Monitoring JBoss AS/EAP Servers

eG Innovations Product Documentation

www.eginnovations.com

Total Performance Visibility

Table of Contents

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: HOW TO MONITOR JBOSS AS/EAP SERVER USING EG ENTERPRISE? 3

2.1 Pre-requisites for Monitoring JBoss EAP 3

2.1.1 Determining the Port Number for Managing JBoss EAP 3

2.1.2 Creating a New User in the JBoss EAP Server 11

2.1.3 Configuring JVM Monitoring for a JBoss EAP Server 14

2.1.4 Configuring Java Business Transaction Monitoring on JBoss EAP 25

2.2 Managing the JBoss AS/EAP Server 32

CHAPTER 3: MONITORING JBOSS AS/EAP SERVERS 34

3.1 The JVM Layer 35

3.2 The JBoss Server Layer 35

3.2.1 JBoss Connectors Test 37

3.2.2 JBoss Datasources Test 39

3.2.3 JBoss EJBs Test 45

3.2.4 JBoss MQ Queues Test 48

3.2.5 JBoss MQ Topics Test 50

3.2.6 JBoss Servlet Test 53

3.2.7 JBoss Transactions Test 54

3.2.8 JBoss XA-Datasources Test 59

3.2.9 JBoss JPA Test 66

3.2.10 Web Service Test 69

3.3 The Application Transactions Layer 82

3.4 The Java Transactions Layer 82

3.4.1 Java Business Transactions Test 82

ABOUT EG INNOVATIONS 84

Table of Figures

Figure 1.1: The general classification of the JBoss AS 7 server 1

Figure 1.2: The JBoss AS 7 running in Domain mode 2

Figure 2.1: How to determine the operating mode of JBoss EAP 4

Figure 2.2: Selecting the Management Model option from Tools menu 5

Figure 2.3: Determining the configuration file of the monitored server 6

Figure 2.4: Determining the management-http port number of the JBoss EAP server 7

Figure 2.5: Selecting the Management Model option from the Tools menu in the management console of the domain controller 8

Figure 2.6: Finding the profile name of the target JBoss EAP server 9

Figure 2.7: Determining the port number of a JBoss EAP server operating in the domain mode 10

Figure 2.8: Port-offset and its impact on the application/HTTP port configuration of a server 10

Figure 2.9: How to determine the operating mode of JBoss EAP 15

Figure 2.10: Selecting the Management Model option from Tools menu 16

Figure 2.11: Determining the configuration file of the monitored server 17

Figure 2.12: Determining the JMX Remote Port of the JBoss EAP server 18

Figure 2.13: Selecting the Management Model option from the Tools menu in the management console of the domain controller 19

Figure 2.14: Finding the profile name of the target JBoss EAP server 20

Figure 2.15: Determining the port number of a JBoss EAP server operating in the domain mode 20

Figure 2.16: Port-offset and its impact on the application/HTTP port configuration of a server 21

Figure 2.17: Selecting the Management Model option from the Tools menu in the management console of the domain controller 23

Figure 2.18: Finding the profile name of the target JBoss EAP server 24

Figure 2.19: Disabling the management endpoint for the domain 25

Figure 2.20: How to determine the operating mode of JBoss EAP 26

Figure 2.21: Editing the start-up script to BTM-enable a JBoss EAP server that is monitored in an agent-based manner 29

Figure 2.22: Editing the domain.xml file or standalone.conf file 29

Figure 2.23: Editing the start-up script to BTM-enable a JBoss EAP server on Unix that is monitored in an agent-based manner 31

Figure 2.24: Editing the domain.xml file or standalone.conf file 32

Figure 2.25: Adding a JBoss AS/EAP for monitoring 33

Figure 3.1: The layer model of the JBoss AS/EAP server 34

Figure 3.2: The tests mapped to the JVM layer 35

Figure 3.3: The tests mapped to the JBoss Server layer 36

Figure 3.4: Configuring the WebService test 76

Figure 3.5: The WebService URL Configuration page 77

Figure 3.6: Configuring the Web Service Operation 79

Figure 3.7: Specifying the value for the chosen operation 80

Figure 3.8: The value that appears when the operation is performed successfully 81
Figure 3.9: An Error appearing during value conversion 81

Chapter 1: Introduction

The JBoss Application Server is a widely used Java application server that provides a J2EE certified platform for developing and deploying enterprise Java applications, web applications, and portals, and also offers extended enterprise services such as clustering, caching, and persistence.

Beginning with the JBoss AS 7, the internal file system of the JBoss AS 7 is basically divided into two core parts: the dichotomy reflects the distinction between **standaloneservers** and **domain** servers. Figure 1.1 clearly explains the server distinction of the JBoss AS 7.

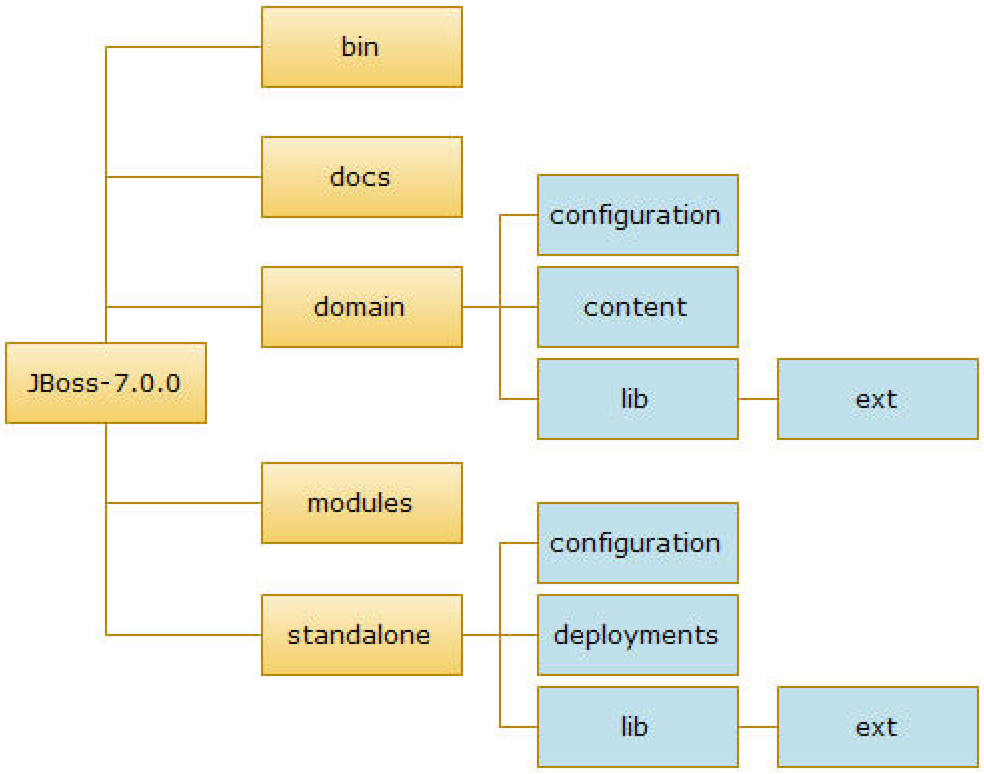


Figure 1.1: The general classification of the JBoss AS 7 server

A JBoss domain is used to manage and coordinate a set of application server instances. JBoss AS 7 in **domain** mode spawns multiple JVMs which build up the domain. Besides the AS instances, two more processes are created: the **Domain Controller** which acts as management control point of the domain and the **Host Controller** which interacts with the domain Controller to control the lifecycle of the AS instances.

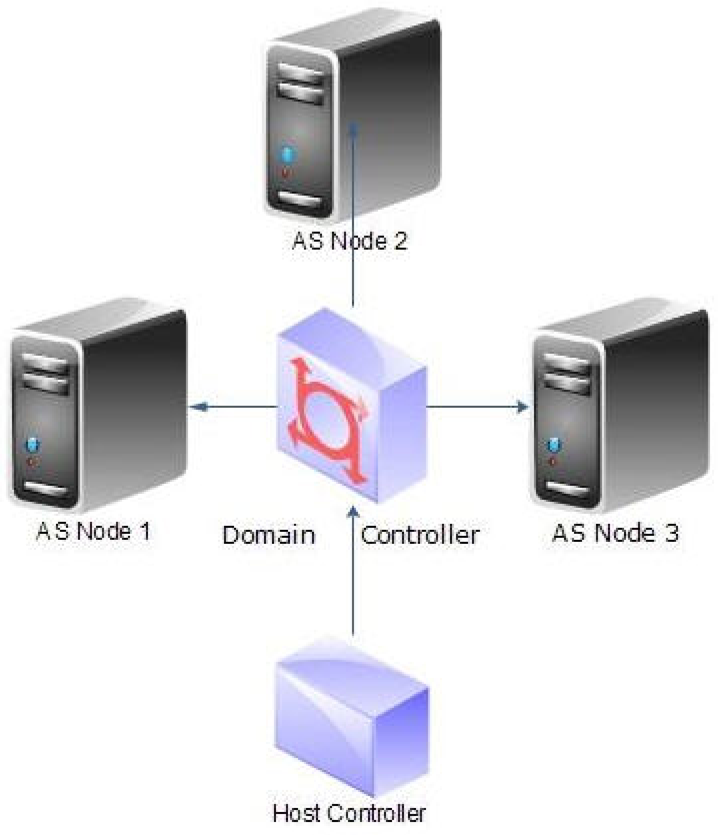


Figure 1.2: The JBoss AS 7 running in Domain mode

At a coarse level, the JBoss AS 7 server consists of two main elements:

- A core manageable service container based on modular classloading
- Extensions to that core that provide the kind of functionality most users associate with an application server, like handling HTTP requests and managing transactions

The AS distribution also includes two clients for the management interfaces it exposes (a CLI tool and a web-based admin console).

Chapter 2: How to Monitor JBoss AS/EAP Server Using eG Enterprise?

To monitor the JBoss AS/EAP Server, eG Enterprise employs an agent-based approach. The broad steps for monitoring the server using this approach are as follows:

1. Deploy an eG agent on the target host that is to be monitored. Use the installation procedure detailed in the *eG Installation Guide* to install the eG agent.
2. Ensure that the pre-requisites for monitoring JBoss EAP are fulfilled. To know what are the pre-requisites and how to fulfill them, refer to Pre-requisites for Monitoring JBoss EAP.
3. Manage the target JBoss EAP server using eG administrative interface. See Section **2.2**.
4. Finally, start the eG agent. To know how to start the eG agent, refer to the *eG Installation Guide*.

2.1 Pre-requisites for Monitoring JBoss EAP

Before attempting to manage and monitor JBoss EAP, make sure that the following pre-requisites are fulfilled:

- Determine the correct port using which the target JBoss EAP server has to be managed in the eG admin interface. To know what port to use and where to find it, refer to Section **2.1.1** topic.
- The eG tests have to be configured with the **credentials of a special user** who is authorized to access the web-based Management Console and remote instances of the Management CLI. The eG agent connects to the target EAP server and monitors it using this user's credentials only. To know the type of user required for this purpose and how to create such a user, refer to Section **2.1.2** topic.
- To enable the eG agent to collect JVM-related metrics, make sure the requirements for JVM monitoring detailed in the Section **2.1.3** topic are satisfied.
- By default, the eG agent cannot perform Java business transaction monitoring on a JBoss EAP server. For that, you need to enable the eG Java Business Transaction Monitor (BTM) on the EAP server. To know how to achieve this, refer to Section **2.1.4** topic.

2.1.1 Determining the Port Number for Managing JBoss EAP

When adding a JBoss EAP server using the eG admin interface, you need to specify a **Port number** for the target server. By default, this is set to *9990*. Depending upon the operating mode of the target

JBoss EAP server, you may have to override this default setting by specifying a different port number here.

Typically, JBoss EAP operates in one of the following modes:

- **Standalone server** operating mode for managing discrete instances
- **Managed domain** operating mode for managing groups of instances from a single control point

To know in which mode the JBoss EAP server you want to monitor is operating, login to the JBoss Management Console and click the **Runtime** option in its main menu. When Figure 1 appears, take a look at the options displayed in the left panel of Figure 1.

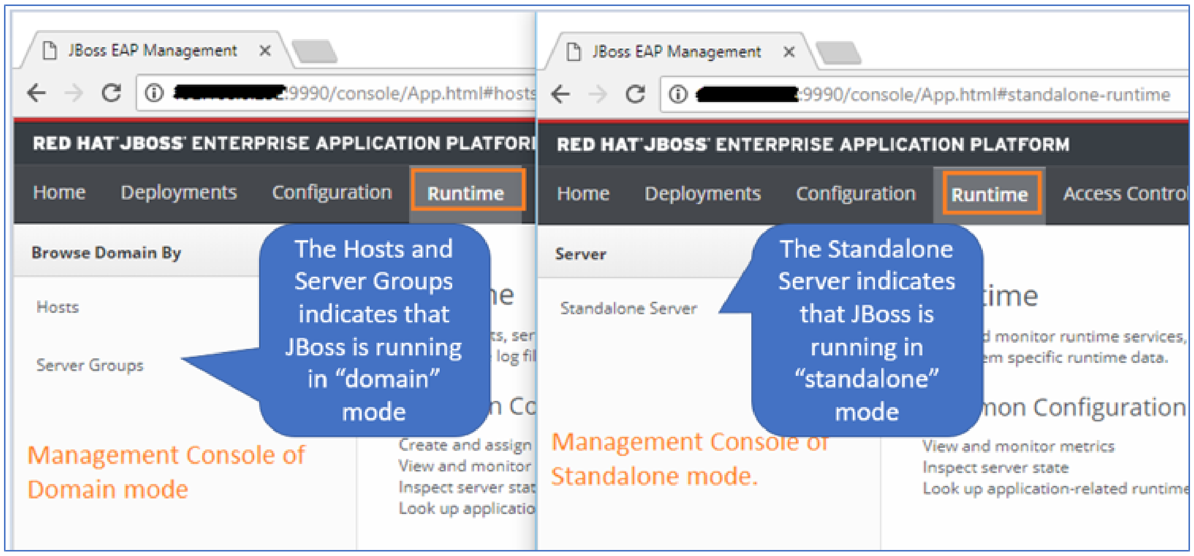

JBoss EAP Management
[redacted]:9990/console/App.html#hosts
RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM
Home Deployments Configuration Runtime
Browse Domain By
Hosts
Server Groups
The Hosts and Server Groups indicates that JBoss is running in "domain" mode
Management Console of Domain mode
JBoss EAP Management
[redacted]:9990/console/App.html#standalone-runtime
RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM
Home Deployments Configuration Runtime Access Control
Server
Standalone Server
The Standalone Server indicates that JBoss is running in "standalone" mode
Management Console of Standalone mode.

Figure 2.1: How to determine the operating mode of JBoss EAP

If you see **Hosts and Server Groups** displayed therein, it indicates that the target JBoss EAP server is running in the *domain* mode. If you see **Standalone Server** there instead, it implies that the target server is running in the *standalone* mode.

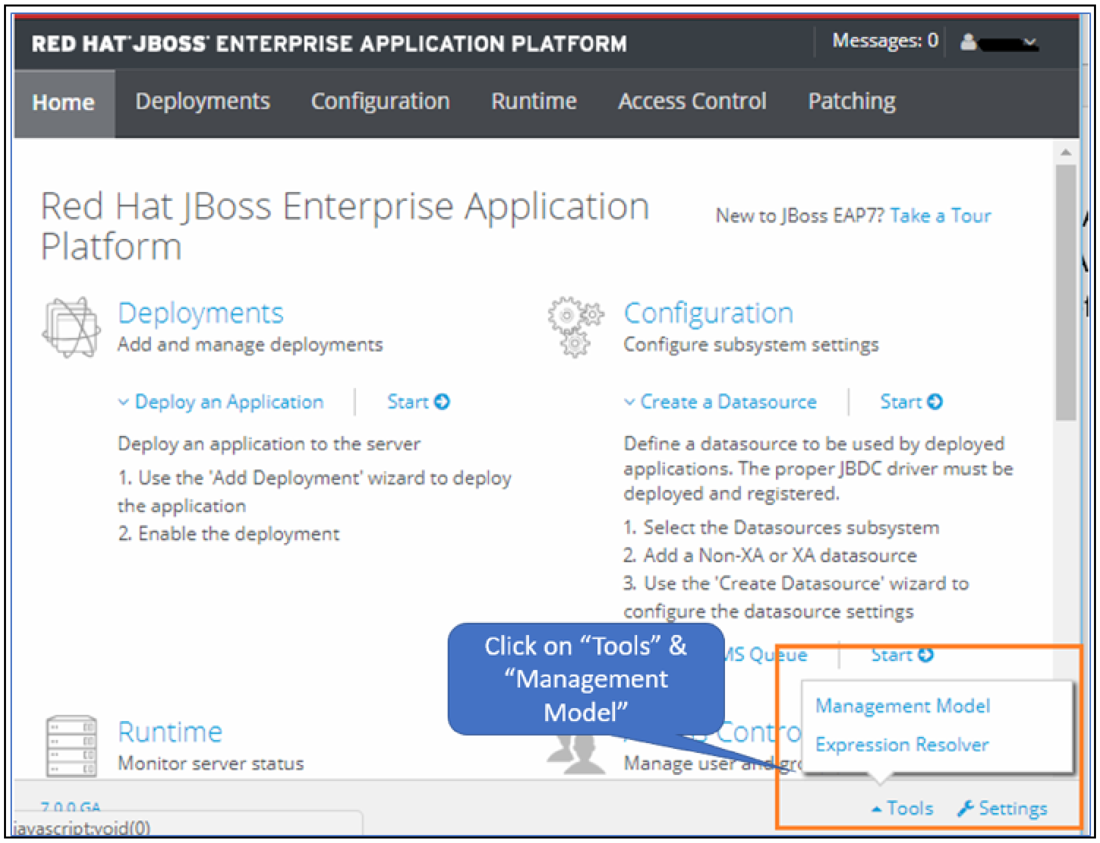
If the server to be monitored is operating in the *standalone* mode, then follow the steps below to determine what port the standalone server has to be configured with:

1. Typically, a JBoss EAP server operating in the standalone mode should be managed using the port on which the HTTP REST API is enabled. By default, this is port *9900*. To verify whether this default setting really holds good or has to be changed, first determine the profile on which the JBoss server is running. In the standalone operating mode, each profile is associated with a configuration file (*.xml). This configuration file is where the various port bindings are

configured. The table below lists the profile names and the configuration (XML) file that corresponds to each:

Profile Name	**Configuration File Name**
Default	standalone.xml
full	standalone-full.xml
ha	standalone-ha.xml
Full-ha	standalone-full-ha.xml

2. Next, determine which configuration file is in use on the target server and zoom into the contents of that file to identify the port number on which the HTTP REST API is enabled. To know the configuration file, login to the JBoss Management Console of the target server, click on the **Tools** menu at the bottom of the console and choose the **Management Model** option (as indicated by Figure 2).



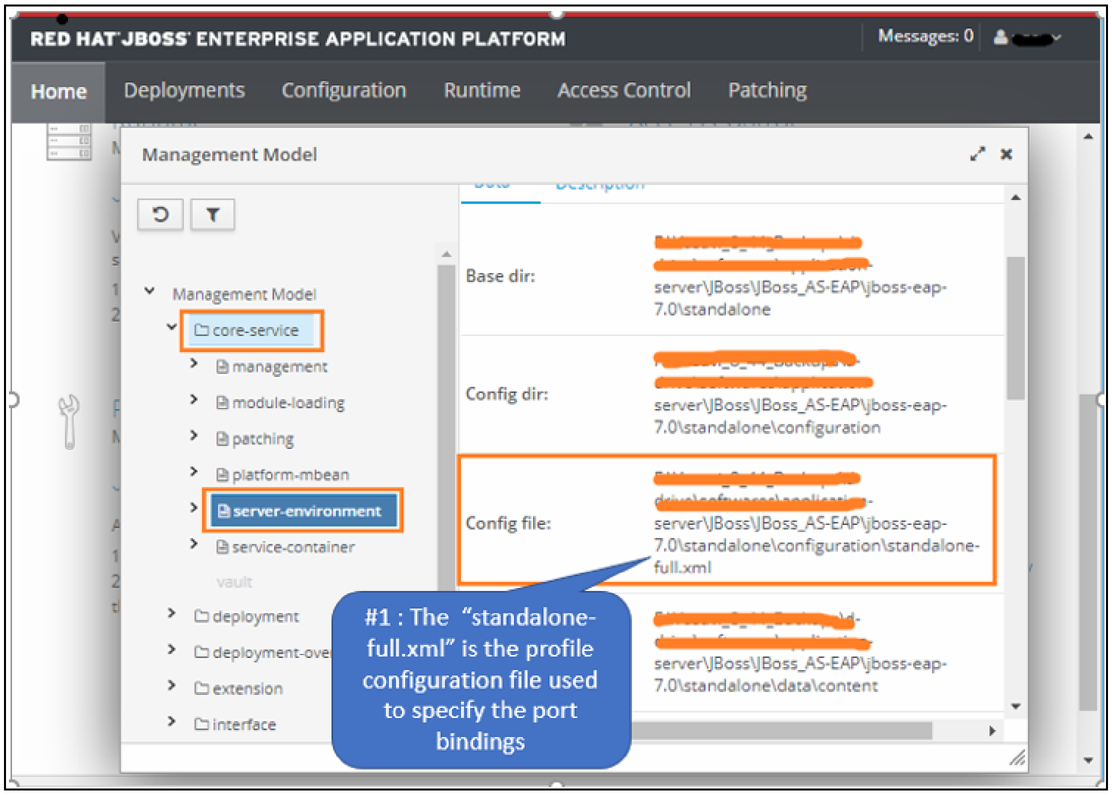
Deploy an application to the server
1. Use the 'Add Deployment' wizard to deploy the application
2. Enable the deployment

Define a datasource to be used by deployed applications. The proper JBDC driver must be deployed and registered.
1. Select the Datasources subsystem
2. Add a Non-XA or XA datasource
3. Use the 'Create Datasource' wizard to configure the datasource settings

Figure 2.2: Selecting the Management Model option from Tools menu

3. Then, in the tree-structure that appears in the left panel (see Figure 3), expand the

Management Model node. Expand the **core-service** sub-node within, and finally click the **server-environment** sub-node (see Figure 3). The right panel will then change to display the details about the target server. Browsing these details will lead you to the **Config file** of the standalone server (see Figure 3).



