TIBCO Silver® Fabric WebLogic Enabler Guide

Software Release 5.6
March 2017

Two-Second Advantage®

TIBCO®

Important Information

Contents

TIBCO Documentation and Support Services

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, visit:

https://docs.tibco.com

Product-Specific Documentation

The following documents for this product can be found on the TIBCO Documentation site:

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

TIBCO products may include some or all of the following:

Software developed by Terence Parr.

Software developed by the Apache Software Foundation (http://www.apache.org/).

This product uses c3p0. c3p0 is distributed pursuant to the terms of the Lesser General Public License. The source code for c3p0 may be obtained from https://sourceforge.net/projects/c3p0/. For a period of time not to exceed three years from the Purchase Date, TIBCO also offers to provide Customer, upon written request of Customer, a copy of the source code of c3p0.

Software developed by MetaStuff, Ltd.

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support:

- 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.

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts. It is 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 the following web address:

https://www.tibcommunity.com

About the WebLogic Enabler

A Silver Fabric enabler runs an external application or application platform, such as a J2EE application server in a TIBCO Silver® Fabric software environment. The WebLogic enabler for Silver Fabric provides integration between Silver Fabric and Oracle WebLogic server. The enabler automatically provisions, orchestrates, controls and manages a WebLogic standalone or clustered environment. WebLogic server instances are distributed and run on Silver Fabric engines, either as standalone administration servers or as managed servers belonging to a cluster. The management of the WebLogic environment through Silver Fabric is dynamic, centralized, and largely automated. This centralized management adapts the structure of the WebLogic environment, and manages the life cycle process state of standalone or managed application servers, as specified in Silver Fabric by policies that are directed by business objectives associated with the target applications.

Architecture

The Silver Fabric engine manages a WebLogic standalone or managed server instance. The Silver Fabric broker and the Silver Fabric engine instance collaborate to perform the following actions:

- Automatically provision all WLS-related and Java JDK software to a machine with a Silver Fabric engine. All the software that is automatically provisioned is encapsulated within archive libraries (.zip files) and distributed to computers running Silver Fabric engines.
- In a clustered environment, automatically configure a managed server within the WebLogic domain of the Admin server.
- Automatically start and stop standalone or managed WebLogic application servers on a node, as required by policies specified within Silver Fabric.
- Optionally, start and manage an instance of a WebLogic Admin server to support a WebLogic clustered environment.
- Create partitions inside a domain of a standalone or managed WebLogic application server.
- Monitor the health of the WebLogic server and retrieve statistics.

Installing the WebLogic Enabler

The following procedure describes how to install the WebLogic enabler:

Prerequisites

- These instructions presume a strong familiarity with your particular version of WebLogic. If you are uncertain of how to achieve a particular task, consult your WebLogic documentation.
- This procedure assumes a Silver Fabric broker is running with at least one engine installed, and that you have the broker's host name, a user name, and password for the Silver Fabric Administration Tool. If this is not true, see the *Silver Fabric Installation Guide*, or contact your administrator.
- See the included Silver Fabric Readme for the latest prerequisites required for this enabler.
- If engines used to run WebLogic server instances are configured to use a shared grid libraries directory, that directory need to have a sibling directory named `temp_jars` and the engines in question needs to be able to write to that directory. For more about the use of a shared directory for grid libraries, see "Resource Synchronization" in the *Silver Fabric Installation Guide*.
- The engine's JDK version must be 1.8.0.

Procedure

1. Get the required grid Libraries for the installation.

   The WebLogic enabler consists of an enabler Runtime Grid Library and a distribution 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.

   Runtime grid libraries are downloaded from the TIBCO download site. Distribution grid libraries are created using the directions in Creating Distribution Grid Libraries.

   The following required grid libraries for each version of the WebLogic enabler are listed:

WebLogic Enabler Version	Required Grid Libraries
10.3.1	`TIB_silver_fabric_5.6.0_weblogic10_3_1_distribution-gridlib.zip` `TIB_silver_fabric_5.6.0_weblogic_10_3_1_enabler.tar.gz`
10.3.5	`TIB_silver_fabric_5.6.0_weblogic10_3_5_distribution-gridlib.zip` `TIB_silver_fabric_5.6.0_weblogic_10_3_5_enabler.tar.gz`

12.1.1	`TIB_silver_fabric_5.6.0_weblogic12_1_1_distribution-gridlib.zip`
	`TIB_silver_fabric_5.6.0_weblogic_12_1_1_enabler.tar.gz`
	`j2sdk-`*`platform`*`-1_7_0_`*`patchnum`*`-gridlib.zip (optional)`

WebLogic Enabler Version	Required Grid Libraries

12.1.3	`TIB_silver_fabric_5.6.0_weblogic12_1_3_distribution-gridlib.zip`
	`TIB_silver_fabric_5.6.0_weblogic_12_1_3_enabler.tar.gz`
	`j2sdk-`*`platform`*`-1_7_0_`*`patchnum`*`-gridlib.zip (optional)`

12.2.1	`TIB_silver_fabric_5.6.0_weblogic12_2_1_distribution-gridlib.zip`
	`TIB_silver_fabric_5.6.0_weblogic_12_2_1_enabler.tar.gz`
	`j2sdk-`*`platform`*`-1_8_0_`*`patchnum`*`-gridlib.zip (optional)`

2. Copy the desired WebLogic enabler version grid library files to the *SF_HOME*`/webapps/livecluster/deploy/resources/gridlib` directory in the following order:
   1. Distribution
   2. Enabler Runtime

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 is displayed in the list.

WebLogic Server version 12.2.1 onwards, the enabler must install the WebLogic Server in an empty or non existent directory, *ORACLE_HOME*, inside the engine instance's working directory. The default name of this folder is `oracle_home`. You can modify this value later, through a runtime context variable.

Configuring the WebLogic Enabler

The following topics provide instructions on configuring the WebLogic enabler.

Running WebLogic Applications in Standalone Mode

The following procedure describes how to run a WebLogic component in standalone mode.

In standalone mode, Silver Fabric engines run individual self-contained servers and do not require an Admin server. Instead, each instance of the component running on an engine runs as a WebLogic Admin server. On activation, the engine deploys archives contained in the component to the server.

To define the component in Silver Fabric:

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
2. Select **Create New J2EE Component** from the **Global Actions** list.
3. If prompted, select the WebLogic Enabler and the desired Enabler Version from the list.
4. Enter a component name in the **Name** field.
5. Click **Next**.
   The Configure Component Features screen is displayed.
6. Make sure that the **Clustered Managed Server Support** and **Clustered Admin Server Support** options are not selected.
7. If you want to create partitions inside a domain, select the **Domain Partitioning Support** option.
8. You can further customize component features as needed by selecting the feature and clicking **Edit**. When finished, click **Next**.
   The Configure Component Options screen is displayed.
9. Enter any desired options and click **Next** until the **Add/Remove Archive Files** screen is displayed.
10. Click **Add** and upload the J2EE archives that you want to deploy and run on the WebLogic server.
11. Configure the options on the remaining screens as needed. Additional options are also configurable through context variables.
12. If you selected **Domain Partitioning Support** option earlier then see the following steps:
    a) On the Add/override/edit Enabler and Component-specific Content Files screen, add `ds_WLSPartition.py` script from the enabler. This script will be a template. Refer to the oracle documentation for using the WebLogic Scripting Tool to create your own script.
    b) Edit and save the customized `ds_WLSPartition.py` script.
13. Click **Finish**.
14. Go to **Stacks > Components**.
15. Select **Publish Component** from the corresponding Actions list of the component you created.
16. Create a stack and add the component as described in the *Silver Fabric Cloud Administration Guide*. Activate the stack when desired.

Running Clustered WebLogic Applications Using an External Admin Server

The following procedure describes how to run a clustered WebLogic application from Silver Fabric, using an external Admin server.

