BEAProducts

Domain Template Reference

BEA WebLogic Server® Version 10.0
Revised: March 28, 2008

Contents

Domain Template Reference

Domain Template Reference

This document provides general information about templates. This document discusses the following topics:

- Types of Templates
- Location of Installed Templates
- Template Tools
- Template Summary
- Relationships Between Templates
- Files Typically Included in a Template
- Basic WebLogic Server Domain Template
- WebLogic Beehive Extension Template
- WebLogic Advanced Web Services Extension Template
- Avitek Medical Records Sample Domain Template
- BEA Workshop for WebLogic Extension Template
- Workshop for WebLogic 10.2 Extension Template
- WebLogic Server Default Domain Extension Template
- WebLogic Server Examples Extension Template
- WebLogic Integration BPM Extension Template

- WebLogic Integration Worklist Extension Template
- WebLogic Integration Worklist (Compatibility) Extension Template
- WebLogic Portal Extension Template
- WebLogic Portal Collaboration Repository
- WebLogic Portal GroupSpace Application

Types of Templates

The term *template* refers to a Java Archive (JAR) file that contains the files and scripts required to create or extend a domain. The types of template include:

- *Domain template*—defines the full set of resources within a domain, including infrastructure components, applications, services, security options, and general environment and operating system options.

  The product installation includes a predefined Basic WebLogic Server Domain template. This template defines the core set of resources within a domain, including an Administration Server and basic configuration information. For more information on Basic WebLogic Server Domain template, see "Basic WebLogic Server Domain Template" on page 1-13.

  You can also create a custom domain template from an existing domain by using the Domain Template Builder or the `pack` command. By using the Domain Template Builder, you can also create a custom domain template from an existing template.

- *Extension template*—defines the applications and services that you can add to an existing domain, including product component functionality and resources such as JDBC or JMS.

  The product installation includes several predefined extension templates. For a summary of extension templates, see Template Summary.

  You can also create a custom extension template from an existing domain or template using the Domain Template Builder.

  You can use the Worklist extension templates to add 8.1.x backward compatibility to a 10.2 Worklist domain. For more information, see WebLogic Integration Worklist Extension Template, and WebLogic Integration Worklist (Compatibility) Extension Template.

- *Managed Server template* – defines the subset of resources within a domain that are required to create a Managed Server domain directory on a remote machine. You can create a custom Managed Server template by using the `pack` command. Complete details are provided in *Creating Templates and Domains Using the pack and unpack Commands*.

Location of Installed Templates

The following table identifies the location of the predefined templates provided with your product installation, where *`BEA_HOME`* represents the product installation directory.

Table 1 Location of Templates

Type of Template	Directory Location
Domain	*`BEA_HOME`*`\common\templates\domains`
Extension	*`BEA_HOME`*`\common\templates\applications`

Template Tools

The following table identifies the tools with which you can create templates and the tools with which you can use templates to create or extend a domain.

Table 2 Template Tools

To . . .	Use this tool . . .
Create a new domain	• Configuration Wizard • WLST Offline • `unpack` command
Extend an existing domain	• Configuration Wizard • WLST Offline
Create a new Managed Server domain on a remote machine	`unpack` command
Create a domain template	• Domain Template Builder • `pack` command • WLST Offline
Create an extension template	Domain Template Builder
Create a Managed Server template	`pack` command

Note: All the tools used to create or extend a domain leverage a common underlying infrastructure, referred to as the *Configuration Wizard framework*.

- For information about using the Configuration Wizard, see *Creating WebLogic Domains Using the Configuration Wizard*.
- For information about using the WLST Offline, see *WebLogic Scripting Tool*.
- For information about using the `pack/unpack` commands, see in *Creating Templates and Domains Using the pack and unpack Commands.*
- For information about using the Domain Template Builder, see *Creating Templates Using the Domain Template Builder*.

Template Summary

The following tables summarizes the predefined templates that may be provided in your product installation.

Table 3 Summary of AquaLogic Template

Template	Filename	Description
AquaLogic Service Bus Extension Template	`wlsb.jar`	Extends the base WebLogic Server domain by providing the resources required to support AquaLogic Service Bus. For more information on the AquaLogic Service Bus Extension template, see the AquaLogic Service Bus Deployment Resources in the *Deployment Guide*.

