TIBCO Silver® Fabric

JBoss Enterprise Application Platform Enabler Guide

Software Release 5.6
November 2015

Two-Second Advantage®

Important Information

Contents

Preface

TIBCO Silver® Fabric combines the flexibility and scalability of the public cloud with the security and control of your own data center. It brings the elasticity of cloud computing to your organization – supporting existing solutions within your current infrastructure while automatically scaling resources to meet demand.

Topics

Related Documentation

This section lists documentation resources you may find useful.

For the latest version of documentation, including any changes or additions made since the last product release, please visit http://docs.tibco.com.

TIBCO Silver Fabric Documentation

The following documentation is included with Silver Fabric in Adobe Acrobat (PDF) format. To view the guides, log in to the Administration Tool and go to **Admin > Documentation**. The PDF files are also on the Broker at *SF_HOME*`/webapps/livecluster/admin/docs`. The following documents form the Silver Fabric documentation set:

- *Introducing Silver Fabric* Contains an introduction to Silver Fabric, including definitions of key concepts and terms, such as Enablers, Stacks, Components, Engines, and Brokers. Read this first if you are new to Silver Fabric.
- *Silver Fabric Installation Guide* Covers installation of Silver Fabric for Windows and Unix, including Brokers, Engines, and pre-installation planning.
- *Silver Fabric Cloud Administration Guide* Covers Silver Fabric cloud administration, configuration of Engines, Enablers, and Components, and configuration and use of Skyway. Also covers security, general maintenance, performance tuning, and database administration.
- *Silver Fabric Developer's Guide* Developer-related topics such as logging and debugging, using the Admin API, and the Enabler SDK.
- *Silver Fabric Developer's Tutorial* Tutorials for developers, such as how to write Enablers and Asset Managers.
- *Silver Fabric User's Guide* Covers Silver Fabric use and operation, including management of Engines, Enablers, Components, and Stacks.
- *Silver Fabric Skyway User's Guide* Covers usage of Skyway, which enables users to quickly and easily provision and manage their Silver Fabric Stacks.
- *Silver Fabric Tomcat Enabler Guide* Covers installation and configuration of applications run on the Tomcat Enabler.
- *Silver Fabric Command Line Enabler Guide* Covers installation and configuration of applications run on the Command Line Enabler.

Other Documentation and Help

Additional help and information is available from the following sources:

- *Silver Fabric Administration Tool Help* Context-sensitive help is provided throughout the Silver Fabric Administration Tool by clicking the Page Help button located on any page.
- *API Reference* Silver Fabric API reference information is available in the Silver Fabric SDK in the `api` directory in JavaDoc format. You can also view and search them from the Silver Fabric Administration Tool; log in to the Administration Tool and go to **Admin > Documentation**.

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME	Many TIBCO products must be installed within the same home directory. This directory is referenced in documentation as *TIBCO_HOME*. The default value of *TIBCO_HOME* depends on the operating system. For example, on Windows systems, the default value is `C:\tibco`.
SF_HOME	TIBCO Silver® Fabric installs into a directory within *TIBCO_HOME*. This directory is referenced in documentation as *SF_HOME*. The default value of *SF_HOME* depends on the operating system. For example on Windows systems, the default value is `C:\tibco\fabric`.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [enable | disable]`
`italic font`	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms. For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*

Table 1 General Typographical Conventions (Continued)

Convention	Use
Key combinations	Key names separated by a plus sign indicates keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical `OR` that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand param1 | param2 | param3`

Table 2 Syntax Typographical Conventions (Continued)

Convention	Use

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	
	`MyCommand {param1 param2} | {param3 param4}`
	
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	
	`MyCommand {param1 | param2} {param3 | param4}`
	
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	
	`MyCommand param1 [param2] {param3 | param4}`

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts, a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access All TIBCO Documentation

After you join TIBCOmmunity, you can access the documentation for all supported product versions here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows.

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Introduction

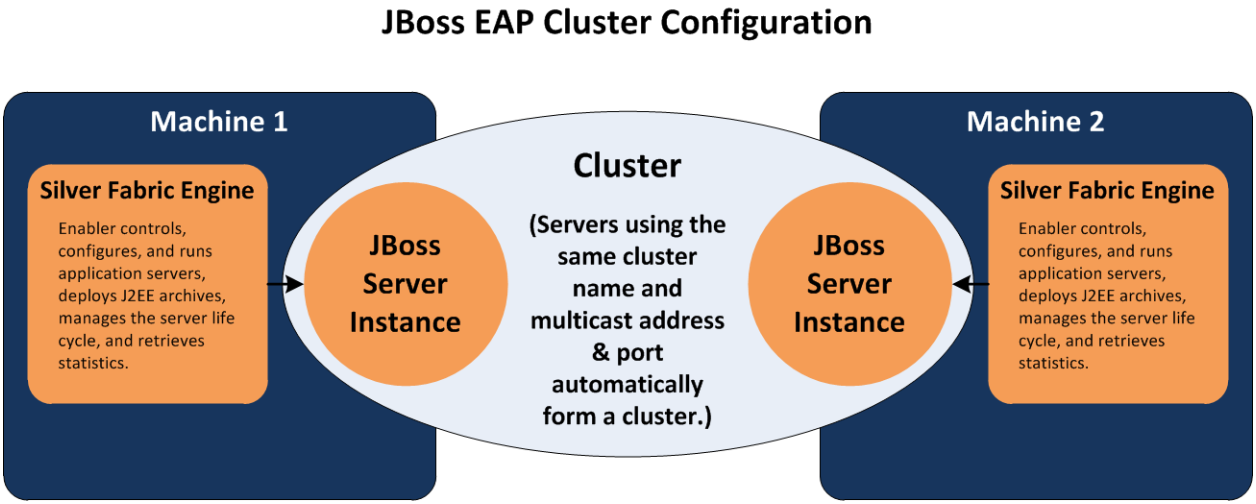
A Silver Fabric Enabler allows an external application or application platform, such as a J2EE application server, to run in a TIBCO Silver® Fabric software environment. The JBoss Enterprise Application Platform (EAP) Enabler for Silver Fabric provides integration between Silver Fabric and the JBoss Enterprise Application Platform (EAP) from Red Hat. The Enabler automatically provisions, orchestrates, controls and manages a JBoss EAP standalone or clustered environment. JBoss EAP server instances are distributed and run on Silver Fabric Engines, either as standalone servers or as servers belonging to a JBoss EAP partition (cluster). The management of the JBoss EAP environment through Silver Fabric is dynamic, centralized, and largely automated. This centralized management adapts the structure of the JBoss EAP environment, and manages the life cycle process state of application servers, as specified in Silver Fabric by Policies that are directed by business objectives associated with the target applications.