Procedure

1. Set up the WebLogic Domain. Repeat this task for each WebLogic Domain.
   a) In WebLogic, create a new WebLogic Domain and configure it with a name and port for the Admin Server of this new Domain. It is a good practice to use a naming scheme when using several Domains.
   b) Create a custom cluster configuration using one of the following use cases.
      - Recommended Use Case - Do not configure any Managed Servers. Server entries are created by the enabler dynamically at run time.
      - Advanced Use Case - Add servers with the following naming convention: *`prefix_hostname-instance`*. Note that any defined ports dynamically change based on the `HTTP_PORT`, `HTTPS_PORT`, and `ADMIN_LISTEN_PORT` variables. Add new Managed Servers following the above naming scheme based on the hostname/instance of the engines you are using. The prefix is server by default, but you can customize it as a context variable.
   c) Create a new cluster. Enter unique multicast address and ports for different applications. It is a good practice to enter unique names for clusters.
      - In the Recommended Use Case, ensure that there is at least one server added to the cluster. You cannot use WebLogic for the deployment of modules to empty clusters.
      - In the Advanced Use Case, assign Servers to Clusters by selecting all of the defined Managed Servers and adding them to the cluster that you just created.
   d) Configure any optional items your application requires in the JDBC components and messaging areas.
   e) Configure a user name and the password for the Admin server. Note these credentials, as they are required to configure your Silver Fabric component.
   f) Ensure that your settings are set as desired.
   g) Create the WebLogic domain using a unique domain name.
2. Deploy Your Application
   a) Start your WebLogic Administration server and login to the console.
   b) Access the area where you deploy a new application or module.
   c) Select and upload your .EAR or .WAR/.JAR files.
   d) Select the uploaded file and click Target Application for .EAR files or Target Module for .WAR/.JAR files.
   e) When prompted for deployment targets, locate the cluster you defined previously and click the **All servers in the cluster** option.
   f) Confirm the settings, accept the default page settings, and deploy the application.
3. Define the Application in Silver Fabric:
   a) Go to **Stacks > Components**.
   b) Select **Create New J2EE Component** from the **Global Actions** list.
   c) If prompted, select the WebLogic enabler and the desired enabler version from the list.
   d) Enter a component name in the **Name** field.
   e) Click **Next**.
   The **Configure Component features** screen appears.
   f) Remove **Archive Management Support** from the **Add feature** list. It is not required because in this mode, archives are deployed from the WebLogic Admin server.

g) Select **Clustered Managed Server Support** from the Add feature list.
The Enter feature values screen appears.
h) Enter the URL or a comma-delimited list of possible URLs of the administration server in the **Admin URLs** field, and click OK.
i) Click **Next**.
The **Configure Component Options** screen appears.
j) Continue configuring the options as needed.
k) On the **Add/override/edit Enabler and Component-specific runtime context variables** screen, add the credentials you entered previously as Environment type context variables. The variables are: `WLS_USER` and `WLS_PW`. If desired, you can define the `WLS_PW` variable as an Encrypted type variable. Alternatively, you can edit `WLS_USER` and `WLS_PW` through the Enabler Wizard.
l) Click **Finish**.

Running Clustered WebLogic Applications Using a Silver Fabric-Controlled Admin Server

The following procedure describes how to run a WebLogic application in clustered mode using an Administration server running on a Silver Fabric engine.

In this mode, two Silver Fabric components are configured and run. One is the Administration component and is configured to run a single instance. The other component runs Managed servers that connect to the Administration server. Similarly to standalone mode, archives contained in the Administration server component are deployed during activation. However, in this mode, they deploy onto the cluster instead of the Administration server.

Procedure

1. Create a component for the Administration server:
   a) In the Silver Fabric Administration Tool, go to **Stacks > Components**.
   b) Select **Create New J2EE Component** from the **Global Actions** list.
   c) If prompted, select the WebLogic enabler and the desired Enabler Version from the list.
   d) Enter a component name in the **Name** field.
   e) Click **Next**.
   The **Configure Component features** screen is displayed. **HTTP Support** and **Archive Support** are added to the feature list by default.
   f) If you want to create partitions inside a domain, select the **Domain Partitioning Support** option.
   g) Select **Clustered Admin Server Support** from the **Add Feature** list to add it to the component.
   The **Enter Feature Values** screen is displayed.
   h) Enter the name of the WebLogic cluster and the associated Cluster Multicast Address and Port parameters. The cluster can be pre-defined, or created dynamically at run time. If desired, you can pre-define the cluster by customizing the enabler's `config.xml` file. If **Cluster Name** is left blank, the component name is used. If either of the **Cluster Multicast** settings are left blank, they are not automatically configured at run time and the WebLogic default or pre-configured values from `config.xml` are used.
   i) Click **OK**. You can further customize component features as needed by selecting the feature and clicking **Edit**.
   j) Click **Next**.
   The **Configure Component Options** screen is displayed. Make changes as needed.
   k) Click **Next** until the **Add/Remove Archive Files** screen is displayed.
   l) Click **Add** and upload the J2EE archives that you want to deploy and run on the WebLogic cluster.
   m) Configure the options on the remaining screens as needed. Additional options are also configurable through context variables.
   n) Click **Finish**.

2. Create a component for the Managed server
   a) Go to **Stacks > Components**.
   b) Select **Create New J2EE Component** from the **Global Actions** list.
   c) If prompted, select the WebLogic enabler and the desired enabler version from the list.
   d) Enter a component name in the **Name** field.
   e) Click **Next**.
   The **Configure Component features** screen is displayed.
   f) Remove **Archive Management Support** from the **Add feature** list. It is not required because in this mode, archives are deployed from the WebLogic Admin server.
   g) Select **Clustered Managed Server Support** from the **Add feature** list.
   The **Enter feature values** screen is displayed.
   h) Enter the URL or a comma-delimited list based on the engines on which the Administration server component can run. By default, this includes all hosts on which you are running Silver Fabric engines. To restrict the engines on which the Clustered Administration server component can run, add engine Filter rules as appropriate.
   i) Click **OK**.
   j) On the **Add/override/edit Enabler and Component-specific runtime context variables** screen, add the credentials as defined in your Clustered Admin server component. The variables are: `WLS_USER` and `WLS_PW`. The default value for the user name is `weblogic`; the default password is `welcome1`. If desired, you can define the `WLS_PW` variable as an Encrypted type variable. Alternatively, you can edit `WLS_USER` and `WLS_PW` through the Enabler Wizard.
   k) On the Add/override/edit Enabler and Component-specific Content Files screen, add `ds_WLSPartition.py` script from the enabler. This script will be a template. Refer to the oracle documentation for using the WebLogic Scripting Tool to create your own script.
   l) Click **Finish**.
3. Publish the components:
   a) Go to **Stacks > Components**.
   b) Select **Publish Component** from the corresponding Actions lists of the two components you created in Tasks 1 and 2.
4. Create a stack:
   a) Go to **Stacks > Stacks** in the Silver Fabric Administration Tool.
   b) Select **Create New Stack** from the **Global Actions** list.
   c) Enter a stack name in the **Name** field, and optionally a description.
   d) Select the **Clustered Admin Server Component** and the **Clustered Managed Server Component** from the **Available Components** list and add them to the stack by clicking the **>>** button.
   e) Under **Policies**, customize the minimum and maximum number of instances for each component. For the Admin server component, enter a minimum value of 1 and a maximum value of 1. A WebLogic cluster requires only one Administration server.
   f) To restrict the engines on which the Clustered Administration server component can run, add Resource Preference rules as appropriate (optional).
   g) Add a Component Dependency Rule to the Managed server component in the Policies section such that it depends on your Admin server component. This ensures the Admin server component is activated before the Managed server component.
   h) Click **Finish**.
   i) On the **Stacks > Stacks** page, select **Publish Stack** from the Actions list for the stack you just created.

Result

Your clustered WebLogic stack is now ready. To run the stack, you must activate it. You can do this manually by going to **Stacks > Stacks** and selecting **Run Stack in Manual Mode** from the new stack's Actions list, or automatically by selecting **Run Stack in Auto Mode** and relying on schedule-based

allocation. For more information on creating and using Schedules, see the *Silver Fabric Cloud Administration Guide*.

Running WebLogic Applications in Fault Tolerance Mode

Store the WebLogic Server files in a shared drive and use it as fault tolerance directory. This helps in preventing loss of data during a component restart.

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
2. Select **Create New J2EE Component** from the **Global Actions** list.
3. If prompted, select the WebLogic Enabler and the desired Enabler Version from the list.
4. Enter a component name in the **Name** field.
5. Click **Next**.
   The Configure Component Features screen is displayed. HTTP Support and Archive Support are added to the feature list by default.
6. Select the **FT Directory Support** option and the other features according to the type of server to configure.
7. Add the FT Directory path and click **OK**.
8. You can further customize component features as needed by selecting the feature and clicking **Edit**. When finished, click **Next**.
   The Configure Component Options screen is displayed.
9. Enter any desired options and click **Next** until the **Add/Remove Archive Files** screen is displayed.
10. Click **Add** and upload the J2EE archives that you want to deploy and run on the WebLogic server.
11. Configure the options on the remaining screens as needed. Additional options are also configurable through context variables.
12. Click **Finish**.
13. Go to **Stacks > Components**.
14. Select **Publish Component** from the corresponding Actions list of the component you created.
15. Create a stack and add the component as described in the *Silver Fabric Cloud Administration Guide*. Activate the stack when desired.

