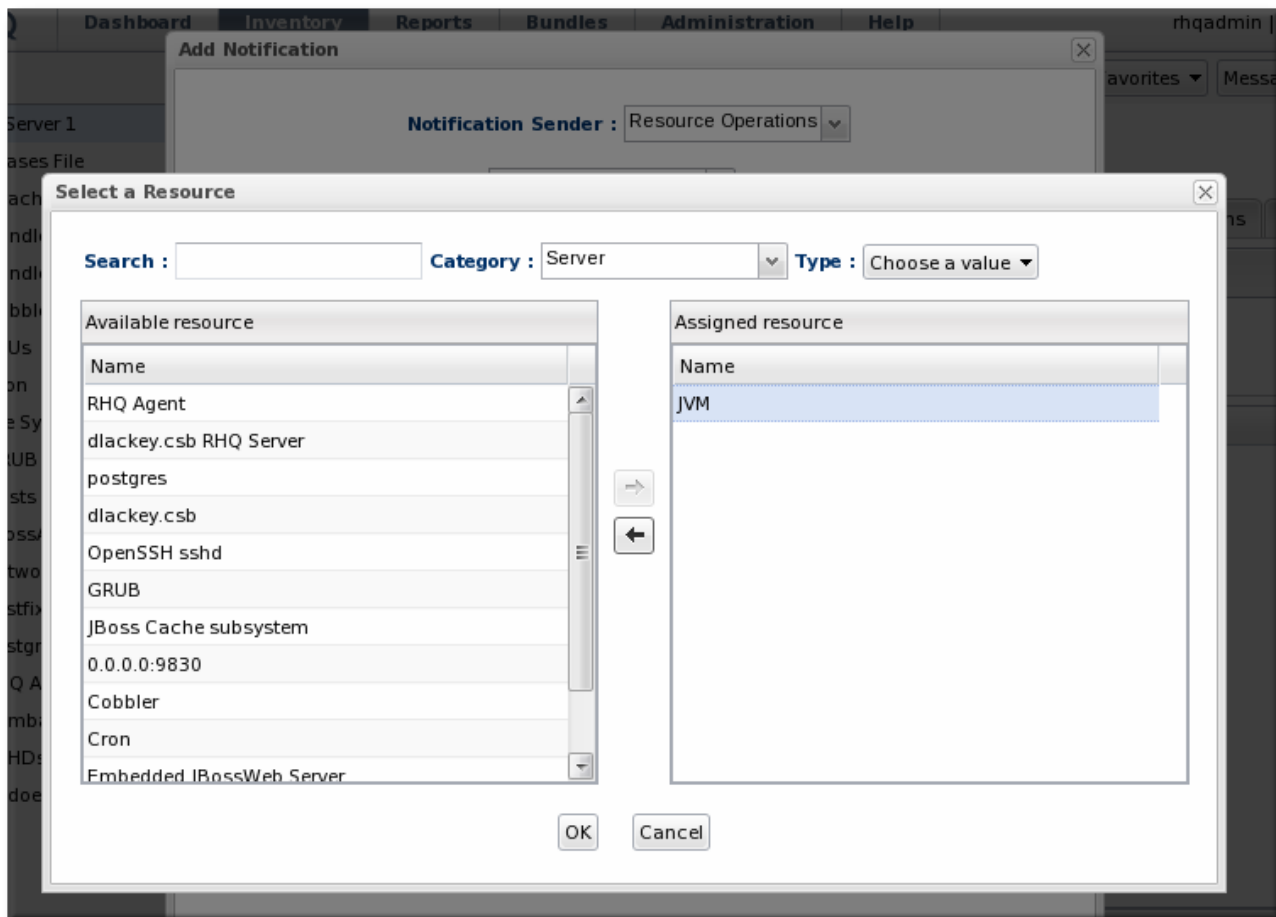
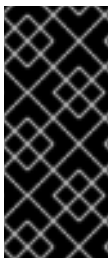


Figure 30. Resource Selection



IMPORTANT

If you select a **relative** resource and *do not* enter a specific resource name, then the operation will run on *every* resource which matches that resource type in the relative path. If no resource matches, then it is logged into the audit trail and the alert process proceeds.

For a relative resource, the resource name is not required. For a specific resource, it is.

The second half is selecting the operation type. The available operations and their configuration parameters depend on the type of resource selected as the target of the operation.

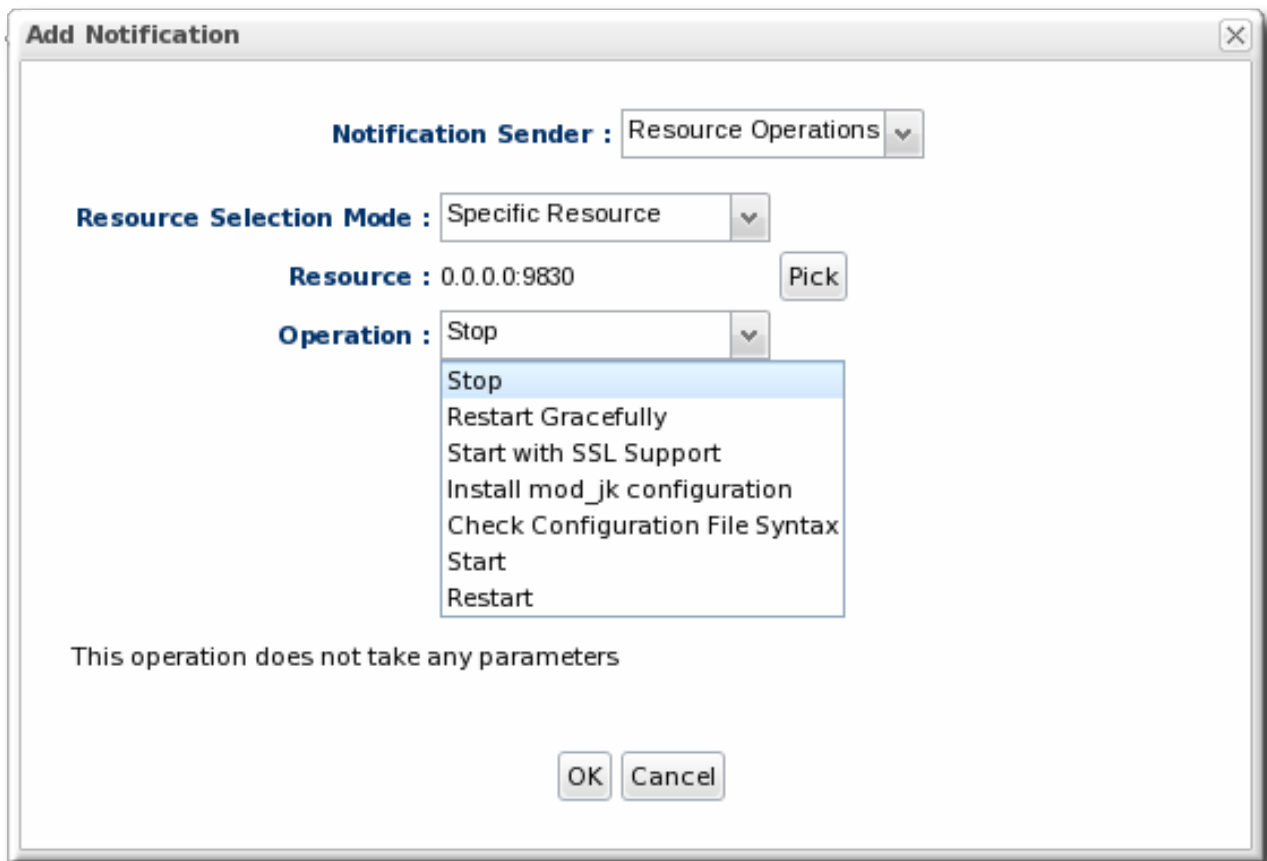


Figure 31. Operation Settings

13.3. *Detailed Discussion*: Initiating Resource Scripts

Scripts, such as shell and bat scripts, can be imported into the JBoss ON inventory and managed as resources. These scripts can include start scripts, configuration scripts, or diagnostic scripts.

As described in [Section 13.2.2, “Setting Alert Operations”](#), an alert notification can run an operation on a resource (even a different resource than the one which triggered the alert). A specific usage scenario is to run a resource script as an operation.

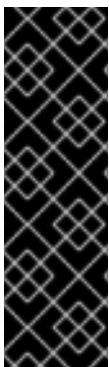


NOTE

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

Running a resource script is the same as running an operation. The resource script is selected as the resource for the operation, and then the start or restart operation for the script is set and, optionally, any command-line arguments to pass to the script.

Figure 32. Resource Script Settings



IMPORTANT

If you select a **relative** resource and *do not* enter a specific script name in the name filter field, then the operation will run on *every* script resource that is in the relative path with the command arguments that are given. If no script matches, then it is logged into the audit trail and the alert process proceeds.

For a relative resource, the resource name is not required. For a specific resource, it is. To limit script execution to a single specific script, select the specific resource option and select the precise script from the list.

13.4. Detailed Discussion: Launching JBoss ON CLI Scripts from an Alert

JBoss ON has its own command-line client that can be used to manage server instances in the same way that the web UI manages servers. Much like running a script resource or launching an operation in response to an alert condition, a server CLI script can be run in response to an alert condition.



NOTE

Unlike resource scripts, CLI scripts are not treated as resources in the inventory. These are tools available to and used by the JBoss ON server itself (not limited or associated with any given resource).

Server CLI scripts must be uploaded to the server as content within a repository before it can be run.

The CLI script must use the proper API to perform the operation on the server. JBoss ON has several different API sets, depending on the task being performed. To connect to a server and run a script requires the [remoting API](#), which allows commands to be executed on the server remotely. Writing CLI scripts is covered more in [Running JBoss ON Command-Line Scripts](#).



NOTE

The CLI script can actually reference an alert object for the alert which triggers the script by using a pre-defined `alert` variable.

The `alert` variable implicitly identifies the alert definition and specific alert instance which has been fired. This allows you to create a proxy resource definition in the script that could be applied to any resource which uses that alert script.

```
var myResource =
ProxyFactory.getResource(alert.alertDefinition.resource.id)
```

The resource ID is pulled from the fired alert, and the script can reference the resource from there.