Table 4 Summary of WebLogic Server, WebLogic Integration, Portal, and BEA Workshop for WebLogic Platform Templates

Template	Filename	Description
Domain Template		
Basic WebLogic Server Domain Template	`wls.jar`	Creates a base WebLogic Server domain.
Extension Templates		

Table 4 Summary of WebLogic Server, WebLogic Integration, Portal, and BEA Workshop for WebLogic Platform Templates (Continued)

Template	Filename	Description
WebLogic Beehive Extension Template	`weblogic-beehive.jar`	Extends the base WebLogic Server domain to create a WebLogic Beehive domain. Adds required Beehive libraries to support run-time use of controls. **Note:** Resources from the WebLogic Advanced Web Services Extension template are required to create a complete WebLogic Beehive domain.
WebLogic Advanced Web Services Extension Template	`wls_webservice.jar`	Extends an existing WebLogic Server domain to add functionality required for advanced Web Services, including WSRM, Buffering, and JMS Transport.
Avitek Medical Records Sample Domain Template	`medrec.jar`	Extends the Basic WebLogic Server domain to create the Avitek Medical Records sample domain. This domain is a WebLogic Server sample application suite that demonstrates all aspects of the J2EE platform.
BEA Workshop for WebLogic Extension Template	`workshop_wl.jar`	Extends the Basic WebLogic Server domain to create BEA Workshop for WebLogic domain.
Workshop for WebLogic 10.2 Extension Template	`workshop_wl_10_2.jar`	Extends the Basic WebLogic Server domain to create Workshop for WebLogic 10.2 domain.

Table 4 Summary of WebLogic Server, WebLogic Integration, Portal, and BEA Workshop for WebLogic Platform Templates (Continued)

Template	Filename	Description
WebLogic Server Default Domain Extension Template	`wls_default.jar`	Extends the Basic WebLogic Server domain with a web application designed to guide new users through an introduction to WebLogic Server. When running the web application, users can review informative content on various topics, including highlights of WebLogic Server functionality. From the web application, users can also run several preconfigured, precompiled examples. Resources from this extension template are required for a WebLogic Server Examples domain.
WebLogic Server Examples Extension Template	`wls_examples.jar`	Extends the WebLogic Server domain containing resources from the base WebLogic Server domain template and the WebLogic Server Default Domain extension template to create a complete WebLogic Server Examples domain. The WebLogic Server Examples domain contains a collection of examples that illustrate best practices for coding individual J2EE and WebLogic Server APIs.
WebLogic Integration BPM Extension Template	`wli_jpd.jar`	Imports the resources needed to support the development of WebLogic Integration applications.
WebLogic Integration Worklist Extension Template	`wli_worklist.jar`	
WebLogic Integration Worklist (Compatibility) Extension Template	`wli_worklist81x.jar`	
WebLogic Portal Extension Template	`wlp.jar`	Extends a WebLogic Server domain to allow for WebLogic Portal application development.

Table 4 Summary of WebLogic Server, WebLogic Integration, Portal, and BEA Workshop for WebLogic Platform Templates (Continued)

Template	Filename	Description
WebLogic Portal Collaboration Repository	`wlp_groupspacedb.jar`	Extends a WebLogic Portal domain to allow for development and hosting of GroupSpace applications. This template extends a WebLogic Portal domain by adding additional Datasources and creating an additional database schema.
WebLogic Portal GroupSpace Application	`wlp_groupspace.jar`	Extends a WebLogic Portal GroupSpace enabled domain by adding a preconfigured GroupSpace application to the domain.
WebLogic Personalization Extension	`p13n.jar`	Extends an existing WebLogic Server domain to add Weblogic Personalization functionality.
WebLogic Content Extension	`content.jar`	Extends an existing WebLogic Server domain to add WebLogic Content Management functionality.
WebLogic Simple Producer (Portal) Extension	`wsrp-simple-producer .jar`	Extends an existing WebLogic domain to add WebLogic Simple Producer (Portal) functionality. Domains extended with this template support WebLogic Simple Producer functionality. Use this template to enable the development of WebLogic Simple Producer applications in existing domains.