Removing Servers in Clustered Mode

In Clustered mode, if servers do not already exist on an Admin server, they are automatically added when the Clustered Managed server is activated on an engine.

To control the cleanup behavior of the engine when the Managed server component is deactivated, use the `SERVER_REMOVAL_POLICY` runtime context variable. The possible values for this variable are:

- `NO_ACTION` - No action is taken when a Managed server component instance is deactivated. The server remains configured on the Admin server indefinitely. This is the default.
- `REMOVE_FROM_CLUSTER` - On deactivation, the server remains on the Admin server, but its Cluster attribute is set to None. This mode is useful because in general, WebLogic server start up times are shorter when there are a smaller number of servers configured as members of a cluster, regardless of the number that are currently running.
- `DESTROY_SERVER` - On deactivation, the configured server is removed from the Admin server.

The SERVER_REMOVAL_POLICY only applies when the engine actually adds a server entry to the Admin server on startup. If the server already exists when the component is activated, it is not cleaned up when deactivated.

Overriding Server Names in Clustered Mode

By default, server names are automatically generated based on the host name and instance (*server_prefix_hostname-instance*). In some cases, you might want to instead provide a static server name for one or more servers.

For example, you might need to pre-configure a resource target that requires a static server name, such as for a JMS server. Because server names must be unique within a WebLogic Domain, to use a custom server name, you must configure a separate Managed server component for use with your main Domain and configure it through a policy to only run one instance.

To use a custom server name:

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
2. Select **Copy Component** from the Actions list adjacent to your Clustered Managed server component.
   You are prompted to name the copy.
3. Provide a name for the new component and click **OK**.
   The copy of the component is displayed in the list of components.
4. Select **Edit Component** from the Actions list adjacent to the copy of the component.
5. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables**.
6. Select the **String** variable from the **Add Variable** list.
   The **Add Variable** screen is displayed.
7. Complete the screen as follows:
   a) Enter CLUSTER_SERVER_NAME_OVERRIDE in the name field.
   b) Enter the desired static server name in the value field.
   c) Select **None** as the **Auto Increment Type**.
8. Click **OK** and then click **Finish**.
9. Select **Publish Component** from the corresponding Actions list adjacent to the new component you created.
10. Go to **Stacks > Stacks** and edit the Stack containing your existing Clustered Managed server component to add the newly created component. Use a minimum of 1 and maximum of 1 engine to avoid conflicts resulting from using the same static server name on multiple engines simultaneously.
11. If desired, you can use multiple static cluster server names simultaneously. To do so, follow the previous steps and create a separate component for each custom server name.

Configuring SSL

You can configure SSL on your component in a variety of ways. You can control whether the Admin server listens on HTTP and/or HTTPS, whether the Managed server listens on HTTP and/or HTTPS, and whether client certificates are requested and used.

The relevant connections for SSL are:

- Managed server to Admin server

- Silver Fabric engine to Managed server
- Silver Fabric engine to Admin server
- Clients (including VirtualRouter) to Managed server

To use SSL, you must import certificates from WebLogic into the JRE used by the engines. This involves upgrading engines to a JRE that you have packaged, that includes a Java cacerts file with the certificates imported. To do this:

1. Follow the "Updating Engine JREs" procedure from the *Silver Fabric Cloud Administration Guide*. Instead of obtaining a new JRE and modifying it, you can use the current JRE, after performing the following step.
2. Use `keytools` to import the WebLogic certificate into the `cacerts` file:

   ```
   keytool -import -alias alias -keystore cacerts_file\
          -trustcacerts -file certificate_filename
   ```

   where alias is an alias for the certificate, *`cacerts_file`* is*`jre_location`*`/jre/lib/security/cacerts` and *`certificate_filename`* is *`wlserver_dist`*`/server/lib/DemoIdentity.jks`.

Configuring 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). If the CA is trusted by the client in its local trust store, and the certificate is validated, or if the client is configured to accept the self-signed certificate, the connection is established.

Follow the applicable instructions in one of the following topics to enable SSL in your Silver Fabric component in either standalone or clustered mode.

Prerequisites

- You must import any relevant certificates into the engine's trust store file, `ssl.keystore`, located in the root `DSEngine` directory.
- For context variables that require a fully-qualified path name, use the following environment variables:

Variable	Description
`${DS_WEBLOGIC_BASE}`	Expands to the base directory of the WebLogic distribution on the engine.
`${WL_DOMAIN_DIR}`	Expands to the base directory of the component's runtime directory located in the engine's work directory.

- The engine `ssl.keystore` must import the cert from Weblogic. For example, in the case of demo cert, run the following command:

  ```
  keytool -export -keystore DSEngine/resources/fabric_gridlib/weblogic10_3_1-
  distribution-gridlib/wlserver_10.3/server/lib/DemoTrust.jks -file wl100.cer -
  alias certgenca
  keytool -import -file wl100.cer -alias certgena -keystore DSEngine/ssl.keystore -
  storepass changeit
  ```

Configuring One Way SSL in Standalone Mode

The following procedure is used to configure one way SSL in standalone mode.

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
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.

   

   If both are checked, the engine favors SSL to connect to the server instance.
6. Click **OK**.
7. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables** and configure the following Environment variables as necessary:

Variable	Description
`KEYSTORE_CONFIGURATION`	The configuration rules for finding the server's identity and trust keystores, either 'DemoIdentityAndDemoTrust' or 'CustomIdentityAndCustomTrust'.
`TRUST_KEY_STORE_TYPE`	Optional, defaults to jks.
`IGNORE_HOSTNAME_VERIFICATION`	Specifies if host names are verified. The default is false.
`CUSTOM_TRUST_KEY_STORE_FILE_NAME`	If using `CustomIdentityAndCustomTrust`, specifies the fully qualified path to the trust keystore.
`CUSTOM_TRUST_KEY_STORE_PASS_PHRASE`	If using `CustomIdentityAndCustomTrust`, specifies the password for the trust keystore.
`CUSTOM_IDENTITY_KEY_STORE_FILE_NAME`	If using `CustomIdentityAndCustomTrust`, specifies the fully qualified path to the identity keystore.
`CUSTOM_IDENTITY_KEY_STORE_TYPE`	If using `CustomIdentityAndCustomTrust`, specifies the type of the identity keystore.
`CUSTOM_IDENTITY_KEY_STORE_PASS_PHRASE`	If using `CustomIdentityAndCustomTrust`, specifies the password for the identity keystore.
`SERVER_PRIVATE_KEY_ALIAS`	Alias of the private key entry in the identity keystore.
`SERVER_PRIVATE_KEY_PASS_PHRASE`	Pass phrase of the private key entry.
`SSL_TIMEOUT`	The timeout, in milliseconds, for SSL connections. If required, set this to a higher value to compensate for the latency of your network.

8. Click **Finish**.
9. Select **Publish Component** from the corresponding Actions list adjacent to the component.

Configuring One Way SSL in Clustered Mode With an External Admin Server

The following procedure is used to configure one way SSL in clustered mode with an external Admin server.

Procedure

1. Start your WebLogic Administration server and log in to the console.
2. Access the General configuration page for the Admin server.
3. Enable the SSL Listen Port, and specify the port appropriately. Optionally, disable the HTTP listen port.
4. To use a custom identity or custom trust store, configure those on the appropriate page of the server configuration section.
5. Save your changes and restart the Administration server if necessary.
6. In the Silver Fabric Administration tool, go to **Stacks > Components**.
7. Select **Edit Component** from the Actions list adjacent to your component.
8. Select **Add/Edit Component Features**.
9. Select the **HTTP Support** feature and click **Edit**.
10. 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.
11. Click **OK**.
12. Select the **Clustered Managed Server Support** feature and click **Edit**.
13. Ensure that the Admin URLs field contains the HTTPS protocol and the port specified above.
14. Click **OK**.
15. Select **Publish Component** from the corresponding Actions list adjacent to the component.
16. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables** and configure the following Environment variables as necessary:

Variable	Description
`KEYSTORE_CONFIGURATION`	The configuration rules for finding the server's identity and trust keystores, either 'DemoIdentityAndDemoTrust' or 'CustomIdentityAndCustomTrust'.
`TRUST_KEY_STORE_TYPE`	Optional, defaults to jks.
`IGNORE_HOSTNAME_VERIFICATION`	Specifies if host names are verified. The default is false.
`CUSTOM_TRUST_KEY_STORE_FILE_NAME`	If using `CustomIdentityAndCustomTrust`, specifies the fully qualified path to the trust keystore.
`CUSTOM_TRUST_KEY_STORE_PASS_PHRASE`	If using `CustomIdentityAndCustomTrust`, specifies the password for the trust keystore.
`CUSTOM_IDENTITY_KEY_STORE_FILE_NAME`	If using `CustomIdentityAndCustomTrust`, specifies the fully qualified path to the identity keystore.