Base dir: [redacted] server\JBoss\JBoss_AS-EAP\jboss-eap-7.0\standalone

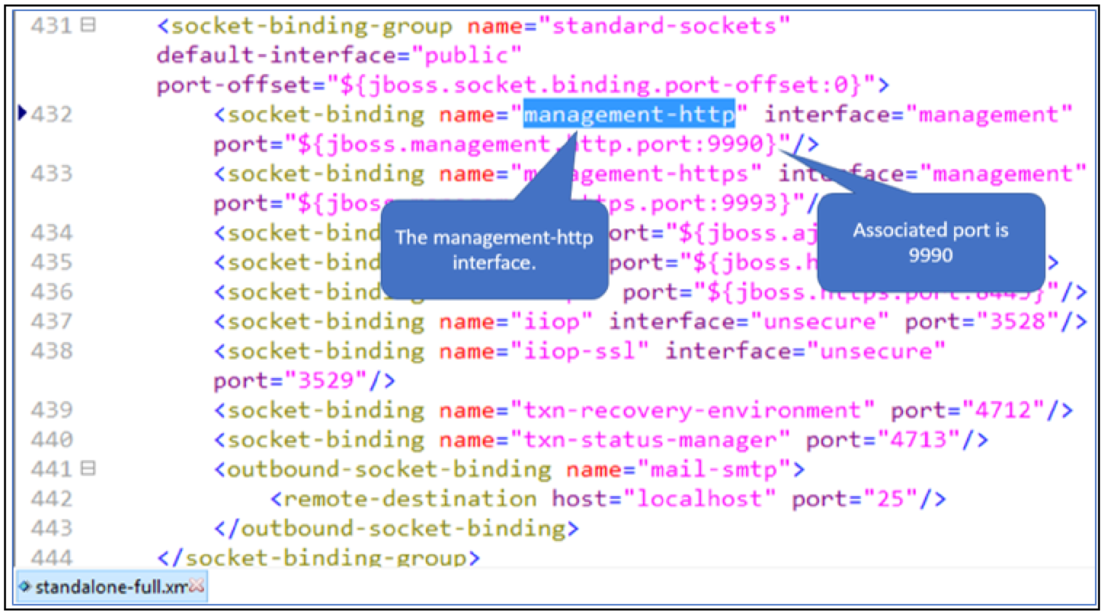
Config dir: [redacted] server\JBoss\JBoss_AS-EAP\jboss-eap-7.0\standalone\configuration

Config file: [redacted] server\JBoss\JBoss_AS-EAP\jboss-eap-7.0\standalone\configuration\standalone-full.xml

[redacted] server\JBoss\JBoss_AS-EAP\jboss-eap-7.0\standalone\data\content

Figure 2.3: Determining the configuration file of the monitored server

4. Next, open the indicated **Config file** using a text/XML editor (see Figure 4). In the file, look for the "socket-binding-group" attribute. Within that attribute, search for the "management-http" socket binding. Typically, in most systems, the HTTP REST API is enabled on the "management- http" socket binding only. In the example of Figure 4, the port used for management-http is set as *9990*. **This is the port that you should use when managing the JBoss EAP server using the eG admin interface.**



```
<socket-binding-group name="standard-sockets"
default-interface="public"
port-offset="${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management"
    port="${jboss.management.ttp.port:9990}"/>
    <socket-binding name="m    agement-https" int    face="management"
    port="${jbos              ps.port:9993}"/
    <socket-bind              ort="${jboss.aj
    <socket-bind              port="${jboss.h            >
    <socket-bind                port="${jboss.                  "/>
    <socket-binding name="iiop" interface="unsecure" port="3528"/>
    <socket-binding name="iiop-ssl" interface="unsecure"
    port="3529"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
```

Figure 2.4: Determining the management-http port number of the JBoss EAP server

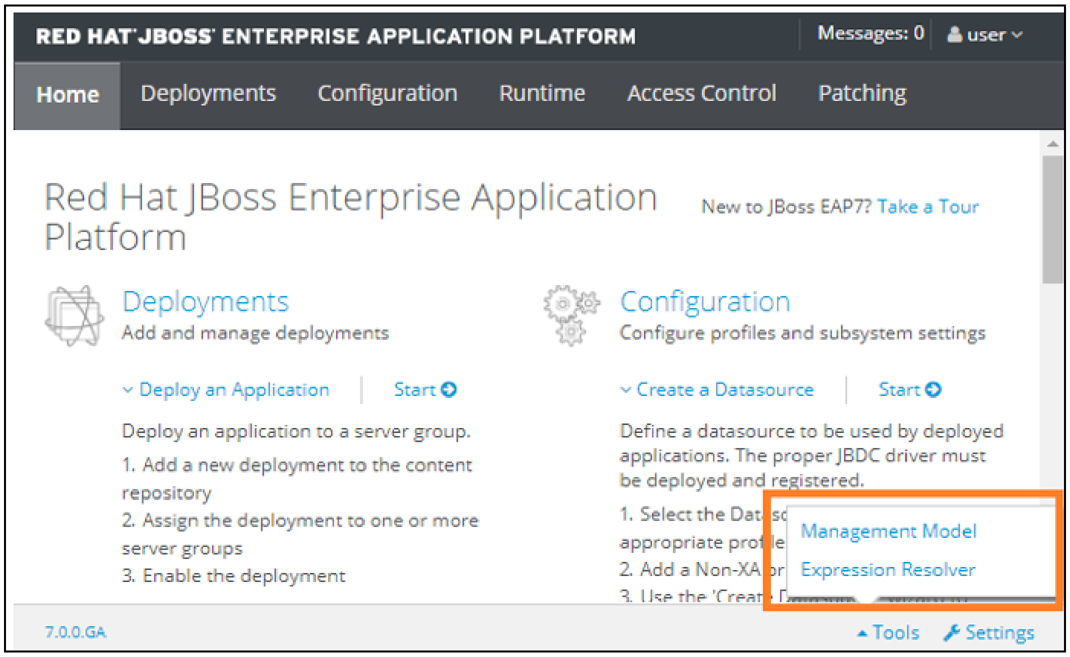
Note:

In some environments, the *port-offset* will also be used to indicate the port. In the environment illustrated by Figure 4 above, the port offset value is set to 0. So the management-http port remains same as “9990”. In case the port-offset is configured with a non-zero value, then you will have to add this port-offset value with the management-http port, and configure the resultant sum as the port number of the target JBoss server. For instance, if the port-offset value is 10 and the management-http port is 9990, then you will have to configure 10000 (9990+10) as the port number of the target JBoss EAP server in the eG admin interface. “10000”. Also, remember that the port-offset sometimes is set via Java Options.

In the *domain* operating mode, there can be ‘N’ server instances hosted on ‘M’ hosts. Within the M hosts, one acts as master (domain controller) and the remaining are slaves (host controllers). The management console is available only in *domain controller*.

If the JBoss EAP server to be monitored is operating in the *domain* mode, then, when adding that server to the eG Enterprise system, make sure that you configure the server with the **application/HTTP port**. To determine the application/HTTP port of a JBoss EAP server, follow the steps below:

1. First, login to the target server and identify the profile on which it runs. For that, login to the management console of the domain controller. From the **Tools** button at the bottom of the console, select the **Management Model** option (see Figure 5).

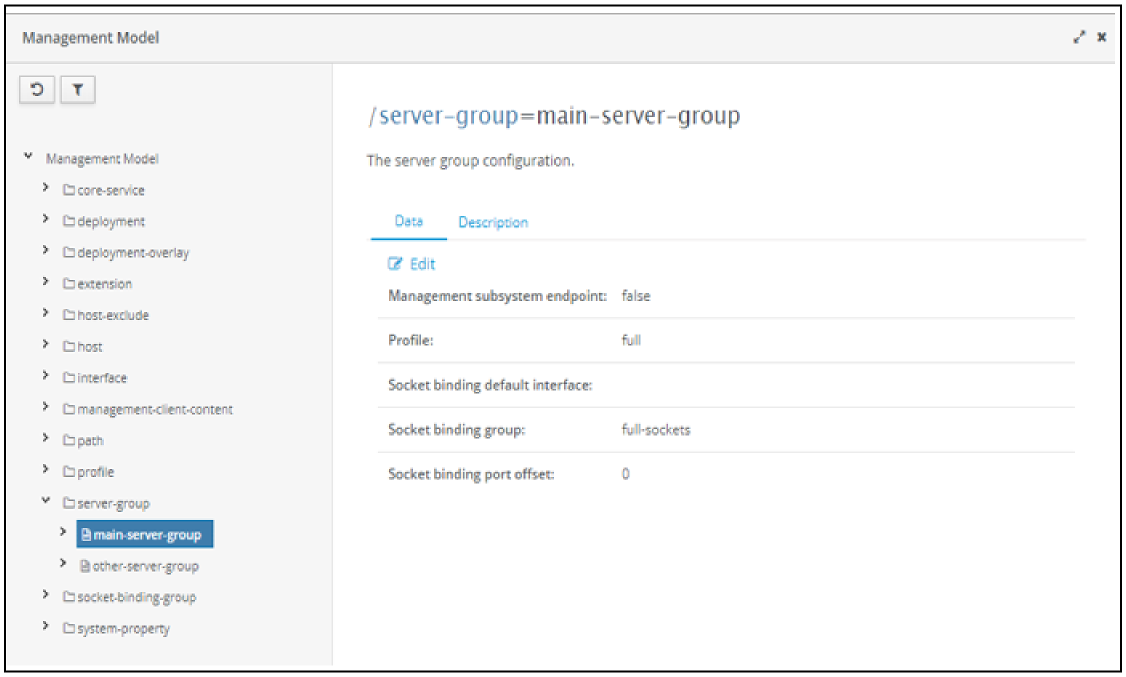


Deploy an application to a server group.
1. Add a new deployment to the content repository
2. Assign the deployment to one or more server groups
3. Enable the deployment

Define a datasource to be used by deployed applications. The proper JBDC driver must be deployed and registered.

Figure 2.5: Selecting the Management Model option from the Tools menu in the management console of the domain controller

2. Figure 6 then appears, where you will find a tree-structure in the left panel. Expand the **Management Model** node in the tree-structure, and then expand the **server-group** sub-node within. Finally, click the **main-server-group** sub-node. The right panel will change to display the details of the **main-server-group**. The **Profile** name will be displayed as part of these details. In the case of the example illustrated by Figure 6, full is the **Profile** name of the target JBoss EAP server.



/server-group=main-server-group

The server group configuration.

Management subsystem endpoint: false

Profile: full

Socket binding default interface:

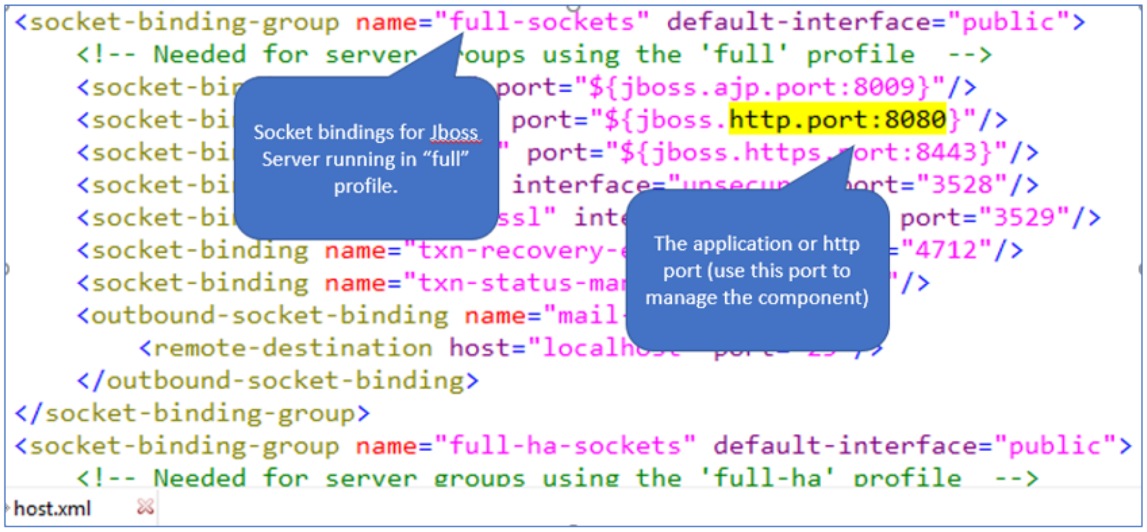
Socket binding group: full-sockets

Socket binding port offset: 0

Figure 2.6: Finding the profile name of the target JBoss EAP server

3. Next, open the <EAP_HOME>/domain/configuration/host.xml file on the target server, using a text/XML editor. Typically, each JBoss server host that is managed by a domain controller is associated with a host.xml file; this is the file from which each host reads its configuration.

4. In the host.xml file, look for a "socket-binding-group name" that corresponds to the profile name of the target server. For instance, if the target server runs on the "full" profile, then look for a "socket-binding-group name " that is set to "full-sockets" (see Figure 5).


Socket bindings for Jboss Server running in “full” profile.
The application or http port (use this port to manage the component)
host.xml

```
<socket-binding-group name="full-sockets" default-interface="public">
    <!-- Needed for server    oups using the 'full' profile  -->
    <socket-bi                     port="${jboss.ajp.port:8009}"/>
    <socket-bi                      port="${jboss.http.port:8080}"/>
    <socket-bi                     " port="${jboss.https  ort:8443}"/>
    <socket-bi                      interface="unsecur       ort="3528"/>
    <socket-bi                     ssl" inte               port="3529"/>
    <socket-binding name="txn-recovery-                    ="4712"/>
    <socket-binding name="txn-status-ma                    "/>
    <outbound-socket-binding name="mail
        <remote-destination host="localh
    </outbound-socket-binding>
</socket-binding-group>
<socket-binding-group name="full-ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full-ha' profile  -->
```

Figure 2.7: Determining the port number of a JBoss EAP server operating in the domain mode

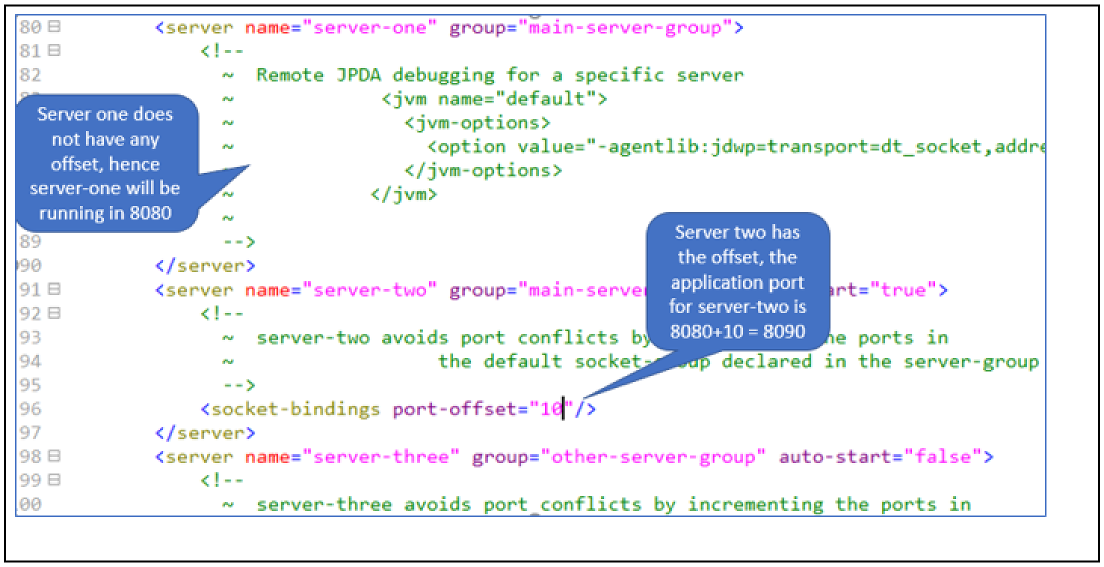
5. In the "full-sockets" section, you will find a specification of the following format (as indicated by Figure 5):

   "${jboss.http.port:<port_number>}"

   The <*port_number*> in this specification, is the application/HTTP port of the target JBoss server. When managing the JBoss EAP server using the eG admin interface, **make sure you configure the server with this port number.**

   Note:

   In some environments, the *port-offset* setting will also be used to indicate the port of a server. In the host.xml file, you will find separate sections for each server managed by the domain controller. Each of these sections will be headed , <*server name="<name of the server managed by the domain controller>"*. The port-offset setting will be part of some/all of these server-specific sections. For instance, in Figure 6 below, you can see 2 server-specific sections - one for *server-one* and another for *server-two*.



```
        <server name="server-one" group="main-server-group">
            <!--
              ~  Remote JPDA debugging for a specific server
              ~          <jvm name="default">
              ~            <jvm-options>
              ~              <option value="-agentlib:jdwp=transport=dt_socket,addre
              ~            </jvm-options>
              ~          </jvm>
              ~
              -->
        </server>
        <server name="server-two" group="main-serve              art="true">
            <!--
              ~  server-two avoids port conflicts by              he ports in
              ~                the default socket-     up declared in the server-group
              -->
            <socket-bindings port-offset="10"/>
        </server>
        <server name="server-three" group="other-server-group" auto-start="false">
            <!--
              ~  server-three avoids port conflicts by incrementing the ports in
```

Server one does not have any offset, hence server-one will be running in 8080
Server two has the offset, the application port for server-two is 8080+10 = 8090

Figure 2.8: Port-offset and its impact on the application/HTTP port configuration of a server

If the *port-offset* parameter is unavailable in a server-specific section, then go with the application/HTTP port shown by Figure 5. For instance, *server-one* in Figure 6 does not have a port-offset setting; this means that the application/HTTP port of that server will only be 8080 (as indicated by Figure 5). On the other hand, if a port-offset parameter is available in a server-specific section and is configured with a non-zero value, then you will have to add this port-offset value to the application/HTTP port of Figure 5, and configure the resultant sum as the port number of the target JBoss server. For instance, *server-two* in Figure 6 is configured with the port-offset value of 10. If the application/HTTP port is 8080, then you will have to configure 8090 (8080+10) as the port number of the target JBoss EAP server in the eG admin interface. Also, remember that the *port-offset* sometimes is set via Java Options.

2.1.2 Creating a New User in the JBoss EAP Server

The management interfaces in a JBoss Enterprise Application Platform are secured by default, and hence there is no default user. This is a security precaution, to prevent security breaches from remote systems due to simple configuration errors. Without a user, administrators may not be able to use the web-based Management Console of the JBoss AS/EAP server. It is therefore mandatory to create an initial administrative user, who will be able to use the web-based Management Console and remote instances of the Management CLI to configure and administer JBoss from remote systems. This user can be either the **Management user** or the **Application user**. A **Management user** is added to the **ManagementRealm** of the JBoss AS/EAP server and is authorized to perform management operations using the web-based management console or the Management CLI. On the other hand, the **Application user** is added to the **ApplicationRealm** and this user has no particular permissions and is provided for use with applications.

In order to monitor the JBoss AS/EAP server, the eG agent has to be configured with the credentials of a Management/Application user, so that it can access the JBoss management console and management CLI for running commands and pulling out desired metrics. While you can use any existing management/application user for this purpose, **it is recommended that you create a new user.** Typically, the type of user you need to create will vary according to the operating mode (standalone or domain) of the target JBoss EAP server.

If the server being monitored is running in the standalone mode, then you need to create a Management user in the ManagementRealm, and pass the credentials of this user to the eG tests. On the other hand, if the server being monitored is running in the domain mode, then create a new Application user in the ApplicationRealm, and configure eG tests with the credentials of this user.

To create a Management user on a standalone server, follow the steps below:

1. In order to add a user to the JBoss AS/EAP server, you will require either one of the following files available in the **<JBOSS_INSTALL_DIR\bin>** location:

   - **add-user.sh**
   - **add-user.bat**

2. Execute the **add-user.bat** file in case the JBoss AS/EAP server is installed on a Windows environment and execute the **add-user.sh** file in case the JBoss AS/EAP server is installed on a Linux environment.
3. Once the file is executed, you will be required to choose the type of the user that you wish to add. To add a **Management User**, specify **a**.

```
What type of user do you wish to add?
Management User (mgmt-user.properties)
Application User (application-users.properties)
(a): a
```

4. Specify the credentials of the user that you wish to add. **Make a note of the 'Username' and 'Password' you provide here, as these are the credentials that you will have to pass to the eG tests.**

```
Username: elvis
Password:*****
Re-enter Password:****
```

5. If the **Management User** option is chosen at step 3, then the user will be added to the **ManagementRealm** of the JBoss AS/EAP server. Specify **yes** to confirm the same.

```
About to add user ‘elvis’ for realm ‘ManagementRealm’
Is this correct yes/no? yes
```

6. Now, the user will be added to the **mgmt-users.properties** of the JBoss AS/EAP server installation. At the next prompt, indicate whether the user being added represents another instance of JBoss EAP, which must be able to authenticate to join a cluster as a member. If you specify **yes** here, then the user you are adding will be designated for this purpose. If you specify **no** here, then the user will not be able to communicate with other server instances in a cluster setup.

```
Is this new user going to be used for one AS process to connect to another AS process?
e.g. slave domain controller? yes/no? no
```

7. If you specify **yes** in Step 7, a secret value will appear which needs to be copied and stored separately for future reference. Whenever a new JBoss AS/EAP instance is added in a domain, specifying the secret value while configuring the new instance will let the new instance be the slave of the JBoss AS/EAP installation in a cluster setup i.e., a user will be allowed to communicate with all the associated instances once the secret value is shared.

```
To represent the user add the server-identities definition (secret
value=”AWEStanW4cmziQ”).
```

To create an Application user on a server operating in the domain mode, follow the steps below:

1. In order to add a user to the JBoss AS/EAP server, you will require either one of the following files available in the **<JBOSS_INSTALL_DIR\bin>** location:

   - **add-user.sh**
   - **add-user.bat**

2. Execute the **add-user.bat** file in case the JBoss AS/EAP server is installed on a Windows environment and execute the **add-user.sh** file in case the JBoss AS/EAP server is installed on a Linux environment.

3. Once the file is executed, you will be required to choose the type of the user that you wish to add. To add an **ApplicationUser**, specify **b**.

```
What type of user do you wish to add?
Management User (mgmt-user.properties)
Application User (application-users.properties)
(a): b
```

4. Specify the credentials of the user that you wish to add. **Make a note of the 'Username' and 'Password' you provide here, as these are the credentials that you will have to pass to the eG tests if the target server is operating in the domain node.**

```
Username: elvis
Password:*****
Re-enter Password:****
```

5. If the **Application User** option is chosen at step 3, then the user will be added to the **ApplicationRealm** of the JBoss AS/EAP server. Specify **yes** to confirm the same.

```
About to add user ‘elvis’ for realm ‘ApplicationRealm’
Is this correct yes/no? yes
```

6. If you wish to associate the user with a role, then you can provide a comma-separated list of roles. If you do not wish to associate the user with any role, then simply press Enter at this prompt. This will trigger user creation.

```
What roles do you want this user to belong to? (Please enter a comma separated list,
or leave blank for none):
```

7. Now, the user will be added to the **application-users.pr o perties** and the **application-roles.properties** of the JBoss AS/EAP server installation. At the next prompt, indicate whether the user being added represents another instance of JBoss EAP, which must be able to

authenticate to join a cluster as a member. If you specify **yes** here, then the user you are adding will be designated for this purpose. If you specify **no** here, then the user will not be able to communicate with other server instances in a cluster setup.

```
Is this new user going to be used for one AS process to connect to another AS process?
e.g. slave domain controller? yes/no? yes
```

8. If you specify **yes** in Step 7, a secret value will appear which needs to be copied and stored separately for future reference. Whenever a new JBoss AS/EAP instance is added in a domain, specifying the secret value while configuring the new instance will let the new instance be the slave of the JBoss AS/EAP installation in a cluster setup i.e., a user will be allowed to communicate with all the associated instances once the secret value is shared.

```
To represent the user add the server-identities definition (secret
value="AWEStanW4cmziQ").
```

2.1.3 Configuring JVM Monitoring for a JBoss EAP Server

To collect JVM-related metrics, the eG agent connects to the JRE of the JBoss EAP Server via JMX. To enable JVM monitoring therefore, JMX support is required.

By default, JMX support is enabled on JBoss EAP. However, to enable the eG agent to use JMX, the following requirements have to be fulfilled:

- The JVM tests that the eG agent runs should be configured with a JMX Remote Port - this is the port at which the JMX listens for requests from remote hosts. You need to determine the correct JMX remote port to configure.
- If the target server is operating in the **domain mode**, then **management-http endpoint has to be disabled on the domain controller**.

Each of the above-mentioned requirements are dealt with elaborately in the topics that follow.