Relationships Between Templates

This section discusses the following topics:

- "WebLogic Server Resources as a Prerequisite" on page 7
- "Relationships Between Templates" on page 8

WebLogic Server Resources as a Prerequisite

WebLogic Server resources must be set up in your domain before you can add resources from an extension template. When you select an extension template, the Configuration Wizard framework checks to make sure the required resources are available for you.

Relationships Between Templates

You can create a base WebLogic domain by using the predefined Basic WebLogic Server domain template, or you can create a Basic WebLogic domain and extend it incrementally using the extension templates. The following table shows the relationships between the templates and the domains created.

Table 5 Relationships Between Templates

This type of domain . . .	Requires resources from these templates . . .
AquaLogic Service Bus	For more information on the AquaLogic Service Bus Extension template, see the AquaLogic Service Bus Deployment Resources in the *Deployment Guide*.
Avitek Medical Records Sample	Basic WebLogic Server Domain template, `wls.jar` + Avitek Medical Records Sample Domain extension template, `medrec.jar`
WebLogic Server (base)	Basic WebLogic Server Domain template, `wls.jar`
WebLogic Server Default	Basic WebLogic Server Domain template, `wls.jar` + WebLogic Server Default Domain extension template, `wls_default.jar`
WebLogic Server Examples	Basic WebLogic Server Domain template, `wls.jar` + WebLogic Server Default Domain extension template, `wls_default.jar` + WebLogic Server Examples extension template, `wls_examples.jar`
WebLogic Integration Worklist Extension	Basic WebLogic Server Domain, `wls.jar` + WebLogic Advanced Web Services Extension, `wls_webservice.jar`, + BEA Workshop for WebLogic Extension, `workshop_wl_10_2.jar` + WebLogic Personalization Extension, `p13n.jar` + Worklist Extension Template, `wli_worklist.jar`.

Table 5 Relationships Between Templates (Continued)

This type of domain . . .	Requires resources from these templates . . .
WebLogic Integration Worklist (Compatibility) Extension	Basic WebLogic Server Domain, `wls.jar` + WebLogic Advanced Web Services Extension, `wls_webservice.jar`, + BEA Workshop for WebLogic Extension, `workshop_wl_10_2.jar` + WebLogic Personalization Extension, `p13n.jar` + Worklist Extension Template, `wli_worklist.jar` +WebLogic Integration Worklist (Compatibility) Extension, `wli_worklist81x.jar`
WebLogic Integration Business Process Management	Basic WebLogic Server Domain, `wls.jar` + WebLogic Advanced Web Services Extension, `wls_webservice.jar`, + BEA Workshop for WebLogic Extension, `workshop_wl_10_2.jar`, + WebLogic Personalization Extension, `p13n.jar`, + WebLogic Integration BPM Extension Templates, `wli_jpd.jar`.
WebLogic Advanced Web Services Extension	Basic WebLogic Server Domain, `wls.jar` + WebLogic Advanced Web Services Extension, `wls_webservice.jar`
WebLogic Beehive Extension	Basic WebLogic Server Domain, `wls.jar` + WebLogic Advanced Web Services Extension, `wls_webservice.jar` + WebLogic Beehive Extension, `weblogic_beehive.jar`
WebLogic Portal	Basic WebLogic Server Domain, `wls.jar` + WebLogic Advanced Web Services Extension, `wls_webservice.jar`, + BEA Workshop for WebLogic Extension, `workshop_wl_10_2.jar` + WebLogic Personalization Extension, `p13n.jar` + WebLogic Content Extension, `content.jar` + WebLogic Portal Extension, `wlp.jar`

Table 5 Relationships Between Templates (Continued)