Example 1. Writing Alert-Relevant CLI Scripts

Server-side scripts are both powerful and versatile. They can touch almost any server functionality and any resource, group, or other object and run a series of commands. This versatility makes CLI scripts very useful to responding to alerts in proactive ways, but it means that the alert should be planned to notify on specific conditions and the script should be designed to respond to that specific set of circumstances.

In other words, make the alert specific and the CLI script relevant.

This script checks the recent monitoring statistics for a web application and restarts the web server database if there are connection problems:

```
var myResource =
ProxyFactory.getResource(alert.alertDefinition.resource.id)

var definitionCriteria = new MeasurementDefinitionCriteria()
definitionCriteria.addFilterDisplayName('Sessions created per Minute')
definitionCriteria.addFilterResourceTypeId(myResource.resourceTypeId)

var definitions =
MeasurementDefinitionManager.findMeasurementDefinitionsByCriteria(definitionCriteria)

if (definitions.empty) {
    throw new java.lang.Exception("Could not get 'Sessions created per Minute' metric on resource "
        + myResource.id)
}

var definition = definitions.get(0)

var startDate = new Date() - 8 * 3600 * 1000 //8 hrs in milliseconds
var endDate = new Date()

var data = MeasurementDataManager.findDataForResource(myResource.id, [
definition.id ], startDate, endDate, 60)

exporter.setTarget('csv', '/the/output/folder/for/my/metrics/' + endDate
+ '.csv')
```

```

exporter.write(data.get(0))

var dataSource = ProxyFactory.getResource(10411)

connectionTest = dataSource.testConnection()

if (connectionTest == null || connectionTest.get('result').booleanValue
== false) {
    //ok, this means we had problems connecting to the database
    //let's suppose there's an executable bash script somewhere on the
server that
    //the admins use to restart the database

java.lang.Runtime.getRuntime().exec('/somewhere/on/the/server/restart-
database.sh')
}

```

Commands, options, and variables for the JBoss ON CLI are listed in [Running JBoss ON Command-Line Scripts](#).

An example alert script is included with the server files in `serverInstallDir/alert-scripts/`.

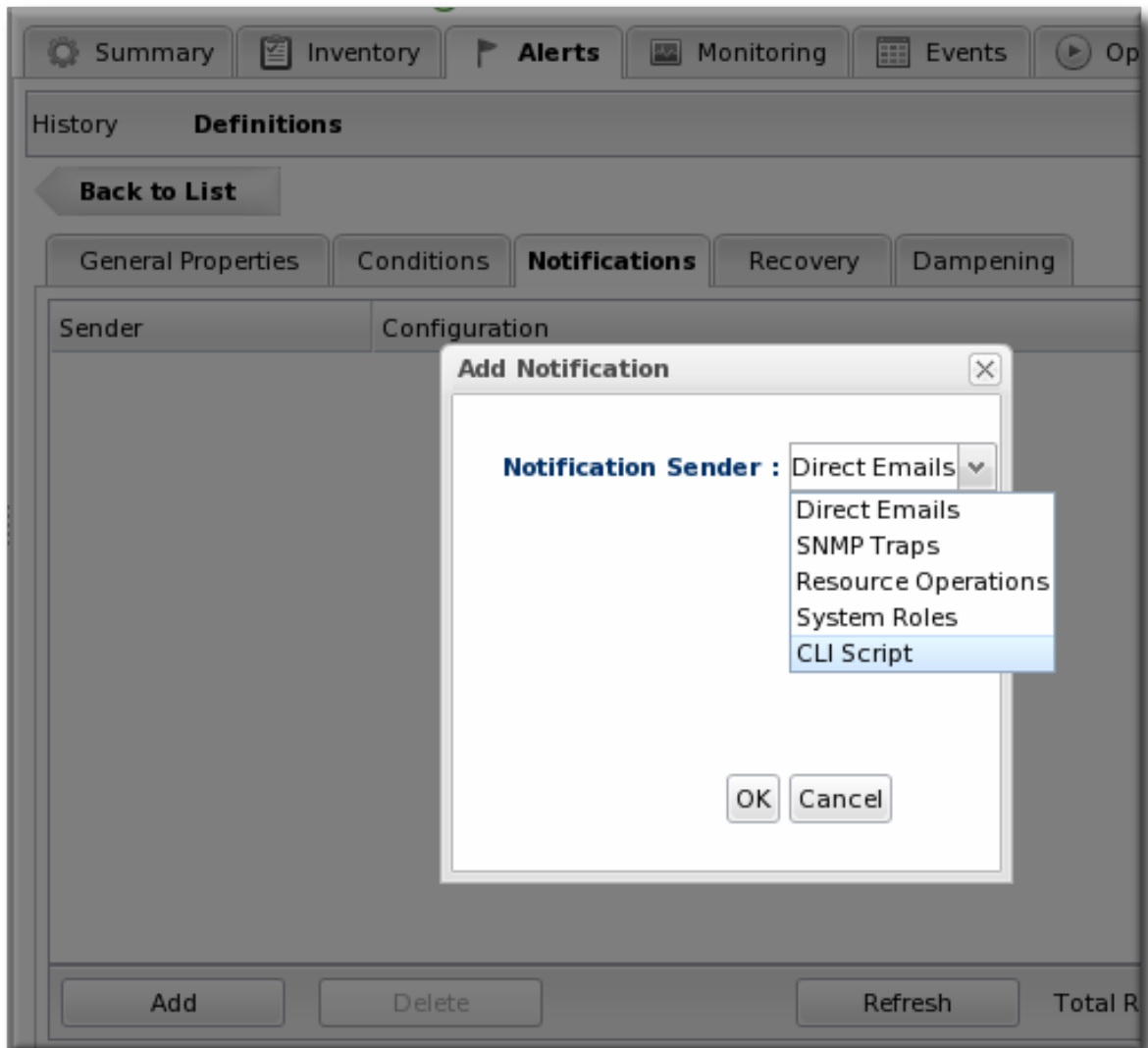
1. Upload the script to a content repository.



NOTE

Create a separate repository for alert CLI scripts.

2. Search for the resource, and configure the basic alert definition, as in [Section 11.2, “Basic Procedure for Setting Alerts for a Resource”](#).
3. In the **Notifications** tab for the alert definition, give the notification method a name, and select the **CLI Script** method from the **Alert Senders** drop-down menu.



4. First, select the JBoss ON user as whom to run the script. The default is as the user who is creating the notification.

Add Notification

Notification Sender: CLI Script

User To Run The Script As

Myself

Another User

User Name : rhqadmin Password : ●●●●●●

Verify

Repository

Select the repository where the script should reside :

cli-scripts

Script

Existing Script

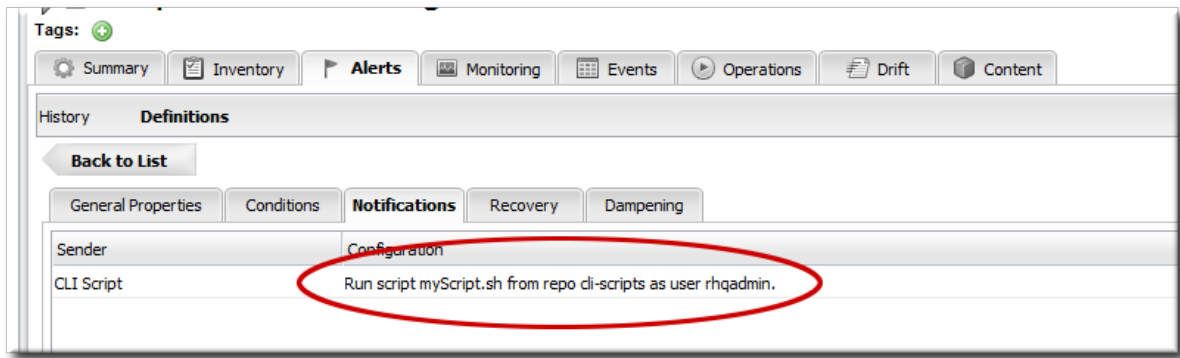
Upload New Script

File : /home/jsmith/dev/scripts/ Browse...

Version : 1.0

OK Cancel

5. Select the repository which contains the CLI script. If you are uploading a new script, this is the repository to which the script will be added.
6. Select the CLI script to use from the drop-down menu, which lists all of the scripts in the specified repository. Alternatively, click the **Upload** button to browse to a script on the local machine.
7. Click **OK** to save the notification. The line in the **Notifications** tab shows the script, the repository, and the user as whom it will run.



13.5. Configuring SNMP for Notifications

Configuring JBoss ON to send SNMP alerts has two parts:

- Configuring the SNMP alert plug-in for the server.
- Configuring the actual alert with an SNMP notification.

13.5.1. JBoss ON SNMP Information

JBoss ON can send SNMP traps to other management stations and systems as part of alerting notifications. 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 2](#), “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)**.

Example 2. Default Alert Object in JBoss ON MIB

```

alertGroup OBJECT-GROUP
  OBJECTS {
    alertName,
    alertResourceName,
    alertPlatformName,
    alertCondition,
    alertSeverity,
    alertUrl }
  STATUS current
  DESCRIPTION "A collection of objects providing information about an
  alert"

```

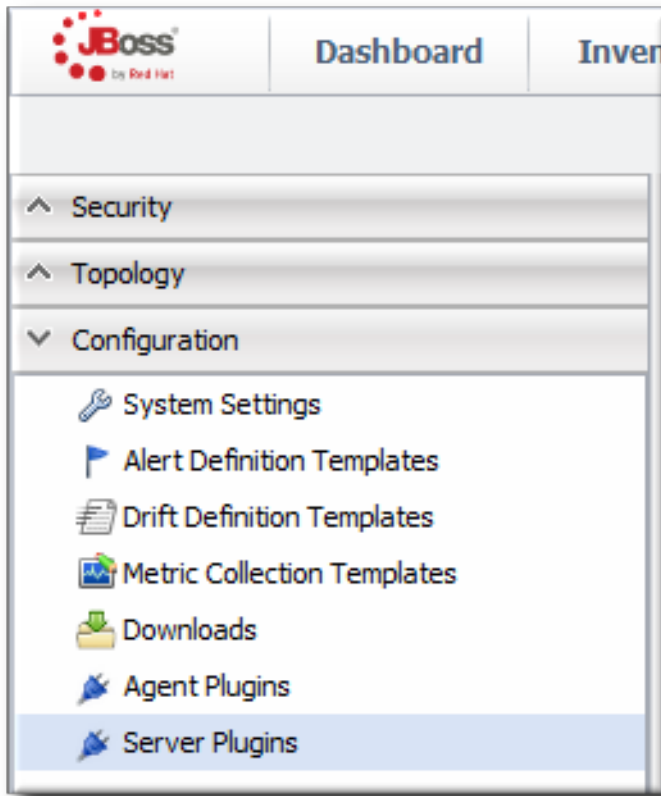
With the default MIB file, each trap sends the alert definition name, resource name, platform, alert conditions, severity, and a URL to the alert details page.