2.1.3.1 Determining the JMX Remote Port

By default, JMX support is enabled on port *9990*. This means that in all the JVM tests that the eG agent runs, 9990 is by default set as the **JMX Remote Port**. However, depending upon the operating mode of the target JBoss EAP server, you may have to override this default setting by specifying a different port number at the time of test configuration.

Typically, JBoss EAP operates in one of the following modes:

- **Standalone server** operating mode for managing discrete instances
- **Managed domain** operating mode for managing groups of instances from a single control point

To know in which mode the JBoss EAP server you want to monitor is operating, login to the JBoss Management Console and click the **Runtime** option in its main menu. When Figure 1 appears, take a look at the options displayed in the left panel of Figure 1.

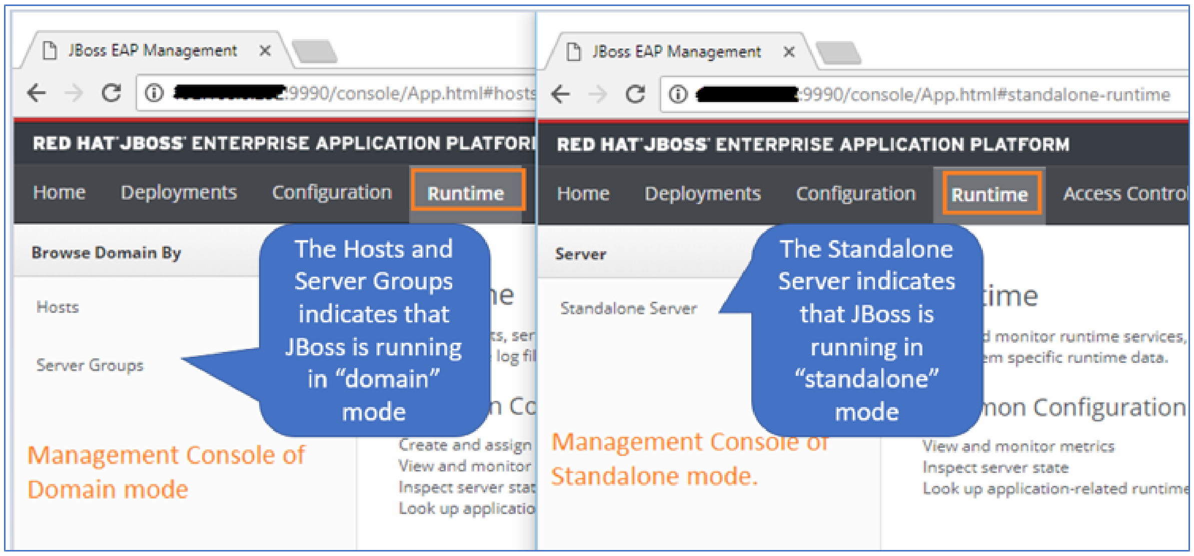

JBoss EAP Management
[redacted]:9990/console/App.html#hosts
RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM
Home
Deployments
Configuration
Runtime
Browse Domain By
Hosts
Server Groups
The Hosts and Server Groups indicates that JBoss is running in "domain" mode
Management Console of Domain mode
JBoss EAP Management
[redacted]:9990/console/App.html#standalone-runtime
RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM
Home
Deployments
Configuration
Runtime
Access Control
Server
Standalone Server
The Standalone Server indicates that JBoss is running in "standalone" mode
Management Console of Standalone mode.
View and monitor metrics
Inspect server state
Look up application-related runtime

Figure 2.9: How to determine the operating mode of JBoss EAP

If you see **Hosts and Server Groups** displayed therein, it indicates that the target JBoss EAP server is running in the *domain* mode. If you see **Standalone Server** there instead, it implies that the target server is running in the *standalone* mode.

If the server to be monitored is operating in the *standalone* mode, then follow the steps below to determine the port on which JMX is enabled:

1. First, determine the profile on which the JBoss server is running. In the standalone operating mode, each profile is associated with a configuration file (*.xml). This configuration file is where the various port bindings are configured. The table below lists the profile names and the configuration (XML) file that corresponds to each:

Profile Name	**Configuration File Name**
Default	standalone.xml
full	standalone-full.xml
ha	standalone-ha.xml
Full-ha	standalone-full-ha.xml

2. Next, determine which configuration file is in use on the target server and zoom into the contents of that file to identify the port number on which JMX is enabled. To know the configuration file, login to the JBoss Management Console of the target server, click on the **Tools** menu at the bottom of the console and choose the **Management Model** option (as indicated by Figure 2).

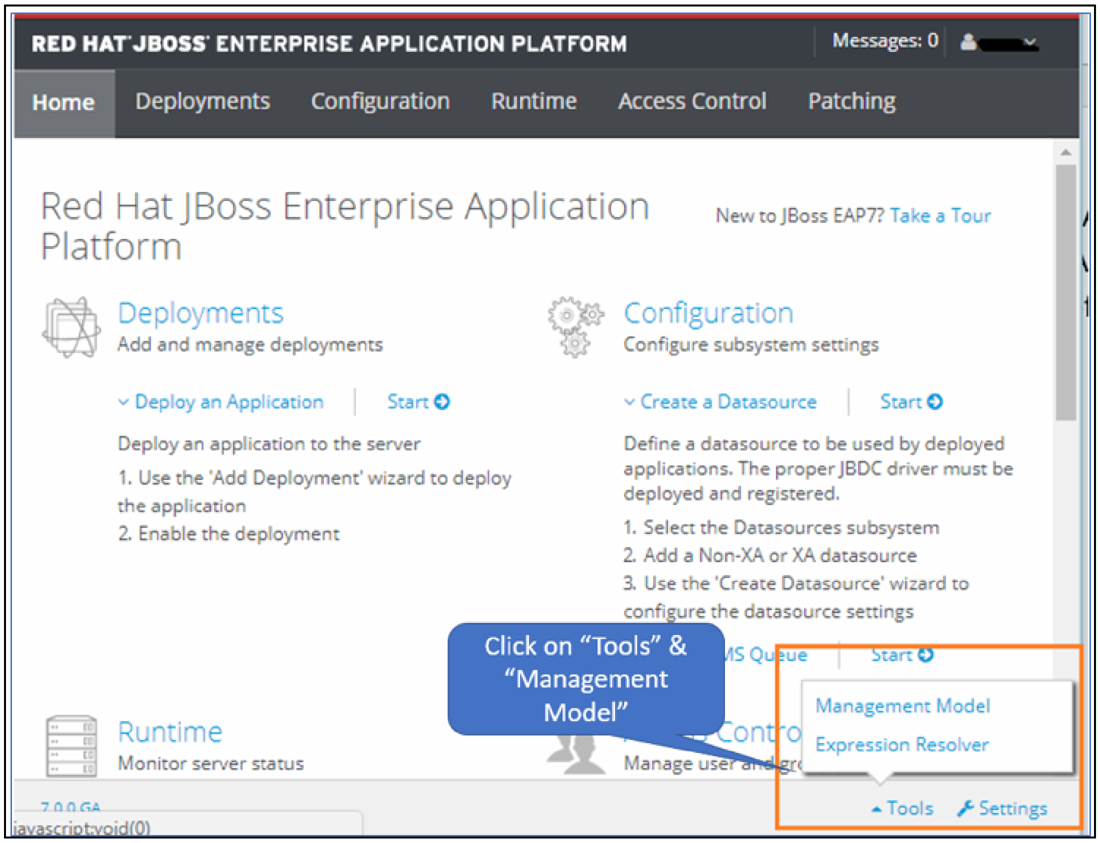
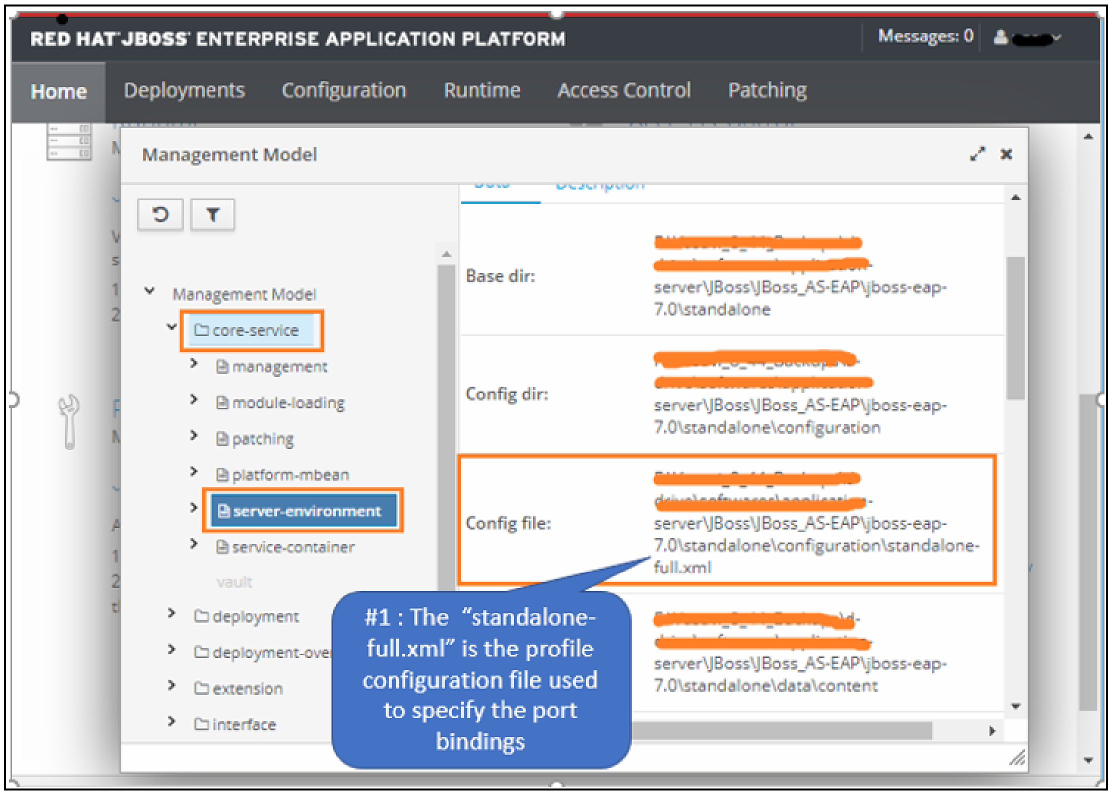

RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM
Messages: 0
Home
Deployments
Configuration
Runtime
Access Control
Patching
Red Hat JBoss Enterprise Application Platform
New to JBoss EAP7? Take a Tour
Deployments
Add and manage deployments
Configuration
Configure subsystem settings
Deploy an Application
Start
Create a Datasource
Start
Deploy an application to the server
1. Use the 'Add Deployment' wizard to deploy the application
2. Enable the deployment
Define a datasource to be used by deployed applications. The proper JBDC driver must be deployed and registered.
1. Select the Datasources subsystem
2. Add a Non-XA or XA datasource
3. Use the 'Create Datasource' wizard to configure the datasource settings
Click on "Tools" & "Management Model"
Management Model
Expression Resolver
Runtime
Monitor server status
Manage user
Tools
Settings
7.0.0.GA
javascript:void(0)

Figure 2.10: Selecting the Management Model option from Tools menu

3. Then, in the tree-structure that appears in the left panel (see Figure 3), expand the **Management Model** node. Expand the **core-service** sub-node within, and finally click the **server-environment** sub-node (see Figure 3). The right panel will then change to display the details about the target server. Browsing these details will lead you to the **Config file** of the standalone server (see Figure 3).



Base dir: [redacted] server\JBoss\JBoss_AS-EAP\jboss-eap-7.0\standalone

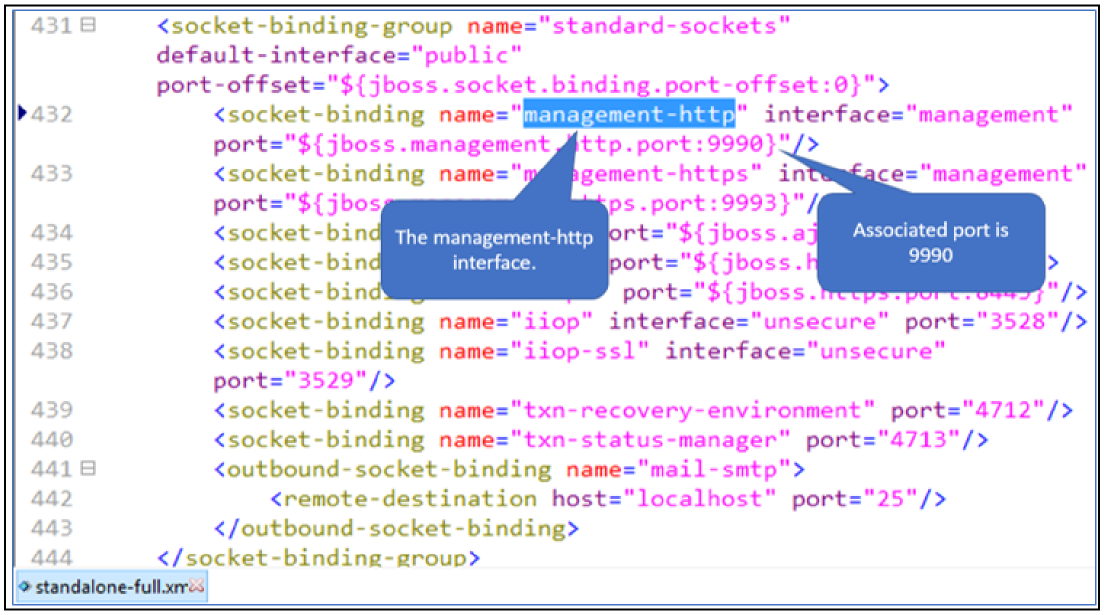
Config dir: [redacted] server\JBoss\JBoss_AS-EAP\jboss-eap-7.0\standalone\configuration

Config file: [redacted] server\JBoss\JBoss_AS-EAP\jboss-eap-7.0\standalone\configuration\standalone-full.xml

[redacted] server\JBoss\JBoss_AS-EAP\jboss-eap-7.0\standalone\data\content

Figure 2.11: Determining the configuration file of the monitored server

4. Next, open the indicated **Config file** using a text/XML editor (see Figure 4). In the file, look for the "socket-binding-group" attribute. Within that attribute, search for the "management-http" socket binding. Typically, in most systems, JMX is enabled on the "management-http" socket binding only - i.e., JMX listens on the management-http port only. In the example of Figure 4, the port used for management-http is set as *9990*. **This is the port that you should use when configuring the JMX Remote Port parameter of the eG tests.**



```
    <socket-binding-group name="standard-sockets"
    default-interface="public"
    port-offset="${jboss.socket.binding.port-offset:0}">
        <socket-binding name="management-http" interface="management"
        port="${jboss.management.  ttp.port:9990}"/>
        <socket-binding name="m     agement-https" int     face="management"
        port="${jbos               ps.port:9993}"/
        <socket-bind                ort="${jboss.aj
        <socket-bind                port="${jboss.h                  >
        <socket-bind                 port="${jboss.                 "/>
        <socket-binding name="iiop" interface="unsecure" port="3528"/>
        <socket-binding name="iiop-ssl" interface="unsecure"
        port="3529"/>
        <socket-binding name="txn-recovery-environment" port="4712"/>
        <socket-binding name="txn-status-manager" port="4713"/>
        <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
        </outbound-socket-binding>
    </socket-binding-group>
```

Figure 2.12: Determining the JMX Remote Port of the JBoss EAP server

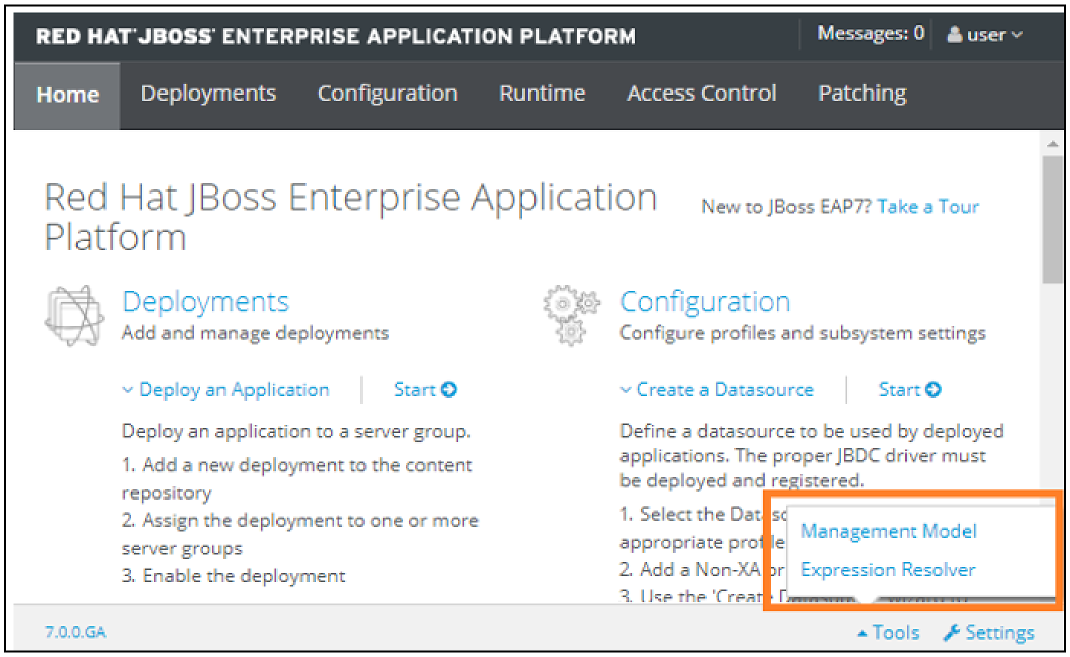
Note:

In some environments, the *port-offset* will also be used to indicate the port. In the environment illustrated by Figure 4 above, the port offset value is set to 0. So the management-http port remains same as “9990”. In case the port-offset is configured with a non-zero value, then you will have to add this port-offset value with the management-http port, and configure the resultant sum as the JMX remote port. For instance, if the port-offset value is 10 and the management-http port is 9990, then you will have to configure 10000 (9990+10) as the JMX remote port. Also, remember that the port-offset sometimes is set via Java Options.

In the *domain* operating mode, there can be ‘N’ server instances hosted on ‘M’ hosts. Within the M hosts, one acts as master (domain controller) and the remaining are slaves (host controllers). The management console is available only in *domain controller*.

If the JBoss EAP server to be monitored is operating in the *domain* mode, then, when enabling JVM monitoring for that server, make sure that you configure the JMX remote port parameter of the eG tests with the **application/HTTP port**. To determine the application/HTTP port of a JBoss EAP server, follow the steps below:

1. First, login to the target server and identify the profile on which it runs. For that, login to the management console of the domain controller. From the **Tools** button at the bottom of the console, select the **Management Model** option (see Figure 5).



Red Hat JBoss Enterprise Application Platform

Deployments
Add and manage deployments

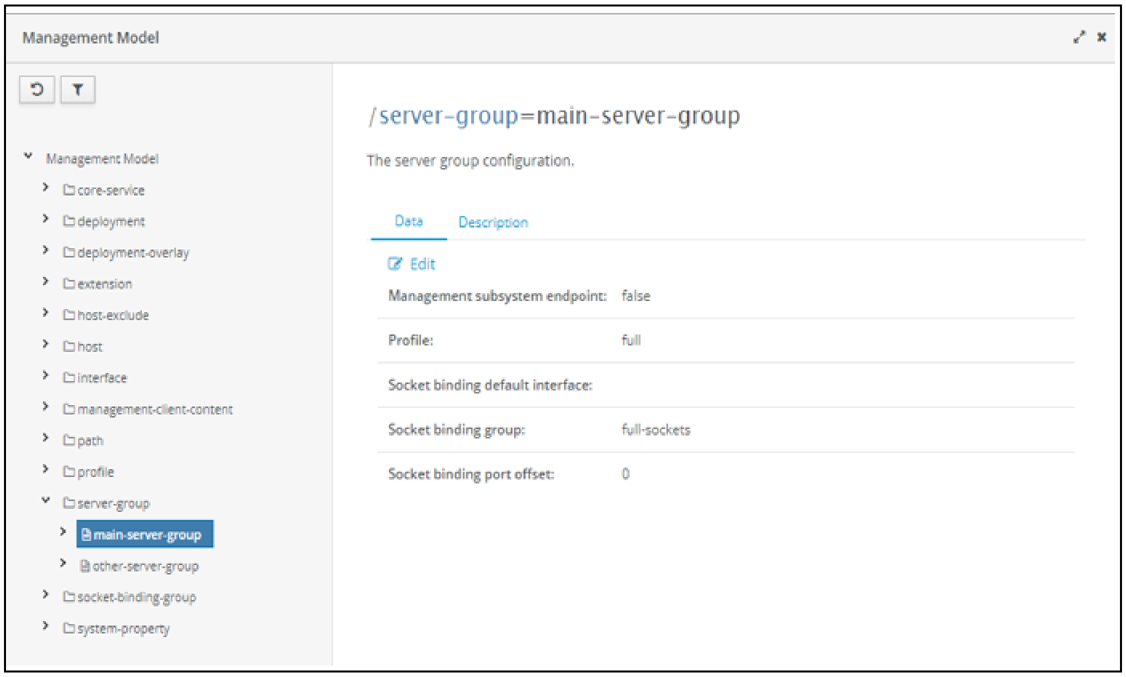
Deploy an application to a server group.
1. Add a new deployment to the content repository
2. Assign the deployment to one or more server groups
3. Enable the deployment

Configuration
Configure profiles and subsystem settings

Define a datasource to be used by deployed applications. The proper JBDC driver must be deployed and registered.
1. Select the Dat
appropriate prof
2. Add a Non-XA
3. Use the 'Creat

Figure 2.13: Selecting the Management Model option from the Tools menu in the management console of the domain controller

2. Figure 6 then appears, where you will find a tree-structure in the left panel. Expand the **Management Model** node in the tree-structure, and then expand the **server-group** sub-node within. Finally, click the **main-server-group** sub-node. The right panel will change to display the details of the **main-server-group**. The **Profile** name will be displayed as part of these details. In the case of the example illustrated by Figure 6, *full* is the **Profile** name of the target JBoss EAP server.



/server-group=main-server-group

The server group configuration.

Management subsystem endpoint: false

Profile: full

Socket binding default interface:

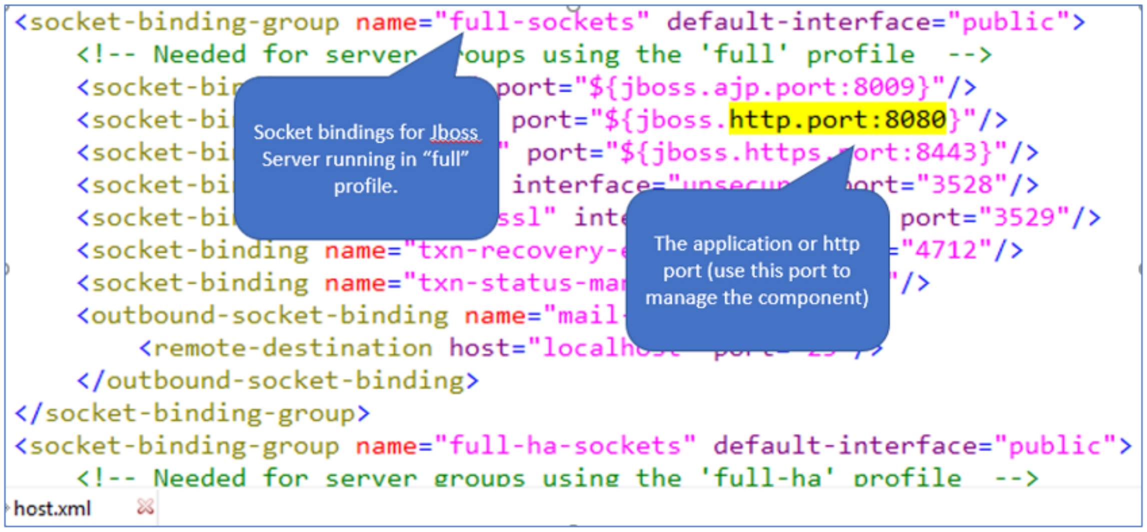
Socket binding group: full-sockets

Socket binding port offset: 0

Figure 2.14: Finding the profile name of the target JBoss EAP server

3. Next, open the <EAP_HOME>/domain/configuration/host.xml file on the target server, using a text/XML editor. Typically, each JBoss server host that is managed by a domain controller is associated with a host.xml file; this is the file from which each host reads its configuration.