Topics

Architecture

The Silver Fabric Engine manages a JBoss EAP standalone or clustered server instance. The Silver Fabric Broker and the Silver Fabric Engine instance collaborate to perform the following actions:

- Automatically provision all JBoss EAP-related and Java-Development-Kit (JDK) software to a machine with a Silver Fabric Engine. All the software that is automatically provisioned is encapsulated within archive libraries (zip or tar.gz files) and distributed to computers running Silver Fabric Engines.
- Automatically start and stop JBoss EAP application servers on a node, as required by schedules specified within Silver Fabric.
- Monitor the health of the JBoss EAP Server and retrieve statistics.



Before Beginning

This guide provides the instructions for installing and configuring the JBoss EAP Enabler for Silver Fabric. This guide presumes a Silver Fabric Broker is running with at least one Engine installed, and that you have the Broker's hostname, a username, and password for the Silver Fabric Administration Tool. If this isn't true, see the *Silver Fabric Installation Guide*, or contact your administrator.

This guide presumes a strong familiarity with your particular version of JBoss EAP. If you are uncertain about how to achieve a particular task, consult your version-specific JBoss EAP documentation.

Requirements

Please see the included Silver Fabric Readme for the latest prerequisites required for this Enabler.

Chapter 2 Installation

This chapter provides information on installing the JBoss EAP Enabler.

Topics

Required Grid Libraries

The JBoss EAP Enabler consists of an Enabler Runtime Grid Library, a Distribution Grid Library, and an optional JDK Grid Library. The Enabler Runtime contains information specific to a Silver Fabric version that is used to integrate the Enabler, and the Distribution contains the application server or program used for the Enabler.

The required Grid Libraries for each version of the JBoss EAP Enabler are listed below.

JBoss EAP Enabler Version	Required Grid Libraries
5.0	`TIB_silver_fabric_5.6.0_jboss-eap_5.0_distribution-gridlib.zip`
	`TIB_silver_fabric_5.6.0_jboss_eap_5.0_enabler.tar.gz`
	`j2sdk-`*`platform`*`-1_6_0_23-gridlib.zip`
5.1	`TIB_silver_fabric_5.6.0_jboss-eap_5.1.1_distribution-gridlib.zip`
	`TIB_silver_fabric_5.6.0_jboss_eap_5.1.1_enabler.tar.gz`
	`j2sdk-`*`platform`*`-1_6_0_23-gridlib.zip`
6.0	`TIB_silver_fabric_5.6.0_jboss-eap_6.0_distribution-gridlib.zip`
	`TIB_silver_fabric_5.6.0_jboss_eap_6.0_enabler.tar.gz`
	`j2sdk-`*`platform`*`-1_6_0_23-gridlib.zip (optional)`
6.4.1	`TIB_silver_fabric_5.6.0_jboss-eap_6.4_distribution-gridlib.zip`
	`TIB_silver_fabric_5.6.0_jboss_eap_6.4_enabler.tar.gz`
	`j2sdk-`*`platform`*`-1_6_0_23-gridlib.zip (optional)`

Required JDK Grid Libraries

Silver Grid includes the Java 7 SDK, but only JBoss EAP 6 supports Java 7. JBoss EAP 5.0 and 5.1 only support the Java 6 SDK, so you must deploy a Grid Library containing the Java 6 SDK. You can also optionally run JBoss EAP 6 with Java 6 if your application requires it.

To run the Enabler on multiple platforms, you must deploy multiple JDK Grid Libraries, one for each platform. Name each Grid Library as shown as above, substituting *platform* with the platform on which Silver Fabric Engines run JBoss EAP Enablers.

For information on how to create a JDK Grid Library, see Chapter 6, Distribution Grid-library.xml Files, on page 37.

Creating a Distribution Grid Library

Distribution Grid Libraries are not provided by TIBCO. For information on how to create them, see Chapter 6, Distribution Grid-library.xml Files, on page 37.

Installing the JBoss EAP Enabler

To install the JBoss EAP Enabler:

1. You cannot upgrade or remove Grid Libraries while the respective Component and Enabler are part of an active Stack.
2. Copy the desired JBoss EAP Enabler version Grid Library files to the *SF_HOME*`/webapps/livecluster/deploy/resources/gridlib` directory in the following order:
   - Distribution
   - Enabler Runtime
   - JDK Grid Library

Copying the Grid Libraries files to this directory also extracts them to the deployed directory *SF_HOME*`/webapps/livecluster/deploy/expanded/`. This overwrites any changes to the existing Grid Library in the staging directory.

3. Verify successful installation by selecting **Stacks > Enablers** in the Silver Fabric Administration tool and ensuring that the Enabler appears in the list.

Chapter 3 Configuration

This chapter provides information on configuring the JBoss EAP Enabler.

The instructions presume you are familiar with your version of JBoss EAP. If you are uncertain about how to achieve a particular task, consult your version-specific JBoss EAP documentation.

Topics

Running JBoss EAP Applications in Standalone Mode

The following section describes how to run a JBoss Component in standalone mode. In this mode, Silver Fabric Engines run individual self-contained servers that do not communicate with each other. On activation, the Engine deploys archives contained in the Component to the server.

In this mode, each JBoss EAP server instance is a unique cluster in which it is the only member. The name of the cluster has the form *hostname-instance*.

To define the Component in Silver Fabric:

1. Enter your Silver Fabric host and port in your browser and log in to the Silver Fabric Administration Tool.
2. Go to **Stacks > Components**.
3. Select **Create New J2EE Component** from the Global Actions list.
4. If prompted, select the JBoss EAP Enabler and the desired Enabler Version from the list.
5. Enter a Component name in the **Name** field.
6. Click **Next**. The Configure Component Features screen appears.
7. HTTP Support and Archive Management Support are added to the feature list by default. Make sure that the Clustering Support option is *not* selected.
   - — You can further customize Component features as needed by selecting the feature and clicking **Edit**. When finished, click **Next**. The Configure Component Options screen appears.
8. Enter any desired options and click **Next** until the Add/Remove Archive Files screen appears.
9. Click **Add** and upload the J2EE archives that you want to deploy and run on the JBoss server.
10. Configure the options on the remaining screens as needed. Additional options are also configurable through context variables. To customize the username and password for the JBoss EAP instance, see “Securing Passwords”.
11. Click **Finish**.
12. Go to **Stacks > Components** in the Silver Fabric Administration Tool.
13. Select **Publish Component** from the corresponding Actions list of the Component you created.