Variable	Description
CUSTOM_IDENTITY_KEY_STORE_TYPE	If using CustomIdentityAndCustomTrust, specifies the type of the identity keystore.
CUSTOM_IDENTITY_KEY_STORE_PASS_PHRASE	If using CustomIdentityAndCustomTrust, specifies the password for the identity keystore.
SERVER_PRIVATE_KEY_ALIAS	Alias of the private key entry in the identity keystore.
SERVER_PRIVATE_KEY_PASS_PHRASE	Pass phrase of the private key entry.
SSL_TIMEOUT	The timeout, in milliseconds, for SSL connections. If required, set this to a higher value to compensate for the latency of your network.

17. Click **Finish**.
18. Select **Publish Component** from the corresponding Actions list adjacent to the component.

Configuring One Way SSL in Clustered Mode With a Silver Fabric-Controlled Admin Server

The following procedure is used to configure one way SSL in clustered mode with a Silver Fabric-controlled Admin server.

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
2. Select **Edit Component** from the Actions list adjacent to your Clustered Admin server 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. Select **Publish Component** from the corresponding Actions list adjacent to the component.
8. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables** and configure the following Environment variables as necessary:

Variable	Description
KEYSTORE_CONFIGURATION	The configuration rules for finding the server's identity and trust keystores, either 'DemoIdentityAndDemoTrust' or 'CustomIdentityAndCustomTrust'.
TRUST_KEY_STORE_TYPE	Optional, defaults to jks.
IGNORE_HOSTNAME_VERIFICATION	Specifies if host names are verified. The default is false.
CUSTOM_TRUST_KEY_STORE_FILE_NAME	If using CustomIdentityAndCustomTrust, specifies the fully qualified path to the trust keystore.
CUSTOM_TRUST_KEY_STORE_PASS_PHRASE	If using CustomIdentityAndCustomTrust, specifies the password for the trust keystore.

Variable	Description
CUSTOM_IDENTITY_KEY_STORE_FILE_NAME	If using CustomIdentityAndCustomTrust, specifies the fully qualified path to the identity keystore.
CUSTOM_IDENTITY_KEY_STORE_TYPE	If using CustomIdentityAndCustomTrust, specifies the type of the identity keystore.
CUSTOM_IDENTITY_KEY_STORE_PASS_PHRASE	If using CustomIdentityAndCustomTrust, specifies the password for the identity keystore.
SERVER_PRIVATE_KEY_ALIAS	Alias of the private key entry in the identity keystore.
SERVER_PRIVATE_KEY_PASS_PHRASE	Pass phrase of the private key entry.
SSL_TIMEOUT	The timeout, in milliseconds, for SSL connections. If required, set this to a higher value to compensate for the latency of your network.

9. Click **Finish**.
10. Select **Publish Component** from the corresponding Actions list adjacent to the component.
11. Select **Edit Component** again from the Actions list adjacent to your Clustered Managed server component.
12. Select **Add/Edit Component Features**.
13. Select the **HTTP Support** feature and click **Edit**.
14. 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.
15. Click **OK**.
16. Select the **Clustered Managed Server Support** feature and click **Edit**.
17. Ensure that the **Admin URLs** field contains the HTTPS protocol and the ports specified by the HTTPS_PORT variable in the Clustered Admin server component.
18. Click **OK**.
19. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables** and configure the following Environment variables as necessary:

Variable	Description
KEYSTORE_CONFIGURATION	The configuration rules for finding the server's identity and trust keystores, either 'DemoIdentityAndDemoTrust' or 'CustomIdentityAndCustomTrust'.
TRUST_KEY_STORE_TYPE	Optional, defaults to jks.
IGNORE_HOSTNAME_VERIFICATION	Specifies if host names are verified. The default is false.
CUSTOM_TRUST_KEY_STORE_FILE_NAME	If using CustomIdentityAndCustomTrust, specifies the fully qualified path to the trust keystore.
CUSTOM_TRUST_KEY_STORE_PASS_PHRASE	If using CustomIdentityAndCustomTrust, specifies the password for the trust keystore.

Variable	Description
CUSTOM_IDENTITY_KEY_STORE_FILE_NAME	If using CustomIdentityAndCustomTrust, specifies the fully qualified path to the identity keystore.
CUSTOM_IDENTITY_KEY_STORE_TYPE	If using CustomIdentityAndCustomTrust, specifies the type of the identity keystore.
CUSTOM_IDENTITY_KEY_STORE_PASS_PHRASE	If using CustomIdentityAndCustomTrust, specifies the password for the identity keystore.
SERVER_PRIVATE_KEY_ALIAS	Alias of the private key entry in the identity keystore.
SERVER_PRIVATE_KEY_PASS_PHRASE	Pass phrase of the private key entry.
SSL_TIMEOUT	The timeout, in milliseconds, for SSL connections. If required, set this to a higher value to compensate for the latency of your network.

20. Click **Finish**.
21. Select **Publish Component** from the corresponding Actions list adjacent to the component.
22. The CUSTOM_TRUST_KEY_STORE_PASS_PHRASE and CUSTOM_IDENTITY_KEY_STORE_PASS_PHRASE variables must be entered in encrypted form when PRODUCTION_MODE is set to true. When PRODUCTION_MODE is false, they might either be encrypted or plaintext.

Configuring Two Way SSL

In two way SSL, the WebLogic 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.

Configuring Two Way SSL in Standalone Mode

This procedure configures Two Way SSL in clustered mode in standalone mode.

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
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 is displayed.
5. Complete the screen as follows:
   a) Enter TWO_WAY_SSL_ENABLED in the **name** field.
   b) Enter true in the **value** field.
   c) Select **None** as the **Auto Increment Type**.
6. Configure the following Environment variables as necessary:

Variable	Description
IDENTITY_CERT_DIR_NAME	The directory where the client certs are stored.
IDENTITY_CERTIFICATE_CHAIN	A comma-delimited list of identity certificate filenames in PEM format.

Variable	Description
SSL_CLIENT_KEY_PASSWORD	The password for the client's SSL key.
SSL_TIMEOUT	The timeout, in milliseconds, for SSL connections. If required, set this to a higher value to compensate for the latency of your network.

7. Click **OK** when finished.
8. You must now upload your client certificates and keys. Select **Add/Override/Customize Enabler and Component-Specific Content Files**.
9. Click **Upload**.
   The **Add File** screen is displayed.
10. Complete the screen as follows:
    a) Enter a name for your file in the **Name** field.
    b) Enter the value you used for IDENTITY_CERT_DIR_NAME in the **Relative Path** field.
    c) Enter your client certificate in the **File** field.
11. Repeat the previous two steps for your client key and any additional client certificates.
12. Click **Finish**.
13. Select **Publish Component** from the corresponding Actions list adjacent to the component.

Configuring Two Way SSL in Clustered Mode With an External Admin Server

This procedure configures Two Way SSL in clustered mode with an External Admin server.

Procedure

1. Start your WebLogic Administration server and log in to the console.
2. Access the SSL configuration page of the Admin server.
3. Click **Advanced** to view additional configuration options.
4. Set the following options as needed:
   - Set Two Way Client Cert Behavior to either **Client Certs Requested But Not Enforced** or **Client Certs Requested and Enforced**.
   - Optionally set **Hostname Verification** to None.
5. Save your changes and restart the Administration server if necessary.
6. Follow the instructions for Configuring Two Way SSL in Standalone Mode to configure your Clustered Managed server component.

Configuring Two Way SSL in Clustered Mode With a Silver Fabric-Controlled Admin Server

This procedure configures Two Way SSL in clustered mode with a Silver Fabric-controlled Admin server.