4. In the host.xml file, look for a "socket-binding-group name" that corresponds to the profile name of the target server. For instance, if the target server runs on the "full" profile, then look for a "socket-binding-group name " that is set to "full-sockets" (see Figure 5).



```
<socket-binding-group name="full-sockets" default-interface="public">
    <!-- Needed for server   oups using the 'full' profile  -->
    <socket-bi                port="${jboss.ajp.port:8009}"/>
    <socket-bi                 port="${jboss.http.port:8080}"/>
    <socket-bi                "  port="${jboss.https  ort:8443}"/>
    <socket-bi                 interface="unsecur     ort="3528"/>
    <socket-bi                ssl" inte               port="3529"/>
    <socket-binding name="txn-recovery-               ="4712"/>
    <socket-binding name="txn-status-ma               '/>
    <outbound-socket-binding name="mail
        <remote-destination host="localh
    </outbound-socket-binding>
</socket-binding-group>
<socket-binding-group name="full-ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full-ha' profile  -->
```

Socket bindings for Jboss Server running in "full" profile.
The application or http port (use this port to manage the component)
host.xml

Figure 2.15: Determining the port number of a JBoss EAP server operating in the domain mode

- In the "full-sockets" section, you will find a specification of the following format (as indicated by Figure 5):

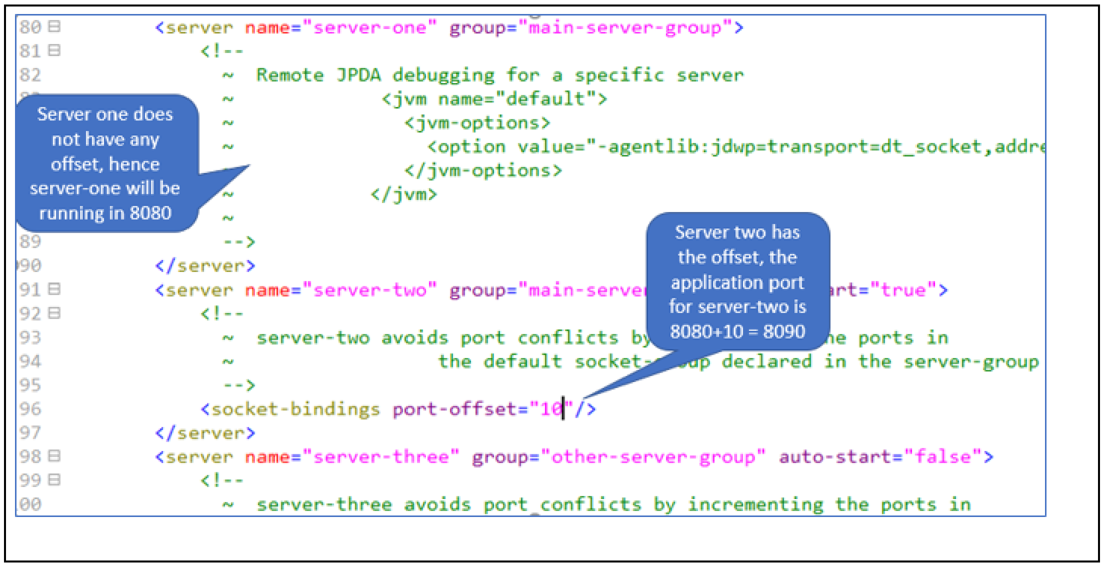
  "${jboss.http.port:<port_number>}"

  The <*port_number*> in this specification, is the application/HTTP port of the target JBoss server. When configuring the eG tests, **make sure you specify this port number as the JMX Remote Port of the target JBoss EAP server.**

  Note:

  In some environments, the combination of *http.port* and *port-offset* setting may have to be used to determine the JMX Remote Port.

  In the host.xml file, you will find separate sections for each server managed by the domain controller. Each of these sections will be headed , <*server name="<name of the server managed by the domain controller>"*. The *port-offset* setting will be part of some/all of these server-specific sections. For instance, in Figure 6 below, you can see 2 server-specific sections - one for *server-one* and another for *server-two*.



```
        <server name="server-one" group="main-server-group">
            <!--
              ~  Remote JPDA debugging for a specific server
              ~             <jvm name="default">
              ~               <jvm-options>
              ~                 <option value="-agentlib:jdwp=transport=dt_socket,addre
              ~               </jvm-options>
              ~            </jvm>
              ~
              -->
        </server>
        <server name="server-two" group="main-serve                  rt="true">
            <!--
              ~  server-two avoids port conflicts by                 ne ports in
              ~                 the default socket-     up declared in the server-group
              -->
            <socket-bindings port-offset="10"/>
        </server>
        <server name="server-three" group="other-server-group" auto-start="false">
            <!--
              ~  server-three avoids port_conflicts by incrementing the ports in
```

Server one does not have any offset, hence server-one will be running in 8080
Server two has the offset, the application port for server-two is 8080+10 = 8090

Figure 2.16: Port-offset and its impact on the application/HTTP port configuration of a server

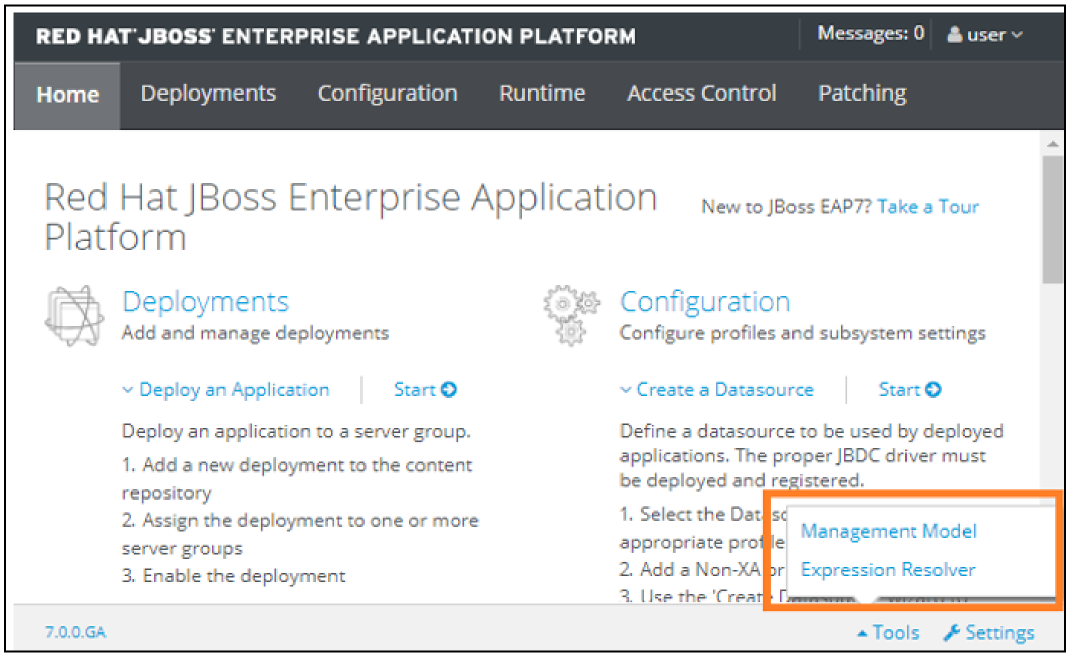
If the *port-offset* parameter is unavailable in a server-specific section, then the JMX remote port will be the *http.port* shown by Figure 5. For instance, *server-one* in Figure 6 does not have a port-offset setting; this means that the JMX remote port of that server will only be 8080 (as indicated by Figure 5). On the other hand, if a port-offset parameter is available in a server-specific section and is configured with a non-zero value, then you will have to add this port-offset value to the http.port setting of Figure 5, and configure the resultant sum as the JMX remote port of the target JBoss server. For instance, *server-two* in Figure 6 is configured with the port-offset value of 10. If the http.port is set as 8080, then you will have to configure 8090 (8080+10) as the JMX remote port. Also, remember that the *port-offset* sometimes is set via Java Options.

2.1.3.2 Disabling the management-http Endpoint for a Server Operating in the Domain Mode

In case of a standalone server, all management activities are performed via the management-http port on that server. Moreover, since JMX also listens on that port, JVM monitoring for a standalone server is also performed only via its management-http port.

In case of a domain/cluster setup, all management activities are performed centrally on the domain controller. Clients interact with the individual servers/server instances in the domain only via the application/HTTP port on each server/instance; not via any management-http port. This is why, the eG agent should use the application/HTTP port of each server/server instance to monitor the health of its JVM. For this purpose, you need to make sure that the inbound and outgoing JMX connections for a domain do not use the management-http port. To achieve this, you need to disable the management-http endpoint on the remoting connector for the JMX sub-system of the entire domain. The steps to achieve this are given below:

1. First, login to the domain controller and identify the profile on which it runs. For that, login to the management console of the domain controller. From the **Tools** button at the bottom of the console, select the **Management Model** option (see Figure 5).

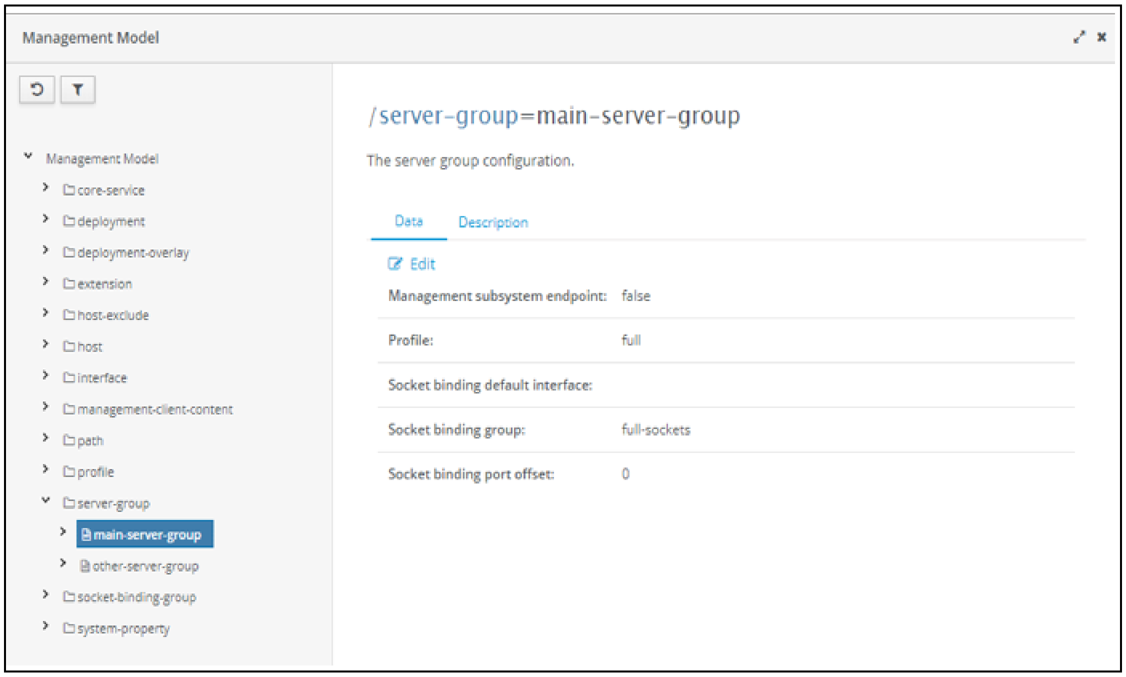


Deploy an application to a server group.
1. Add a new deployment to the content repository
2. Assign the deployment to one or more server groups
3. Enable the deployment

Define a datasource to be used by deployed applications. The proper JBDC driver must be deployed and registered.

Figure 2.17: Selecting the Management Model option from the Tools menu in the management console of the domain controller

2. Figure 6 then appears, where you will find a tree-structure in the left panel. Expand the **Management Model** node in the tree-structure, and then expand the **server-group** sub-node within. Finally, click the **main-server-group** sub-node. The right panel will change to display the details of the **main-server-group**. The **Profile** name will be displayed as part of these details. In the case of the example illustrated by Figure 6, *full* is the **Profile** name of the target JBoss EAP server.



/server-group=main-server-group

The server group configuration.

Management subsystem endpoint:	false
Profile:	full
Socket binding default interface:	
Socket binding group:	full-sockets
Socket binding port offset:	0

Figure 2.18: Finding the profile name of the target JBoss EAP server

3. Next, open the <EAP_HOME>/domain/configuration/domain.xml file using a text/XML editor. In the file, first locate the section that corresponds to the profile of the server. For instance, if the profile is "full", then look for the section headed *<profile name="full">*.

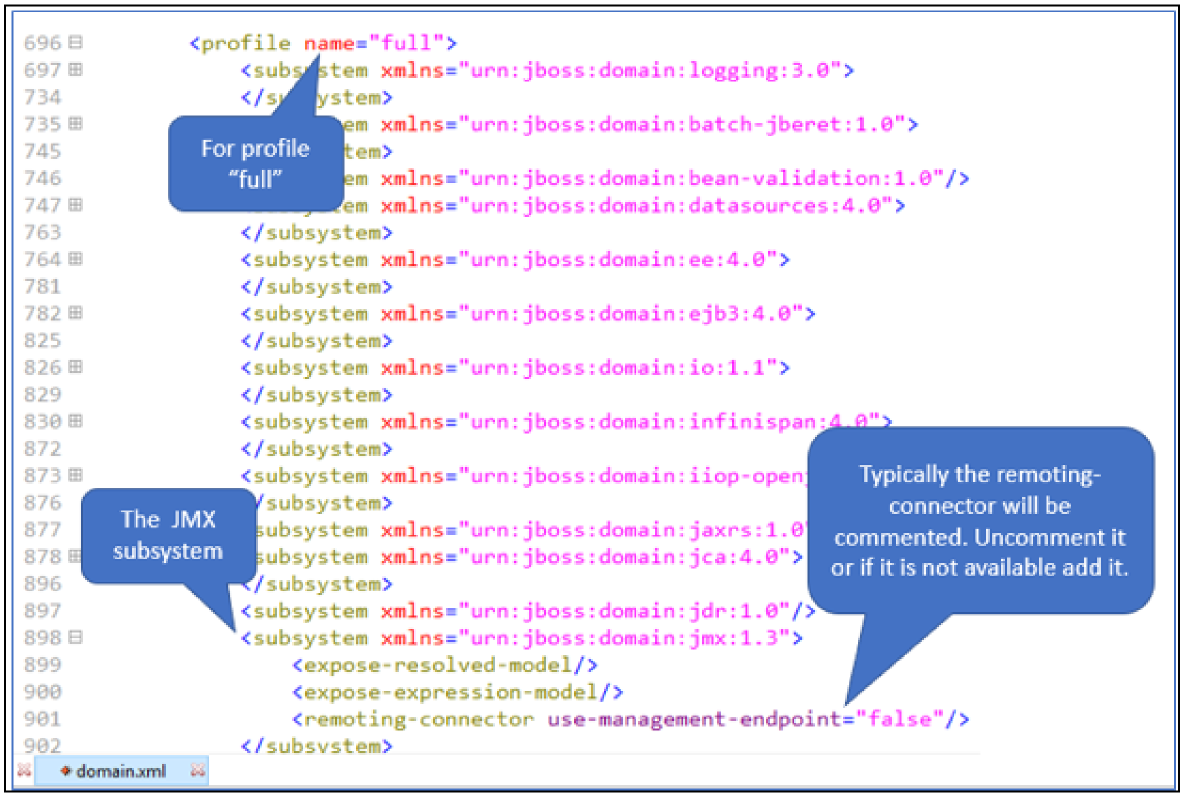
4. In that section, look for the following entry:

   <subsystem xmlns="urn:jboss:domain:jmx:1.3">

5. What follows the above entry (in the file) is the JMX configuration of the domain. Therefore, search for the following entry in the JMX configuration:

   <remoting-connector use-management-endpoint="false">

6. Typically, the *remoting-connector* entry will be commented. If so, uncomment it. If you do not find such an entry at all, then add one as indicated by Figure 11.


For profile "full"
The JMX subsystem
Typically the remoting-connector will be commented. Uncomment it or if it is not available add it.
domain.xml

```
        <profile name="full">
            <subs   stem xmlns="urn:jboss:domain:logging:3.0">
            </s   ystem>
                   em xmlns="urn:jboss:domain:batch-jberet:1.0">
                   tem>
                   em xmlns="urn:jboss:domain:bean-validation:1.0"/>
                   em xmlns="urn:jboss:domain:datasources:4.0">
            </subsystem>
            <subsystem xmlns="urn:jboss:domain:ee:4.0">
            </subsystem>
            <subsystem xmlns="urn:jboss:domain:ejb3:4.0">
            </subsystem>
            <subsystem xmlns="urn:jboss:domain:io:1.1">
            </subsystem>
            <subsystem xmlns="urn:jboss:domain:infinispan:4.0">
            </subsystem>
            <subsystem xmlns="urn:jboss:domain:iiop-open
             /subsystem>
             subsystem xmlns="urn:jboss:domain:jaxrs:1.0
             subsystem xmlns="urn:jboss:domain:jca:4.0">
             /subsystem>
            <subsystem xmlns="urn:jboss:domain:jdr:1.0"/>
            <subsystem xmlns="urn:jboss:domain:jmx:1.3">
                <expose-resolved-model/>
                <expose-expression-model/>
                <remoting-connector use-management-endpoint="false"/>
            </subsystem>
```

Figure 2.19: Disabling the management endpoint for the domain

7. Finally, save the file.

2.1.4 Configuring Java Business Transaction Monitoring on JBoss EAP

eG Enterprise' Java Business Transaction Monitor (BTM) tracks individual transactions to JBoss EAP in real-time, captures the time taken by each transaction, traces the journey of each transaction and the path it takes, and accurately pinpoints where and why a transaction slowed down.

By default, the eG BTM capability is disabled on JBoss EAP. To enable this capability, you first need to determine the operating mode of the target server.

Typically, JBoss EAP operates in one of the following modes:

- **Standalone server** operating mode for managing discrete instances
- **Managed domain** operating mode for managing groups of instances from a single control point

To know in which mode the JBoss EAP server you want to monitor is operating, login to the JBoss Management Console and click the **Runtime** option in its main menu. When Figure 1 appears, take a look at the options displayed in the left panel of Figure 1.

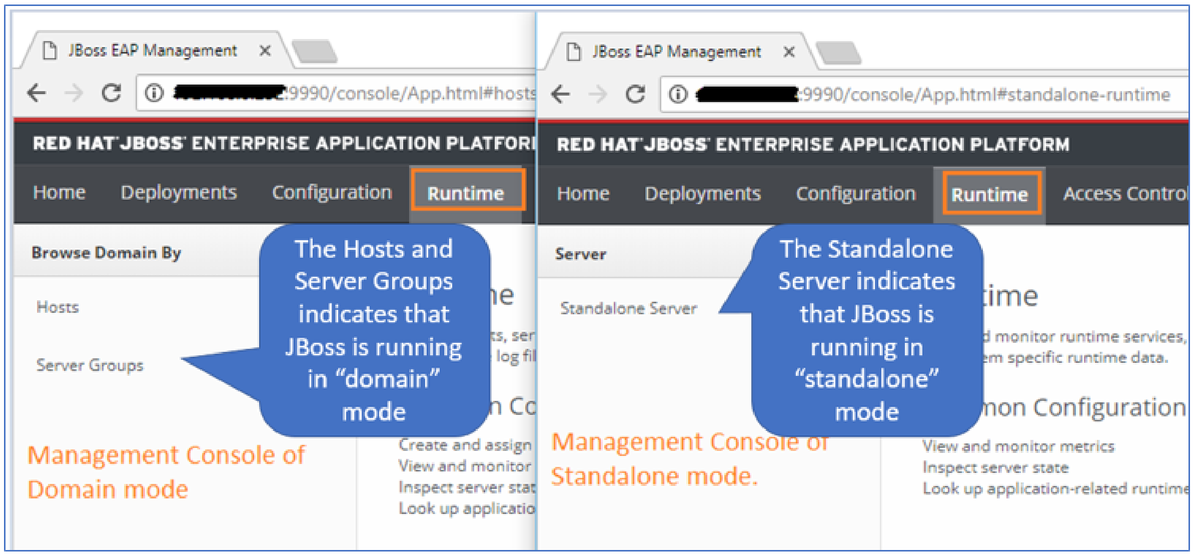

JBoss EAP Management
[redacted]:9990/console/App.html#hosts
RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM
Home Deployments Configuration Runtime
Browse Domain By
Hosts
Server Groups
The Hosts and Server Groups indicates that JBoss is running in "domain" mode
Management Console of Domain mode
JBoss EAP Management
[redacted]:9990/console/App.html#standalone-runtime
RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM
Home Deployments Configuration Runtime Access Control
Server
Standalone Server
The Standalone Server indicates that JBoss is running in "standalone" mode
Management Console of Standalone mode.

Figure 2.20: How to determine the operating mode of JBoss EAP

If you see **Hosts and Server Groups** displayed therein, it indicates that the target JBoss EAP server is running in the *domain* mode. If you see **Standalone Server** there instead, it implies that the target server is running in the *standalone* mode.

Now, proceed to BTM-enable the server.

To BTM-enable a JBoss EAP server operating on a Windows host, follow the steps detailed below.

1. Manage the JBoss EAP server using the eG administrative interface. When managing, make a note of the **Nick name** and **Port number** that you provide.
2. If multiple JBoss EAP server instances are operating on a single host, and you want to BTM-enable all the instances, then you will have to manage each instance as a separate JBoss EAP server using the eG administrative interface. When doing so, make a note of the **Nick name** and **Port number** using which you managed each instance.
3. In the **<EG_AGENT_INSTALL_DIR>\lib\btm** directory, you will find the following files:

   - **eg_btm.jar**
   - **btmLogging.props**
   - **btmOther.props**
   - **exclude.props**

4. Next, create a new directory under the **<EG_AGENT_INSTALL_DIR>\lib\btm**. Take care to name this directory in the following format: <*Managed_Component_NickName*>_<*Managed_Component_Port*>. For instance, if you have managed the JBoss EAP server using the nick name *JBoss1* and the port number *9990*, the new directory under the **btm** directory should be named as *JBoss1_9990*.
5. If you have managed multiple JBoss EAP server instances running on a single host, then you will have to create multiple sub-directories under the **btm** directory- one each for every instance. Each of these sub-directories should be named after the **Nick name** and **Port number** using which the corresponding instance has been managed in eG.
6. Once the new directory is created, copy the following files from the **btm** directory to the new directory. If multiple directories have been created as described in step 5 above, then the following files should be copied to all directories:
   - **btmLogging.props**
   - **btmOther.props**
   - **exclude.props**
7. Next, edit the **btmOther.props** file. You will find the following lines in the file:

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Below property is BTM Server Socket Port, through which eG Agent Communicates
# Restart is required, if any changes in this property
# Default port is "13931"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
BTM_Port=13931
#
```

By default, the**BTMPort** parameter is set to 13931. If you want to enable eG Java BTM on a different port, then specify the same here. In this case, when configuring the **Java Business Transactions** test or the **Key Java Business Transactions** test for the JBoss EAP server, make sure you configure the **BTM** port parameter of the test with this port number.

Note:

When BTM-enabling multiple instances on the same server, make sure you configure a different **BTM Port** for each instance.

Also, by default, the **Designated_Agent** parameter will be empty; do not disturb this default setting. In this case therefore, the eG Java BTM will treat the host from which the very first 'measure request' comes in as the **Designated_Agent**.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Below property is used to specify IP address of eG Agent which collectes BTM Data.
# Default is None
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
Designated_Agent=
#
```

Note:
In case a specific **Designated_Agent** is not provided, and the eG Java BTM treats the host from which the very first 'measure request' comes in as the **Designated_Agent**, then if such a **Designated_Agent** is stopped or uninstalled for any reason, the eG Java BTM will wait for a maximum of 10 measure periods for that 'deemed' **Designated_Agent** to request for metrics. If no requests come in for 10 consecutive measure periods, then the eG Java BTM will begin responding to 'measure requests' coming in from any other eG agent.