This type of domain . . .	Requires resources from these templates . . .
WebLogic Portal Collaboration Repository	Basic WebLogic Server Domain, `wls.jar` + WebLogic Advanced Web Services Extension, `wls_webservice.jar`, + BEA Workshop for WebLogic Extension, `workshop_wl_10_2.jar` + WebLogic Personalization Extension, `p13n.jar` + WebLogic Content Extension, `content.jar` + WebLogic Portal Extension, `wlp.jar` + WebLogic Portal Collaboration Repository Extension, `wlp_groupspacedb.jar`
Weblogic Portal GroupSpace Application	Basic WebLogic Server Domain, `wls.jar` + BEA Workshop for WebLogic Extension, `workshop_wl_10_2.jar` + WebLogic Advanced Web Services Extension, `wls_webservice.jar` + WebLogic Personalization Extension, `p13n.jar` + WebLogic Content Extension, `content.jar` + WebLogic Portal Extension, `wlp.jar` + WebLogic Portal Collaboration Repository Extension, `wlp_groupspacedb.jar` +WebLogic Portal GroupSpace Application Extension, `wlp_groupspace.jar`
BEA Workshop for WebLogic Platform	Basic WebLogic Server Domain, `wls.jar` + Advanced Web Services Extension, `wls_webservice.jar` + BEA Workshop for WebLogic Extension, `workshop_wl.jar`
Workshop for WebLogic Platform 10.2	Basic WebLogic Server Domain, `wls.jar` + Advanced Web Services Extension, `wls_webservice.jar` + Workshop for WebLogic 10.2, `workshop_wl_10_2.jar`

Files Typically Included in a Template

The basic files included in any template are `config.xml` and `template-info.xml`. There are additional files in the predefined templates, and a domain is created or extended based on these files. The following table describes the files typically included in a domain or extension template.

Table 6 Files Included in a Template

Filename	Description
product component files	Various files used to complete the domain setup for a specific BEA product component. Such files may provide information for security and default database settings.
`*-jdbc.xml`	Sets up or extends a domain with JDBC system resources required by a product component. In a template, the `*-jdbc.xml` files must be located in the `config\jdbc` directory.
`*-jms.xml`	Sets up or extends a domain with JMS system resources required by a product component. In a template, the `*-jms.xml` files must be located in the `config\jms` directory.
`clusters.script`	Used to modify the Configuration Wizard framework's default auto-configuration of a cluster. By default, resources are targeted to the cluster. You can `unassign` a resource from the cluster and then `assign` it to another component. To specify a target, you can use the following replacement variables: • `%AManagedServer%` — Any Managed Server • `%AllManagedServers%` — Comma-separated list of all Managed Servers • `%AdminServer%` — Administration Server name • `%Cluster%` — Cluster name • `%ProxyServer%` — Proxy server name • `%HTTPProxyApp%` — http proxy application definition Note the following additional considerations: • You must use the name attribute of an object that is to be replaced. • You can use an asterisk (*) as a wildcard for "All." In a template, the `clusters.script` file must be located in the `script` directory. **Note:** AquaLogic Service Bus overrides this behavior. For more information on the AquaLogic Service Bus Extension template, see the AquaLogic Service Bus Deployment Resources in the *Deployment Guide*.
`config.xml`	Sets up or extends the domain configuration. In a template, the `config.xml` file must be located in the `config` directory.

Table 6 Files Included in a Template (Continued)

Filename	Description
`jdbc.index`	Identifies the locations of SQL scripts used to set up a database. The file lists the scripts in the order in which they must be run. If the scripts are not contained in the template, but are located in the product installation directory, that directory can be represented by a tilde (~) in the pathname for the scripts, as shown in the following example: `~/integration/common/dbscripts/oracle/reporting_runtime.sql` Specifically, the tilde represents the directory path identified by the `$USER_INSTALL_DIR$` variable in the `stringsubs.xml` file. In a template, a `jdbc.index` file must be located in the `_jdbc_\`*`dbtype`*`\`*`dbversion`* directory, where *`dbtype`* is the type of database, such as Oracle, and *`dbversion`* is the database version, such as 9i. In addition to listing the SQL files related to a data source, the `jdbc.index` file contains information about the categories associated with the data source. The default `dbCategories` that are available are: • 'Drop/Create P13N Database Objects' category associated with the `p13nDataSource` data source, which is a part of the `p13n.jar` domain template • 'Drop/Create Portal Database Objects' category associated with the "p13nDataSource" data source, which is a part of the `wlp.jar` domain template • 'Drop/Create GroupSpace Database Objects' category associated with the `appsGroupSpaceDataSource` data source, which is a part of the `wlp_groupspacedb.jar` domain template All these template jar files are located in the *`BEA_HOME`*`\wlserver_10.2\common\templates\applications` directory.
`security.xml`	Used to create user groups and roles that establish identity and access to domain resources. You can create the default Admin user only through the `security.xml` in a *domain* template. However, you can create user groups and roles through the `security.xml` included in either a domain or an extension template.
`startmenu.xml`	Used to create Windows start menu entries.
`startscript.xml`	Used to create the `*.cmd` and `*.sh` files that are placed into a domain's root and `bin` directories.