14. Create a Stack and add the Component as described in the Silver Fabric documentation. Activate the Stack when desired.

Running Clustered JBoss EAP Applications

You can configure Silver Fabric to run with multiple JBoss EAP Partitions (clusters). In this configuration, Engines running a JBoss EAP J2EE Component act as nodes in a cluster. Components with the same cluster name are each a node on a JBoss EAP cluster. Providing different cluster names for J2EE Components creates multiple JBoss EAP partitions. The name provided while creating a Component using the Component Wizard is a system property to the JBoss EAP process and is the name used in the `cluster-service.xml` file. These multiple clusters can reside on the same network as well.

To define a clustered JBoss EAP application:

1. Start Silver Fabric and log into the Silver Fabric Administration Tool.
2. Go to **Stacks > Components**.
3. Select **Create New J2EE Component** from the Global Actions list.
4. If prompted, select the JBoss EAP Enabler and the desired version.
5. Enter a Component name.
6. Click **Next**. The Configure Component features screen appears.
7. Select the **Clustering Support** option from the Add feature list. The Enter feature values screen appears.
8. Enter the correct values in the appropriate fields and click **OK**.
9. Click **Next** again. The Configure Component options screen appears.
10. Enter any desired options and click **Next**.
11. Click **Next** until you reach the Add/Remove Archive Files screen.
12. Click **Add** and select the archives to add to the Component.
13. Click **OK** to add the archives to the J2EE Component.
14. Configure the options on the remaining screens as needed. Additional options are also configurable through context variables.
15. Click **Finish.**
16. Select **Publish Component** from the Actions list to publish the Component.

When running multiple clustered JBoss EAP Components concurrently, configure each Component with a unique cluster name, and a unique multicast address.

Multicast Address

You can set the multicast address on the Add/Edit Component Features page by selecting and editing the Clustering Support feature.

Note that JBoss uses several different multicast ports for different services. For convenience, the `JBOSS_PARTITION_UDPPORT` runtime context variable may be set to control the socket on which the HA-JNDI service listens for cluster auto-discovery requests.

Securing Passwords

JBoss EAP Server 5.0 and 5.1 contains a secured web console and JMX console. The JBoss EAP Enabler includes variables for the user name and password. Those variables are represented at runtime by the following runtime context variables:

- SECURITY_PRINCIPAL (User name. Default value: admin)
- SECURITY_CREDENTIALS (Password. Default value: admin123)

Activation stores the MD5 hash of the password in the appropriate JBoss EAP configuration files. The engine uses the password to authenticate when connecting to the JMX server. If you access the Web Console of the JBoss EAP server on an engine, you also must enter the username and password. By default, the password is in plain text in the `container.xml` file and in Engine log files.

For further security, encrypt the password rather than storing it as plain text. To encrypt, change the type of the SECURITY_CREDENTIALS variable to Encrypted.

Configuring Password Encryption in the Component

To configure the Component to encrypt passwords:

1. Go to the **Stacks > Components** page.
2. Select your Component and select **Edit Component**.
3. Click **Add/override/edit Enabler and Component-Specific Runtime Context Variables.**
4. Choose **Encrypted** from the **Add Variable** drop down.
5. For the variable, enter SECURITY_CREDENTIALS. For the value, enter the desired password. Optionally, add a description.
6. Click **OK > Finish.** Republish the Component.

Configuring Password Encryption in the Enabler

To configure the password at the Enabler level:

1. Go to the **Stacks > Enablers** page.
2. Choose **Edit Enabler** for the JBoss EAP Enabler.
3. Click **Add, Remove** or **Edit** runtime variables.
4. Click the SECURITY_CREDENTIALS variable.

5. Click **Delete**.
6. Choose **Encrypted** from the **Add Variable** drop down.
7. For the variable, enter `SECURITY_CREDENTIALS`. For the value, enter the password you want to use. Optionally, add a description.
8. Click **OK > Finish.** Republish the Enabler.

Securing Communication Using SSL

You can configure one-way or two-way SSL on your Component. By default, the JBoss EAP Server listens on both an HTTP port and an HTTPS port. However, you can control whether the Engine communicates with the JBoss EAP Server over HTTP or HTTPS. You can also control whether client certificates are requested and used. The SSL connections are:

- Silver Fabric Engine to JBoss EAP Server
- Clients (including VirtualRouter) to JBoss EAP Server

The JBoss EAP Enabler included with Silver Fabric uses a default keystore with certificates signed by the TIBCO certificate. The certificates are in the same certificate chain, so Engines trust them without modification.

By default, the locations of the server keystore and server trust store are in the same file:

```
${JBOSS_SERVER_BASE_DIR}/${SERVER_NAME}/conf/server.keystore
```

This default keystore enables all Silver Fabric components to trust the application server as well.

When Broker-Engine communication is over SSL, provide Java SSL properties before the Engine starts. Go to **Config > Engines** to specify Engine JVM command-line arguments. For example, `-Djavax.net.ssl.keyStore=`*`yourKeystoreFile`*.

One-Way SSL

One-way SSL is the most common, standard implementation of SSL in client/server connections. In this mode, when a client attempts to connect with the server, the server offers the client a signed certificate. This certificate can be self signed or signed by a Certificate Authority (CA). The connection succeeds if the client configuration accepts self-signed certificates or when the client trusts the CA in its local trust store and validates the certificate.

To enable SSL in your Silver Fabric Component:

1. Go to **Stacks > Components** in the Silver Fabric Administration Tool.
2. Select **Edit Component** from the Actions list adjacent to your Component.
3. Select **Add/Edit Component Features**.
4. Select the **HTTP Support** feature and click **Edit**.
5. Select the **HTTPS Enabled** option. Optionally, you can clear the HTTP Enabled option to create a pure SSL configuration. Note that if both are checked, the Engine favors SSL to connect to the server instance.