8. Then, you need to configure the JBoss EAP server with the path to the **eg_btm.jar** and **.props** files. To achieve this, in case of a server operating in the standalone mode, you need to edit the standalone.bat file in the <EAP_HOME>\bin directory of the JBoss EAP server. In case of a server operating in the domain mode, edit the host.xml file in the <EAP_ HOME>\domain\configuration directory on the targeet server.

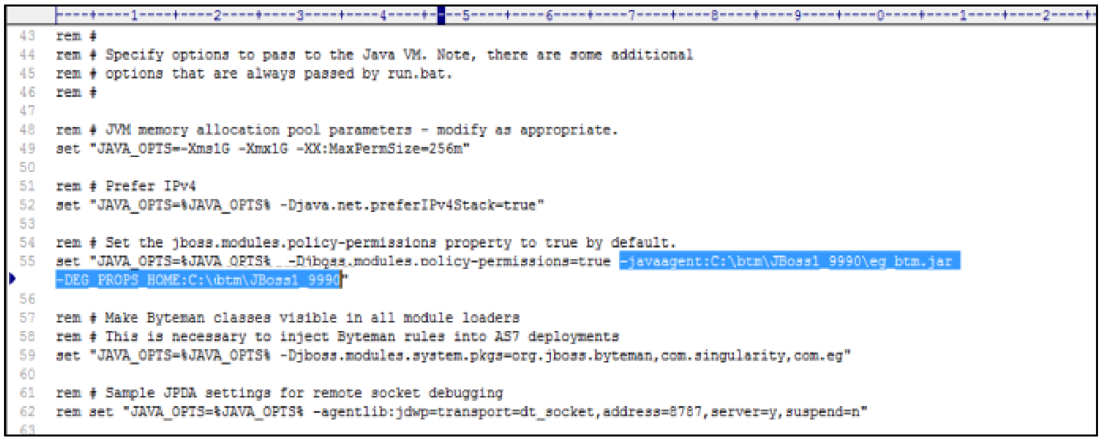
9. Then, in the file, enter the following lines, as depicted by Figure 2.21.

```
-javaagent:<EG_AGENT_INSTALL_DIR>\lib\btm\eg_btm.jar
-DEG_PROPS_HOME=<<PATH OF THE LOCAL FOLDER CONTAINING THE .PROPS FILES>>
```

For instance, if the .props files had been copied to the **<EG_ AGENT_ INSTALL_ DIR>\lib\btm\JBoss1_9990** directory, the above specification will be:

```
-javaagent:<EG_AGENT_INSTALL_DIR>\lib\btm\eg_btm.jar
```

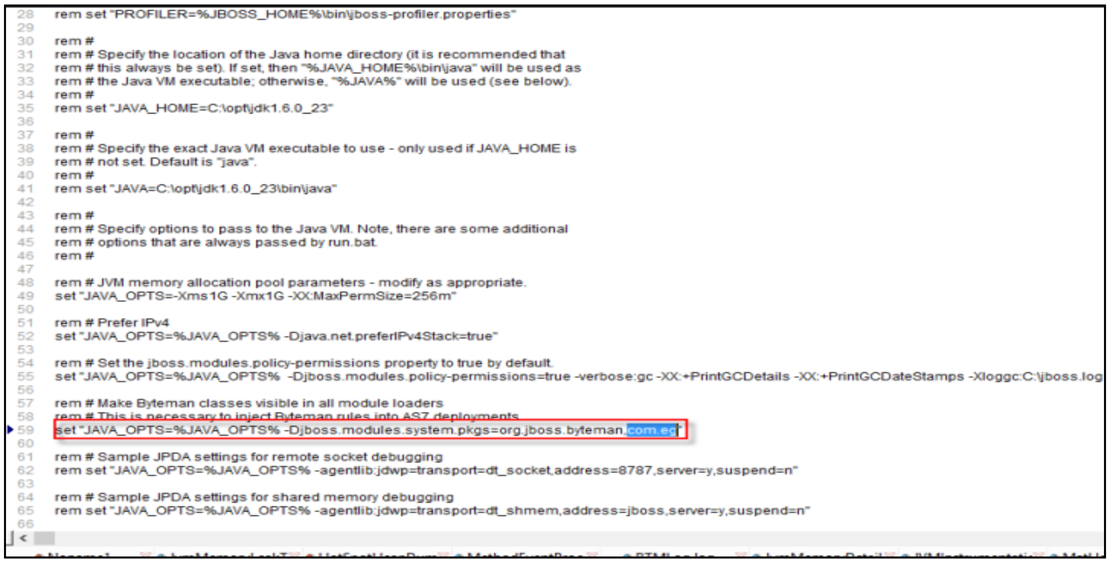
```
-DEG_PROPS_HOME=<EG_AGENT_INSTALL_DIR>\lib\btm\JBoss1_9990
```



```
rem #
rem # Specify options to pass to the Java VM. Note, there are some additional
rem # options that are always passed by run.bat.
rem #

rem # JVM memory allocation pool parameters - modify as appropriate.
set "JAVA_OPTS=-Xms1G -Xmx1G -XX:MaxPermSize=256m"

rem # Prefer IPv4
set "JAVA_OPTS=%JAVA_OPTS% -Djava.net.preferIPv4Stack=true"

rem # Set the jboss.modules.policy-permissions property to true by default.
set "JAVA_OPTS=%JAVA_OPTS% -Djboss.modules.policy-permissions=true -javaagent:C:\btm\JBoss1_9990\eg_btm.jar
-DEG_PROPS_HOME:C:\btm\JBoss1_9990"

rem # Make Byteman classes visible in all module loaders
rem # This is necessary to inject Byteman rules into AS7 deployments
set "JAVA_OPTS=%JAVA_OPTS% -Djboss.modules.system.pkgs=org.jboss.byteman,com.singularity,com.eg"

rem # Sample JPDA settings for remote socket debugging
rem set "JAVA_OPTS=%JAVA_OPTS% -agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n"
```

Figure 2.21: Editing the start-up script to BTM-enable a JBoss EAP server that is monitored in an agent-based manner

10. Next, if the JBoss server is running in domain mode, open the **domain.xml** file in <EAP_HOME>\domain\configuration directory. If the JBoss server is running in standalone mode, open the **standalone.conf** file in the **JBoss_Home\bin** directory.



```
rem set "PROFILER=%JBOSS_HOME%\bin\jboss-profiler.properties"

rem #
rem # Specify the location of the Java home directory (it is recommended that
rem # this always be set). If set, then "%JAVA_HOME%\bin\java" will be used as
rem # the Java VM executable; otherwise, "%JAVA%" will be used (see below).
rem #
rem set "JAVA_HOME=C:\opt\jdk1.6.0_23"

rem #
rem # Specify the exact Java VM executable to use - only used if JAVA_HOME is
rem # not set. Default is "java".
rem #
rem set "JAVA=C:\opt\jdk1.6.0_23\bin\java"

rem #
rem # Specify options to pass to the Java VM. Note, there are some additional
rem # options that are always passed by run.bat.
rem #

rem # JVM memory allocation pool parameters - modify as appropriate.
set "JAVA_OPTS=-Xms1G -Xmx1G -XX:MaxPermSize=256m"

rem # Prefer IPv4
set "JAVA_OPTS=%JAVA_OPTS% -Djava.net.preferIPv4Stack=true"

rem # Set the jboss.modules.policy-permissions property to true by default.
set "JAVA_OPTS=%JAVA_OPTS%  -Djboss.modules.policy-permissions=true -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:C:\jboss.log

rem # Make Byteman classes visible in all module loaders
rem # This is necessary to inject Byteman rules into AS7 deployments
set "JAVA_OPTS=%JAVA_OPTS% -Djboss.modules.system.pkgs=org.jboss.byteman,com.eg"

rem # Sample JPDA settings for remote socket debugging
rem set "JAVA_OPTS=%JAVA_OPTS% -agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n"

rem # Sample JPDA settings for shared memory debugging
rem set "JAVA_OPTS=%JAVA_OPTS% -agentlib:jdwp=transport=dt_shmem,address=jboss,server=y,suspend=n"
```

Figure 2.22: Editing the domain.xml file or standalone.conf file

11. Append **",com.eg"** to the following line in the file, as depicted by Figure 2.22:

```
set "JAVA_OPTS=%JAVA_OPTS% -Djboss.modules.system.pkgs="org.jboss.byteman"
```

12. Finally, save the file. and restart the JBoss EAP server.

If a JBoss EAP server is running on Unix, then follow the steps below to BTM-enable that server:

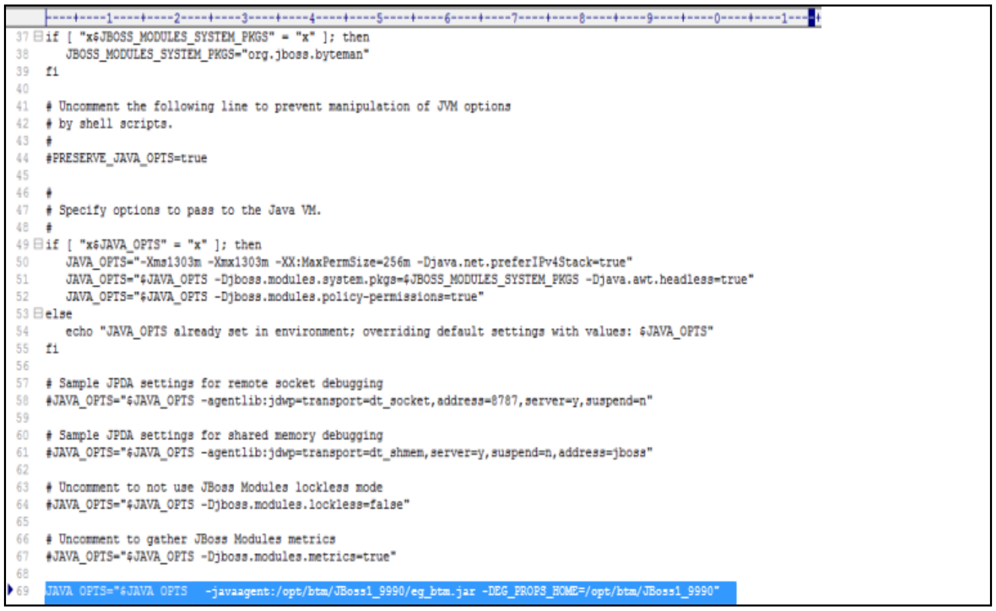
1. Follow steps 1 - 7 above. While doing so, note that the jar and .props files will be available in the **/opt/egurkha/lib/btm** directory on the eG agent host.
2. Then, you need to configure the JBoss EAP server with the path to the **eg_btm.jar** and **.props** files. To achieve this, in case of a standalone EAP server, you need to edit the **<EAP_ HOME>/bin /standalone.sh** file of the JBoss EAP server. In case of a server operating in the domain mode, edit the **host.xml** file in the **<EAP_HOME>/domain/configuration** directory.
3. Then, in the file, enter the following lines, as depicted by Figure 2.23.

```
JAVA_OPTS="$JAVA_OPTS -javaagent:<<PATH TO the eg_btm.jar>> -DEG_PROPS_HOME=<<PATH
TO LOCAL FOLDER CONTAINING THE
```

```
.PROPS FILES>>"
```

For instance, if the .props files had been copied to the **/opt/egurkha/lib/btm/JBoss1_9990** directory, the above specification will be:

```
JAVA_OPTS="$JAVA_OPTS -javaagent:/opt/egurkha/lib/btm/eg_btm.jar -DEG_PROPS_
HOME=/opt/egurkha/lib/btm/JBoss1_9990"
```



```
if [ "x$JBOSS_MODULES_SYSTEM_PKGS" = "x" ]; then
   JBOSS_MODULES_SYSTEM_PKGS="org.jboss.byteman"
fi

# Uncomment the following line to prevent manipulation of JVM options
# by shell scripts.
#
#PRESERVE_JAVA_OPTS=true

#
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms1303m -Xmx1303m -XX:MaxPermSize=256m -Djava.net.preferIPv4Stack=true"
   JAVA_OPTS="$JAVA_OPTS -Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -Djava.awt.headless=true"
   JAVA_OPTS="$JAVA_OPTS -Djboss.modules.policy-permissions=true"
else
   echo "JAVA_OPTS already set in environment; overriding default settings with values: $JAVA_OPTS"
fi

# Sample JPDA settings for remote socket debugging
#JAVA_OPTS="$JAVA_OPTS -agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n"

# Sample JPDA settings for shared memory debugging
#JAVA_OPTS="$JAVA_OPTS -agentlib:jdwp=transport=dt_shmem,server=y,suspend=n,address=jboss"

# Uncomment to not use JBoss Modules lockless mode
#JAVA_OPTS="$JAVA_OPTS -Djboss.modules.lockless=false"

# Uncomment to gather JBoss Modules metrics
#JAVA_OPTS="$JAVA_OPTS -Djboss.modules.metrics=true"

JAVA_OPTS="$JAVA_OPTS   -javaagent:/opt/btm/JBoss1_9990/eg_btm.jar -DEG_PROPS_HOME=/opt/btm/JBoss1_9990"
```

Figure 2.23: Editing the start-up script to BTM-enable a JBoss EAP server on Unix that is monitored in an agent-based manner

4. In Unix environments, if the eG agent is deployed on the same host as the JBoss EAP server, then both the agent and the server will be running using different user privileges. In this situation, by default, the eG Java BTM logs will not be created. In order to create the same, insert the following entry after the -DEG_PROPS_HOME specification, but before the closing quotes.

   ```
   -DEG_LOG_HOME=<<LogFile_Path>>
   ```

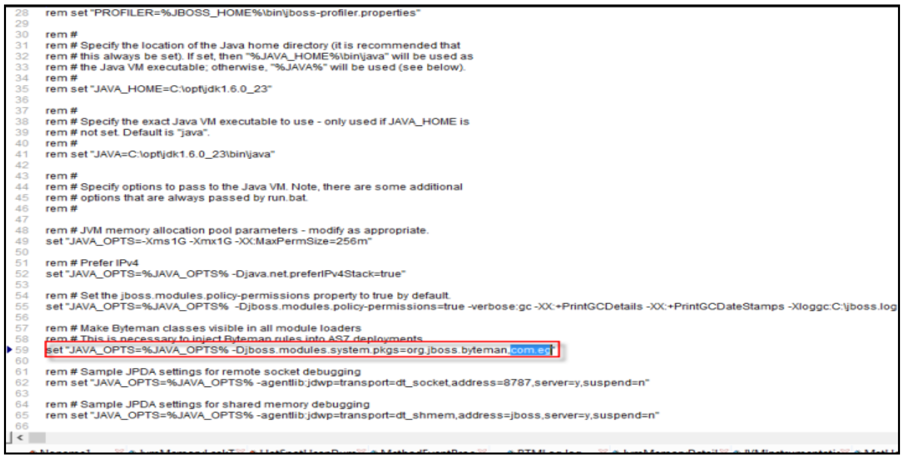
   Before providing this specification, make sure you create a folder for BTM logs - say, **eGBTMLogs** - in any directory to which the target application server has access. Then, against, -DEG_LOG_HOME, provide the full path to the **eGBTMLogs** directory. Where multiple instances on the same server are to be BTM-enabled, you can use the same directory for writing log files of all instances.

   For example, to create log files in the **/App001/eGBTMLogs** directory, the complete specification will be as follows:

   ```
   JAVA_OPTS="$JAVA_OPTS -javaagent:/opt/egurkha/lib/btm/eg_btm.jar -DEG_PROPS_
   HOME=/opt/egurkha/lib/btm/JBoss1_9990 -DEG_LOG_HOME=/App001/eGBTMLogs"
   ```

5. Then, save the file.

6. Next, if the JBoss server is running in domain mode, open the **domain.xml** file in **JBoss_ Home/domain/configuration** directory. If the JBoss server is running in standalone mode, open the **standalone.conf** file in the **JBoss_Home/bin** directory.



```
rem set "PROFILER=%JBOSS_HOME%\bin\jboss-profiler.properties"

rem #
rem # Specify the location of the Java home directory (it is recommended that
rem # this always be set). If set, then "%JAVA_HOME%\bin\java" will be used as
rem # the Java VM executable; otherwise, "%JAVA%" will be used (see below).
rem #
rem set "JAVA_HOME=C:\opt\jdk1.6.0_23"

rem #
rem # Specify the exact Java VM executable to use - only used if JAVA_HOME is
rem # not set. Default is "java".
rem #
rem set "JAVA=C:\opt\jdk1.6.0_23\bin\java"

rem #
rem # Specify options to pass to the Java VM. Note, there are some additional
rem # options that are always passed by run.bat.
rem #

rem # JVM memory allocation pool parameters - modify as appropriate.
set "JAVA_OPTS=-Xms1G -Xmx1G -XX:MaxPermSize=256m"

rem # Prefer IPv4
set "JAVA_OPTS=%JAVA_OPTS% -Djava.net.preferIPv4Stack=true"

rem # Set the jboss.modules.policy-permissions property to true by default.
set "JAVA_OPTS=%JAVA_OPTS%  -Djboss.modules.policy-permissions=true -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:C:\jboss.log

rem # Make Byteman classes visible in all module loaders
rem # This is necessary to inject Byteman rules into AS7 deployments
set "JAVA_OPTS=%JAVA_OPTS% -Djboss.modules.system.pkgs=org.jboss.byteman,com.eg"

rem # Sample JPDA settings for remote socket debugging
rem set "JAVA_OPTS=%JAVA_OPTS% -agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n"

rem # Sample JPDA settings for shared memory debugging
rem set "JAVA_OPTS=%JAVA_OPTS% -agentlib:jdwp=transport=dt_shmem,address=jboss,server=y,suspend=n"
```

Figure 2.24: Editing the domain.xml file or standalone.conf file

7. Append **",com.eg"** to the following line in the file, as depicted by Figure 2.22:

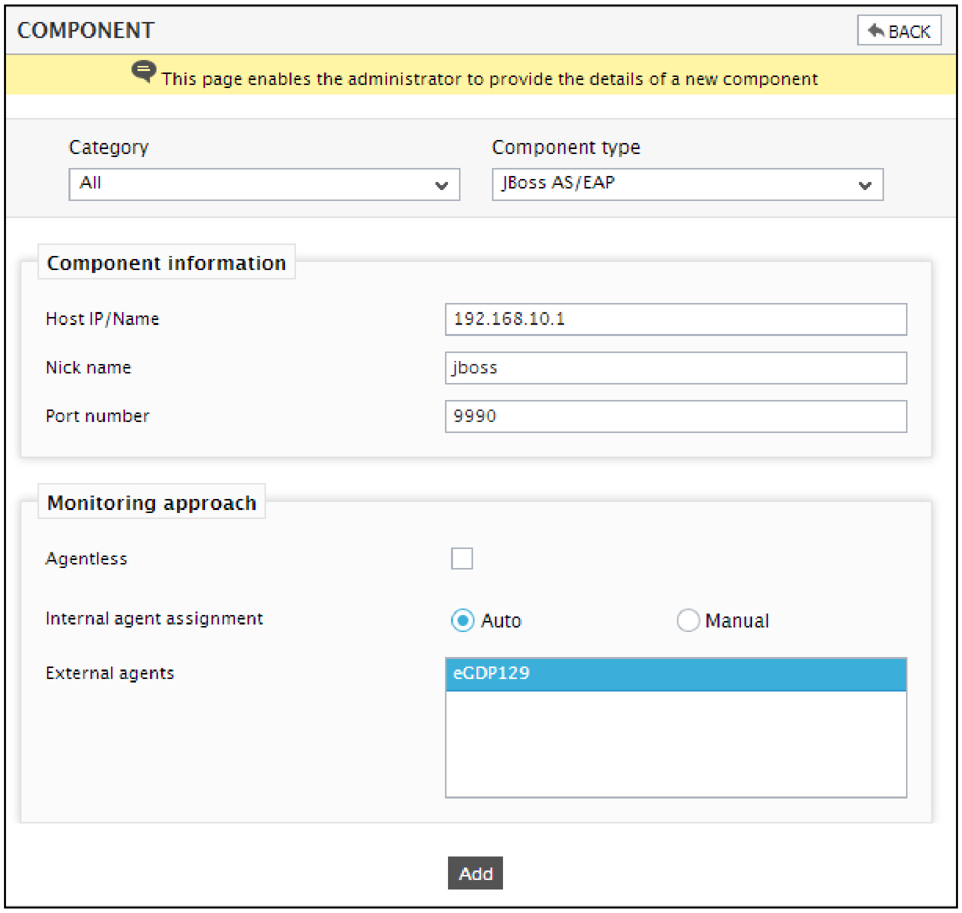
```
set "JAVA_OPTS=%JAVA_OPTS% -Djboss.modules.system.pkgs="org.jboss.byteman"
```

8. Finally, save the file and restart the JBoss EAP server instance.

2.2 Managing the JBoss AS/EAP Server

To manage a target JBoss AS/EAP Server using eG Enterprise, do the following:

1. Login to the eG administrative interface.
2. Invoke the Admin tile menu and select the Add/Modify option from the Components menu of the Infrastructure tile.
3. Then, select the *JBoss AS/EAP* as the **Component type** and click the **Add New Component** button. Figure 2.25 will then appear.



This page enables the administrator to provide the details of a new component

Category	All
Component type	JBoss AS/EAP

Component information

Host IP/Name	192.168.10.1
Nick name	jboss
Port number	9990

Monitoring approach

Agentless	☐
Internal agent assignment	[✓ Auto] Manual
External agents	eGDP129

Figure 2.25: Adding a JBoss AS/EAP for monitoring

4. In Figure 2.25, provide the **Host IP/Name** of the JBoss AS/EAP server to be monitored. Then, provide a **Nick name** for the server.

5. The **Port number** will be set as *9990* by default. Depending upon the operating mode of the target JBoss EAP server, you may have to override this default setting by specifying a different port number here. To determine the exact port number that you should specify, follow the procedure detailed in Section **2.1.1** topic.

6. Finally, click the **Add** button to add the server for monitoring.

Chapter 3: Monitoring JBoss AS/EAP servers

The eG agent periodically executes tests on the JBoss AS 7 or above using the JBoss AS/EAP server, collects the necessary statistics, and reports them to the eG manager. These tests are mapped to specific layers of the JBoss AS/EAP server's layer model (see Figure 3.1).