Table 6 Files Included in a Template (Continued)

Filename	Description
`stringsubs.xml`	Identifies string substitution values and files that will receive string substitutions during domain creation or extension. The files that will receive string substitutions must already be prepared with replacement variables. During domain creation or extension, the Configuration Wizard framework runs macros to replace variables with the appropriate string substitution, using information from *`BEA_HOME`*`\common\lib\macrorules.xml`, where *`BEA_HOME`* is the product installation directory.
`template-info.xml`	Provides template identification information, such as the template name, software version, type of template (domain or application), author, description, and so on.

Basic WebLogic Server Domain Template

Your product installation provides one predefined Basic WebLogic Server domain template. All other predefined templates are extension templates that you may use to add resources, services, and applications to a Basic WebLogic Server domain. You can easily create or extend a domain by using these predefined templates with the Configuration Wizard or WLST.

Generated Domain Output

The Basic WebLogic Server Domain template allows you to create a simple WebLogic Server domain. By default, when using the Basic WebLogic Server Domain template, you generate a domain that contains only the required components: an Administration Server and a single administrative user. Any required applications must be created and configured within the domain.

The following table defines the default directory structure and files generated by the Basic WebLogic Server Domain template. Unless otherwise specified, by default, the Configuration Wizard framework creates the domain in the *`BEA_HOME`*`\user_projects\domains\base_domain` directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 7 Output Generated from the Basic WebLogic Server Domain Template

Directory	File	Description
`user_projects\applications\base_domain\`		
	n.a.	Directory designated as the repository for any custom application files that you create.

Table 7 Output Generated from the Basic WebLogic Server Domain Template (Continued)

Directory	File	Description
`user_projects\domains\base_domain\`		
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.

Table 7 Output Generated from the Basic WebLogic Server Domain Template (Continued)

Directory	File	Description
`config\`	`config.xml`	File containing the configuration information used by the Administration Server. For more information, see Domain Configuration Files in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
`config\jms\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.

Table 7 Output Generated from the Basic WebLogic Server Domain Template (Continued)

Directory	File	Description
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 7 Output Generated from the Basic WebLogic Server Domain Template (Continued)

Directory	File	Description

`security\`	`DefaultAuthenticatorInit.ldift`
	`DefaultRoleMapperInit.ldift`
	`XACMLRoleMapperInit.ldift`
		Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information.
		Note: WebLogic domains created with this release use the XACML providers, by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111.

	`SerializedSystemIni.dat`	File containing encrypted security information.
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.

Resources and Services Configured for WebLogic Server Domain Template

The following table identifies the resources and services configured in a domain created with the Basic WebLogic Server Domain template.

Table 8 Resources Configured in a Basic WebLogic Server Domain

Resource Type	Name	Notes

Administration Server	`AdminServer`	When using the Configuration Wizard or WLST Offline to create a new domain, and you want the Administration Server name to be different from the default name, `AdminServer`, you must configure the name manually. You cannot change the name later when applying an extension template.
		
		For information about customizing the Administration Server name while creating a domain with the Configuration Wizard, see *Creating WebLogic Domains Using the Configuration Wizard*.
		
		For information about customizing the Administration Server name while creating a domain with WLST Offline, see “Creating and Configuring WebLogic Domains Using WLST Offline” in *WebLogic Scripting Tool*.
		