6. Click **OK**.
7. The server uses the demo keystore bundled with JBoss EAP by default. If this is sufficient for your needs, click **Finish**. Otherwise, specify runtime context variables in the next step.
8. Select **Add/override/edit Enabler and Component-Specific Runtime Context Variables** and configure the following Environment variables as necessary:
   - `IGNORE_HOSTNAME_VERIFICATION` - Specifies if hostnames are verified. The default is false.
   - `SERVER_KEY_STORE_FILE` - Server key store file location for incoming SSL connections.
   - `SERVER_KEY_STORE_PASSWORD` - Password for the server key store.
   - `CLIENT_TRUST_STORE_FILE` - Client trust store file location for validating the certificate received from the JBoss EAP server.
   - `CLIENT_TRUST_STORE_PASSWORD` - Password for the client trust store.

   For JBoss EAP 6, also set the following if needed:
   - `SERVER_KEY_ALIAS` - Private key entry's alias in the server key store. Change this if you changed the location of the server key store file with `SERVER_KEY_STORE_FILE`.
9. Click **Finish**.
10. Select **Publish Component** from the Actions list to deploy the Component.

For context variables that require a fully-qualified pathname, you can use the following environment variables:

- `${JBOSS_HOME}` – Expands to the base directory of the JBoss EAP distribution on the Engine.
- `${JBOSS_SERVER_BASE_DIR}` – Expands to the base directory of the Component's runtime directory located in the Engine's work directory.

Two-Way SSL

In two-way SSL, the JBoss EAP server additionally tries to establish trust with the connecting client by requesting a certificate from the client, and either accepting or rejecting it based on its own trust settings.

To enable two way SSL in your Silver Fabric Component:

1. Go to **Stacks > Components** in the Silver Fabric Administration Tool.
2. Select **Edit Component** from the Actions list adjacent to your Component.

3. Select **Add/override/edit Enabler and Component-Specific Runtime Context Variables**.
4. Select the **Environment** variable from the Add Variable list. The Add Variable screen appears.
5. Complete the screen as follows:
   - Enter `TWO_WAY_SSL_ENABLED` in the name field.
   - Enter **true** in the value field.
   - Select **None** as the Auto Increment Type.
6. Click **OK**.
7. Configure the following Environment variables as necessary:
   - `CLIENT_KEY_STORE_FILE` — The client key store file used when connecting to the JBoss MBean server.
   - `CLIENT_KEY_STORE_PASSWORD` — The client key store password used when connecting to the JBoss MBean server.

   For JBoss EAP 6, also set the following if needed:

   - `SERVER_KEY_ALIAS` - Private key entry's alias in the server key store. Change this if you changed the location of the server key store file with `SERVER_KEY_STORE_FILE`.

   Click **OK** when finished. You must now upload your key store.
8. Select **Add/override/customize Enabler and Component-specific Content Files**.
9. Click **Upload**. The Config File screen appears.
10. Complete the screen as follows:
    - Enter a name for your file in the **Name** field.
    - Enter the value you used for `CLIENT_KEY_STORE_FILE` in the **Relative Path** field.
    - Enter the keystore file containing your client certificate in the **File** field.
11. Click **Finish**.
12. Select **Publish Component** from the Actions list to deploy the Component.

Archive Management

The JBoss EAP Enabler can dynamically manage the set of application archives that are deployed and running on JBoss EAP servers. This consists of archive detection, context URL detection, archive scaling, and continuous deployment operations.

- **Archive Detection** The JBoss EAP Enabler regularly detects the current set of archives that are deployed and running on the server. On the Broker, this list is reflected in the `ActivationInfo` associated with the Engine. The `ARCHIVE_DETECTION_ENABLED` and `ARCHIVE_DETECTION_FREQUENCY` runtime context variables can be modified to control whether or not polling of archives is performed, and how often this detection happens.
- **Context URL Detection** The JBoss EAP JMX server is queried for the current set of context URLs associated with running web applications. This set of URLs is reported to the Broker and updated in VirtualRouter.
- **Archive Scaling** Archive scaling is a method of dynamically adding or removing archives from a running environments, without affecting what is already running; these archives can then scale up or down as per archive-specific allocation rules. See the *Silver Fabric Administration Guide* for more details on creating Scaling rules.
- **Continuous Deployment** Users can deploy, start, stop, and undeploy application archives to JBoss EAP by using the archive-level API operations provided by the Silver Fabric Broker through REST, Ant, and the Command Line Interface (CLI). See the *Silver Fabric Administration Guide* for more details on using these tools to manage archives.

Continuous Deployment to Clustered JBoss Applications

When using continuous deployment to deploy archives directly to Components and JBoss clustering is enabled, the default behavior is that the deployment and running of an archive on one node of the cluster does not affect other nodes.

Some versions of the JBoss Enabler can support the JBoss farming service. This enables you to deploy an archive to a single Engine, and when you start that archive, it will start on each Component in the cluster. Likewise, you can stop the archive across all nodes in the cluster, and undeploying the archive from the original Engine will undeploy it from all Engines in the cluster.

Supported Versions

Continuous deployment to clustered JBoss Applications is supported in JBoss EAP 5.0.0 and JBoss EAP 5.1.1 Enablers only. JBoss clustering must be enabled. See Running Clustered JBoss EAP Applications on page 12 to enable clustering.

Using the Farming Service

Deploying and Starting Archives

To deploy and start an archive on all nodes:

1. Deploy an archive to a single Engine using REST, the CLI or an Ant task.
2. On that same Engine, start the archive using REST, the CLI, or an Ant task, and set the property named `START_ARCHIVE_CLUSTER` to `true`.

   All Engines in the cluster will display the archive under Deployed Archives and Running Archives in the Engine details. The context will also be detected for all the nodes in the cluster.

Note that in order to start the archive across the cluster, it must be started on the same Engine where it was deployed, and it must not be running on that node.

Stopping Archives

To stop an archive that is running on all nodes:

1. Stop the archive on any Engine using REST, the CLI, or an Ant task, and set the property named `STOP_ARCHIVE_CLUSTER` to `true`.

Undeploying Archives

To undeploy a stopped archive on all nodes:

1. On the Engine where you deployed the archive, undeploy the archive using REST, the CLI, or an Ant task.

This will only undeploy a stopped archive, and it must be done on the Engine where the archive was deployed. If you attempt to undeploy from any other Engine, you will get a warning in the Engine log that the archive cannot be undeployed because the archive does not exist. However, REST, the CLI, or the Ant task will get a response that the archive was undeployed.