13.5.2. Configuring the SNMP Alert Plug-in

The SNMP alert sender plug-in is the only alert notification plug-in that requires additional configuration before the notification method can be used. The SNMP plug-in has to be configured with the appropriate SNMP version and SNMP agent information.

1. In the top menu, select the **Administration** tab.

- In the **System Configuration** menu, select the **ServerPlugins** item.



- Click the name of the SNMP plug-in in the list.

Name ^	Description	Last Updated	Enabled?	Deployed?
Alert:CLI	Alert sender plugin that can execute a CLI script.	May 18, 2012 3:34:40 PM	✓	✓
Alert:Email	Alert sender plugin that sends alert notifications via email	May 18, 2012 3:34:40 PM	✓	✓
Alert:IRC	Alert sender plugin that sends alert notifications via IRC	May 2, 2012 1:21:48 AM	!	✓
Alert:Microblog	Alert sender plugin that sends alert notifications via Microblog services (twitter, laconi.ca, ..)	May 2, 2012 1:21:48 AM	!	!
Alert:Mobicents	Alert sender plugin that sends alert notifications via Mobicents (voice, sms, ..)	May 2, 2012 1:21:48 AM	✓	✓
Alert:Operations	Alert sender plugin that can run operations on resources in RHQ inventory.	May 18, 2012 3:34:40 PM	✓	✓
Alert:Roles	Alert sender plugin that sends alert notifications to RHQ roles	May 18, 2012 3:34:40 PM	✓	✓
Alert:SNMP	Alert sender plugin that sends alert notifications via SNMP traps	May 18, 2012 3:34:40 PM	✓	✓
Alert:Subject	Alert sender plugin that sends alert notifications to RHQ subjects	Jun 4, 2012 7:49:36 PM	✓	✓
Ant Bundle Processor	Processes bundles whose recipes are Ant scripts	May 18, 2012 3:34:40 PM	✓	✓
Disk Content	Provides the ability to obtain content from a local file system	May 18, 2012 3:34:40 PM	✓	✓

Below the table, there are several control buttons: Scan For Updates, Restart Master Plugin Container, Hide Undeployed, Upload Plugin: [input field] Browse..., Upload (?), Enable, Disable, Undeploy, Purge, Refresh, and Total Rows: 18 (selected).

- In the plug-in details page, expand the **Plugin Configuration** section.

Plugin: Alert:SNMP (alert-snmp)

Version: 4.0.0-SNAPSHOT	AMPS Version:
Enabled: true	Type: SERVER
MD5: 8688ebff13a89b4f7d59ca8f8ec205a4	Description: Alert sender plugin that sends alert notifications via SNMP traps
Path: alert-snmp-4.0.0-SNAPSHOT.jar	

[Configure 'Alert:SNMP'](#)

Plugin Help

Used to send notifications to SNMP trap receivers.

5. All SNMP versions require information about the JBoss ON MIB OID and selected version. Fill in the appropriate values.

[Back to List](#)

Details

Display Name: Alert:SNMP	Name: alert-snmp
Version: 4.4.0.JON310ER4	AMPS Version:
MD5: b8d64d2ddd9a3e48b3c679c49f9fa220	Kind: Server
Description: Alert sender plugin that sends alert notifications via SNMP traps	Path: alert-snmp-4.4.0.JON310ER4.jar
Last Updated: May 18, 2012 3:34:40 PM	Enabled?: <input checked="" type="checkbox"/>
Type: org.rhq.enterprise.server.xmlschema.generated.serverplugin.alert.AlertPluginDescriptorType	

Help

Plugin Configuration

Save Reset

Jump to Section

General Properties

Property	Unset?	Value	Description
SNMP protocol version	<input type="checkbox"/>	<input checked="" type="radio"/> 1 <input type="radio"/> 2c <input type="radio"/> 3	
Trap OID	<input type="checkbox"/>	<input type="text"/>	OID for the trap sent
Community	<input type="checkbox"/>	<input type="text" value="public"/>	Community - v1 and v2c only
SNMP version 1 properties			
SNMP version 3 properties			

6. SNMP version 1 and version 3 both require additional configuration. Expand the version-specific configuration section and fill in the information about the SNMP agent.

Plugin Configuration

Save Reset

Jump to Section

SNMP version 1 properties

SNMP version 3 properties

Property	Unset?	Value	Description
Auth Protocol		<input type="radio"/> none <input checked="" type="radio"/> MD5 <input type="radio"/> SHA	
Privacy Protocol		<input type="radio"/> DES <input checked="" type="radio"/> AES <input type="radio"/> AES192 <input type="radio"/> AES256	
Target Context Name	<input checked="" type="checkbox"/>	<input type="text"/>	
Auth Passphrase	<input checked="" type="checkbox"/>	<input type="text"/>	Auto Passphrase is required with authorization enabled
Privacy Passphrase	<input checked="" type="checkbox"/>	<input type="text"/>	Privacy Passphrase is required with privacy enabled
Security Name	<input checked="" type="checkbox"/>	<input type="text"/>	

It may be necessary to unselect the **Unset** checkbox to allow the fields to be edited.

13.5.3. Configuring the SNMP Alert Notification

Before JBoss ON can send any SNMP notifications, SNMP traps have to be configured for the server.

When configuring the alert, select the SNMP Trap notification type, and fill in the JBoss ON SNMP information.

Add Notification

Notification Sender : SNMP Traps

Property	Unset?	Value	Description
Host		<input type="text" value="snmp.example.com"/>	Trap target host
Port	<input checked="" type="checkbox"/>	<input type="text"/>	Trap target port
Oid		<input type="text" value="1.3.6.1.4.1.18016.2.1"/>	OID of the trap sent

OK Cancel

Figure 33. JBoss ON SNMP Trap Information

- 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**.

14. VIEWING ALERT DATA

Once an alert notification is triggered, JBoss ON records it in an *alert history* for the resource. The history includes the time the alert was sent and the condition which triggered it. Crucially, JBoss ON also shows whether an alert has been acknowledged and, if so, what user addressed it.

While monitoring and alerting work together to make administrators immediately aware of a problem with a resource, the alert history gives a much broader view into resource performance *and* IT staff responses. Knowing, historically, what kind of alerts have been triggered and when can help with root cause analysis, incident response, and maintenance policies.

14.1. Viewing the Alert Definitions Report

While the alert definitions for a specific resource are always available by viewing that resource entry, it is also possible to view all of the alert definitions configured in JBoss ON in the Alert Definitions Report.

1. Select the **Reports** tab in the top navigation bar.
2. In the **Subsystems** menu box on the left, select **Alert Definitions**.
3. The definitions report shows a list of all configured definitions, for all resources in the inventory.

Name	Description	Enabled	Priority	Resource	Ancestry
System load 1	System load changes	✓	🚩	Linux Server 1	
postgres avail		✓	🚩	postgres	postgres < Linux Server 1
samba cxn probs		✓	🚩	dlackey.csb	Linux Server 1

The results table provide the most basic information for the definitions:

- The resource (**Name**).
- The parent or *ancestry*. 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 description of the alert.
- Whether it is active (enabled).

**NOTE**

A user may have the write to create and edit an alert definition, but that does not mean that the user has the right to delete an item from the alert history.

Deleting elements in the history requires the manage inventory permission.

**NOTE**

Reports can be exported to CSV, which can be used for office systems or further data manipulation.

To export a report, simply click the **Export** button. The report will automatically be downloaded as **alertDefinitions.csv**.

14.2. Viewing Alerts

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

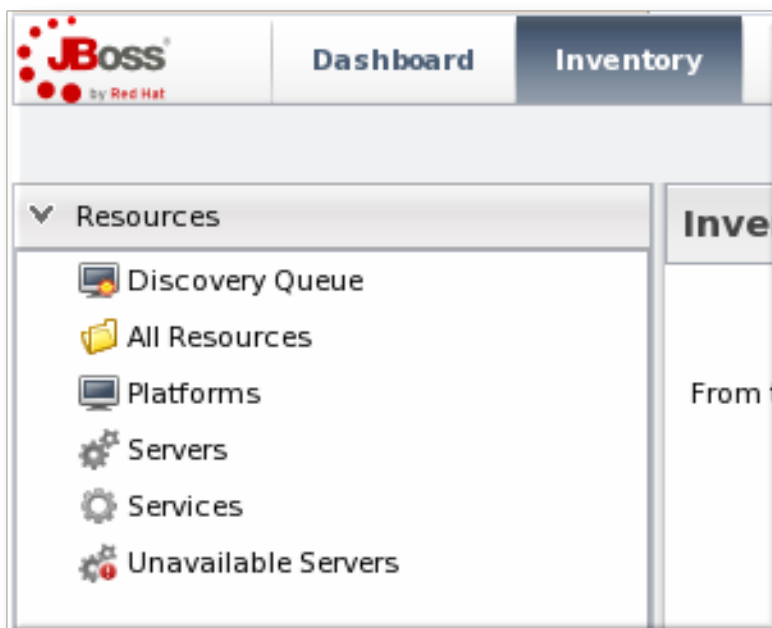
14.2.1. Viewing Alert Details for a Specific Resource

**NOTE**

A user may have the write to create and edit an alert definition, but that does not mean that the user has the right to delete an item from the alert history.