		The following sample WLST Offline code snippet shows how to change the default Administration Server name, `AdminServer`, to `MedRecServer`.
		
		`#-----------------------------------`
		`#Read the Basic WebLogic Server Domain template`
		`readTemplate``('d:/bea/wlserver_10.2/common/templates/domains/`**`wls.jar`**`')`
		`#Change the Administration Server name.`
		`cd('Servers/AdminServer')`
		`set('Name', 'MedRecServer')`
		`#-----------------------------------`

Security realm	`myrealm`	n.a.

WebLogic Beehive Extension Template

By using the Configuration Wizard or WLST, you can easily extend a base WebLogic Server domain to include the resources required for using WebLogic Beehive. You accomplish this by adding the resources and services provided in the WebLogic Beehive and WebLogic Advanced Web Services extension templates to a base WebLogic Server domain.

Generated Domain Output

The following table defines the default directory structure and files generated after applying the WebLogic Beehive and WebLogic Advanced Web Services extension templates to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the `*BEA_HOME*\user_projects\domains\base_domain` directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 9 Base Domain After Applying the WebLogic Beehive and WebLogic Advanced Web Services Extension Templates

Directory	File	Description
`user_projects\applications\base_domain\`		
	n.a.	Directory serving as a placeholder for any custom application files that you create.
`user_projects\domains\base_domain\`		
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`URLs.dat`	File containing the URL for the JDBC database.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.

Table 9 Base Domain After Applying the WebLogic Beehive and WebLogic Advanced Web Services Extension Templates (Continued)

Directory	File	Description
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the Administration Server. For more information, see Domain Configuration Files in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).

Table 9 Base Domain After Applying the WebLogic Beehive and WebLogic Advanced Web Services Extension Templates (Continued)

Directory	File	Description
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`cgDataSource-jdbc.xml`	Global XA JDBC Data Source module for the domain configured for conversational Web services.
	`cgDataSource-nonXA-jdbc.xml`	Global non-XA JDBC Data Source module for the domain configured for conversational Web services.
`config\jms\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
	`conversational-jms.xml`	Global JMS module for the domain configured for conversational Web services.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.

Table 9 Base Domain After Applying the WebLogic Beehive and WebLogic Advanced Web Services Extension Templates (Continued)

Directory	File	Description
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 9 Base Domain After Applying the WebLogic Beehive and WebLogic Advanced Web Services Extension Templates (Continued)

Directory	File	Description

`security\`	`DefaultAuthenticatorInit.ldift`
	`DefaultRoleMapperInit.ldift`
	`XACMLRoleMapperInit.ldift`
		Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information.
		Note: WebLogic domains created with this release use the XACML providers by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111.

	`SerializedSystemIni.dat`	File containing encrypted security information.

Table 9 Base Domain After Applying the WebLogic Beehive and WebLogic Advanced Web Services Extension Templates (Continued)

Directory	File	Description
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://edocs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the WebLogic Beehive and WebLogic Advanced Web Services extension templates.

Table 10 Resources Configured in a WebLogic Beehive Domain

Resource Type	Name	Extension Result
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `cgServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.
Libraries Deployed	`beehive-netui-1.0#1.0@1.0`	Adds the Apache Beehive NetUI Version 1.0 libraries provided by the WebLogic Beehive extension template and targets them to the Administration Server, `AdminServer`. These libraries support pageflow development, and depend on the libraries contained in `struts-1.1.war` and `weblogic-beehive-1.0.ear`.
	`struts-1.1#1.1@1.0`	Adds the Apache Struts Version 1.1 libraries provided by the WebLogic Beehive extension template and targets them to the Administration Server, `AdminServer`.
	`struts-1.2#1.2@1.0`	Adds the Apache Struts Version 1.2 libraries provided by the WebLogic Beehive extension template and targets them to the Administration Server, `AdminServer`.

WebLogic Advanced Web Services Extension Template

By using the Configuration Wizard or WLST, you can easily extend a base WebLogic Server domain to include the resources required for advanced Web services. You accomplish this by adding the resources and services provided in the WebLogic Advanced Web Services extension template to a base WebLogic Server domain.