To undeploy a running archive on all nodes:

1. On the Engine where you deployed the archive, undeploy the archive using REST, the CLI or an Ant task, and set the property named

`STOP_ARCHIVE_CLUSTER` set to `true`, and the property `FORCE_UNDEPLOY` to `true`.

As with the undeploy above, if the Engine where the archive was deployed is not used a warning will be displayed in the Engine log.

With regards to archive scaling, an archive that is running on the cluster can be scaled up to another Component that is not in the cluster.

Chapter 4 Running Your Component

After configuring and activating your Silver Fabric JBoss EAP Component, you can access the applications running on the JBoss EAP Servers. To further verify successful Component configuration, access the Web application through the VirtualRouter Status page as described below.

Topics

Verifying Your Component Configuration

To verify your Component configuration:

1. Go to **Engines > Engines** in the Silver Fabric Administration Tool and ensure that at least one instance of the Component is running on an Engine.
2. Go to **Admin > VirtualRouter**. The VirtualRouter page contains a table for each VirtualRouter client that is currently running. Select the **VirtualRouter Properties** action next to each client to show the Component each host is running.
3. When you find the client running your Component, select the **Status Page** action. The VirtualRouter Status page is shown:
4. In the **Relative URLs** column, entries for each web application are listed. Click a URL to connect to the web application. HTTP requests made when the URL is clicked in the Component-level **Relative URLs** column, are sent to VirtualRouter, which forwards the request to an Engine running the Component. HTTP requests made when the URL is clicked on the Engine-level Relative URLs column are sent directly to that Engine.
   - Alternatively, you can mimic the behavior of the VirtualRouter Status page by directing your browser to the address/port of your Silver Fabric Broker with a relative URL from your Component, such as `http://myserver:8080/myapp/index.html` (where `myapp` is one of the deployed J2EE application). If you cannot access your Web application, reverify the Component configuration based on the steps in Chapter 3, Configuration, page 9.
   - For testing, you can also directly access JBoss EAP servers running on Silver Fabric Engines. The listen port of the server is generally the value of the relevant runtime context variable for the Component (`HTTP_PORT` or `HTTPS_PORT`) plus the Engine instance number. For example, `HTTP_PORT` is by default 9080, so a JBoss EAP server running on an Engine mymachine-2 is listening on port 9082.
     Note that you must access the JMX console application of a JBoss EAP server directly on an Engine. For user name and password, see the runtime context variables `SECURITY_PRINCIPAL` and `SECURITY_CREDENTIALS` defined in the Enabler or optionally defined in and overridden by the Component.

Load Balancing

Load balancing is recommended for proxying requests from clients and routing them to Engines running the appropriate Component. Achieve load balancing through mappings between relative URLs and Components.

VirtualRouter

VirtualRouter is Silver Fabric's load balancer for HTTP-enabled Components. An instance of VirtualRouter runs by default on each Silver Fabric Broker and can also run externally. Additionally, VirtualRouter maintains HTTP Session mappings between clients and JBoss EAP servers by examining the standard `JSESSIONID` cookie. For more information about using VirtualRouter, see the Silver Fabric documentation.

Custom External Load Balancer

You can create a custom load balancer to distribute requests across your Component. Accommodating the dynamic nature of Silver Fabric allocation requires a mechanism to update the routing list of the custom balancer when the Silver Fabric Component allocation changes. Notification of Component activation/deactivation events can occur in the following ways:

- ServerHook events
- SNMP traps
- Web Service calls

Refer to the Silver Fabric documentation for more information on configuring and using these methods.

Chapter 5 Statistics and Variables

This chapter provides information on statistics and runtime context variables available in the JBoss EAP Enabler.

Topics

Statistics

This section describes the default statistics supported by JBoss EAP Enabler. The Enabler uses JMX to retrieve statistic values from MBean attributes on the JBoss EAP Server. You can select and track these statistics from the Component Wizard. Tracked statistics are available for report output. You can also create policy rules based on any tracked statistic.

Table 3 JBoss EAP Statistics

Name	Description	MBean/Attribute	Units
JBoss Thread Count	The number of busy threads.	jboss.web:name=http-${JBOSS_BIND_ADDRESS}-${HTTP_PORT},type=ThreadPool/currentThreadsBusy	Threads
JBoss HTTPS Thread Count	The number of busy threads.	jboss.web:name=http-${JBOSS_BIND_ADDRESS}-${HTTPS_PORT},type=ThreadPool/currentThreadsBusy	Threads
JBoss Throughput	The number of requests processed per second.	jboss.web:name=http-${JBOSS_BIND_ADDRESS}-${HTTP_PORT},type=GlobalRequestProcessor/requestCount	Requests per second
JBoss HTTPS Throughput	The number of requests processed per second.	jboss.web:name=http-${JBOSS_BIND_ADDRESS}-${HTTPS_PORT},type=GlobalRequestProcessor/requestCount	Requests per second
JBoss Data Throughput	The number of bytes sent per second.	jboss.web:name=http-${JBOSS_BIND_ADDRESS}-${HTTP_PORT},type=GlobalRequestProcessor/bytesSent	Bytes per second
JBoss HTTPS Data Throughput	The number of bytes sent per second.	jboss.web:name=http-${JBOSS_BIND_ADDRESS}-${HTTPS_PORT},type=GlobalRequestProcessor/bytesSent	Bytes per second
JBoss Active Sessions	The number of active sessions.	jboss.web:host=localhost,path=/,type=Manager/activeSessions	Sessions
JBoss Cache Size	The size of the cache.	jboss.web:host=localhost,path=/,type=Cache/cacheSize	MB
JBoss Cache Hit Frequency	The number of cache hits per second.	jboss.web:host=localhost,path=/,type=Cache/hitsCount	Hits per second
JBoss Free Memory	Free heap memory.	jboss.system:type=ServerInfo/FreeMemory	MB

Table 3 JBoss EAP Statistics (Continued)

Name	Description	MBean/Attribute	Units
JBoss Max Memory	Maximum heap memory.	jboss.system:type=ServerInfo/MaxMemory	MB
JBoss Active Thread Count	The number of active threads.	jboss.system:type=ServerInfo/ActiveThreadCount	Threads

The following statistics are reported at the archive level for finer grain reporting information that can be used for archive scaling.