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
2. Select **Edit Component** from the Actions list adjacent to your Clustered Admin server 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 is displayed.
5. Complete the screen as follows:
   a) Enter `TWO_WAY_SSL_ENABLED` in the **name** field.
   b) Enter `true` in the **value** field.
   c) Select **None** as the **Auto Increment Type**.
6. Click **OK**.
7. Click **Finish**.
8. Select **Publish Component** from the corresponding Actions list adjacent to the component.
9. Select **Edit Component** from the Actions list adjacent to your Clustered Managed server component.
10. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables**.
11. Configure the following Environment variables as necessary:

Variable	Description
`IDENTITY_CERT_DIR_NAME`	The directory where the client certs are stored.
`IDENTITY_CERTIFICATE_CHAIN`	A comma-delimited list of identity certificate file names in PEM format.
`SSL_CLIENT_KEY_PASSWORD`	The password for the client's SSL key.
`SSL_TIMEOUT`	The timeout, in milliseconds, for SSL connections. If required, set this to a higher value to compensate for the latency of your network.

12. Click **OK** when finished.
13. You must now upload your client certificates and keys. Select **Add/Override/Customize Enabler and Component-Specific Content Files**.
14. Click **Upload**.
    The **Add File** screen is displayed.
15. Complete the screen as follows:
    a) Enter a name for your file in the **Name** field.
    b) Enter the value you used for `IDENTITY_CERT_DIR_NAME` in the **Relative Path** field.
    c) Enter your client certificate in the **File** field.
16. Repeat the previous two steps for your client key and any additional client certificates.
17. Click **Finish**.
18. Select **Publish Component** from the corresponding Actions list adjacent to the component.

Enabling the Administration Port

You can use WebLogic server to configure and use a special port for administration traffic, separate from application traffic. To use an Administration Port with Silver Fabric components, first ensure that SSL is configured properly in your components, then follow the instructions that apply to the mode you are using.

In the enabler, the default value of the `STANDALONE_ADMIN_PORT_ENABLED` variable is false and the default value of the `ADMIN_LISTEN_PORT` variable is 9000.

As an alternative to adding these variables to the component, you can use the Enabler Wizard to edit the enabler and set these values to suit your needs.

Enabling the Administration Port in Standalone Mode

You can enable the administration port in standalone mode by using the following procedure:

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
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 is displayed.
5. Complete the screen as follows:
   a) Enter `STANDALONE_ADMIN_PORT_ENABLED` in the **name** field.
   b) Enter true in the **value** field.
   c) Select None as the Auto Increment Type.
   d) Click OK.
6. To specify the ports used for the Administration Port, add another Environment type variable with the following settings:
   a) Enter `ADMIN_LISTEN_PORT` in the **name** field.
   b) Enter the base port to use for the admin listen port in the **value** field. Select Numeric as the Auto Increment Type.
7. Click **OK** and then click **Finish**.
8. Select **Publish Component** from the corresponding Actions list adjacent to the component.

Enabling the Administration Port in Clustered Mode With an External Admin Server

You can enable the administration port in clustered mode with an external Admin server by using the following procedure:

Procedure

1. Start your WebLogic Administration server and log in to the console.
2. Access the base configuration page for your Domain.
3. Select the **Enable the Administration Port** option, set the port to the desired value and save your changes. This might require a restart of the Administration server.
4. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
5. Select **Edit Component** from the Actions list adjacent to your component.
6. Select **Add/Edit Component Features**.
7. Select the **Clustered Managed Server Support** feature and click **Edit**.
8. Ensure that the **Admin URLs** field contains the HTTPS protocol and the port specified on the WebLogic Administration server.
9. Click **OK**.
10. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables**.
11. To specify the ports used for the Administration Port on the Managed servers, add another Environment type variable with the following settings:
    a) Enter `ADMIN_LISTEN_PORT` in the **name** field.
    b) Enter the base port to use for the admin listen port in the **value** field.

c) Select **Numeric** as the Auto Increment Type. The actual port used by engines is the base port plus the instance number of the engine.

The `STANDALONE_ADMIN_PORT_ENABLED` variable is not used in this mode.

12. Click **OK** and then click **Finish**.
13. Select **Publish Component** from the corresponding Actions list adjacent to the component.

Enabling the Administration Port in Clustered Mode With a Silver Fabric-controlled Admin Server

You can use the following procedure to enable the administration port in clustered mode with a Silver Fabric-controlled Admin server.

Procedure

1. In the Silver Fabric Administration Tool, go to **Stacks > Components**.
2. Select **Edit Component** from the Actions list adjacent to your Clustered Admin server component.
3. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables**.
4. Add an Environment type variable with the following settings:
   a) Enter `STANDALONE_ADMIN_PORT_ENABLED` in the **name** field.
   b) Enter true in the **value** field.
   c) Select None as the Auto Increment Type.
5. To specify the ports used for the Administration Port on the Admin server, add another Environment type variable with the following settings:
   a) Enter `ADMIN_LISTEN_PORT` in the **name** field.
   b) Enter the port to use for the admin listen port in the **value** field.
   c) Select **None** as the **Auto Increment Type**.
   d) Click **OK** and then click **Finish**.
6. Select **Edit Component** from the Actions list adjacent to your Clustered Managed server component.
7. Select **Add/Edit Component Features**.
8. Select the **Clustered Managed Server Support** feature and click **Edit**.
9. Ensure that the **Admin URLs** field contains the https protocol and the port specified in step 5.
10. Click **OK**.
11. To specify the ports used for the Administration Port on the Managed servers, add another Environment type variable with the following settings:
    - Enter `ADMIN_LISTEN_PORT` in the **name** field.
    - Enter the base port to use for the admin listen port in the **value** field.
    - Select Numeric as the Auto Increment Type. The actual port used by Engines is the base port plus the instance number of the Engine.
12. Click **OK** and then click **Finish**.
13. Select **Publish Component** from the corresponding Actions list adjacent to the Component.

Configuring the WebLogic Load Balancing Plug-in

The WebLogic Load Balancing Plug-in can be used with Silver Fabric.

Procedure

1. Download your version-specific WebLogic documentation for installation and setup from the Oracle Technology Network.
2. In the configuration file for the application server on which you are running the plug-in, create mappings from the URL patterns used by your Silver Fabric components to the locations of your Virtual Router servers.
   This is an example Apache `httpd.conf`:

```
<LocationMatch /FooService*> <!-- Change this line as appropriate to match the
relative URL you want -->
        SetHandler weblogic-handler
        ConnectRetrySecs 1
        ConnectTimeoutSecs 60
        WebLogicCluster example60:8000,example61:8000 <!-- Configure this line
to be the addresses of the primary, failover Brokers, external VirtualRouter -->
</LocationMatch>
```

   The two important parameters are `WebLogicCluster` (which is a parameter common to all plug-ins) and the URL pattern match (`LocationMatch` for Apache, `ppath` for Netscape Enterprise Server Plug-In).
3. When configuring your component to use SSL, the WebLogic plug-in must establish trust with both VirtualRouter and WebLogic instance servers so that it can communicate with both of them. Update the trust store file accordingly for the plug-in.

Archive Management

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

- **Archive Detection:** The WebLogic 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. Note that the **`ARCHIVE_DETECTION_ENABLED`** and **`ARCHIVE_DETECTION_FREQUENCY`** runtime context variables can be modified, respectively, to control whether or not polling of archives is performed, and how often this detection happens.
- **Context URL Detection:** The WebLogic 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:** It is a method of dynamically adding or removing archives from 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 WebLogic 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. Note that for the WebLogic Enabler, archive operations can be targeted to either an admin server component instance or a managed server component instance. In the former case, deploy/start/stop/undeploy operations are done at the cluster level and operate on the cluster configured in the Clustered Admin Server Support feature. In the latter case, the operate directly on the server being run by the managed server component instance.

To use Archive scaling in your Component, you must define Scaling rules to specify when archives are added or removed from Component instances. The WebLogic Enabler has three Archive-level statistics that can be used to create rules. See Weblogic Statistics for a listing of statistics

Verifying Your Component Configuration

You can verify your component configuration.