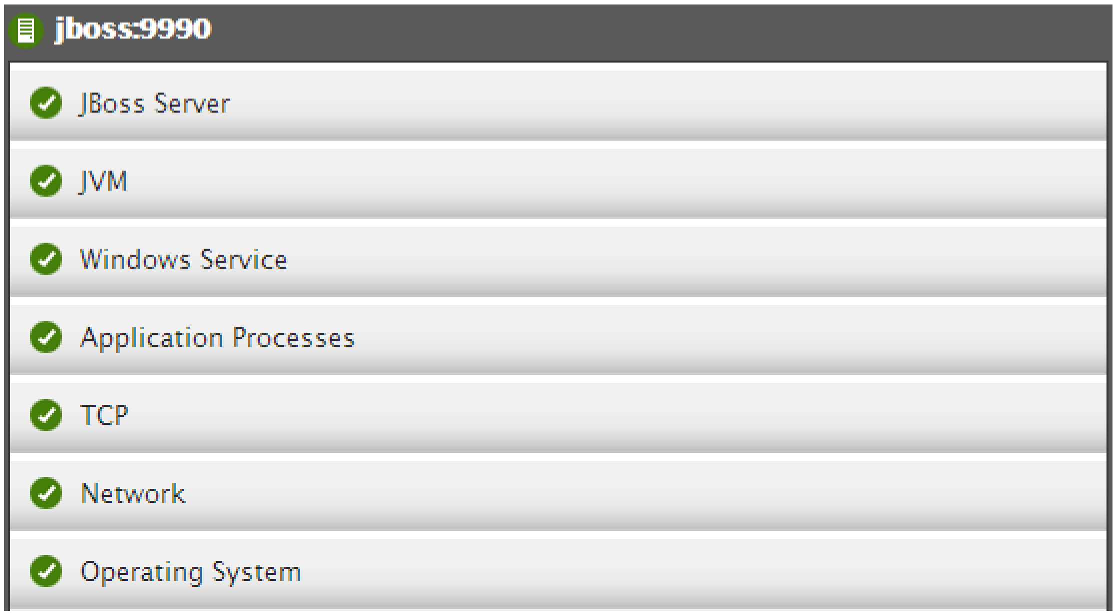


Figure 3.1: The layer model of the JBoss AS/EAP server

Using the metrics reported, administrators can find quick and accurate answers for the following performance questions:

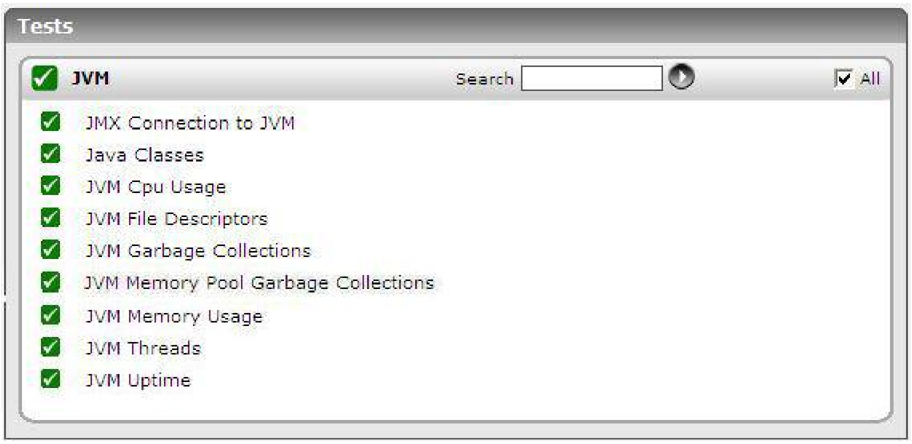
- How well each connector is processing the incoming requests?
- Are enough connections available in the connection pool of the datasources and XA-datasources, or do more connections have to be allotted to the pool?
- Are there adequate prepared statements available? How well the prepared statements are accessed from the prepared statement cache?
- How frequently was the servlet invoked?
- Does the servlet take too long to execute?
- How frequently are EJBs created/removed?
- Are there any EJBs that are ready to service clients?
- How many EJBs are currently in the EJB pool?

- Are too many messages have been enqueued?
- How quickly does the queue process messages within?
- Does the topic take too long to process messages?
- What type of messages are currently available in the topic - durable or non-durable?
- How many durable subscribers are there to a topic?
- How well each type of transaction is processed?

The **Operating System**, **Network**, **TCP**, **Application Processes** and **Windows Services** layers of the JBoss AS/EAP monitoring model is similar to that of a Windows Generic server model. Refer to the *Monitoring Unix and Windows Servers* document to know more about the tests pertaining to these layers. The JVM layer of this server is similar to that of the Java Application server monitoring model. Since the tests pertaining to these layers have been dealt with in the *Monitoring Java Application servers* document, Section **3.2** focuses on the **JBoss Server** layer.

3.1 The JVM Layer

This layer collectively reports the resource usage and overall health of the JBoss JVM.



- ✓ JMX Connection to JVM
- ✓ Java Classes
- ✓ JVM Cpu Usage
- ✓ JVM File Descriptors
- ✓ JVM Garbage Collections
- ✓ JVM Memory Pool Garbage Collections
- ✓ JVM Memory Usage
- ✓ JVM Threads
- ✓ JVM Uptime

Figure 3.2: The tests mapped to the JVM layer

All the tests displayed in Figure 3.2 have been dealt with in the *Monitoring Java Applications* document. For these tests to run and report metrics, make sure that the pre- requisites for JVM monitoring detailed in Section **2.1.3** are followed.

3.2 The JBoss Server Layer

This layer helps the administrators find quick and accurate performance measures for the following:

- The number of incoming requests processed by each connector;
- The number of connections available in the connection pool of the datasources and XA-datasources, the utilization of the connections by the datasource and the XA datatsource;
- The availability of the prepared statements; the number of prepared statements accessed from the prepared statement cache of the datasource and the XA datasource?
- The number of times the servlet was invoked;
- The time duration taken by the servlet for execution;
- The number of EJBs that are ready to service clients;
- The number of EJBs currently in the EJB pool;
- The number of messages that are enqueued in the queue;
- The time taken by the queue to process messages;
- The time taken by the topic to process the messages;
- The type of message are currently available in the topic;
- The number of durable subscribers to a topic;
- The number of transactions of various types that were processed; etc

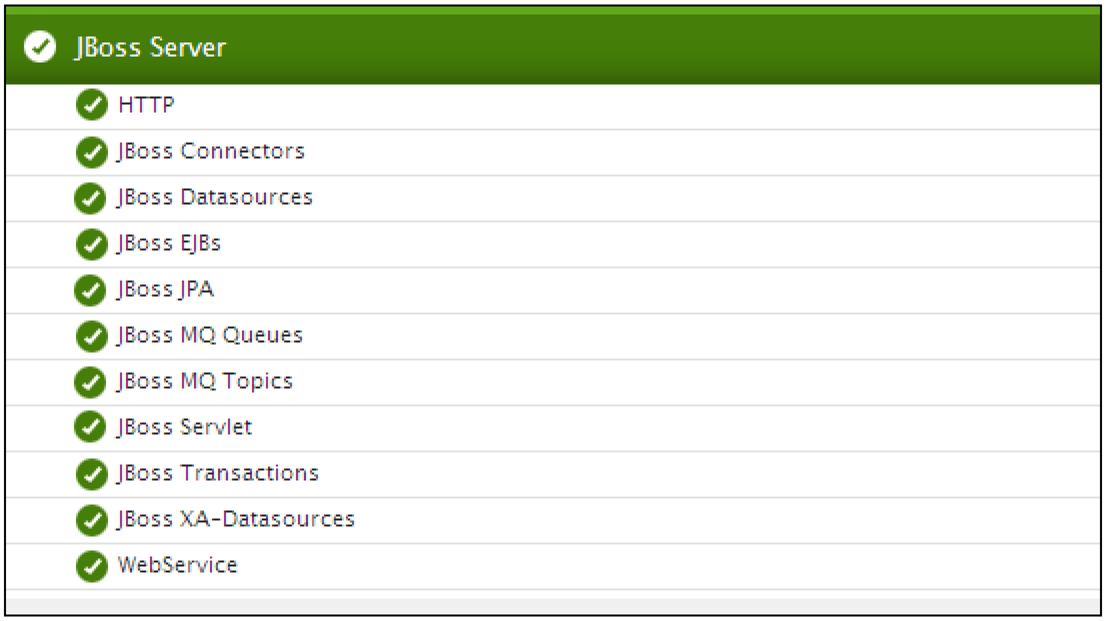


Figure 3.3: The tests mapped to the JBoss Server layer

3.2.1 JBoss Connectors Test

In a typical JBoss AS/EAP architecture, there are two main web connectors namely the Java I/O Connector and the AJP Connector. While the Java I/O connector uses the Java I/O to serve HTTP/HTTPS connections directly to the platform, the AJP Connector uses Apache's Portable Runtime (APR) native code library. Often, the connectors of the JBoss AS/EAP server receive a large number of requests for processing. In such cases, it becomes inevitable to identify how many requests are processed by the connector, how long it takes to process a request etc. The **JBoss Connectors** test exactly helps you figure out your concerns! This test not only monitors the number of incoming requests processed by each connector but also evaluates the time taken for processing the requests and the maximum time taken to process a single request. This test also sheds light on the amount of data used in processing the requests and errors encountered while processing the requests.

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each connector on the target JBoss AS/EAP that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance

Parameter	Description
	name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.
JBoss Server Instance Name	Specify the name of the server instance that is to be monitored. By default, *none* will be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test

Measurement	Description	Measurement Unit	Interpretation
Request count	Indicates the number of incoming requests processed by this connector.	Number	A high value is desired for this measure.
Processing time	Indicates the total time taken by this connector to process the incoming requests.	Mins	A low value is desired for this measure. The **maxThreads** parameter of the JBoss AS/EAP is set to 200 by default. This implies that the **maxThreads** parameter actually creates a thread pool behind the connector and processes the incoming requests. If no threads are available or the maximum limit for the number of threads is reached, processing of new requests is delayed which eventually leads to the piling up of requests. Also, a request may take too long to be processed if the request is malign or if adequate resources are not available for processing.
Max time	Indicates the maximum time taken by this	Mins	An abnormally high value clearly indicates a performance issue or lack

Measurement	Description	Measurement Unit	Interpretation
	connector to process an incoming request.		of threads that process the requests.
Bytes sent	Indicates the number of bytes sent by this connector for processing the requests.	Number	Comparing the value of these measures across the connectors will help you identify the connector that is using the maximum bytes for processing the requests.
Bytes received	Indicates the number of bytes received by this connector for processing the requests.	Number	
Error count	Indicates the number of errors encountered by this connector while processing the requests.	Number	Ideally, the value of this measure should be zero.

3.2.2 JBoss Datasources Test

A **Datasource** is a Java Naming and Directory Interface (JNDI) object used to obtain a connection from a connection pool to a database. When connecting to a data source, JBoss Enterprise Application Platform must allocate resources and de-allocate resources for every connection. This is quite expensive in terms of time and system resources. Connection pooling reduces the cost of data source connections by creating a number ("pool") of data source connections available to be shared by applications. Pooling data source connections is much more efficient than allocating and de-allocating resources on demand. Often, administrators may find it tedious to analyze how well connections from a connection pool have been utilized by the datasource and how efficiently the datasource has been channelizing the connections. This is where the **JBoss Datasource** Test helps! This test monitors the datasources and reports the following:

- The number of connections that are currently available and the number of active connections;
- The number of connections that were created and destroyed using this datasource;
- The time taken to create a connection on this datasource and the maximum time taken for creating a connection;
- How many connections are available in a connection pool and how well the connections of a connection pool are utilized?
- The prepared statement cache size and the number of prepared statements added

- How many times prepared statements are accessed and how well the prepared statement cache caters to the requests for accessing the prepared statements?

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each datasource of the target JBoss AS/EAP that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.
JBoss Server Instance Name	Specify the name of the server instance that is to be monitored. By default, *none* will be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test

Measurement	Description	Measurement Unit	Interpretation
Active count	Indicates the number of active connections through this datasource.	Number	A high value is desired for this measure.
Available count	Indicates the number of connections that are currently available for use in this datasource.	Number	A low value for this measure indicates that adequate connections are not available in the datasource.
Average blocking time	Indicates the average time spent blocking on obtaining an exclusive lock on the connection pool during the last measurement period.	Secs	
Average creation time	Indicates the average time spent for creating a physical connection during the last measurement period.	Secs	Comparing the value of this measure across the datasources will reveal the datasource that is the slowest when creating a physical connection.
Created count	Indicates the number of connections that were created using this datasource.	Number	
Destroyed count	Indicates the number of connections that were destroyed in this datasource since the start of the server.	Number	
Max creation time	Indicates the maximum time taken to create a connection in this datasource.	Secs	Compare the value across the datasources to identify the datasource that is taking too long to create a connection i.e., you can identify the datasource that is the slowest.
Max used count	Indicates the maximum	Number	

Measurement	Description	Measurement Unit	Interpretation
	number of connections that were used in this datasource.		
Max wait time	Indicates the maximum time a user had to wait for a connection through this datasource.	Secs	A low value is desired for this measure. An abnormally high value or a sudden increase in the value of this measure is a cause of concern which requires further investigation.
Timed out Connections	Indicates the number of timed out connections.	Number	Normally, a connection would time out if it has been waiting in the queue for a longer time.
Total blocking time	Indicates the total time taken for a connection to fail or time out.	Secs	
Total creation time	Indicates the time taken to create a connection through this datasource.	Secs	A low value is desired for this measure.
Maximum pool size	Indicates the maximum number of connections that can be handled by the connection pool of this datasource.	Number	The *max-pool-size* parameter defines the maximum size of the connection pool. This parameter is more important because it limits the number of active connections to the datasource and thus the concurrent activity on the data source. If this value is set too low it's likely that the platform's hardware resources will be underutilized.
Minimum pool size	Indicates the minimum number of connections that the connection pool of this datasource can contain.	Number	The `min-pool-size` data source parameter defines the minimum size of the connection pool. The default minimum is zero connections, so if a minimum pool size is not specified, no connections will be created in the connection pool when the platform starts. As data source transactions occur, connections will be requested but because the pool defaults to zero on start up, none will be available. The

Measurement	Description	Measurement Unit	Interpretation
			connection pool examines the minimum and maximum parameters, creates connections and places them in the pool. Users of any affected application will experience a delay while this occurs. During periods of inactivity the connection pool will shrink, possibly to the minimum value, and later when transactions occur application performance will again suffer.
PreparedStatement cache size	Indicates the number of prepared statements per connection to be kept open and reused in subsequent requests.	Number	When using the JPA annotations for queries, the result is prepared statements that will be executed against the database. Prepared statements have two phases for execution: preparation of the statement, and execution of the statement. Statement preparation involves significant CPU load so to improve throughput prepared statements can be cached. Statements can be cached either via the JDBC driver or configuration of the data source. To enable caching of prepared statement, add the following two lines to the data source configuration file, a file with the pattern *-ds.xml (where the * is usually your database, such as oracle, mysql, db2, etc.) in the directory: `JBOSS_EAP_DIST/jboss-as/server/PROFILE/deploy`. Note that the minimal configuration does not support data sources. <prepared-statement-cache-size>100</prepared-statement-cache-size> <shared-prepared-statements>true</shared-prepared-statements>

Measurement	Description	Measurement Unit	Interpretation
			The first line enables the prepared statement cache and sets the size of the cache. This should be large enough to hold all the prepared statements across any and all deployed applications using this particular data source (multiple data sources can be configured). The second line states that if two requests are executed in the same transaction the prepared statement cache should return the same statement. This will avoid the use of CPU cycles in preparing the same statements over and over again, increasing throughput and decreasing response times. The actual improvement will vary according to specific circumstances but is well worth the effort.
PreparedStatement cache access count	Indicates the number of times the prepared statement cache was accessed.	Number	
PreparedStatement cache add count	Indicates the number of new prepared statements added i.e., cached in the prepared statement cache.	Number	
PreparedStatement cache current size	Indicates the number of prepared statements that are currently available in the cache.	Number	If the value of this measure is close to the *PreparedStatement cache* size measure, then it indicates that the prepared statement cache is almost running out of space.
PreparedStatement cache delete count	Indicates the number of prepared statements that are deleted from the prepared statement cache.	Number	

Measurement	Description	Measurement Unit	Interpretation
PreparedStatement cache hit count:	Indicates the number of times the statement was accessed from the prepared statement cache.	Number	A high value is desired for this measure.
PreparedStatement cache miss count	Indicates the number of times a request for a statement was not serviced from the prepared statement cache.	Number	A high value for this measure is a cause of concern.

3.2.3 JBoss EJBs Test

A bean is usually an application class that contains business logic. It may be called directly from Java code, or it may be invoked via the Unified EL i.e., Unified Expression language. A bean may access transactional resources. Dependencies between beans are managed automatically by the container. The lifecycle of a bean is always managed by the container. The beans are classified as follows:

- stateless session beans
- stateful session beans
- message driven beans, and
- entity beans

This test auto-discovers the EJBs deployed on the JBoss server and reports critical measures relating to each of the EJB in detail. Using this test, you can figure out the number of times each bean has been invoked and the instances of each bean that can be accommodated in the pool. This way, you could figure out the effective utilization of the pool and the bean in particular.

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each EJB of the target JBoss AS/EAP that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.
JBoss Server Instance Name	Specify the name of the server instance that is to be monitored. By default, *none* will be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of invocations	Indicates the number of times this bean has been invoked since the start of the server.	Number	

Measurement	Description	Measurement Unit	Interpretation
Average execution time	Indicates the average time taken to execute the method of this bean during the last measurement period.	Secs	A higher value for this measure is a cause for concern.
Peak concurrent invocations	Indicates the maximum number of times the method of this bean has been invoked.	Number	
Pool available count	Indicates the number of instances of this bean that is currently available for use in the pool.	Number	A high value is desired for this measure.
Pool creation count	Indicates the number of bean instances that were created in the pool.	Number	
Pool remove count	Indicates the number of bean instances that were removed from the pool.	Number	
Pool maximum size	Indicates the maximum number of instances of this bean that can be accommodated in the pool.	Number	
Current pool size	Indicates the number of instances of this bean that are currently utilized in the pool.	Number	If the value of this measure is equal to the *Pool maximum size* measure, then you can increase the *Pool maximum size* measure to a more desirable value so that the bean instances are readily available for use.
Pool utilization	Indicates the percent of beans that have been utilized from the pool.	Number	A high value is indicative of maximum utilization of the bean instances from the pool.

3.2.4 JBoss MQ Queues Test

Clients that are in the point-to-point paradigm typically use queues. They expect that message sent to a queue will be received by only one other client once and only once. If multiple clients are receiving messages from a single queue, the messages will be load balanced across the receivers. Queue objects, by default, will be stored under the JNDI queue/ sub context.

This test auto-discovers the queues on the JBoss AS/EAPserver, and monitors each queue for the size, number, and type of messages it holds, so that impending overloads and probable delivery bottlenecks can be proactively isolated and corrected.

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each queue of the target JBoss AS/EAP server that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.
JBoss Server	Specify the name of the server instance that is to be monitored. By default, *none* will

Parameter	Description
Instance Name	be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test

Measurement	Description	Measurement Unit	Interpretation
Max queue depth	Indicates the max value of the number of messages that were in this queue since the start of the queue.	Number	
Current queue depth	Indicates the number of messages that are currently in this queue.	Number	A high value is indicative of server workload, or a delivery bottleneck.
Queue occupied	Indicates the percentage of queue length that is occupied by messages.	Percent	A high value is a cause for concern as it could indicate a bottleneck in message delivery, which may be heavily populating the queue with messages.
Messages delivered	Indicates the number of messages in this queue that have been delivered.	Number	A high value is desired for this measure.
Messages scheduled	Indicates the number of messages in this queue that are currently scheduled to be delivered.	Number	
Messages added	Indicates the number of messages that were added to this queue during the last measurement period.	Number	

Measurement	Description	Measurement Unit	Interpretation
Last message sent time	Indicates the time that elapsed since the last message was sent from this queue.	Secs	Ideally, the value of this measure should be low. A high value is indicative of a delay in message delivery or a message processing bottleneck.
Subscriber count	Indicates the number of subscribers registered with this queue.	Number	
Receivers count:	Indicates the number of clients who are currently configured as receivers of messages from this queue.	Number	
Message processing rate	Indicates the rate at which messages are being processed by this queue.	Msgs/sec	

3.2.5 JBoss MQ Topics Test

Topics are used in the publish-subscribe paradigm. When a client publishes a message to a topic, he/she expects that a copy of the message will be delivered to each client that has subscribed to the topic. However, if the client is not up, running and receiving messages from the topics, it will miss messages published to the topic. To get around this problem of missing messages, clients can start a durable subscription. This is like having a VCR record a show you cannot watch at its scheduled time so that you can see what you missed when you turn your TV back on. Similarly, messages meant for a durable subscriber are stored in the persistent cache even when the subscriber is inactive. These messages are delivered to durable subscribers when they connect to the server. Using the **JBoss MQ Topics** test, continuous monitoring of the topics is possible. This test auto discovers the topics on the JBoss AS/EAP server and reports the number of durable messages and the durable subscribers of the topic. This test also helps the administrators to keep track on the messages added into each topic and the messages delivered from each topic along with the processing rate of the messages from the topic. This way, administrators can figure out the topic that is currently subscribed by most of the subscribers!

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each topic of the JBoss AS/EAP server that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.
JBoss Server Instance Name	Specify the name of the server instance that is to be monitored. By default, *none* will be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test

Measurement	Description	Measurement Unit	Interpretation
Durable messages count	Indicates the number of durable messages	Number	

Measurement	Description	Measurement Unit	Interpretation
	currently in the topic.		
Durable subject count	Indicates the number of durable subscribers to the topic.	Number	Messages meant for a durable subscriber are stored in the persistent cache even when the subscriber is inactive. These messages are delivered to durable subscribers when they connect to the server.
Max topic depth	Indicates the maximum value of the number of messages in the topic since the start of the queue.	Number	
Message processing rate	Indicates the rate at which messages were being processed by the topic in the last measurement period.	Msgs/Sec	
Non-durable messages count	Indicates the number of non-durable messages currently in the topic.	Number	If the total number of durable subscribers is high, then we can expect the total durable messages to be stored on the server also to be relatively on the higher side.
Non-durable subject count	Indicates the number of subscribers to the topic who are currently non-durable.	Number	
Total subject count	Indicates the total number of subscribers to a topic.	Number	
Messages added count	Indicates the number of messages that were added to this topic during the last measurement period.	Number	
Message delivered count	Indicates the number of messages in this topic that have been delivered.	Number	A high value is desired for this measure.

3.2.6 JBoss Servlet Test

This test monitors the servlets deployed on the JBoss AS/EAP server and reports the following:

- The number of incoming requests to each servlet;
- The time taken to load each servlet;
- The time taken by each servlet to to process the requests and the rate at which the requests are processed

This way, the administrators can identify the servlet that is handling the maximum number of requests.

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each servlet of the JBoss AS/EAP server that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.

Parameter	Description
JBoss Server Instance Name	Specify the name of the server instance that is to be monitored. By default, *none* will be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test:

Measurement	Description	Measurement Unit	Interpretation
Request count	Indicates the number of incoming requests that were handled by this servlet.	Number	This value is a good indicator on the load of the servlet.
Load time	Indicates the total time taken to load this servlet.	Secs	A low value is desired for this measure.
Processing time	Indicates the time taken to process the incoming requests to this servlet during the last measurement period.	Secs	If the value taken to process the requests is higher, then it may be due to a request taking too long to be processed, a connection issue where the servlet failed to load, etc.
Request processing rate	Indicates the rate at which the requests were processed by this servlet during the last measurement period.	Number/Sec	A high value is desired for this measure.