Table 4 JBoss EAP Archive Scaling Statistics

Name	Description	MBean/Attribute	Units
Name: JBoss Archive Throughput (JBoss EAP 5.0 & JBoss EAP 5.1.1)	Request throughput per archive.	Attribute: "requestCount" MBean: jboss.web:j2eeType=WebModule,J2EEApplication=none,J2EEServer=none,name=//localhost/< context-root >	Requests per second
Name: Active Archive Sessions	JBoss Active Archive Sessions.	Attribute: "activeSessions" MBean: jboss.web:type=Manager,path=<context-path >,host=localhost	Session count
Name: JBoss Archive Throughput (JBoss EAP 6)	Request throughput per archive.	retrieved with jboss-dmr API Note that the calculated archive throughput only includes requests to servlets within the webapp defined by that archive, not requests for static content or JSPs.	Requests per second

Runtime Context Variables

The following is a comprehensive list of all runtime context variables used by the JBoss EAP Enabler.The first table contains JBoss EAP 5.0 and 5.1 runtime context variables, while the second contains variables for JBoss EAP 6:

Table 5 Runtime Context Variables

Variable	Type	Description
ARCHIVE_DEPLOYMENT_TIMEOUT	String	The amount of time in seconds to allow for each archive deployment to finish in standalone mode. Only valid if VERIFY_ARCHIVE_DEPLOYMENT_SUCCESS is true.
ARCHIVE_DETECTION_ENABLED	String	Archive detection will be enabled if this variable is set to true.
ARCHIVE_DETECTION_FREQUENCY	String	Archive detection frequency in seconds. Default frequency of 30 seconds will be used if a value less than 30 is specified.
ARCHIVE_DETECTION_IGNORED_ARCHIVES	String	Comma separated list of archives to be ignored by archive detection mechanism.
ARCHIVE_DETECTION_IGNORED_URLS	String	Comma separated list of context URLs to be ignored by archive detection mechanism.
AUTH_CONFIG_FILE	String	The location of the Auth Config file
BIND_ON_ALL_LOCAL_ADDRESSES	Environment	Set to true to have JBoss bind to all local IP addresses.
CAPTURE_EXCLUDES	String	Directories/files excluded from capture. Generally useful to exclude are log file directories, tmp directories, etc.
CAPTURE_INCLUDES	String	Directories that need to be captured. Generally these directories contain configuration, applications, etc.
CLIENT_KEY_STORE_FILE	String	Client key store file for connecting to the JBoss MBean server when TWO_WAY_SSL_ENABLED is true

Table 5 Runtime Context Variables (Continued)

Variable	Type	Description
CLIENT_KEY_STORE_PASSWORD	String	Client key store password for connecting to the JBoss MBean server when TWO_WAY_SSL_ENABLED is true
CLIENT_TRUST_STORE_FILE	String	Client trust store file used when connecting to the JBoss MBean server
CLIENT_TRUST_STORE_PASSWORD	String	Client trust store password used when connecting to the JBoss MBean server
DEFAULT_SSL_ALGORITHM	String	Default algorithm for the SSL/TLS Connector
EJB3_DEPLOYER_DefaultEjb3Connector_PORT	String	JBoss EJB3 Deployer port
HAJNDI_SERVICE_XML_RmiPort_PORT	String	Socket HA-JNDI service uses to receive RMI requests from client proxies
HAJNDI_SERVICE_XML_autoDiscovery_PORT	Environment	Port for multicast socket on which HA-JNDI listens for auto-discovery requests from clients
HAJNDI_SERVICE_XML_autoDiscovery_hostName	Environment	Hostname for multicast socket on which HA-JNDI listens for auto-discovery requests from clients
HAJNDI_SERVICE_XML_listen_PORT	String	The listening socket for the HA-JNDI service
HA_LEGACY_SERVICE_XML_jrmpha_PORT	String	Socket for high availability version of the legacy RMI/JRMP invoker
HA_LEGACY_SERVICE_XML_pooledha_PORT	String	Socket for high availability version of the legacy Pooled invoker
HSQLDB_DS_XML_hypersonic_socket_PORT (JBoss EAP 5.0 only)	String	TCP/IP socket for remote connection to Hypersonic database
HTTP_PORT	Environment	HTTP listen port
HTTP_TO_AJP_PORT_OFFSET	String	AJP listen port offset from HTTP_PORT
HTTP_TO_HTTPS_PORT_OFFSET	String	HTTPS listen port offset from HTTP_PORT
IGNORE_HOSTNAME_VERIFICATION	Environment	Ignore hostname verification when connecting to the JBoss MBean server
IIOP_SERVICE_XML_CorbaORB_PORT	String	IIOP socket for the Corba ORB

Table 5 Runtime Context Variables (Continued)

Variable	Type	Description
INITIAL_CONTEXT_FACTORY	String	Initial context factory to create the JMX connection; must support JNDI access over HTTP
INVOKER_JNDI_FACTORY	String	Invoker servlet for JNDI Factory
INVOKER_JNDI_FACTORY_SSL	String	Invoker servlet for SSL JNDI Factory
J2EE_ARCHIVE_DEPLOY_DIRECTORY	String	The JBoss archive deployment directory
JAVA_OPTS	Environment	Additional Java process options
JBOSSMQ_SERVICE_XML_InvocationLayer_PORT (JBoss EAP 5.0 only)	String	UIL2 socket for JBossMQ
JBOSS_HOME	Environment	The JBoss home directory
JBOSS_JGROUPS_UDP_BIND_PORT (JBoss EAP 5.1.1 only)	String	The port used for the JGroups 'udp' stack
JBOSS_SERVER_BASE_DIR	Environment	The JBoss server base directory
JBOSS_SERVER_BASE_URL	Environment	The JBoss server base URL
JBOSS_SERVER_CONFIG_NAME	Environment	The JBoss server configuration to use
JBOSS_SERVER_HOME_DIR	Environment	The JBoss server home directory
JBOSS_SERVER_HOME_URL	Environment	The JBoss server home URL
JBOSS_SERVICE_XML_Naming_RmiPort_PORT	String	Socket Naming service uses to receive RMI requests from client proxies
JBOSS_SERVICE_XML_Naming_jboss_bind_address_PORT	String	The listening socket for the Naming service
JBOSS_SERVICE_XML_WebService_PORT	String	Socket for dynamic class and resource loading
JBOSS_SERVICE_XML_remoting_PORT	String	Socket for JBoss Remoting Connector used by UnifiedInvoker
JDK_NAME	String	The name of the required JDK
JDK_VERSION	String	The version of the required JDK