Procedure

1. Go to **Engines > Engines** in the Silver Fabric Administration Tool and ensure that at least one instance of your 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 displayed.
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 applications). If you cannot access your Web application, verify your component configuration. See Configuring the JBoss EAP Enabler.
    - For testing purposes, you can also directly access applications running on Silver Fabric engines. The listen port of the server is generally the value of the relevant run time context variable for the component (`HTTP_PORT` or `HTTPS_PORT`) plus the engine instance number. For example, `HTTP_PORT` is by default 9090, so a server running on an engine `mymachine-2` is listening on port 9092.

Upgrading a Component

Weblogic Server version 12.2.1 onwards, you can upgrade the components from previous version of WebLogic Server. Change an older enabler to the latest release to take advantage of many new features even if the component is already running on an engine.

Prerequisites

Back up the Oracle installation in a shared drive before upgrade. The folders are listed below:

- coherence
- common
- oracle_common
- wlserver
- logs
- oui
- user_projects
- inventory

Procedure

1. Using the TIBCO Silver Fabric Administrator Components page identify the component you wish to upgrade. Click the Component Actions menu icon, choose **Change Enabler** option, select `WebLogic Server 12.2.1` and click **OK**.
2. Edit the component and go to the Configure Component Features screen. Select the **Domain Upgrade Support** option.
3. Add the backup path and click **OK**.
4. If you want to use the fault tolerance mode,
   - Select the **FT Directory Support** option.
   - Add the FT Directory path and click **OK**.
5. Click **Finish**.
6. Go to **Stacks > Components**.
7. Select **Publish Component** from the corresponding Actions list of the component you edited.
8. Switch to the Engines page and click the Engine Actions menu icon that needs a component restart, and then click **Restart Component**.

Load Balancing

Load balancing is recommended for proxying requests from clients and routing them to Engines running the appropriate Component. Load balancing is achieved 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 WebLogic servers by examining the standard `JSESSIONID` cookie. For more information about using VirtualRouter, see the Silver Fabric Cloud Administration Guide.

WebLogic Plug-In

You can use Oracle's WebLogic load balancing plug-in (described in Configuring the WebLogic Load Balancing Plug-in) to provide access to the WebLogic servers running within Silver Fabric. The WebLogic plug-in initially routes requests through VirtualRouter, and thereafter forwards them directly to the WebLogic server of the cluster.

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 Cloud Administration Guide* for more information on configuring and using these methods.

WebLogic Statistics

The following are the default statistics supported by WebLogic Enablers. The Enablers use JMX to retrieve statistic values from MBean attributes on the WebLogic 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.

Weblogic Statistics

Name	Description	MBean/Attribute	Units
WebLogic Queue Size	The number of waiting requests in the queue.	ThreadPoolRuntime/ PendingUserRequestCount	Messages
WebLogic Throughput	The number of requests that have been processed by this queue.	ThreadPoolRuntime/ CompletedRequestCount	Messages per second
WebLogic Active Thread Count	The number of active queue threads.	ThreadPoolRuntime/ ExecuteThreadTotalCount, ThreadPoolRuntime/ ExecuteThreadIdleCount (subtract operation)	Threads
WebLogic Free Heap Size	Free heap memory.	JVMRuntime/ HeapFreeCurrent	MB
WebLogic Current Heap Size	Maximum heap memory.	JVMRuntime/ HeapSizeCurrent	MB

WebLogic Archive Statistics

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

Name	Description	MBean/Attribute	Units
WebLogic Archive Throughput	Request throughput per archive.	**MBean** `com.bea:ServerRuntime=<server_name>,Name=default,Type=WorkManagerRuntime,ApplicationRuntime=<module_name>` **Attribute** CompletedRequests	Requests per second

Name	Description	MBean/Attribute	Units
WebLogic Archive Pending request Count	Number of pending requests per archive	**MBean** `com.bea:ServerRuntime=<server_name>,Name=default,Type=WorkManagerRuntime,ApplicationRuntime=<module_name>` **Attribute** PendingRequests	Requests
WebLogic Archive Stuck Thread Count	Number of stuck threads per archive.	**MBean** `com.bea:ServerRuntime=<server_name>,Name=default,Type=WorkManagerRuntime,ApplicationRuntime=<module_name>` **Attribute** StuckThreadCount	Number of stuck threads

WebLogic Runtime Context Variables

The following table is a comprehensive list of all runtime context variables used by the WebLogic enabler.

Variable	Type	Description
ADMIN_LISTEN_PORT	String	Admin listen port
ADMIN_RETRY_INTERVAL	String	The amount of time in seconds to wait between attempting to contact an admin server.
ADMIN_RETRY_MAX	String	The maximum number of retries to attempt to contact an admin server.
ARCHIVE_DETECTION_ENABLED	String	Archive detection is enabled only if this variable is defined and it's value is set to true.
ARCHIVE_DETECTION_FREQUENCY	String	Frequency in seconds at which archive detection is triggered. This variable is used only when ARCHIVE_DETECTION_ENABLED is true. Minimum allowed value is 30, if the value specified is lower than the minimum allowed, it resets to the minimum (30 seconds).
CAPTURE_EXCLUDES	String	Directories/files excluded from capture. Generally useful to exclude are log file directories, tmp directories, and so on.
CAPTURE_INCLUDES	String	Directories that must be captured. Generally these directories contain configuration, applications, and so on.
CLUSTER_SERVER_NAME_PREFIX	String	Prefix used to generate server names in clustered mode
CUSTOM_IDENTITY_KEY_STORE_FILE_NAME	Environment	Specifies the fully qualified path to the identity keystore
CUSTOM_IDENTITY_KEY_STORE_PASS_PHRASE	Environment	Pass phrase for the identity keystore
CUSTOM_IDENTITY_KEY_STORE_TYPE	Environment	Specifies the type of the identity keystore
CUSTOM_TRUST_KEY_STORE_FILE_NAME	Environment	Specifies the fully qualified path to the trust keystore
CUSTOM_TRUST_KEY_STORE_PASS_PHRASE	Environment	Pass phrase for the custom trust keystore

Variable	Type	Description
DS_WEBLOGIC_BASE	Environment	BEA Home or the location of WebLogic distribution
HTTPS_PORT	Environment	HTTPS listen port
HTTP_PORT	Environment	HTTP listen port
IDENTITY_CERTIFICATE_CHAIN	String	Comma delimited list of identity certs for clients (only if two way SSL is enabled)
IDENTITY_CERT_DIR_NAME	String	Directory containing identity certs for clients (only if two way SSL is enabled)
IGNORE_HOSTNAME_VERIFICATION	Environment	Specifies if host names are verified
J2EE_ARCHIVE_DEPLOY_DIRECTORY	String	WebLogic archive deployment directory
JAVA_HOME	Environment	Java Home
JAVA_OPTIONS	Environment	Additional Java process options
JAVA_PROTOCOL_HANDLER_PKGS	String	A list of package names containing protocol handler classes
JDK_NAME	String	The name of the required JDK
JDK_VERSION	String	The version of the required JDK
KEYSTORE_CONFIGURATION	Environment	The configuration rules for finding the server's identity and trust keystores, either 'DemoIdentityAndDemoTrust' or 'CustomIdentityAndCustomTrust'
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.
NATIVE_IO_ENABLED	Environment	Whether or not to enable the WebLogic native I/O performance pack.
ORACLE_HOME	String	This value is set in the response file needed for the silent installation of the Weblogic Server.

Variable	Type	Description
PRODUCTION_MODE	Environment	If the server should run in production mode or development mode
SERVER_PRIVATE_KEY_ALIAS	Environment	Alias of the private key entry in the identity keystore
SERVER_PRIVATE_KEY_PASS_PHRASE	Environment	Pass phrase of the private key entry
SERVER_REMOVAL_POLICY	String	Determines the cleanup behavior of a clustered managed server dynamically added to an Admin Server. Valid options are NO_ACTION, REMOVE_FROM_CLUSTER, and DESTROY_SERVER.
SSL_CLIENT_KEY_PASSWORD	String	Password for client key (only if two way SSL is enabled)
SSL_DEBUG	String	Whether or not to enable SSL Debug tracing
SSL_TIMEOUT	String	Specifies the number of milliseconds that the WebLogic Server waits for an SSL connection before timing out. SSL connections take longer to negotiate than regular connections.
STANDALONE_ADMIN_PORT_ENABLED	String	Whether or not to enable the Administration port
STANDALONE_SERVER_NAME	String	Server name for stand-alone Weblogic application servers
TRUST_KEY_STORE_TYPE	String	Specifies the type of the trust keystore
TWO_WAY_SSL_ENABLED	Environment	Specifies if client certs are requested and enforced
USE_PRECONFIGURED_LISTEN_ADDRESS	String	If true, use the managed server listen address that is preconfigured on the admin server rather than the engine's listen address.
WLS_PW	Environment	Password for authentication
WLS_USER	Environment	Username for authentication
WL_DOMAIN	Environment	Name of the WebLogic domain

Variable	Type	Description
WL_DOMAIN_DIR	Environment	WebLogic domain directory
WL_HOME	Environment	Base directory of the WebLogic application server installation

About WebLogic Distribution Grid-library.xml Files

The following topics provide `grid-library.xml` files for use in creating distribution grid libraries required for installation, and how to create the distribution grid libraries.