Deleting elements in the history requires the manage inventory permission.

1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



- Click the resource in the list.
- Click the **Alerts** tab, and make sure that the **History** subtab is selected.
- In the list, click the timestamp or alert definition name for the fired alert.

Creation Time	Name	Condition Text	Priority	Status
Jun 14, 2011 1:46:29 PM	avail	Availability Change [Came up]	MEDIUM	No Ack

- The alert page has tabs for each detail for the alert, including which alert definition was triggered, the conditions that triggered, and any operations that were launched as a result.

Condition	Value
Availability Change [Came up]	UP

14.2.2. Viewing the Fired Alerts Report

- Select the **Reports** tab in the top navigation bar.
- In the **Subsystems** menu box on the left, select **Recent Alerts**.

Creation Time	Name	Condition Text	Priority	Status	Resource	Ancestry
Jun 14, 2011 1:46:29 PM	avail	Availability Change [Came up]	MEDIUM	No Ack	0.0.0.0:80	Linux Server 1

All of the alerts for all resources in JBoss ON are listed in the results table. Several elements are useful for analysis:

- The resource (**Name**)

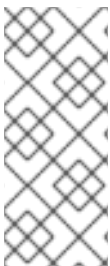
- The parent (ancestor)
- The name of the definition which triggered the alert
- The condition which triggered the alert
- The value of the resource at the time the alert was sent
- The date, which is very useful for correlating the alert notification to an external event



NOTE

A user may have the write to create and edit an alert definition, but that does not mean that the user has the right to delete an item from the alert history.

Deleting elements in the history requires the manage inventory permission.



NOTE

Reports can be exported to CSV, which can be used for office systems or further data manipulation.

To export a report, simply click the **Export** button. The report will automatically be downloaded as **recentAlerts.csv**.

14.2.3. Viewing Alerts in the Dashboard

All of the recently-fired alerts, by default, are listed on the **Dashboard** page of JBoss ON in the recent alerts portlet.

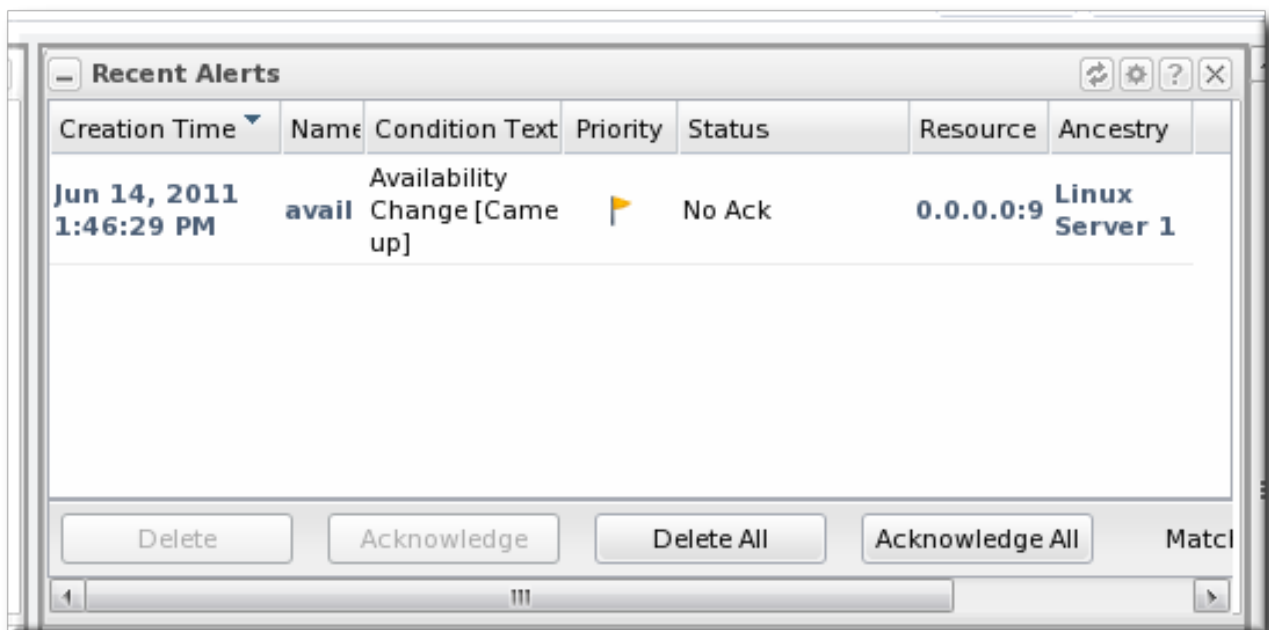


Figure 34. Recent Alerts Portlet

The alerts displayed in the portlet can be filtered for different conditions:

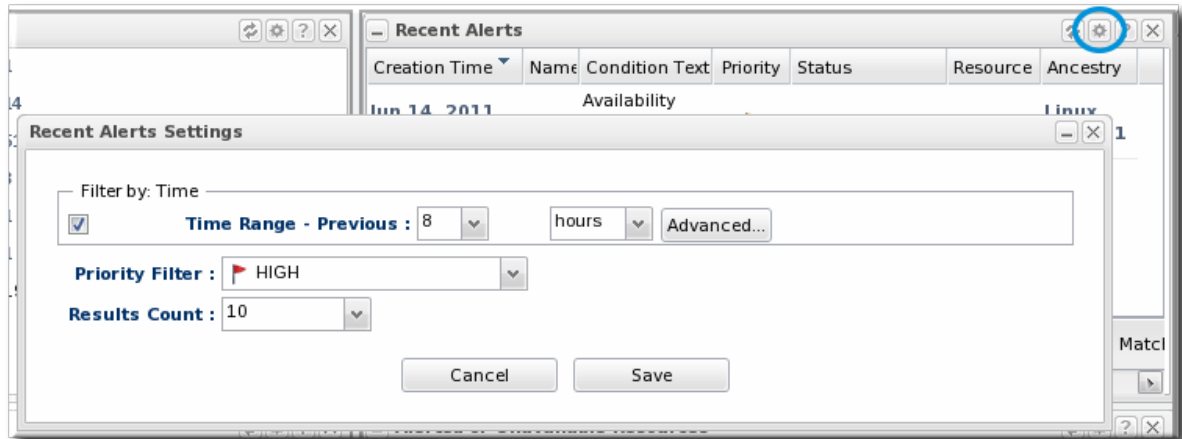
- A time range for when the alert was fired

- The alert priority (which is initially configured in the alert definition)

These conditions are evaluated *in order*, meaning that alerts are filtered first based on time, then priority.

To set the conditions for the alerts portlet in the **Dashboard** page:

1. In the top menu, click **Dashboard**.
2. In the **Recent Alerts** portlet, click the gear icon to open the portlet configuration page.



3. Change the display criteria as desired.

14.3. Acknowledging an Alert

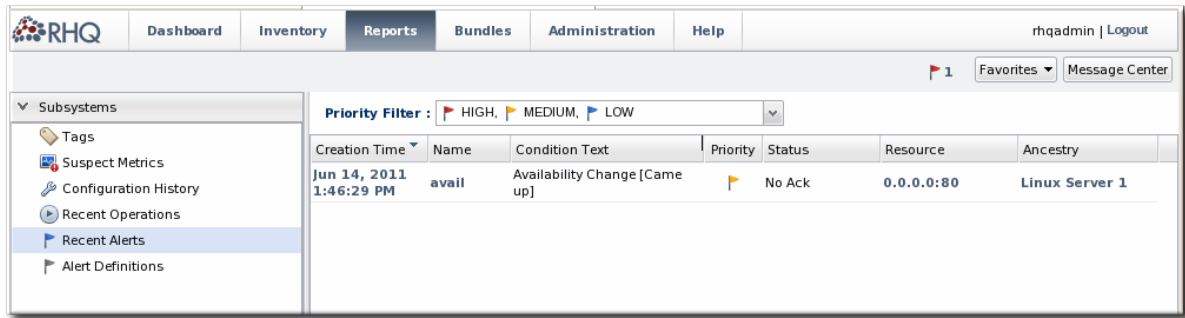
Acknowledging an alert is a way of identifying that the condition which triggered the alert has been addressed in some way. When an alert is acknowledged, the name of the user who acknowledged the alert is recorded. Recording the acknowledger's name allows the action to be audited later if necessary.

There are several different ways to acknowledge an alert:

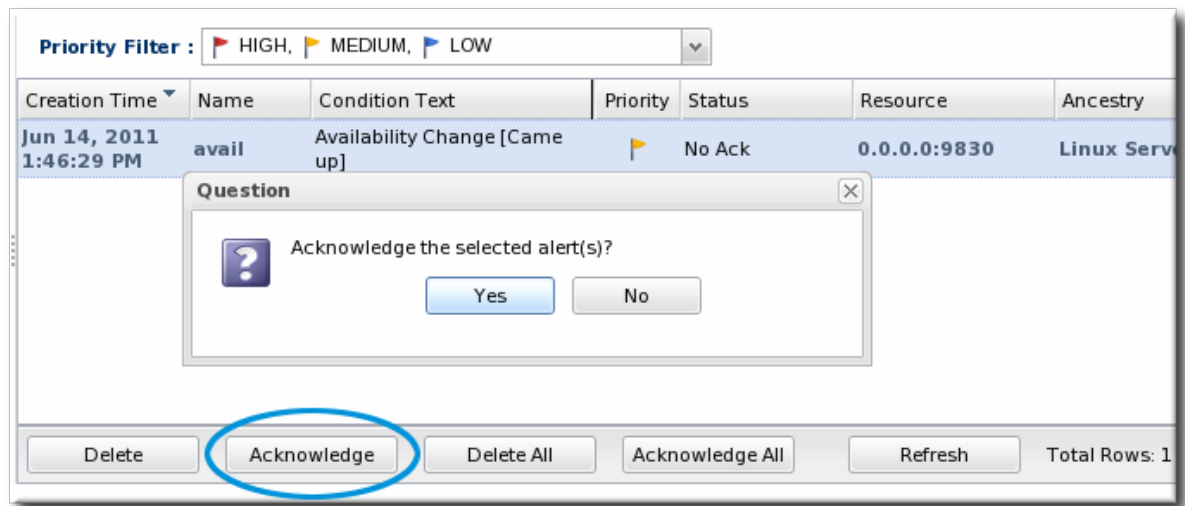
- Through the Recent Alerts Report
- Through a group
- Through the resource entry

Using the Recent Alerts Report is useful because you can acknowledge multiple alerts at the same time and for multiple resource types, which could be simpler if a known outage triggered many alerts. Acknowledging an alert is not a requirement to close the alert, but it can be useful as part of auditing an incident response or making sure that issues have been addressed.