3.2.7 JBoss Transactions Test

A transaction is a unit of work containing one or more operations involving one or more shared resources having ACID properties. ACID is an acronym for atomicity, consistency, isolation and durability, the four important properties of transactions. These terms are defined below in detail:

- **Atomicity**: A transaction must be atomic. This means that either all the work done in the transaction must be performed, or none of it must be performed. Doing part of a transaction is not allowed.
- **Consistency**: When a transaction is completed, the system must be in a stable and consistent condition.
- **Isolation**: Different transactions must be isolated from each other. This means that the partial work done in one transaction is not visible to other transactions until the transaction is committed, and that each process in a multi-user system can be programmed as if it was the only process accessing the system.
- **Durability**: The changes made during a transaction are made persistent when it is committed. When a transaction is committed, its changes will not be lost, even if the server crashes afterwards.

Transactions are performed using the Java transaction API. The Java Transaction API consists of three elements: a high-level application transaction demarcation interface, a high-level transaction manager interface intended for application server, and a standard Java mapping of the X/Open XA protocol intended for transactional resource manager. All of the JTA classes and interfaces occur within the **javax.transaction** package, and the corresponding JBossJTA implementations within the **com.arjuna.ats.jta** package. The **UserTransaction** interface provides applications with the ability to control transaction boundaries. It has methods for beginning, committing, and rolling back top-level transactions. The **TransactionManager** interface allows the application server to control transaction boundaries on behalf of the application being managed.

The Transaction Manager maintains the transaction context association with threads as part of its internal data structure. A thread's transaction context is either null or it refers to a specific global transaction. Multiple threads may be associated with the same global transaction.

Each transaction context is encapsulated by a Transaction object, which can be used to perform operations which are specific to the target transaction, regardless of the calling thread's transaction context. Sometimes, there may be too many transactions being performed on the target application server and administrators may lose count on the number of transactions of each type. This is exactly where the **JBoss Transactions** test helps!

This test measures the transaction activity performed by the JBoss AS/EAP server. Using this test, administrators can easily track the numerical statistics of each transaction type and identify which type of transaction is executed too often on the target JBoss AS/EAP server.

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the target JBoss AS/EAP server that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.
JBoss Server Instance Name	Specify the name of the server instance that is to be monitored. By default, *none* will be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test:

Measurement	Description	Measurement Unit	Interpretation
Number of	Indicates the number of	Number	This measure is a good indicator of the

Measurement	Description	Measurement Unit	Interpretation
transactions	transactions on this server.		load on the server.
Aborted transactions	Indicates the total number of transactions that were aborted i.e., interrupted in this server during the last measurement period.	Number	
Application transactions rolled back	Indicates the number of transactions that were rolled back due to application errors during the last measurement period.	Number	
Committed transactions	Indicates the number of transactions that were committed to be processed by the Transaction manager during the last measurement period.	Number	If the number of transactions that are being committed is very high, it signifies load on the server. It might be caused when some locked transactions are released suddenly.
Heuristics transactions	Indicates the number of transactions that were terminated with heuristic outcome during the last measurement period.	Number	A heuristic completion (or heuristic decision) occurs when a resource makes a unilateral decision during the completion stage of a distributed transaction to commit or rollback updates. This can leave distributed data in an indeterminate state. Network failures or resource timeouts are possible causes for heuristic completion. In the event of an heuristic completion, one of the following heuristic outcome exceptions may be thrown: **HeuristicRollback**—one resource participating in a transaction decided to autonomously rollback its work, even though it agreed to prepare itself and

Measurement	Description	Measurement Unit	Interpretation
			wait for a commit decision. If the Transaction Manager decided to commit the transaction, the resource's heuristic rollback decision was incorrect, and might lead to an inconsistent outcome since other branches of the transaction were committed. **HeuristicCommit**—one resource participating in a transaction decided to autonomously commit its work, even though it agreed to prepare itself and wait for a commit decision. If the Transaction Manager decided to rollback the transaction, the resource's heuristic commit decision was incorrect, and might lead to an inconsistent outcome since other branches of the transaction were rolled back. **HeuristicMixed**—the Transaction Manager is aware that a transaction resulted in a mixed outcome, where some participating resources committed and some rolled back. The underlying cause was most likely heuristic rollback or heuristic commit decisions made by one or more of the participating resources. **HeuristicHazard**—the Transaction Manager is aware that a transaction might have resulted in a mixed outcome, where some participating resources committed and some rolled back. But system or resource failures make it impossible to know for sure whether a Heuristic Mixed outcome definitely occurred. The underlying

Measurement	Description	Measurement Unit	Interpretation
			cause was most likely heuristic rollback or heuristic commit decisions made by one or more of the participating resources.
Inflight transactions	Indicates the number of transactions that have begun but are yet to be terminated during the last measurement period.	Number	A significantly high value may denote a load on the server. This may indicate that specific transactions are taking too long to process requests.
Nested transactions	Indicates the number of nested i.e., sub transactions that were created during the last measurement period.	Number	
Resource transactions rolled back	Indicates the number of transactions that rolled back due to resource failure during the last measurement period.	Number	A high value indicates a problem with the application or with some other dependent servers (e.g. Database).
Timed-out transactions	Indicates the number of transactions that rolled back due to timeout during the last measurement period.	Number	A steady increase in the value of this measure could be due to the problem with the application or with some other dependent server like the database.

3.2.8 JBoss XA-Datasources Test

An XA Datasource is needed to execute a distributed transaction. Generally, a distributed transaction spans 2 or more different datasources. An XA datasource is used instead of the datasource if the target Jboss application:

- Uses the Java Transaction API (JTA)
- Includes multiple database updates within a single transaction

- Accesses multiple resources, such as a database and the Java Messaging Service (JMS), during a transaction
- Uses the same connection pool on multiple servers

Though the XA datasource can use all transaction features of the application server and need fewer physical connections, the XA datasource involves performance overhead along with the risk of a higher deadlock. There may also be a necessity of more physical connections when an emulated XA datasource is used in the target environment. When an XA datasource is used in the target environment, administrators would be required to carefully monitor the XA datasource as the slightest variation in the connections or a higher waiting time for a connection may cause a severe performance overhead on the XA datasource. To overcome such abnormalities, administrators may use the **JBoss XA-Datasources** test. This test helps the administrators figure out the following:

- The number of connections that are currently available and the number of active connections;
- The number of connections that were created and destroyed using this datasource;
- The time taken to create a connection on this datasource and the maximum time taken for creating a connection;
- How many connections are available in a connection pool and how well the connections of a connection pool are utilized?
- The prepared statement cache size and the number of prepared statements added
- How many times prepared statements are accessed and how well the prepared statement cache caters to the requests for accessing the prepared statements?

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each XA datasource of the target JBoss AS/EAP that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.

Parameter	Description
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.
JBoss Server Instance Name	Specify the name of the server instance that is to be monitored. By default, *none* will be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test:

Measurement	Description	Measurement Unit	Interpretation
Active count	Indicates the number of active connections in this XA datasource.	Number	A high value is desired for this measure.
Available count	Indicates the number of connections that are currently available for use in this XA datasource.	Number	A low value for this measure indicates that adequate connections are not available in the datasource.
Average blocking time	Indicates the average time spent blocking on	Secs	

Measurement	Description	Measurement Unit	Interpretation
	obtaining an exclusive lock on the connection pool during the last measurement period.		
Average creation time	Indicates the average time spent for creating a physical connection during the last measurement period.	Secs	Comparing the value of this measure across the XA datasources will reveal the XA datasource that is the slowest when creating a physical connection.
Created count	Indicates the number of connections that were created in this XA datasource.	Number	
Destroyed count	Indicates the number of connections that were destroyed in this XA datasource since the start of the server.	Number	
Max creation time	Indicates the maximum time taken to create a connection in this XA datasource.	Secs	Compare the value across the XA datasources to identify the XA datasource that is taking too long to create a connection i.e., you can identify the XA datasource that is the slowest.
Max used count	Indicates the maximum number of connections that were used in this XA datasource.	Number	
Max wait time	Indicates the maximum time a user has to wait for a connection in this XA datasource.	Secs	A low value is desired for this measure. An abnormally high value or a sudden increase in the value of this measure is a cause of concern which requires further investigation.
Timed out Connections	Indicates the number of timed out connections.	Number	Normally, a connection would time out if it has been waiting in the queue for a longer time.

Measurement	Description	Measurement Unit	Interpretation
Total blocking time	Indicates the total time taken for a connection to fail or time out.	Secs	
Total creation time	Indicates the time taken to create a connection in this XA datasource.	Secs	A low value is desired for this measure.
Maximum pool size	Indicates the maximum number of connections allowed in the connection pool of this XA datasource.	Number	The *max-pool-size* parameter defines the maximum size of the connection pool. This parameter is more important because it limits the number of active connections to the datasource and thus the concurrent activity on the data source. If this value is set too low it's likely that the platform's hardware resources will be underutilized.
Minimum pool size	Indicates the minimum number of connections that the connection pool of this XA datasource can contain.	Number	The *min-pool-size* data source parameter defines the minimum size of the connection pool. The default minimum is zero connections, so if a minimum pool size is not specified, no connections will be created in the connection pool when the platform starts. As data source transactions occur, connections will be requested but because the pool defaults to zero on start up, none will be available. The connection pool examines the minimum and maximum parameters, creates connections and places them in the pool. Users of any affected application will experience a delay while this occurs. During periods of inactivity the connection pool will shrink, possibly to the minimum value, and later when transactions occur application performance will again suffer.
PreparedStatement cache size	Indicates the number of prepared statements per	Number	When using the JPA annotations for

Measurement	Description	Measurement Unit	Interpretation

	connection to be kept open and reused in subsequent requests.		queries, the result is prepared statements that will be executed against the database. Prepared statements have two phases for execution: preparation of the statement, and execution of the statement. Statement preparation involves significant CPU load so to improve throughput prepared statements can be cached. Statements can be cached either via the JDBC driver or configuration of the data source.
			
			To enable caching of prepared statement, add the following two lines to the data source configuration file, a file with the pattern *-ds.xml (where the * is usually your database, such as oracle, mysql, db2, etc.) in the directory: *JBOSS_EAP_DIST/jboss-as/server/PROFILE/deploy*. Note that the minimal configuration does not support data sources.
			
			`<prepared-statement-cache-size>100</prepared-statement-cache-size>`
			
			`<shared-prepared-statements>true</shared-prepared-statements>`
			
			The first line enables the prepared statement cache and sets the size of the cache. This should be large enough to hold all the prepared statements across any and all deployed applications using this particular data source (multiple data sources can be configured). The second line states that if two requests are executed in the same transaction the prepared

Measurement	Description	Measurement Unit	Interpretation
			statement cache should return the same statement. This will avoid the use of CPU cycles in preparing the same statements over and over again, increasing throughput and decreasing response times. The actual improvement will vary according to specific circumstances but is well worth the effort.
PreparedStatement cache access count	Indicates the number of times the prepared statement cache was accessed.	Number	
PreparedStatement cache add count	Indicates the number of new prepared statements that were added i.e., cached in the prepared statement cache.	Number	
PreparedStatement cache current size	Indicates the number of prepared statements that are currently available in the cache.	Number	If the value of this measure is close to the *PreparedStatement cache size* measure, then it indicates that the prepared statement cache is almost running out of space.
vPreparedStatement cache delete count	Indicates the number of prepared statements that are deleted from the prepared statement cache.	Number	
PreparedStatement cache hit count	Indicates the number of times the statement was accessed from the prepared statement cache.	Number	A high value is desired for this measure.
PreparedStatement cache miss count	Indicates the number of times a request for a statement was not	Number	A high value for this measure is a cause of concern.

Measurement	Description	Measurement Unit	Interpretation
	serviced from the prepared statement cache.		

3.2.9 JBoss JPA Test

The Java Persistence API (JPA) is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational database. JPA itself is just a specification, not a product; it cannot perform persistence or anything else by itself. JPA is just a set of interfaces, and requires an implementation. JPA also requires a database to persist to. The JPA allows POJO (Plain Old Java Objects) to be easily persisted without requiring the classes to implement any interfaces or methods. JPA allows an object's object-relational mappings to be defined through standard annotations or XML defining how the Java class maps to a relational database table. JPA also defines a runtime EntityManager API for processing queries and transaction on the objects against the database. JPA defines an object-level query language, JPQL, to allow querying of the objects from the database.

An entity is a lightweight persistence domain object. Typically, an entity represents a table in a relational database, and each entity instance corresponds to a row in that table. The primary programming artifact of an entity is the entity class, although entities can use helper classes.

The persistent state of an entity is represented through either persistent fields or persistent properties. These fields or properties use object/relational mapping annotations to map the entities and entity relationships to the relational data in the underlying data store.

The JPA persistence context contains the entities managed by the persistence provider. The persistence context acts like a first level (transactional) cache for interacting with the datasource. Loaded entities are placed into the persistence context before being returned to the application. Entities changes are also placed into the persistence context (to be saved in the database when the transaction commits).

Collection-valued persistent fields and properties must use the supported Java collection interfaces regardless of whether the entity uses persistent fields or properties. The following collection interfaces may be used:

- java.util.Collection
- java.util.Set

- java.util.List
- java.util.Map

If the entity class uses persistent fields, the type in the preceding method signatures must be one of these collection types. Generic variants of these collection types may also be used. While an administrator is monitoring an application in real-time, he/she may not be aware of a sudden slowdown/malfunctioning of the application due to technical glitches at the backend. This may be due to a sudden change in the entities and collections of the Java Persistence API which when left unnoticed will render the application inaccessible to users. To avoid such a situation, you can use the **JBoss JPA** test. For each entity of an application that is to be monitored, this test reports the number of times each entity was loaded, inserted, fetched, updated, etc. In due course, this test also reports the number of times the collection of each entity was loaded, inserted, deleted, fetched,etc. This way administrators can proactively be alerted to technical failures of the applications and rectify the same before end users start complaining!

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *Application:entity* of the target JBoss AS/EAP that is to be monitored.

Configurable parameters for the test

Parameter	Description
Test period	How often should the test be executed
Host	The host for which the test is to be configured.
Port	The port at which the specified host listens. By default, this is *9990*.
SSL	If the JBoss AS/EAP server being monitored is an SSL-enabled server, then set the SSL flag to **Yes**. If not, then set the sslflag to **No**.
Is JBoss Running in Domain Mode?	Specify whether the server to be monitored is currently running in **Domain Mode** or not. By default, this flag is set to **No** which implies that the server is currently running in **Standalone mode**. If you have started the target JBoss server using the default web profile configuration in domain mode i.e, if you have executed the **./domain.sh** command from the <JBoss_INSTALL_DIR>/bin directory, then specify **Yes** against this flag.
JBoss Host Name	Specify whether the target server to be monitored is a master or a slave in a JBoss

Parameter	Description
	cluster. By default, *none* will be specified here which implies that the target JBoss server is a standalone server. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to know how to identify whether the target server is a master or slave in your environment.
JBoss Server Instance Name	Specify the name of the server instance that is to be monitored. By default, *none* will be specified here. Refer to Identifying the host name and server instance name of the JBoss AS/EAP server running in Domain mode to identify the name of the server instance that is to be monitored.
Management User and Management Password	Specify the credentials of the user who is authorized to access the management consoleof the target JBoss server. To create a new user, refer to Section **2.1.2**.
Confirm Password	Confirm the Management Password by retyping it here.

Measures made by the test

Measurement	Description	Measurement Unit	Interpretation
Entity Load count:	Indicates the number of times this entity was loaded.	Number	Comparing the value of this measure across the entities will help you identify the entity that is the busiest - in terms of loading on the application.
Entity Insert count:	Indicates the number of times this entity was inserted.	Number	
Entity Fetch count:	Indicates the number of times this entity was fetched.	Number	
Entity Update count:	Indicates the number of times this entity was updated.	Number	
Entity Delete count:	Indicates the number of times this entity was deleted.	Number	
Entity Failure count:	Indicates the number of times this entity failed.	Number	A low value is desired for this measure. A sudden/gradual increase in the value of this measure is a cause of

Measurement	Description	Measurement Unit	Interpretation
			concern as this may lead to poor user experience.
Collection Load count:	Indicates the number of times the collection was loaded on this entity.	Number	
Collection Recreate count:	Indicates the number of times the collection was recreated for this entity.	Number	
Collection Fetch count:	Indicates the number of times the collection was fetched for this entity.	Number	
Collection Update count:	Indicates the number of times the collection was updated for this entity.	Number	
Collection Remove count:	Indicates the number of times the collection was removed from this entity.	Number	

3.2.10 Web Service Test

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. A complete web service is, therefore, any service that:

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

The basic web services platform is XML + HTTP. All the standard web services work using the following components:

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of the following:

- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.

The following are the major uses of the Web Services:

- **Reusable application-components**: Often applications need repeated access to application-components like currency conversion, weather reports, or even language translation. In such cases, the web services can be used to offer the application-components as services with ease.
- **Connect existing software**: Web services can help to solve the interoperability problem by giving different applications a way to link their data. With Web services you can exchange data between different applications and different platforms. Any application can have a Web Service component. Web Services can be created regardless of programming language.

In certain environments, administrators are required to keep an eye on the web services that offer repeated access to the application-components i.e., operations so that the work load on the users using those application components can be minimized. If for some reason the web service takes too long to respond or is unavailable to cater to the needs of the users, then the users will be deprived of access to the application- components involved in that particular web service. To avoid such inconvenience caused to the users, administrators are required to continuously monitor the web services. The **Web Service** test helps administrators to perform this task perfectly. By continuously monitoring each operation i.e., application component of a web service that is offered, using the SOAP commands, this test helps administrators identify the availability, response time and response code of the web service and quickly figure out discrepancies if any web service is deemed unavailable. This way, the web services can be kept available round the clock thus helping the users perform their tasks without any difficulty.

Target of the test : A JBoss AS/EAP server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each *WebService:Operation* i.e., application-component performed on the target server that is being monitored.

Configurable parameters for the test

1. **TEST PERIOD -** How often should the test be executed
2. **HOST -** The host for which the test is to be configured
3. **PORT -** The port number at which the specified **HOST** listens
4. **WSDL URL -** This test emulates a user accessing a specific web service(s) on the target server to determine the availability and responsiveness of the server. to enable this emulation, you need to configure the test with the url of the web service that it should access. specify this url against the **WSDL URL** parameter. if required, you can even configure multiple WSDL URLs - one each for every web service that the test should attempt to access. if each WSDL URL configured requires special permissions for logging in, then, you need to configure the test with separate credentials for logging into every WSDL URL. likewise, you need to provide instructions to the test on how to validate the content returned by every WSDL URL, and also set an encoding format for each wsdl url. to enable administrators to easily configure the above per WSDL URL, eg enterprise provides a special interface. to access this interface, click on the encircled '+' button alongside the url text box in the test configuration page. alternatively, you can even click on the encircled '+' button adjacent to the WSDL URL parameter in the test configuration page. to know how to use this special interface, refer to Section **3.2.10.1**.
5. **OPERATIONS** – Once the WSDL URL(s) are specified, the operations that are offered by the web services and those that are to be monitored have to be configured. To select the required operations for monitoring, eG Enterprise provides a special interface. TO access this interface, click on the encircled '+' button alongside the **OPERATIONS** text box in the test configuration page. Alternatively, you can even click on the encircled '+' button adjacent to the **OPERATIONS** parameter in the test configuration page. To know how to use this special interface, refer to Section **3.2.10.1**.
6. **TIMEOUT** – Specify the duration (in seconds) for which this test should wait for a response from the server. If there is no response from the server beyond the configured duration, the test will timeout. By default, this is set to 30 seconds.
7. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

- The eG manager license should allow the detailed diagnosis capability
- Both the normal and abnormalfrequencies configured for the detailed diagnosis measures should not be 0.

Measures made by the test:

Measurement	Description	Measurement Unit	Interpretation
WSDL url availability:	Indicates whether the web service was able to respond successfully to the query made by the test.	Percent	Availability failures could be caused by several factors such as the web service process (es) being down, the web service being misconfigured, a network failure, etc. Temporary unavailability may also occur if the web service is overloaded. Availability is determined based on the response code returned by the service. A response code between 200 to 300 indicates that the service is available.
WSDL response time:	Indicates the time taken by the eG agent to get the configured web service.	Secs	Response time being high denotes a problem. Poor response times may be due to the service being overloaded or misconfigured. If the URL accessed involves the generation of dynamic content by the service, backend problems (e.g., an overload at the application server or a database failure) can also result in an increase in response time.
Port status:	Indicates whether/not the port of the web		The values reported by this measure and the corresponding

Measurement	Description	Measurement Unit	Interpretation
	server is reachable.		numeric equivalents are listed in the table below: **Measure Values** / **Numeric Values** Yes / 1 No / 0 Note: By default, this measure reports the above-mentioned Measure Values to indicate whether the server has been rebooted or not. In the graph of this measure however, the Measure Values are represented using the numeric equivalents only.
TCP connection availability:	Indicates whether the test managed to establish a TCP connection to the server.	Percent	Failure to establish a TCP connection may imply that either the web server process is not up, or that the process is not operating correctly. In some cases of extreme overload, the failure to establish a TCP connection may be a transient condition. As the load subsides, the server may start functioning properly again.
TCP connect time:	This measure quantifies the time for establishing a TCP connection to the web server host.	Secs	Typically, the TCP connection establishment must be very small (of the order of a few milliseconds). Since TCP connection establishment is handled at the OS-level, rather than by the application,

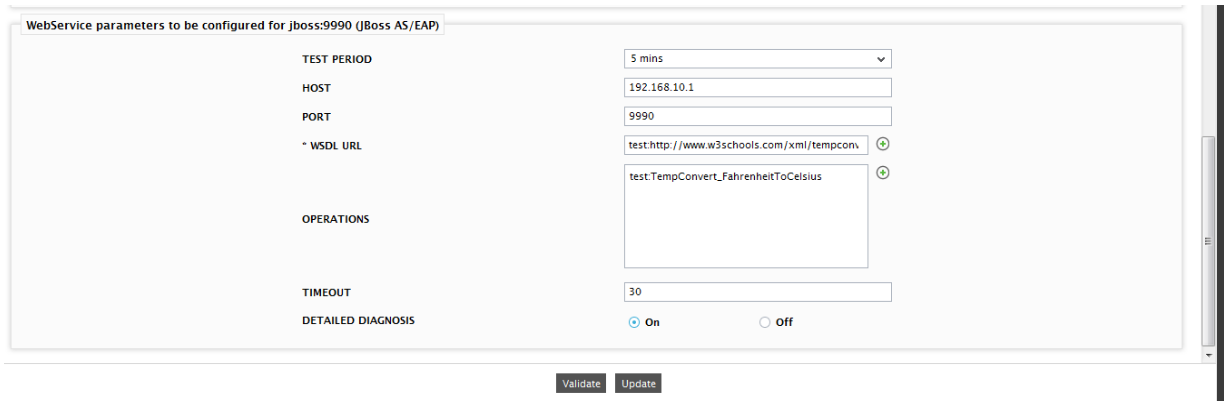
Measurement	Description	Measurement Unit	Interpretation
			an increase in this value signifies a system-level bottleneck on the host that supports the web server.
Server response time:	Indicates the time period between when the connection was established and when the web server sent back a response header to the client.	Secs	While the total response time may depend on several factors, this measure is typically, a very good indicator of a server bottleneck (e.g., because all the available server threads or processes are in use).
Response code:	The response code returned by the web server for the simulated request.	Number	A value between 200 and 300 indicates a good response. A 4xx value indicates a problem with the requested content (eg., page not found). A 5xx value indicates a server error.
Service availability:	Indicates whether/not the web service is available.	Percent	A value of 100 indicates that the web service is available and a value of 0 indicates that the web service is not available.
Operation status:	Indicates whether/not the configured operation is present in the web service.		This measure will not report metrics if the **OPERATION** parameter in the test configuration page is none in the test configuration page.
Operation Content length:	Indicates the response code returned by the server for the simulated request.	Number	A value between 200 and 300 indicates a good response. A 4xx value indicates a problem with the requested content (e.g., page not found). A 5xx value indicates a server error. This measure will not report metrics if the **OPERATION** parameter in the test configuration page is none or if

Measurement	Description	Measurement Unit	Interpretation
			an invalid **Value** is specified or if the Value is not specified in the **HTML View** tab while configuring the operation for monitoring in the test configuration page.
Operation Content validity:	This measure validates whether the operation was successful in executing the request made to it.	Percent	A value of 100% indicates that the content returned by the test is valid. A value of 0% indicates that the content may not be valid. This capability for content validation is especially important for multi- tier web applications. For example, a user may not be able to login to the web site but the server may reply back with a valid HTML page where in the error message, say, “Invalid Login” is reported. In this case, the availability will be 100 % (since we got a valid HTML response). If the test is configured such that the content parameter should exclude the string “Invalid Login”, in the above scenario content validity would have a value 0. This measure will not report metrics if the **OPERATION** parameter in the test configuration page is none or if an invalid **Value** is specified or if the Value is not specified in the **HTML View** tab while configuring the operation for monitoring in the test configuration page.
Operation execution time:	Indicates the time taken to invoke the configured	Secs	This measure will not report metrics if the **OPERATION** parameter in the

Measurement	Description	Measurement Unit	Interpretation
	operation in the web service.		test configuration page is none or if an invalid **Value** is specified or if the Value is not specified in the **HTML View** tab while configuring the operation for monitoring in the test configuration page.