Creating Distribution Grid Libraries

Because the distribution grid library contains the full installation of the WebLogic software, it is not provided by TIBCO; you must create the distribution, using your own copy of WebLogic.
To create a distribution grid ibrary:

Procedure

1. Create a temporary directory named `weblogic-x.y.z-distribution`.
2. Download the Generic installer for WebLogic Server and extract the jar installer into the temporary directory.
3. Create a `grid-library.xml` file using the example from Example WebLogic Distribution Grid-library.xml Files.
4. Put the `grid-library.xml` file at the top level of the temporary directory.
5. Create the `wlfullclient.jar`.
   a) On the command prompt, navigate to `server/lib` directory.
   ```
   cd ORACLE_HOME/server/lib
   ```
   b) Use the following command to create `wlfullclient.jar` in the server/lib directory:
   ```
   java -jar wljarbuilder.jar
   ```
   c) Copy the `wlfullclient.jar` into the temporary directory.
6. Create an archive of the temporary directory. It can be a .zip file archive, or a gzipped TAR archive.
7. Ensure the archive is named `weblogic-x.y.z-distribution.zip` or `.tar.gz`. This is the completed distribution grid library for installation on your broker.

Creating JDK Grid Libraries

To create a JDK grid library:

Procedure

1. Create a temporary directory named `jdk-platform-1.8.0.patchnumgridlib`.
2. Download the JDK 1.8.0 installer for required platform and extract the JDK files in the `j2sdk` folder created in the temporary directory.
3. Create a `grid-library.xml` file using one of the examples from Example WebLogic Distribution Grid-library.xml Files.
4. Put the `grid-library.xml` file at the top level of the temporary directory.
5. Create an archive of the temporary directory, named as `jdk-platform-1.8.0.patchnum-gridlib.tar.gz`.

What to do next

You must also install JCE unlimited-strength policy files for your JRE; see the Oracle website for details.

Example WebLogic Distribution Grid-library.xml Files

See the following example of `grid-library.xml` files for creating an enabler distribution.

For 64-bit versions, change the `OS` attribute of the grid-library element to indicate the correct OS. For example, for 64-bit Windows, change `<grid-library os="win32">` to `<grid-library os="win64">`.

WebLogic 10.3.1

Linux

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library os="linux">
    <grid-library-name>weblogic10_3_1-distribution</grid-library-name>
    <grid-library-version>10.3.1</grid-library-version>
    <!-- No JRE arguments -->
    <!-- No dependencies -->
    <!-- No conflicts -->
    <jar-path>
        <pathelement>wlserver_10.3/server/lib/weblogic.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/api.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wls-api.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wlclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wljmxclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wljmsclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wlcipher.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/webserviceclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/webserviceclient+ssl.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoCore.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoAsn1.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoJcae.jar</pathelement>
        <pathelement>modules/com.bea.core.transaction_2.5.1.0.jar</pathelement>
        <pathelement>modules/
com.bea.core.weblogic.security.wls_1.0.0.0_5-1-0-0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security_1.0.0.0_5-1-0-0.jar</
pathelement>
        <pathelement>modules/com.bea.core.management.core_2.4.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor_1.6.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.workmanager_1.6.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.store_1.4.1.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.rmi.client_1.5.0.0.jar</
pathelement>
        <pathelement>modules/javax.enterprise.deploy_1.2.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.identity_1.1.1.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.utils.full_1.6.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.utils.classloaders_1.5.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.logging_1.5.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.socket.api_1.0.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.digest_1.0.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.weblogic.lifecycle_1.2.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.utils.wrapper_1.4.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.management.jmx_1.2.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor.wl_1.2.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.redef_1.2.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.http.pubsub_1.3.0.0.jar</pathelement>
    </jar-path>
    <!-- No library paths -->
    <!-- No assembly paths -->
    <!-- No command paths -->
    <!-- No hooks paths -->
    <!-- No environment variables -->
    <!-- No system properties -->
</grid-library>
```

Windows

```
<?xml version="1.0" encoding="UTF-8"?>

<grid-library os="win32">
    <grid-library-name>weblogic10_3_1-distribution</grid-library-name>
    <grid-library-version>10.3.1</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->

    <!-- No conflicts -->

    <jar-path>
        <pathelement>wlserver_10.3/server/lib/weblogic.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/api.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wls-api.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wlclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wljmxclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wljmsclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wlcipher.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/webserviceclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/webserviceclient+ssl.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoCore.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoAsn1.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoJcae.jar</pathelement>
        <pathelement>modules/com.bea.core.transaction_2.5.1.0.jar</pathelement>
        <pathelement>modules/
com.bea.core.weblogic.security.wls_1.0.0.0_5-1-0-0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security_1.0.0.0_5-1-0-0.jar</
pathelement>
        <pathelement>modules/com.bea.core.management.core_2.4.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor_1.6.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.workmanager_1.6.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.store_1.4.1.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.rmi.client_1.5.0.0.jar</
pathelement>
        <pathelement>modules/javax.enterprise.deploy_1.2.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.identity_1.1.1.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.utils.full_1.6.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.utils.classloaders_1.5.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.logging_1.5.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.socket.api_1.0.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.digest_1.0.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.weblogic.lifecycle_1.2.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.utils.wrapper_1.4.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.management.jmx_1.2.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor.wl_1.2.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.redef_1.2.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.http.pubsub_1.3.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.logger_1.2.0.0.jar</
pathelement>
    </jar-path>

    <!-- No library paths -->

    <!-- No assembly paths -->

    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->
```

```
</grid-library>
```

WebLogic 10.3.5

Linux

```
<?xml version="1.0" encoding="UTF-8"?>

<grid-library os="linux">
    <grid-library-name>weblogic10_3_5-distribution</grid-library-name>
    <grid-library-version>10.3.5</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->

    <!-- No conflicts -->

    <jar-path>
        <pathelement>wlserver_10.3/server/lib/weblogic.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/api.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wls-api.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wlclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wljmxclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wljmsclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wlcipher.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/webserviceclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/webserviceclient+ssl.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoCore.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoAsn1.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoJcae.jar</pathelement>
        <pathelement>modules/com.bea.core.transaction_2.7.0.0.jar</pathelement>
        <pathelement>modules/
com.bea.core.weblogic.security.wls_1.0.0.0_6-1-0-0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security_1.0.0.0_6-1-0-0.jar</
pathelement>
        <pathelement>modules/com.bea.core.management.core_2.8.0.1.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor_1.9.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.workmanager_1.10.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.store_1.7.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.rmi.client_1.10.0.0.jar</
pathelement>
        <pathelement>modules/javax.enterprise.deploy_1.2.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.identity_1.1.2.1.jar</
pathelement>
        <pathelement>modules/com.bea.core.utils.full_1.9.0.1.jar</pathelement>
        <pathelement>modules/com.bea.core.utils.classloaders_1.9.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.logging_1.8.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.socket.api_1.2.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.digest_1.0.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.weblogic.lifecycle_1.4.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.utils.wrapper_1.4.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.management.jmx_1.4.1.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor.wl_1.3.3.0.jar</pathelement>
        <pathelement>modules/com.bea.core.redef_1.5.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.http.pubsub_1.6.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.logger_1.5.0.0.jar</
pathelement>
    <pathelement>modules/com.oracle.core.weblogic.msgcat_1.1.0.0.jar</pathelement>
        <pathelement>com.bea.core.common.security.utils_1.0.0.0_6-1-0-0.jar</
pathelement>
    </jar-path>

    <!-- No library paths -->

    <!-- No assembly paths -->
```

```
    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->

<system-class-path><pathelement>wlserver_10.3/server/lib/cryptoj.jar</pathelement></
system-class-path>
</grid-library>
```

Windows