1. Select the **Reports** tab in the top navigation bar.
2. In the **Subsystems** menu box on the left, select **Recent Alerts**.



3. Select the alert to acknowledge.
4. Click the **Acknowledge** button, and, when prompted, confirm the action.



NOTE

It is also possible to acknowledge a single alert through the alert details page.

When the alert is acknowledged, the **Status** shows the name of the user who acknowledged (and presumably resolved) the alert.

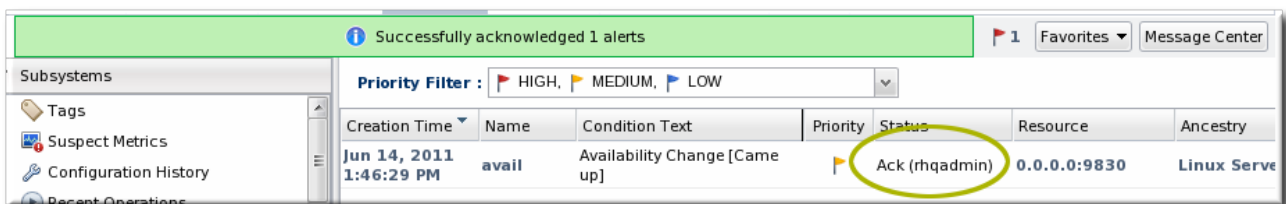


Figure 35. Alert Acknowledgment

14.4. Troubleshooting Alerts

- Q:** I just created an alert definition, and I know that my agent reported data that should have fired an alert immediately. But I don't see an alert. Why not?
- A:** After an alert definition is created, it takes a few seconds to be inserted into the JBoss ON server alert caches and then propagated throughout the JBoss ON server cloud. An alert won't be fired until that alert definition is in the server alert cache.

When the alert definition is inserted into the cache, a message is recored in the JBoss ON server logs:

```
INFO [CacheConsistencyManagerBean] localhost took [51]ms to reload
global cache
INFO [CacheConsistencyManagerBean] localhost took [49]ms to reload
cache for 1 agents
```

It generally takes around 30 seconds for an alert definition to be added to the cache. Wait at least a minute after creating a definition before checking if it fires an alert.

Q: Why do I see alerts triggered on different metric values on different alert definition conditions when they are using the same metric?

A: This can occur due to the nature of how alert conditions are processed when measurement data comes in from the agent. This happens if a single alert definition has multiple conditions that use the same metric and that alert definition uses the "ALL" conjunction. For example, if an alert definition has one condition for "alert if metric X is greater than 5" *and* a separate condition for "alert if metric X less than 10."

The alert condition range works around this by doing range checking. For more information, see [Section 12.2, "Detailed Discussion: Ranges, AND, and OR Operators with Conditions"](#).

15. OPERATIONS: AN INTRODUCTION

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.

The *Resource Reference: Monitoring, Operation, and Configuration Options* contains a complete reference for all of the operations that can be scheduled for each resource type, as well as configurable parameters for the operations. Regardless of the type of operation or resource, the process for scheduling operations is similar for both resources and compatible groups in JBoss ON.

JBoss Operations Network allows administrators to manage resources by scheduling and launching *operations*. Operations manage resources by initiating or even scheduling some basic, specified tasks, 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. The supported operations for a resource are defined by its agent plug-in, and the default operations are listed for each resource type in the *Resource Reference: Monitoring, Operation, and Configuration Options*.

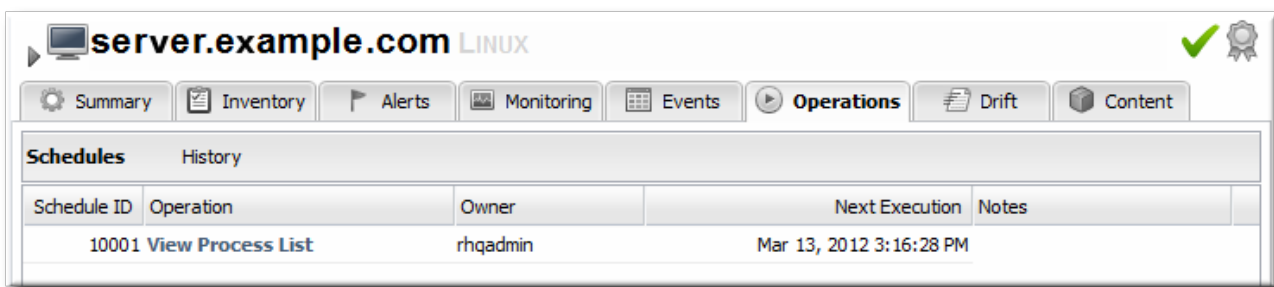
15.1. A Summary of Operation Benefits

Operations provide a way to perform tasks in a consistent way, with a defined order both on resources and in task queuing, and in a way that can be tracked. Because operations are defined by plug-ins, they are extensible. The versatility of running specific tasks through JBoss ON provides several benefits to administrators:

- They allow additional parameters (depending on how the operation is defined in the plug-in), such as command arguments and environment variables.
- They validate any operation parameters, command-line arguments, or environment variables much as JBoss ON validates resource configuration changes.
- They can be run on group of resources as long as they are all of the same type.
- Operations can be ordered to run on group resources in a certain order.
- They can be run on a recurrently schedule or one specific time.
- Operations keep a history of both successes and failures, so that it is possible to audit the operations executed on a resource both for operations run for that specific resource and done on that resources as part of a group.

15.2. About Scheduling Operations

Operations can be executed immediately or scheduled for later execution. Deferred operations can be scheduled for once only execution at a specific time, or operations can be scheduled to occur on a recurring basis starting at a specific time. Recurring operations can continue with no end date or time specified, or they can include a termination date and time.

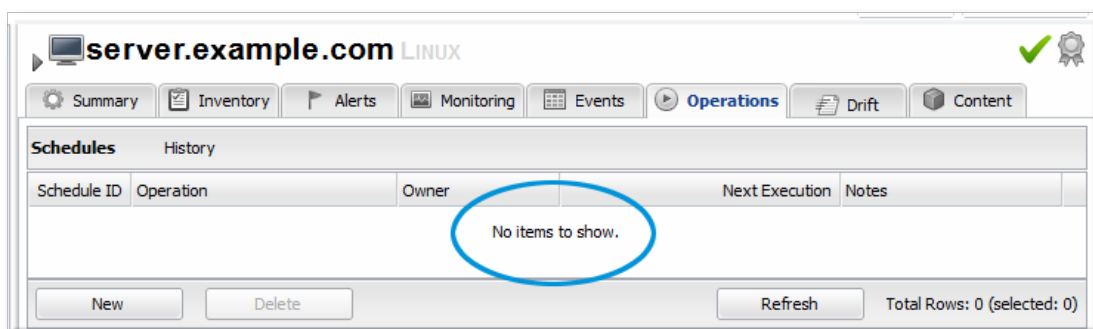


Schedule ID	Operation	Owner	Next Execution	Notes
10001	View Process List	rhqadmin	Mar 13, 2012 3:16:28 PM	

Figure 36. A Scheduled Operation

NOTE

The **Schedules** tab shows a list of scheduled operations, meaning operations which are configured but have not yet been run. If there are no scheduled operations, then the tab has a description that reads *No items to show*. That does not mean that there are no operations available for the resource; it only means that no operations have been scheduled.



Schedule ID	Operation	Owner	Next Execution	Notes
No items to show.				

New Delete Refresh Total Rows: 0 (selected: 0)

When an operation is scheduled, a new operation is added to the history record for the resource, and its state is set to in progress. A message is sent to the agent telling it to invoke a specific operation on a

particular resource with the arguments that were specified when the schedule was created. The agent queues operations so that only one operation is executed on the resource at a time.

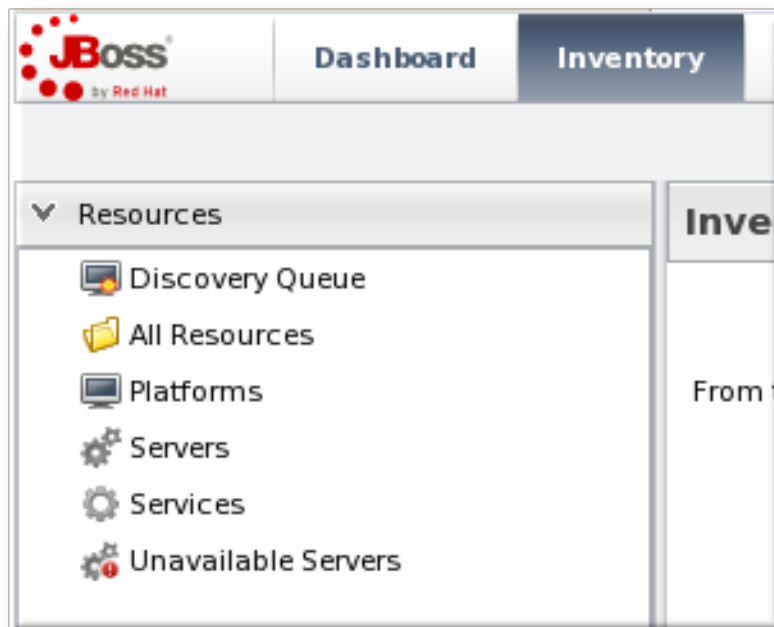
When an operation completes, its raw output is sent back to the agent's resource plug-in, which ultimately parses the output and then generates an appropriate response message. This response message is then sent to the server.

If one operation ever hangs on a resource, then it blocks any other operations from being initiated because only one operation can be run on a resource at a time. Using a timeout setting for the operation enables the agent to kill the hung operation and run other queued operations.