Table 5 Runtime Context Variables (Continued)

Variable	Type	Description
JMX_REMOTING_SAR_rmi_PORT	String	RMI/JRMP socket for connecting to the JMX MBeanServer
LEGACY_INVOKERS_SERVICE_XML_jrmp_PORT	String	Socket for the legacy RMI/JRMP invoker
LEGACY_INVOKERS_SERVICE_XML_pooled_PORT	String	Socket for the legacy Pooled invoker
LISTEN_ADDRESS_NET_MASK	Environment	A comma delimited list of net masks in CIDR notation. The first IP address found that matches one of the net masks is used as the listen address.
MESSAGE_SUCKER_PASSWORD (JBoss EAP 5.0 only)	String	Password for the MessageSucker component
REMOTING_BISOCKET_SERVICE_XML_messaging_1x_PORT	String	Socket for JBoss Messaging 1.x
REMOTING_BISOCKET_SERVICE_XML_messaging_2x_netty-port	String	Socket for JBoss Messaging 2.x
REMOTING_BISOCKET_SERVICE_XML_messaging_2x_netty-ssl-port	String	SSL socket for JBoss Messaging 2.x
SECURITY_CREDENTIALS	String	Password for Web, JMX, and Web Service authentication
SECURITY_PRINCIPAL	String	User for Web, JMX, and Web Service authentication
SERVER_KEY_STORE_FILE	String	Server key store file location for incoming SSL connections
SERVER_KEY_STORE_PASSWORD	String	Password for the server key store
SERVER_PEER_ID	String	Server Peer Id, which must be a unqiue integer between 0 and 1023
SERVER_TRUST_STORE_FILE	String	Server trust store file location for outgoing SSL connections
SERVER_TRUST_STORE_PASSWORD	String	Password for the server trust store
SNMP_ADAPTOR_SAR_snmp_PORT	String	Socket for the SNMP adaptor MBean

Table 5 Runtime Context Variables (Continued)

Variable	Type	Description
SNMP_ADAPTOR_SAR_trapd_PORT	String	Socket for the SNMP trap receiver
TRANSACTION_JBOSS_BEANS_XML_recoveryManager_PORT	String	Socket for JBossTS Recovery Manager
TRANSACTION_JBOSS_BEANS_XML_socketProcessId_PORT	String	Socket used to provide unique process id for JBossTS.
TRANSACTION_JBOSS_BEANS_XML_transactionStatusManager_PORT	String	Socket for JBossTS Transaction Status Manager
TWO_WAY_SSL_ENABLED	Environment	Whether clientAuth is enabled in the SSL/TLS Connector
URL_PKG_PREFIXES	String	Colon-separated list of package prefixes to use when loading in URL context factories. Change only if you are an advanced user and you know what you are doing
VERIFY_ARCHIVE_DEPLOYMENT_SUCCESS	String	Whether or not to verify successful deployment of J2EE archives. This must be set to true for archive ordering.
XA_NODE_IDENTIFIER	String	XA Node Identifier, which must be a unique alphanumeric

The following variables are also available for JBoss EAP 6:

Table 6 JBoss EAP 6 Runtime Context Variables

Variable	Type	Description
ARCHIVE_DEPLOYMENT_TIMEOUT	String	The amount of time in seconds to allow for each archive deployment to finish in standalone mode. Only valid if VERIFY_ARCHIVE_DEPLOYMENT_SUCCESS is true.
ARCHIVE_DETECTION_ENABLED	String	Archive detection will be enabled if this variable is set to true.
ARCHIVE_DETECTION_FREQUENCY	String	Archive detection frequency in seconds. Default frequency of 30 seconds will be used if a value less than 30 is specified.

Table 6 JBoss EAP 6 Runtime Context Variables (Continued)

Variable	Type	Description
BIND_ON_ALL_LOCAL_ADDRESSES	Environment	Set to true to have JBoss bind to all available local IP addresses.
CAPTURE_EXCLUDES	String	Directories/files excluded from capture. Generally useful to exclude are log file directories, tmp directories, etc.
CAPTURE_INCLUDES	String	Directories that need to be captured. Generally these directories contain configuration, applications, etc.
HORNETQ_CLUSTER_PASSWORD	String	Cluster password for the HornetQ Server
HTTPS_MANAGEMENT_INTERFACE_PORT	Environment	HTTPS Management Interface port
HTTPS_PORT	Environment	HTTPS listen port
HTTP_MANAGEMENT_INTERFACE_PORT	Environment	HTTP Management Interface port
HTTP_PORT	Environment	HTTP listen port
JACORB_PORT	Environment	Jacorb port
JACORB_SSL_PORT	Environment	Jacorb SSL port
JAVA_HOME	Environment	The Java home directory
JBOSS_BIND_ADDRESS_MANAGEMENT	Environment	The bind address for the management interface. This can optionally be set to 0.0.0.0 to bind to all local addresses, or to ${JBOSS_BIND_ADDRESS} to use the same address as the public interface, as specified by ${LISTEN_ADDRESS_NET_MASK}. Note that doing so will allow the management interface to be accessible from other network locations, which may entail a security risk.
JBOSS_HOME	Environment	The JBoss home directory
JBOSS_NODE_NAME	Environment	Unique name for the JBoss node, used when clustering is enabled.
JDK_NAME	String	The name of the required JDK
JDK_VERSION	String	The version of the required JDK

Table 6 JBoss EAP 6 Runtime Context Variables (Continued)

Variable	Type	Description
LISTEN_ADDRESS_NET_MASK	Environment	A comma delimited list of net masks in CIDR notation. The first IP address found that matches one of the net masks is used as the listen address.
MESSAGING_PORT	Environment	Messaging port
MESSAGING_THROUGHPUT_PORT	Environment	Messaging throughput port
NATIVE_MANAGEMENT_INTERFACE_PORT	Environment	Native Management Interface port
OSGI_HTTP_PORT	Environment	OSGI HTTP port
SERVER_KEY_ALIAS	String	Private key entry's alias in the server key store, see JBoss bug JBPAPP6-789
REMOTING_PORT	Environment	Remoting port
SERVER_KEY_STORE_FILE	String	Server key store file location for incoming SSL connections
SERVER_TRUST_STORE_FILE	String	Server trust store file location for outgoing SSL connections
SSL_PASSWORD	String	Password for both the truststore and the keystore
START_CONDITION_POLL_PERIOD	String	How often in milliseconds to check if the server has started
TXN_RECOVERY_ENVIRONMENT_PORT	Environment	TXN Recovery Environment port
TXN_STATUS_MANAGER_PORT	Environment	TXN Status Manager port
VERIFY_ARCHIVE_DEPLOYMENT_SUCCESS	String	Whether or not to verify successful deployment of J2EE archives. This must be set to true for archive ordering.