```
<?xml version="1.0" encoding="UTF-8"?>

<grid-library os="win32">
    <grid-library-name>weblogic10_3_5-distribution</grid-library-name>
    <grid-library-version>10.3.5</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->

    <!-- No conflicts -->

    <jar-path>
        <pathelement>wlserver_10.3/server/lib/weblogic.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/api.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wls-api.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wlclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wljmxclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wljmsclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/wlcipher.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/webserviceclient.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/webserviceclient+ssl.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoCore.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoAsn1.jar</pathelement>
        <pathelement>wlserver_10.3/server/lib/EccpressoJcae.jar</pathelement>
        <pathelement>modules/com.bea.core.transaction_2.7.0.0.jar</pathelement>
        <pathelement>modules/
com.bea.core.weblogic.security.wls_1.0.0.0_6-1-0-0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security_1.0.0.0_6-1-0-0.jar</
pathelement>
        <pathelement>modules/com.bea.core.management.core_2.8.0.1.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor_1.9.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.workmanager_1.10.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.store_1.7.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.rmi.client_1.10.0.0.jar</
pathelement>
        <pathelement>modules/javax.enterprise.deploy_1.2.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.identity_1.1.2.1.jar</
pathelement>
        <pathelement>modules/com.bea.core.utils.full_1.9.0.1.jar</pathelement>
        <pathelement>modules/com.bea.core.utils.classloaders_1.9.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.logging_1.8.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.socket.api_1.2.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.digest_1.0.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.weblogic.lifecycle_1.4.0.0.jar</
pathelement>
        <pathelement>modules/com.bea.core.utils.wrapper_1.4.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.management.jmx_1.4.1.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor.wl_1.3.3.0.jar</pathelement>
        <pathelement>modules/com.bea.core.redef_1.5.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.http.pubsub_1.6.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.weblogic.security.logger_1.5.0.0.jar</
pathelement>
```

```
    <pathelement>modules/com.oracle.core.weblogic.msgcat_1.1.0.0.jar</pathelement>
        <pathelement>com.bea.core.common.security.utils_1.0.0.0_6-1-0-0.jar</
pathelement>
    </jar-path>

    <!-- No library paths -->

    <!-- No assembly paths -->

    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->

<system-class-path><pathelement>wlserver_10.3/server/lib/cryptoj.jar</pathelement></
system-class-path>
</grid-library>
```

WebLogic 12.1.1

Linux

```
<?xml version="1.0" encoding="UTF-8"?>

<grid-library os="linux">
    <grid-library-name>weblogic12_1_1-distribution</grid-library-name>
    <grid-library-version>12.1.1</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->

    <!-- No conflicts -->

    <jar-path>
        <pathelement>wlserver_12.1/server/lib/weblogic.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/api.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wls-api.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wlclient.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wljmxclient.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wljmsclient.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wlcipher.jar</pathelement>
        <pathelement>modules/com.bea.core.transaction_3.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security.wls_1.1.0.0_6-2-0-0.jar
        </pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security_1.1.0.0_6-2-0-0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.management.core_3.0.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.workmanager_2.0.0.0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.store_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.rmi.client_2.0.0.0.jar
        </pathelement>
        <pathelement>modules/javax.enterprise.deploy_1.2.1.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security.identity_2.0.0.0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.utils.full_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.utils.classloaders_3.0.0.0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.logging_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.socket.api_2.0.0.0.jar
```

```
        </pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security.digest_2.0.0.0.jar
        </pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.lifecycle_2.0.0.0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.utils.wrapper_2.0.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.management.jmx_2.0.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor.wl_2.0.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.redef_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.http.pubsub.mbean_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security.logger_1.7.0.0.jar
        </pathelement>
        <pathelement>
            modules/com.oracle.core.weblogic.msgcat_1.3.0.0.jar
        </pathelement>
        <pathelement>modules/javax.interceptor_1.1.jar</pathelement>
        <pathelement>
            modules/com.oracle.weblogic.application.archive_1.0.0.0.jar
        </pathelement>
        <pathelement>modules/javax.mail_1.1.0.0_1-4-2.jar</pathelement>
    </jar-path>


    <!-- No library paths -->

    <!-- No assembly paths -->

    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->

</grid-library>
```

Windows

```
<?xml version="1.0" encoding="UTF-8"?>

<grid-library os="win32">
    <grid-library-name>weblogic12_1_1-distribution</grid-library-name>
    <grid-library-version>12.1.1</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->

    <!-- No conflicts -->

    <jar-path>
        <pathelement>wlserver_12.1/server/lib/weblogic.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/api.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wls-api.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wlclient.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wljmxclient.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wljmsclient.jar</pathelement>
        <pathelement>wlserver_12.1/server/lib/wlcipher.jar</pathelement>
        <pathelement>modules/com.bea.core.transaction_3.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security.wls_1.1.0.0_6-2-0-0.jar
        </pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security_1.1.0.0_6-2-0-0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.management.core_3.0.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor_2.0.0.0.jar</pathelement>
```

```
        <pathelement>
            modules/com.bea.core.weblogic.workmanager_2.0.0.0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.store_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.rmi.client_2.0.0.0.jar
        </pathelement>
        <pathelement>modules/javax.enterprise.deploy_1.2.1.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security.identity_2.0.0.0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.utils.full_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.utils.classloaders_3.0.0.0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.logging_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.socket.api_2.0.0.0.jar
        </pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security.digest_2.0.0.0.jar
        </pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.lifecycle_2.0.0.0.jar
        </pathelement>
        <pathelement>modules/com.bea.core.utils.wrapper_2.0.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.management.jmx_2.0.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.descriptor.wl_2.0.0.0.jar</pathelement>
        <pathelement>modules/com.bea.core.redef_2.0.0.0.jar</pathelement>
        <pathelement>
            modules/com.bea.core.http.pubsub.mbean_2.0.0.0.jar
        </pathelement>
        <pathelement>
            modules/com.bea.core.weblogic.security.logger_1.7.0.0.jar
        </pathelement>
        <pathelement>
            modules/com.oracle.core.weblogic.msgcat_1.3.0.0.jar
        </pathelement>
        <pathelement>modules/javax.interceptor_1.1.jar</pathelement>
        <pathelement>
            modules/com.oracle.weblogic.application.archive_1.0.0.0.jar
        </pathelement>
        <pathelement>modules/javax.mail_1.1.0.0_1-4-2.jar</pathelement>
    </jar-path>


    <!-- No library paths -->

    <!-- No assembly paths -->

    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->

</grid-library>
```

WebLogic 12.1.3

Linux

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library os="linux64">
             <grid-library-name>weblogic12_1_3-distribution</grid-library-name>
    <grid-library-version>12.1.3</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->
```

```
    <!-- No conflicts -->

    <jar-path>
        <pathelement>wlserver/server/lib/wlfullclient.jar</pathelement>
                                <pathelement>wlserver/server/lib/cryptoj.jar</
pathelement>
    </jar-path>


    <!-- No library paths -->

    <!-- No assembly paths -->

    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->

</grid-library>
```

Windows

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library os="win32">
             <grid-library-name>weblogic12_1_3-distribution</grid-library-name>
    <grid-library-version>12.1.3</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->

    <!-- No conflicts -->

    <jar-path>
        <pathelement>wlserver/server/lib/wlfullclient.jar</pathelement>
                                <pathelement>wlserver/server/lib/cryptoj.jar</
pathelement>
    </jar-path>


    <!-- No library paths -->

    <!-- No assembly paths -->

    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->

</grid-library>
```

WebLogic 12.2.1

Linux

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library os="linux64">
             <grid-library-name>weblogic12_2_1-distribution</grid-library-name>
    <grid-library-version>12.2.1</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->

    <!-- No conflicts -->
```

```
    <jar-path>
        <pathelement>wlfullclient.jar</pathelement>
    </jar-path>


    <!-- No library paths -->

    <!-- No assembly paths -->

    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->

</grid-library>
```

Windows

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library os="win32">
             <grid-library-name>weblogic12_2_1-distribution</grid-library-name>
    <grid-library-version>12.2.1</grid-library-version>

    <!-- No JRE arguments -->

    <!-- No dependencies -->

    <!-- No conflicts -->

    <jar-path>
        <pathelement>wlfullclient.jar</pathelement>
    </jar-path>


    <!-- No library paths -->

    <!-- No assembly paths -->

    <!-- No command paths -->

    <!-- No hooks paths -->

    <!-- No environment variables -->

    <!-- No system properties -->

</grid-library>
```