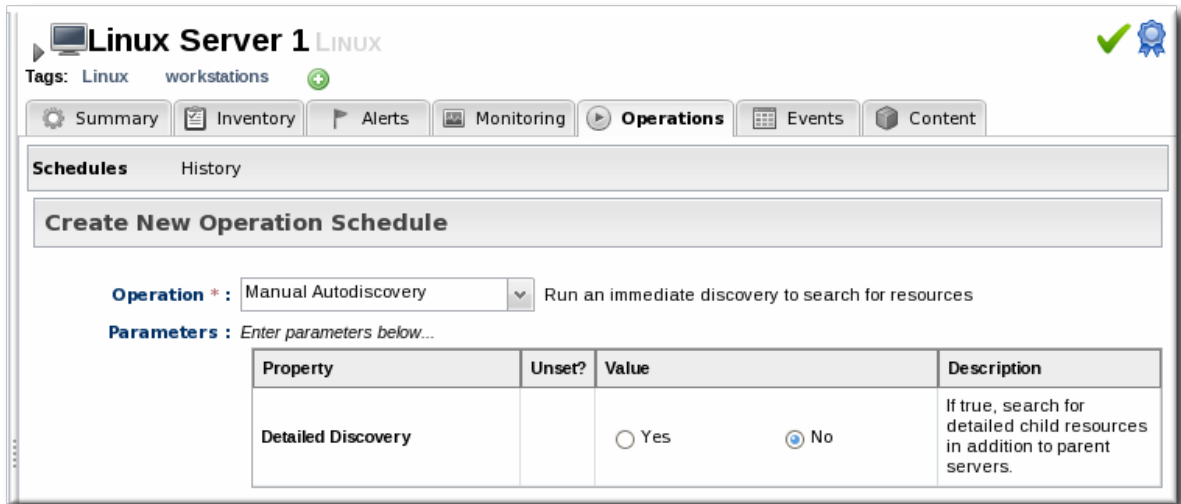
16. MANAGING OPERATIONS: PROCEDURES

16.1. Scheduling Operations

1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the **Operations** tab.
4. In the **Schedules** tab, click the **New** button.



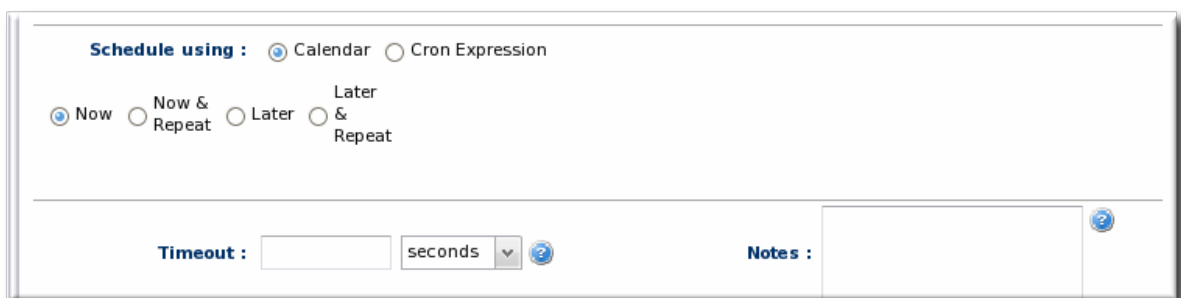
The types of operations that are available vary, depending on the specific type of resource.



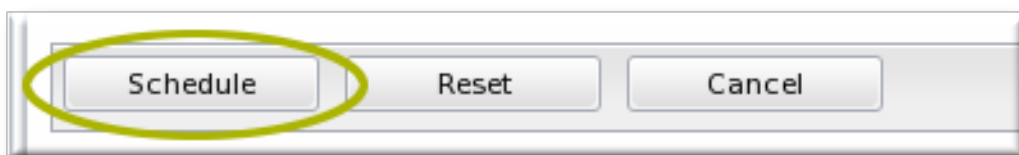
NOTE

The **Schedules** tab shows a list of scheduled operations, meaning operations which are configured but have not yet been run. If there are no scheduled operations, then the tab has a description that reads *No items to show*. That does not mean that there are no operations available for the resource; it only means that no operations have been scheduled.

5. Fill in all of the required information about the operation, such as port numbers, file locations, or command arguments.
6. In the **Schedule** area, set when to run the operation. The operation can run immediately, at a specified time, or on a repeatable schedule.



7. Set other rules for the operations, like a timeout period and notes on the operation itself.
8. Click the **Schedule** button to set up the operation.



If the operation is scheduled to run immediately, the results are available in the **History** subtab as the operation is in progress and then completes. If it was scheduled on a later date or with a recurring schedule, then the operation is listed in the **Schedules** subtab.

16.2. Viewing the Operation History

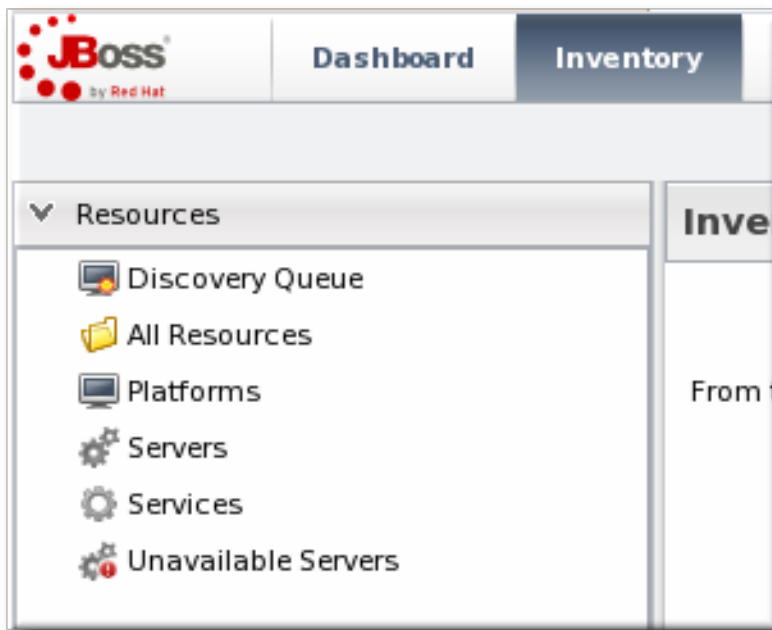


NOTE

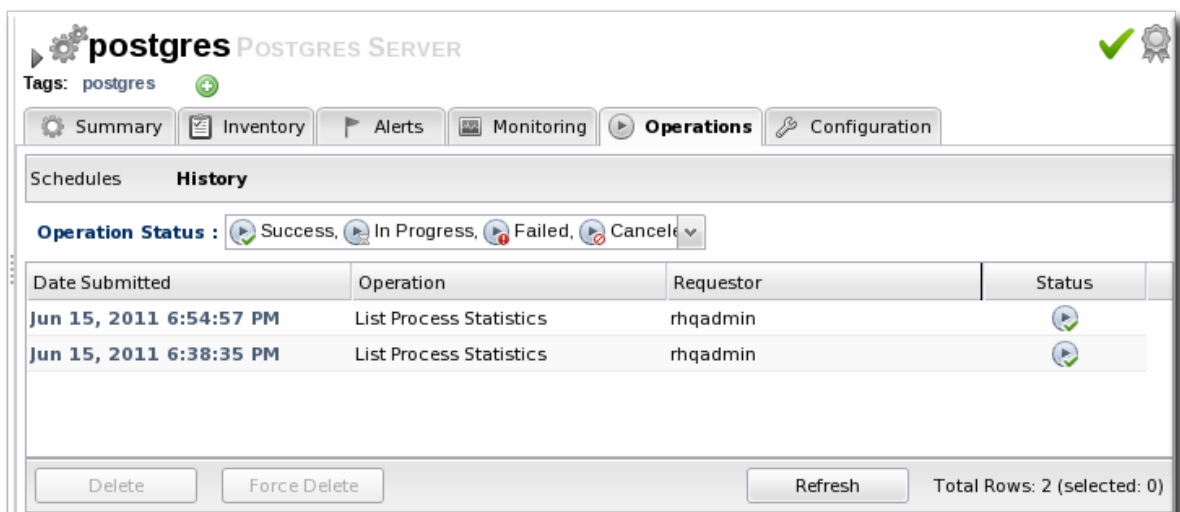
A user may have the write to schedule and edit an operation, but that does not mean that the user has the right to delete an item from the operation history.

Deleting elements in the history requires the manage inventory permission.

1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the **Operations** tab.
4. Click the **History** subtab.



Every completed or in progress operation is listed, with an icon indicating its current status.

- Click the name of the operation to view further details. The history details page shows the start and end times of the operation, the stdout output of the operation or other operation messages, as well as the name of the user who scheduled the operation.

postgres POSTGRES SERVER

Tags: postgres

Summary Inventory Alerts Monitoring **Operations** Configuration

Schedules **History**

[Back to List](#)

Execution ID : 10011

Operation : List Process Statistics

Date Submitted : Jun 15, 2011 6:54:57 PM

Date Completed : Jun 15, 2011 6:54:58 PM

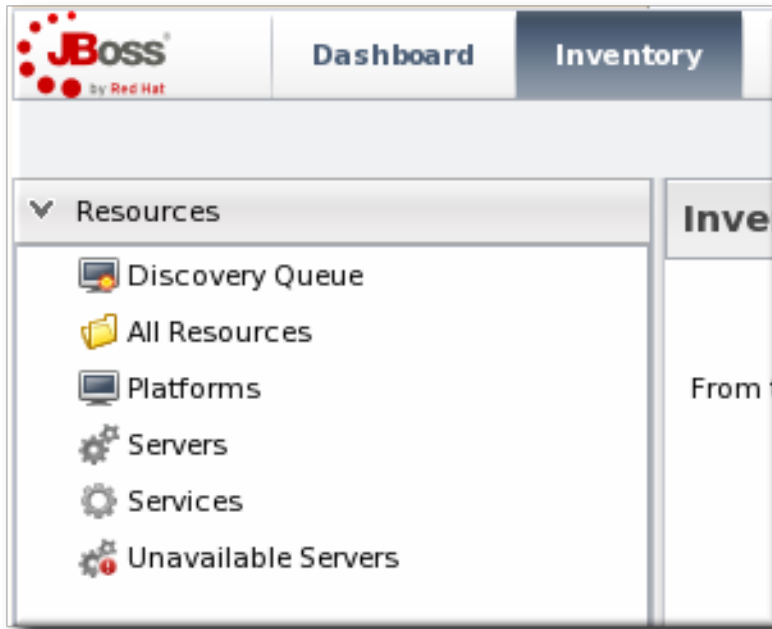
Requestor : rhqadmin

Results

Property	Unset?	Value	Description
Process List	Port	Address	Query
	43424	127.0.0.1	SELECT TRIGGER_STATE, NEXT_FIRE_TIME, JOB_NAME, JOB_GROUP FROM RHQ_QRT...
	35263	127.0.0.1	SELECT * FROM pg_stat_activity ORDER BY current_query desc
	45886	127.0.0.1	
	49535	127.0.0.1	
	44301	127.0.0.1	
	45544	127.0.0.1	
	39379	127.0.0.1	
	35317	127.0.0.1	
	35318	127.0.0.1	

16.3. Canceling Pending Operations

- Click the **Inventory** tab in the top menu.
- Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the **Operations** tab.
4. In the **Schedules** tab, click the line of the operation to cancel.
5. Click **Delete**, and confirm the action.