3.2.10.1 Configuring Multiple WSDL URLs for Monitoring

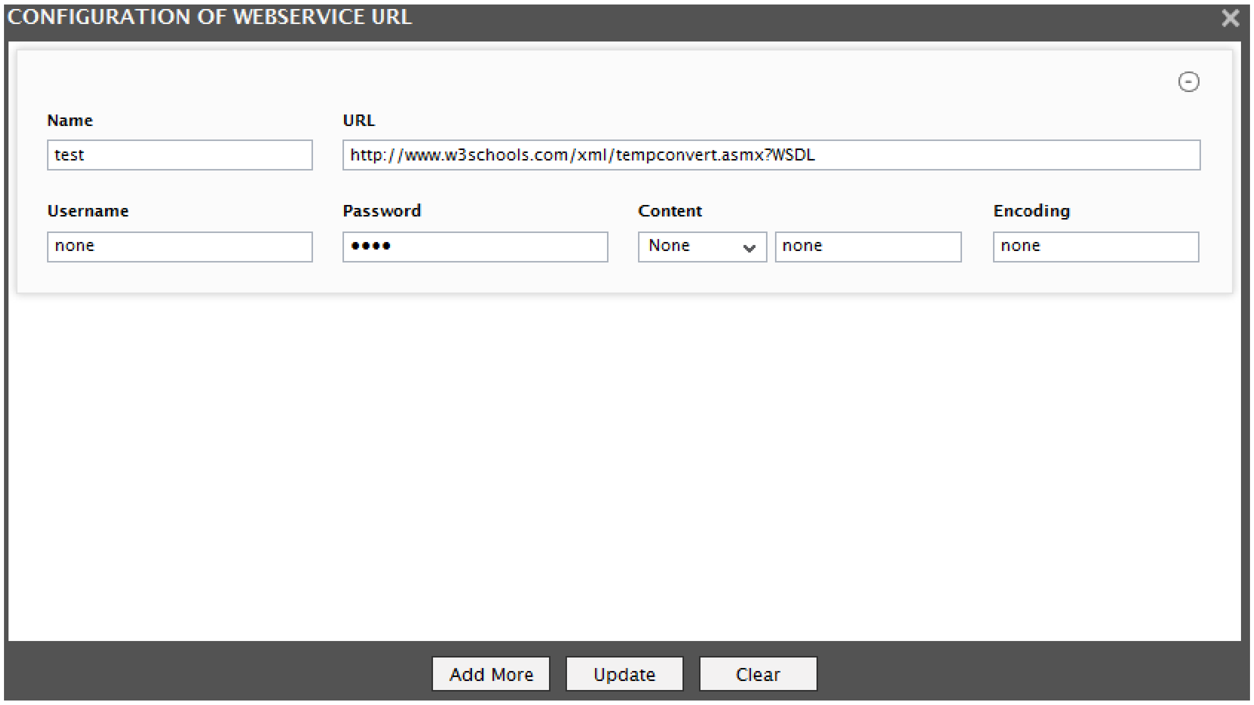
In order to enable the eG agent to connect to multiple WSDL URLs and pull out the required metrics from them, the eG administrative interface provides a special page using which different WSDL URLs and their corresponding operations that need to be monitored can be specified. To configure the WSDL URLs, do the following:


WebService parameters to be configured for jboss:9990 (JBoss AS/EAP)

TEST PERIOD	5 mins
HOST	192.168.10.1
PORT	9990
* WSDL URL	test:http://www.w3schools.com/xml/tempconv
OPERATIONS	test:TempConvert_FahrenheitToCelsius
TIMEOUT	30
DETAILED DIAGNOSIS	[✓ On] Off

Figure 3.4: Configuring the WebService test

1. Click on the encircled '+' button alongside the **WSDL URL** text box. 3.2.10 will then appear.


CONFIGURATION OF WEBSERVICE URL

Name	URL		
test	http://www.w3schools.com/xml/tempconvert.asmx?WSDL		

Username	Password	Content		Encoding
none	●●●●	None	none	none

Figure 3.5: The WebService URL Configuration page

2. Specify the following in Figure 3.5:

- **Name:** Specify a unique name by which the WSDL URL you will be specifiying shortly will be referred to across the eG user interface. This is the name that will appear as the descriptor of this test.
- **URL:** Enter the WSDL URL of the web service that this test should access.
- **Username** and **Password:** These parameters are to be set only if a specific user name / password has to be specified to login to the web service (i.e., WSDL URL) that you have configured for monitoring. In this case, provide valid login credentials using the **Username** and **Password** text boxes. If the server on which **WebService** test executes supports 'Anonymous user access', then these parameters will take either of the following values:
  - A valid **Username** and **Password** for the configured **WSDL URL**
  - none in both the **Username** and **Password** text boxes of the configured WSDL URL, if no user authorization is required
  - Some servers however, support NTLM (Integrated Windows) authentication, where valid login credentials are mandatory. In other words, a none specification will not be supported

by such servers. Therefore, in this case, against each configured **WSDL URL**, you will have to provide a valid **Username** in the format: domainname\username, followed by a valid **Password**.

  - Please be sure to check if your web service requires HTTP authentication while configuring this parameter. HTTP authentication typically involves a separate pop-up window when you try to access the page. Many services use HTTP POST for obtaining the user name and password and validating the user login. In such cases, the username and password have to be provided as part of the POST information and NOT as part of the **CREDENTIALS** specification for the WebService test.

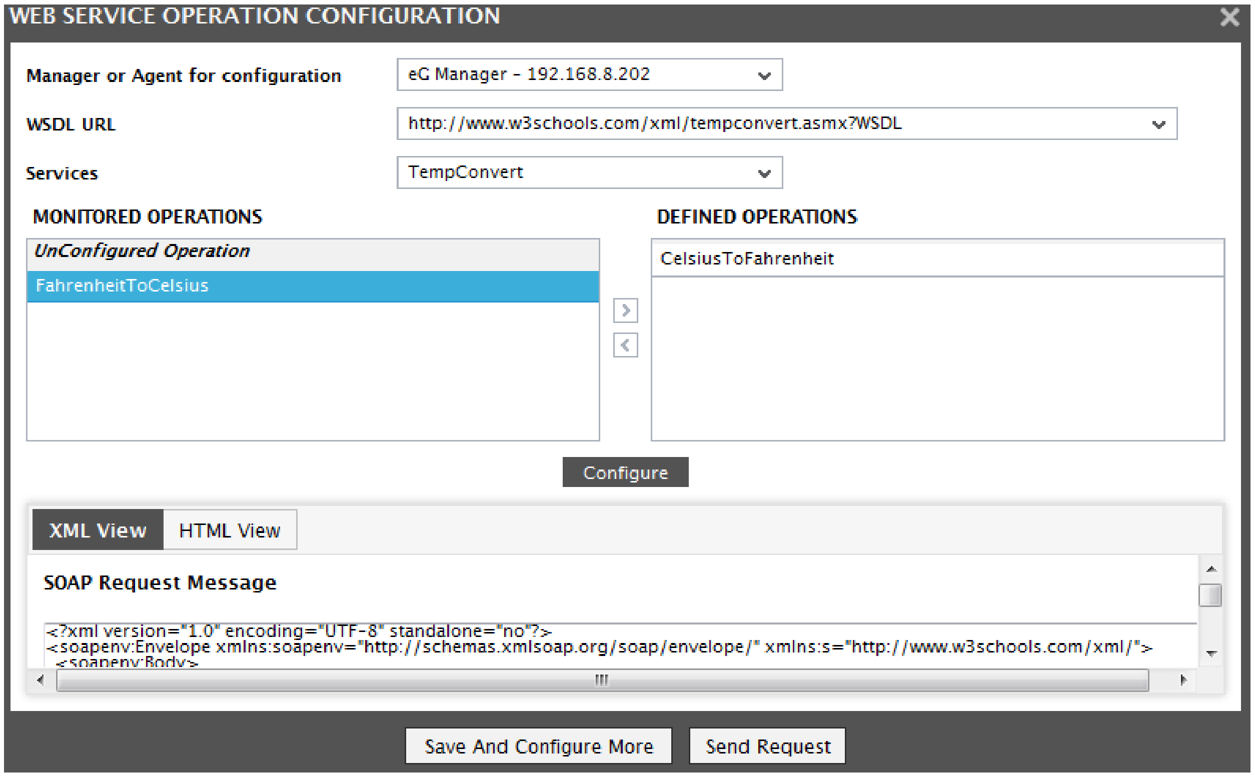
- **Content** : The **Content** parameter has to be configured with an instruction:value pair that will be used to validate the content being returned by the test. If the **Content** value is None, no validation is performed. On the other hand, if you pick the Include option from the **Content** list, it indicates to the test that for the content returned by the web server to be valid, the content must include the specified value (a simple string search is done in this case). This value should be specified in the adjacent text box. Similarly, if the Exclude option is chosen from the **Content** drop-down, it indicates to the test that the server's output is valid if it does not contain the value specified in the adjacent text box. The Include or Exclude value you specify in the text box can include wildcard characters. For example, an Include instruction can be *Home page*.
- **Encoding**: Sometimes the eG agent has to parse the **WSDL URL** content with specific encoding other than the default (ISO-8859-1) encoding. In such a case, specify the type of encoding using which the eG agent can parse the **WSDL URL** content in the **Encoding** text box. By default, this value is none.

3. Similarly, you can add multiple URL specifications by clicking the **Add More** button. To remove a WSDL URL specification, click on the encircled '-' button corresponding to it. To clear all **WSDL URL** specifications, click the **Clear** button. To update all the changes you made, click the **Update** button.
4. Once **Update** is clicked, you will return to the test configuration page as shown in Figure 3.4. The **WSDL URL** text box in the test configuration page will display just the **Name**s - i.e., the unique display names - that you may have configured for the multiple WSDL URLs, as a comma-separated list. To view the complete WSDL URL specification, click the encircled '+' button alongside the **WSDL URL** text box, once again.

3.2.10.2 Configuring Multiple Operations for Monitoring - WebServiceTest

By default, the **WebServiceTest** will be configured with the WSDL URLs that offer the web services that are to be monitored. To configure the operations that are offered by the WSDL URLs, do the following:

1. Click on the encircled '+' button alongside the **OPERATIONS** text box as shown in Figure 3.4. Figure 3.6 will then appear.


WEB SERVICE OPERATION CONFIGURATION

Manager or Agent for configuration: eG Manager - 192.168.8.202

WSDL URL: http://www.w3schools.com/xml/tempconvert.asmx?WSDL

Services: TempConvert

MONITORED OPERATIONS: *UnConfigured Operation*, FahrenheitToCelsius

DEFINED OPERATIONS: CelsiusToFahrenheit

SOAP Request Message

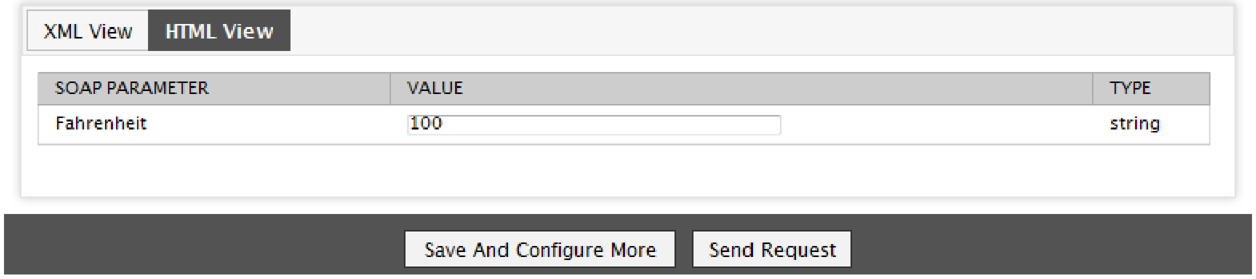
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:s="http://www.w3schools.com/xml/">
  <soapenv:Body>
```

Figure 3.6: Configuring the Web Service Operation

2. Specify the following in Figure 3.6:

   - **Manager/Agent for accessing WSDL URL:** Select the eG agent or the eG Manager that is authorized to access the configured WSDL URL from this list.
   - **WSDL URL:** Once the eG agent/eG Manager is chosen from the **Manager/Agent for accessing WSDL URL** list, this list will be populated automatically with all the WSDL URLs specified in the **WSDL URL** text box (See Figure 3.4). Select the **WSDL URL** of your choice from this list.
   - **Services:** The web services offered by the chosen WSDL URL will then be populated in this list. Select a service of your choice from this list.
     - The operations that are offered by the chosen service will then be populated in the **DEFINED OPERATIONS** list. To monitor a chosen operation, select the operation and click the **<** button. This will move the chosen operation to the **MONITORED OPERATIONS** list.

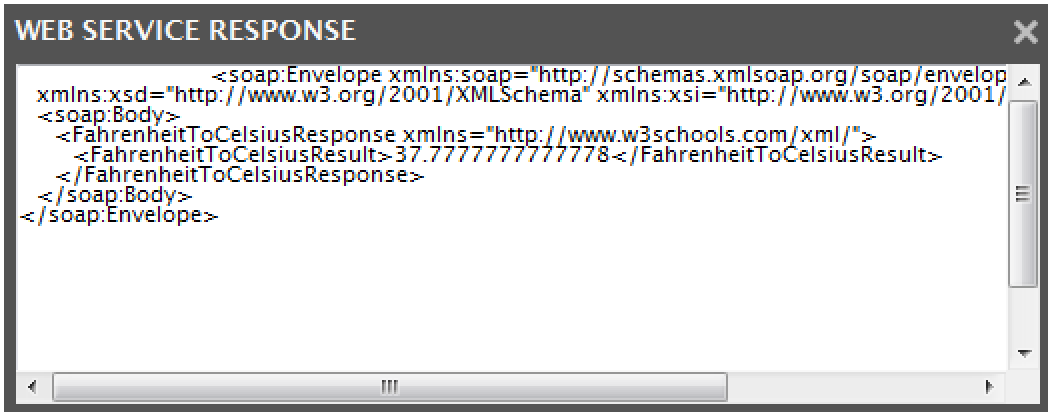
- Click the **Configure** button to save the changes.
- The eG agent uses SOAP requests to obtain the necessary metrics from the web service. Once the operation is configured, the XML View of the SOAP Request corresponding to the chosen operation will be generated and listed in the **XML View** tab. Likewise, the **HTML View** tab lists the **SOAP Parameter** that is passed to collect the required metrics for the chosen operation.
- To obtain operation-level statistics, it is important to specify a valid value in the **VALUE** text box of the **HTML VIEW** tab as shown in Figure 3.7. Each time the test is executed, this value will be provided as an input to the chosen operation.



SOAP PARAMETER	VALUE	TYPE
Fahrenheit	100	string

Figure 3.7: Specifying the value for the chosen operation

- Click the **Save and Cofigure More** button to save the changes made.
- If you wish to verify if the **VALUE** specified in the **HTML View** tab is valid, then you can do so by clicking the **Send Request** button. Figure 3.8 will then appear. If the value specified in the **VALUE** text box is indeed valid, then the operation will be performed on the value and the result will be specified. For example, if your chosen operation is FahrenheittoCelsius, the **SOAP PARAMETER** is Farenheit and the value that you wish to convert is 100, the result will be specified in the **WEB SERVICE RESPONSE** pop up window as below: *<FahrenheitToCelsiusResult>37.7777777777778</FahrenheitToCelsiusResult>*

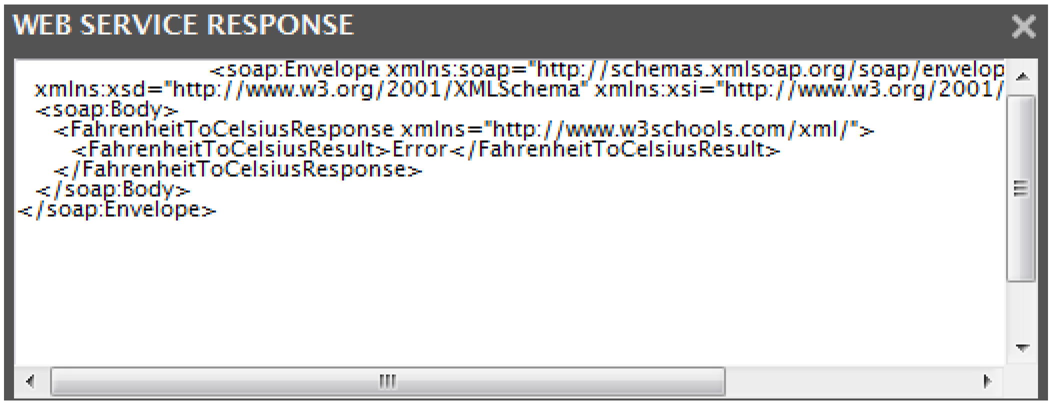


```
                    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelop
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
  <soap:Body>
    <FahrenheitToCelsiusResponse xmlns="http://www.w3schools.com/xml/">
      <FahrenheitToCelsiusResult>37.7777777777778</FahrenheitToCelsiusResult>
    </FahrenheitToCelsiusResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 3.8: The value that appears when the operation is performed successfully

If you have specified an invalid value, then a message as follows will be displayed in the pop up window:

<FahrenheitToCelsiusResult>Error</FahrenheitToCelsiusResult>



```
                    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelop
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
  <soap:Body>
    <FahrenheitToCelsiusResponse xmlns="http://www.w3schools.com/xml/">
      <FahrenheitToCelsiusResult>Error</FahrenheitToCelsiusResult>
    </FahrenheitToCelsiusResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 3.9: An Error appearing during value conversion

If you do not specify a **VALUE** or specify an invalid value, operation-level statistics will not be collected by the eG agent and such metrics will not be available in the eG monitoring interface.

3. Similarly, you can configure multiple Operations by clicking the **Configure** button in Figure 3.6. To remove an operation, select the operation from the **MONITORED OPERATION** list and click the **>** button.
4. Once **Save and Configure More** button is clicked, you will return to the test configuration page (see Figure 3.4). The **OPERATIONS** text box in the test configuration page will display just the operations, as a comma-separated list. To view the complete operation specification, click the encircled '+' button alongside the **OPERATIONS** text box, once again.

3.3 The Application Transactions Layer

This layer will appear only if the eG Java Business Transaction Monitor (BTM) is enabled on the EAP server. To know how to enable Java BTM, refer to Section **2.1.4**.

Once this is done, then a **Java Business Transactions** test will appear in this layer. To know how to configure this test and what metrics it reports, refer to the *Java Business Transaction Monitoring* document.

3.4 The Java Transactions Layer

By default, this layer will not be available for the JBoss EAP server. This is because, the **Java Business Transactions** test mapped to this layer is disabled by default. To enable the test, follow the *Agents -> Tests -> Enable/Disable* menu sequence, select *JBoss AS / EAP* as the **Component type**, *Performance* as the **Test type**, and then select **Java Business Transactions** from the **DISABLED TESTS** list. Click the **Enable** button to enable the selected test, and click the **Update** button to save the changes.

3.4.1 Java Business Transactions Test

The responsiveness of a transaction is the key determinant of user experience with that transaction; if response time increases, user experience deteriorates. To make users happy, a Java business transaction should be rapidly processed by each of the JVM nodes in its path. Processing bottlenecks on a single JVM node can slowdown/stall an entire business transaction or can cause serious transaction errors. This in turn can badly scar the experience of users. To avoid this, administrators should promptly identify slow/stalled/errored transactions, isolate the JVM node on which the slowness/error occurred, and uncover what caused the aberration on that node – is it owing to SQL queries executed by the node? Or is it because of external calls – eg., async calls, SAP JCO calls, HTTP calls, etc. - made by that node? The **Java Business Transactions** test helps with this!

This test runs on a BTM-enabled JVM in an IT infrastructure, tracks all the transaction requests received by that JVM, and groups requests based on user-configured pattern specifications. For each transaction pattern, the test then computes and reports the average time taken by that JVM node to respond to the transaction requests of that pattern. In the process, the test identifies the slow/stalled transactions of that pattern, and reports the count of such transactions and their responsiveness. Detailed diagnostics provided by the test accurately pinpoint the exact transaction URLs that are slow/stalled, the total round-trip time of each transaction, and also indicate when such transaction requests were received by that node. The slowest transaction in the group can thus be identified.

For this test to run and report metrics on a JBoss EAP server, you first need to BTM-enable the JBoss EAP JVM. To know how, refer to the Installing eG Java BTM on JBoss EAP topic in the *Java Business Transaction Monitoring* document.

Then, proceed to configure this test. Refer to the Java Business Transactions Test topic in the *Java Business Transaction Monitoring* document to know how to configure this test and learn about the metrics it reports.

About eG Innovations

eG Innovations provides intelligent performance management solutions that automate and dramatically accelerate the discovery, diagnosis, and resolution of IT performance issues in on-premises, cloud and hybrid environments. Where traditional monitoring tools often fail to provide insight into the performance drivers of business services and user experience, eG Innovations provides total performance visibility across every layer and every tier of the IT infrastructure that supports the business service chain. From desktops to applications, from servers to network and storage, from virtualization to cloud, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations is dedicated to helping businesses across the globe transform IT service delivery into a competitive advantage and a center for productivity, growth and profit. Many of the world's largest businesses use eG Enterprise to enhance IT service performance, increase operational efficiency, ensure IT effectiveness and deliver on the ROI promise of transformational IT investments across physical, virtual and cloud environments.

To learn more visit www.eginnovations.com.

Contact Us

For support queries, email support@eginnovations.com.

To contact eG Innovations sales team, email sales@eginnovations.com.