Chapter 6 Distribution Grid-library.xml Files

This chapter provides `grid-library.xml` files for use in creating distribution Grid Libraries required for installation.

Topics

Creating Distribution Grid Libraries

Because the distribution Grid Library contains the full installation of the JBoss EAP software, it is not provided by TIBCO; you must create the distribution, using your own copy of JBoss EAP.

To create a distribution Grid Library:

1. Create a temporary directory named `jboss-eap`*`x.y.z`*`-distribution-gridlib`.
2. Download the full installation of the JBoss EAP software, and extract the ZIP file into the temporary directory.

   For JBoss EAP 6.4.1, download JBoss EAP 6.4.0 and apply the JBoss EAP 6.4.1 patch using the JBoss path tool:

   a. Run `bin/jboss-cli.sh` or `bin\jboss-cli.bat`.

   b. At the prompt, run the `patch` command:

   `patch apply` */path/to/downloaded-patch*`.zip`
3. The extracted ZIP download of the JBoss EAP software contains a single directory with the name `jboss-eap-`*`x.y.z`*. Rename this directory `jboss-eap`.
4. Create a `grid-library.xml` file using one of the examples from Sample Grid-library.xml Files on page 39.
5. Put the `grid-library.xml`into the top level of the temporary directory.

   The temporary directory will now contain the `grid-library.xml` and a `jboss-eap` directory.
6. Create an archive of the temporary directory. It can be a ZIP archive, or a gzipped TAR archive. Note that the archive should contain the the `grid-library.xml` and `jboss-eap` directory at the top level, and not the temporary directory itself.
7. Ensure the archive is named `jboss-eap`*`x.y.z`*`-distribution-gridlib.zip` or `.tgz`. This is the completed distribution Grid Library for installation on your Broker.

The process for creating a JDK Grid Library is similar to the above, except the naming of the Grid Library would be `j2sdk-`*platform*`-1_6_0_23-gridlib.zip`. You must also install JCE unlimited-strength policy files for your JRE; see the Oracle web site for details.

Sample Grid-library.xml Files

- The following example `grid-library.xml` files are shown below:
- Java SDK on page 39
- JBoss EAP 5.0.0, page 39
- JBoss EAP 5.1.1, page 39
- JBoss EAP 6.0.0, page 39
- JBoss EAP 6.4.1, page 40

Java SDK

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library jre="true" os="platform">
    <grid-library-name>j2sdk-platform</grid-library-name>
    <grid-library-version>1.6.0.patchnum</grid-library-version>
</grid-library>
```

JBoss EAP 5.0.0

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library>
    <grid-library-name>jboss-eap5_0_0-distribution</grid-library-name>
    <grid-library-version>5.0.0</grid-library-version>
    <jar-path>
        <pathelement>jboss/jboss-as/client</pathelement>
    </jar-path>
</grid-library>
```

JBoss EAP 5.1.1

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library>
    <grid-library-name>jboss-eap5_1_1-distribution</grid-library-name>
    <grid-library-version>5.1.1</grid-library-version>
    <jar-path>
        <pathelement>jboss/jboss-as/client</pathelement>
    </jar-path>
</grid-library>
```

JBoss EAP 6.0.0

```
<?xml version="1.0" encoding="UTF-8" ?>
<grid-library>
    <grid-library-name>jboss-eap6_0_0-distribution</grid-library-name>
```

```
    <grid-library-version>6.0.0</grid-library-version>
    <jar-path>
        <pathelement>jboss/modules/org/jboss/as/controller-client/main/jboss-as-con
troller-client-7.1.2.Final-redhat-1.jar</pathelement>
        <pathelement>jboss/modules/org/jboss/as/protocol/main/jboss-as-protocol-7.1
.2.Final-redhat-1.jar</pathelement>
        <pathelement>jboss/modules/org/jboss/dmr/main/jboss-dmr-1.1.1.Final-redhat-
1.jar</pathelement>
       <pathelement>jboss/modules/org/jboss/logging/main/jboss-logging-3.1.1.GA-red
hat-1.jar</pathelement>
        <pathelement>jboss/modules/org/jboss/logmanager/main/jboss-logmanager-1.3.1
.Final-redhat-1.jar</pathelement>
        <pathelement>jboss/modules/org/jboss/marshalling/main/jboss-marshalling-1.3
.14.GA-redhat-1.jar</pathelement>
        <pathelement>jboss/modules/org/jboss/remoting3/main/jboss-remoting-3.2.8.GA
-redhat-1.jar</pathelement>
        <pathelement>jboss/modules/org/jboss/sasl/main/jboss-sasl-1.0.1.Final-redha
t-1.jar</pathelement>
        <pathelement>jboss/modules/org/jboss/threads/main/jboss-threads-2.0.0.GA-re
dhat-1.jar</pathelement>
        <pathelement>jboss/modules/org/jboss/xnio/main/xnio-api-3.0.4.GA-redhat-1.j
ar</pathelement>
        <pathelement>jboss/modules/org/jboss/xnio/nio/main/xnio-nio-3.0.4.GA-redhat
-1.jar</pathelement>
        <pathelement>jboss/modules/org/codehaus/jackson/jackson-mapper-asl/main/jac
kson-mapper-asl-1.9.2-redhat-1.jar</pathelement>
    </jar-path>
</grid-library>
```

JBoss EAP 6.4.1

```
<?xml version="1.0" encoding="UTF-8" ?>
<grid-library>
   <grid-library-name>jboss-eap6_4_1-distribution</grid-library-name>
   <grid-library-version>6.4.1</grid-library-version>
   <jar-path>
      <pathelement>jboss/bin/client/jboss-cli-client.jar</pathelement>
   </jar-path>
</grid-library>
```