NOTE

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

16.4. Ordering Group Operations

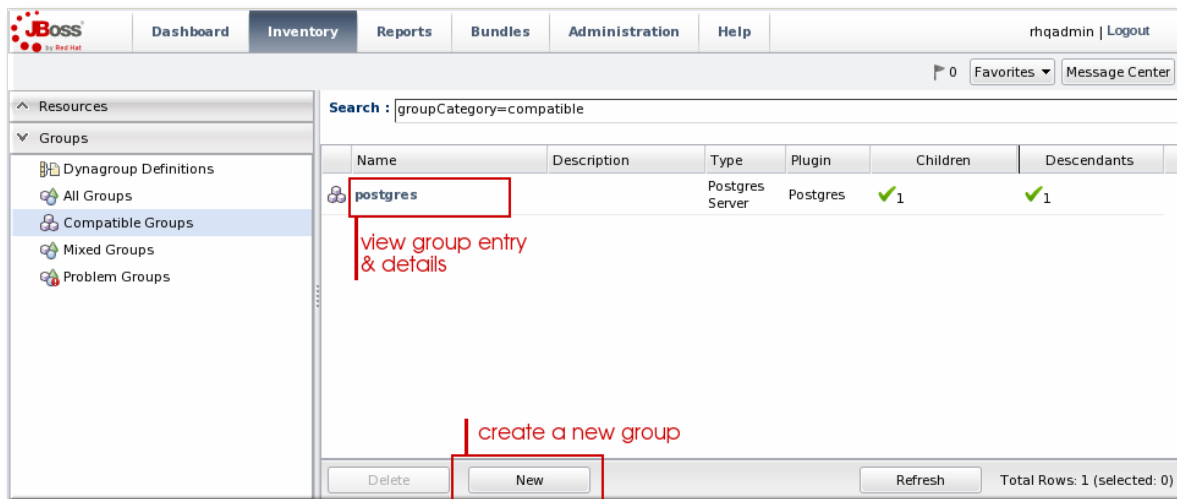
Group operations can be scheduled. This is useful when operations need to be performed in a particular order.



NOTE

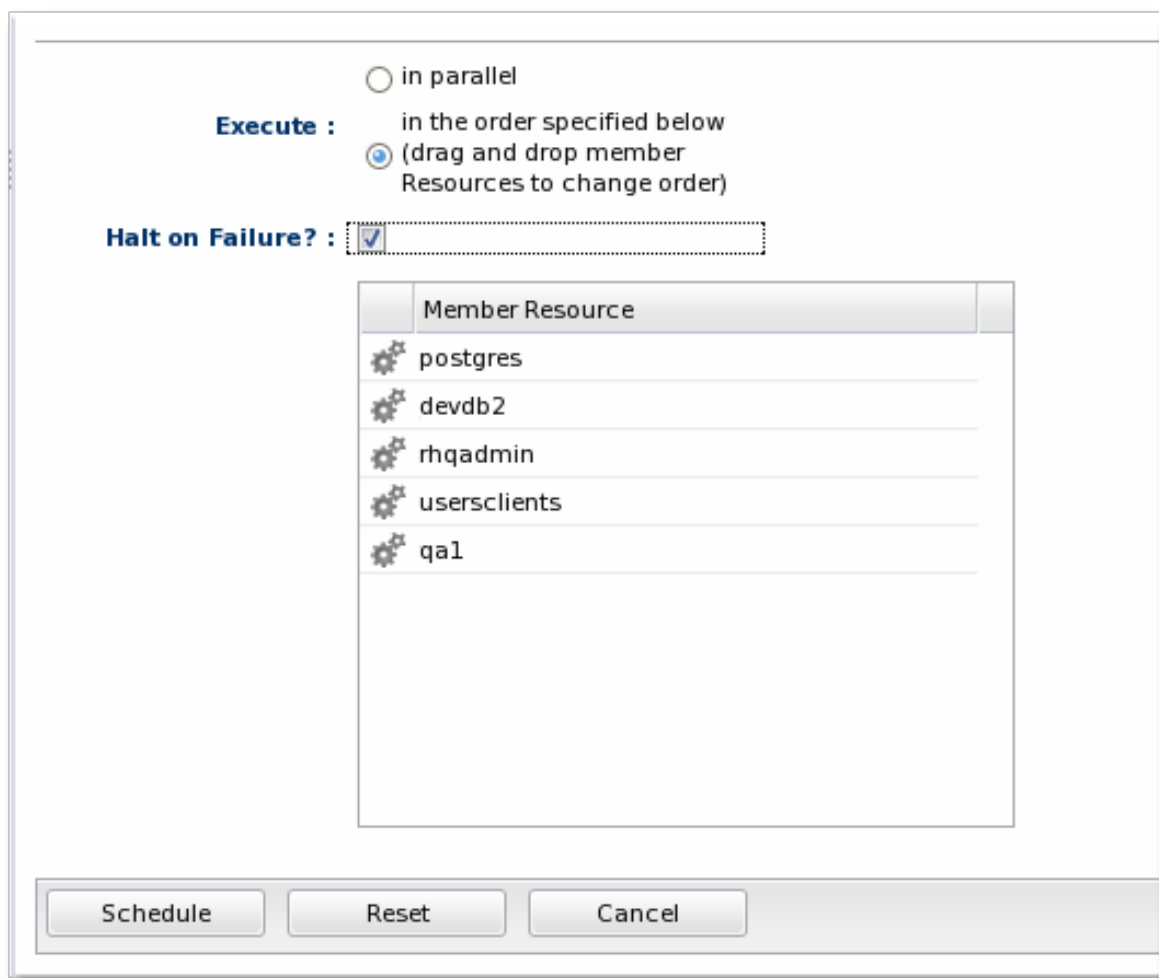
This procedure assumes groups are already set up.

1. In the **Inventory** tab in the top menu, select the **Compatible Groups** item in the **Groups** menu on the left.



2. Click the name of the group to run the operation on.
3. Configure the operation, as in [Section 16.1, “Scheduling Operations”](#).
4. In the **Resource Operation Order** area, set the operation to execute on all resources at the same time (in parallel) or in a specified order. If the operation must occur in resources in a certain order, then all of the group members are listed in the **Member Resources** box, and they can be rearranged by dragging and dropping them into the desired order.

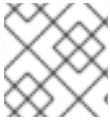
Optionally, select the **Halt on failure** checkbox to stop the group queue for the operation if it fails on any resource.



16.5. Running Scripts as Operations for JBoss Servers

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 and Linux
- **.pl** scripts for Unix and Linux



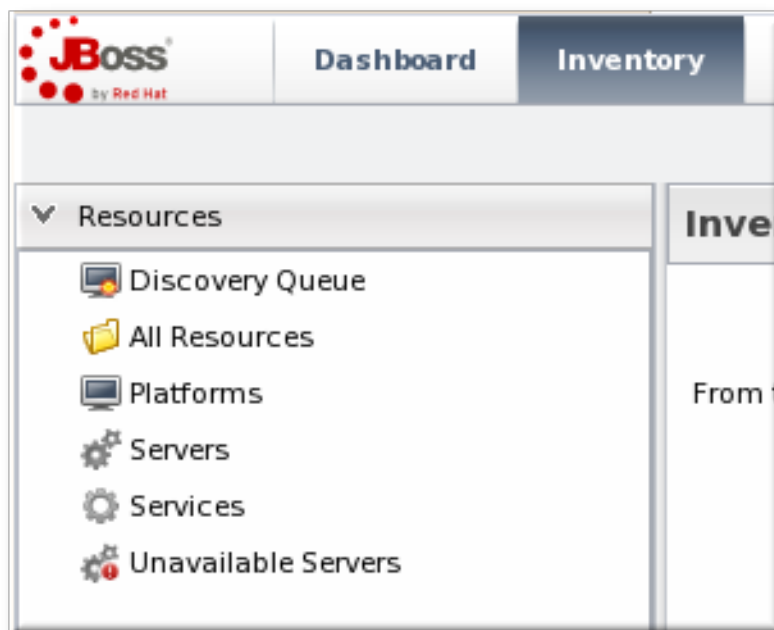
NOTE

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.

To execute a script as an operation:

1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



3. Click the **Operations** tab.
4. In the **Schedules** tab, click the **New** button.
5. Select **Execute script** as the operation type from the **Operation** drop-down menu.

The screenshot shows the 'Create New Operation Schedule' form. At the top, there are tabs for 'Summary', 'Inventory', 'Alerts', 'Monitoring', and 'Operations'. Below the tabs, there are sections for 'Schedules' and 'History'. The main form area is titled 'Create New Operation Schedule'. It includes a dropdown menu for 'Operation' set to 'Execute Script' with a note: 'execute the script; NOTE: environment variables to be passed to the script can be configured via this Script service's connection properties (under its Inventory>Connection tab)'. Below this is a 'Parameters' section with the instruction 'Enter parameters below...'. A table with four columns: 'Property', 'Unset?', 'Value', and 'Description' is shown. The first row has 'Command Line Arguments' in the 'Property' column, a checked checkbox in the 'Unset?' column, an empty text box in the 'Value' column, and a detailed description in the 'Description' column: 'the command-line arguments (if any) to pass to the script; each command-line argument must be on a new line; the arguments can contain properties with the syntax %propertyName%; the script plugin will interpolate these with the current values of the corresponding properties from the script's parent JBossAS server's connection properties'.



NOTE

The **Execute script** option is only available for JBoss AS and JBoss AS 5 resources, by default, and only if a script is available to execute.

6. Enter any command-line arguments in the **Parameters** text box.

Each new argument has the format *name=value*; and is added on a new line. If the variable's value contains properties with the syntax **%propertyName%**, then JBoss ON interprets the value as the current values of the corresponding properties from the script's parent resource's connection properties.

7. Finish configuring the operation, as in [Section 16.1, "Scheduling Operations"](#).

16.6. 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-3.1.2.GA1/bin/rhq-server.properties
```

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

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

16.7. Operation History Report

Every resource tracks its own individual operations history, as in [Section 16.2, “Viewing the Operation History”](#).

JBoss ON also keeps a master list of all operations, for all resources. This is displayed in the **Operation History Report**, in the **Reports** tab.

As with the resource-level operation history, the Operation History shows the operation name, the date it was submitted, and its status. Because all resources are listed, the **Operation History Report** also shows the resource name and its parent (and grandparent) to help disambiguate on which resource the operation ran.

Date Submitted	Operation	Requestor	Status	Resource	Ancestry
May 8, 2012 10:46:18 PM	Execute Availability Scan	rhqadmin	Success	RHQ Agent	server.example.com
May 8, 2012 10:23:56 PM	Execute Availability Scan	rhqadmin	Success	RHQ Agent	server.example.com
May 7, 2012 2:42:14 PM	View Process List	rhqadmin	Success	server.exam...	
May 7, 2012 11:47:12 AM	Install RHQ user	rhqadmin	Success	EAP Domain Controller (127.0.0.1:99...	server.example.com

Figure 37. Operation History Report

The **Operation History Report** can be filtered (not just sorted) by two criteria: the operation status and the date or date range that the operation was submitted.

NOTE

Reports can be exported to CSV, which can be used for office systems or further data manipulation.

Only the information displayed for the report is exported. If the **Operation History Report** is filtered by date or status, only the matching operations are included in the report.

To export a report, simply click the **Export** button. The report will automatically be downloaded as **configurationHistory.csv**.

INDEX

A

alerts

- acknowledging, [Acknowledging an Alert](#)
- assigning operations, [Detailed Discussion: Initiating an Operation](#)
- available operation tokens, [Using Tokens with Alert Operations](#)
- conditions, [Reasons for Firing an Alert](#)
- configuring, [Planning Alerts](#)
- configuring and managing, [Alert Conditions](#), [Alert Responses](#)
- configuring for groups, [Configuring Group Alerts](#)
- creating definition templates, [Creating Alert Definition Templates](#)
- definitions report, [Viewing the Alert Definitions Report](#)
- enabling and disabling definitions, [Enabling and Disabling Alert Definitions](#)
- fired alerts report, [Viewing the Fired Alerts Report](#)
- groups and templates, [Group Alerting and Alert Templates](#)
- histories and acknowledgments, [Viewing Alert Data](#)
- initiating CLI scripts, [Detailed Discussion: Launching JBoss ON CLI Scripts from an Alert](#)
- initiating resource scripts, [Detailed Discussion: Initiating Resource Scripts](#)
- notification methods, [Notifying Administrators and Responding to Alerts](#)
- operations, [Detailed Discussion: Initiating an Operation](#), [Setting Alert Operations](#)
- setting, [Basic Procedure for Setting Alerts for a Resource](#)
- setting dashboard display, [Viewing Alerts in the Dashboard](#)
- SNMP, [Configuring SNMP for Notifications](#), [Configuring the SNMP Alert Plug-in](#)
- SNMP notifications, [Configuring the SNMP Alert Notification](#)
- tokens and operations, [Using Tokens with Alert Operations](#)
- using templates and group alerts, [Group Alerting and Alert Templates](#)
- viewing, [Viewing Alerts](#)
- viewing alerts for specific resource, [Viewing Alert Details for a Specific Resource](#)

Apache

- configuring the SNMP module, [Configuring the Apache SNMP Module](#)

auto discovery

- Tomcat, [Configuring Tomcat/EWS Servers for Monitoring](#)

availability, [Availability](#)

- charts, [Viewing a Resource's Availability Charts](#)
- troubleshooting, [Core "Up and Down" Monitoring](#)
- uptime percentage, [Core "Up and Down" Monitoring](#)

B

baselines

recalculating, [Recalculating Baseline Values](#)

C

call-time data, [URL Response Time Monitoring](#)

CLI

running for alerts, [Detailed Discussion: Launching JBoss ON CLI Scripts from an Alert](#)

configuration

alerts, [Alert Conditions](#), [Alert Responses](#)

Apache SNMP module, [Configuring the Apache SNMP Module](#)

response time filters for Tomcat, [Configuring Response Time Filters for Tomcat](#)

D**dashboard**

alerts settings, [Viewing Alerts in the Dashboard](#)

data storage

monitoring, [Storing Monitoring Data](#)

E**events**

defining new events, [Defining a New Event](#)

overview, [Events](#)

viewing, [Viewing Events](#)

G**groups**

alerts and templates, [Group Alerting and Alert Templates](#)

configuring alerts, [Configuring Group Alerts](#)

J**JBoss EAP 6/AS 7**

configuring response time metrics, [Configuring Response Time Metrics for JBoss EAP 6/AS 7](#)

JBoss EAP/AS 5

configuring response time metrics, [Configuring Response Time Metrics for JBoss EAP/AS 5](#)

JBoss ON

and SNMP, [JBoss ON SNMP Information](#)

M

metrics, [Metrics and Measurements](#)

baselines, [Baselines and Out-of-Bounds Metrics](#)

monitoring

call-time data, [URL Response Time Monitoring](#)

changing defaults, [Changing Metrics Templates](#)

changing the resource availability scan period, [Changing the Agent's Availability Scan Period](#)

collection schedules, [Collection Schedules](#)

configuring response time filters for JBoss EAP 6/AS 7 servers, [Configuring Response Time Metrics for JBoss EAP 6/AS 7](#)

configuring response time filters for JBoss EAP/AS 5 servers, [Configuring Response Time Metrics for JBoss EAP/AS 5](#)

configuring Tomcat, [Configuring Tomcat/EWS Servers for Monitoring](#)

enabling and disabling metrics, [Enabling and Disabling Metrics for a Specific Resource](#)
operations, [Reports and Data](#)

out of bounds (OOB), [Baselines and Out-of-Bounds Metrics](#)

problem metrics, [Baselines and Out-of-Bounds Metrics](#)

recalculating baselines, [Recalculating Baseline Values](#)

resources with special configuration, [Resources Which Require Special Configuration for Monitoring](#)

response time filters, [URL Response Time Monitoring](#)

setting collection intervals, [Setting Collection Intervals for a Specific Resource](#)

storage limits for data, [Storing Monitoring Data](#)

traits, [Resource Traits](#)

N

notifications

SNMP and alerts, [Configuring the SNMP Alert Notification](#)

O

operations, [Operations: An Introduction](#)

alerts, [Detailed Discussion: Initiating an Operation](#)

alerts and tokens, [Using Tokens with Alert Operations](#)

canceling, [Canceling Pending Operations](#)

executing scripts, [Running Scripts as Operations for JBoss Servers](#)

ordering group, [Ordering Group Operations](#)

scheduling, [About Scheduling Operations](#), [Scheduling Operations](#)

scripts, [Running Scripts as Operations for JBoss Servers](#)

setting alert operations, [Setting Alert Operations](#)

timeout defaults, [Setting an Operation Timeout Default](#)

timeouts, [About Scheduling Operations](#)
viewing history, [Viewing the Operation History](#)

R

reports

alert definitions, [Viewing the Alert Definitions Report](#)

resources

enabling and disabling metric collection, [Enabling and Disabling Metrics for a Specific Resource](#)

monitoring, [Reports and Data](#)

monitoring and recalculating baselines, [Recalculating Baseline Values](#)

monitoring collection intervals, [Setting Collection Intervals for a Specific Resource](#)

setting alerts, [Basic Procedure for Setting Alerts for a Resource](#)

viewing alerts for specific resources, [Viewing Alert Details for a Specific Resource](#)

response time filters

configuring JBoss EAP 6/AS 7, [Configuring Response Time Metrics for JBoss EAP 6/AS 7](#)

configuring JBoss EAP/AS 5, [Configuring Response Time Metrics for JBoss EAP/AS 5](#)

configuring Tomcat, [Configuring Response Time Filters for Tomcat](#)

response-time filters, [URL Response Time Monitoring](#)

S

scripts

as operations, [Running Scripts as Operations for JBoss Servers](#)

from alerts, [Detailed Discussion: Initiating Resource Scripts](#)

initiating from alerts

CLI, [Detailed Discussion: Launching JBoss ON CLI Scripts from an Alert](#)

SNMP

alert notification, [Configuring the SNMP Alert Notification](#)
and JBoss ON, [JBoss ON SNMP Information](#)

configuring alerts, [Configuring the SNMP Alert Plug-in](#)

configuring for alerts, [Configuring SNMP for Notifications](#)

configuring the Apache module, [Configuring the Apache SNMP Module](#)

T

tokens

alerts and operations, [Using Tokens with Alert Operations](#)

Tomcat

configuring response time filters, [Configuring Response Time Filters for Tomcat](#)

monitoring configuration, [Configuring Tomcat/EWS Servers for Monitoring](#)

traits, [Resource Traits](#)

U

uptime percentage, [Core "Up and Down" Monitoring](#)

[1] This is mean in the statistical sense. It is the middle data point of all collected uptime lengths.