Generated Domain Output

The following table defines the default directory structure and files generated after applying the WebLogic Advanced Web Services extension template to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the `*BEA_HOME*\user_projects\domains\base_domain` directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 11 Base Domain After Applying the WebLogic Advanced Web Services Extension Template

Directory	File	Description
`user_projects\applications\base_domain\`		
	n.a.	Directory serving as a placeholder for any custom application files that you create.
`user_projects\domains\base_domain\`		
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`URLs.dat`	File containing the URL for the JDBC database.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.

Table 11 Base Domain After Applying the WebLogic Advanced Web Services Extension Template (Continued)

Directory	File	Description
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the "Domain Configuration Files" Administration Server. For more information, see in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).

Table 11 Base Domain After Applying the WebLogic Advanced Web Services Extension Template (Continued)

Directory	File	Description
config\jms\	readme.txt	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
	`wseejmsmodule-jms.xml`	Global JMS module for the domain configured for advanced Web Services.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.

Table 11 Base Domain After Applying the WebLogic Advanced Web Services Extension Template (Continued)

Directory	File	Description
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 11 Base Domain After Applying the WebLogic Advanced Web Services Extension Template (Continued)

Directory	File	Description

`security\`	`DefaultAuthenticatorInit.ldift`
	`DefaultRoleMapperInit.ldift`
	`XACMLRoleMapperInit.ldift`
		Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information.
		Note: WebLogic domains created with this release use the XACML providers by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111

	`SerializedSystemIni.dat`	File containing encrypted security information.

Table 11 Base Domain After Applying the WebLogic Advanced Web Services Extension Template (Continued)

Directory	File	Description
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the WebLogic Advanced Web Services extension template.

Table 12 Resources Configured in a WebLogic Advanced Web Services Domain

Resource Type	Name	Extension Result
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `cgServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.
JMS Queues	`WseeMessageQueue`	Adds the JMS queue to the JMS server, `WseeJmsServer`.
	`WseeCallbackQueue`	Adds the JMS queue to the JMS server, `WseeJmsServer`.
JMS Server	`WseeJmsServer`	Adds the JMS server as a system resource and targets it to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.

Avitek Medical Records Sample Domain Template

By using the Configuration Wizard or WLST, you can easily extend a base WebLogic Server domain to create an Avitek Medical Records Sample domain. You accomplish this by adding the resources and services provided in the Avitek Medical Records Sample domain extension template to a base WebLogic Server domain.

For more information about the Avitek Medical Records sample application, see Sample Application Examples and Tutorials for BEA WebLogic Server 10.0.

Generated Domain Output

The following table defines the default directory structure and files generated after applying the Avitek Medical Records Sample Domain extension template to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the *BEA_HOME*\user_projects\domains\base_domain directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 13 Base Domain After Applying the Avitek Medical Records Sample Extension Template

Directory	File	Description
user_projects\applications\base_domain\		
build\	Various	Includes Avitek Medical Records split directory deployments.
console-extension\	Various	Includes sub-directories containing various files used to demonstrate extending the WebLogic Server Administration Console with a different look and feel.
dist\	Various	Includes sub-directories containing various files of the Avitek Medical Records applications in an exploded (unarchived) directory format.
doc\	Various	Directory and files containing the Avitek Medical Records online documentation.
lib\	Various	Includes sub-directories containing library files supporting the Avitek Medical Records sample.
setup\	build.xml	Ant build file used with corresponding scripts to set up a database for the Avitek Medical Records sample.
setup\db\	medrec_mysql.ddl medrec_mysql_data.sql medrec_oracle.ddl medrec_oracle_data.sql medrec_pointbase.ddl medrec_pointbase_data.sql	SQL scripts used to set up different databases that can be used with the Avitek Medical Records sample.
src\	Various	Includes sub-directories containing Avitek Medical Records source code including various Java, XML, JSP, HTML files, and so on.
user_projects\domains\base_domain\		

Table 13 Base Domain After Applying the Avitek Medical Records Sample Extension Template (Continued)

Directory	File	Description
	`democa.pem`	Provides sample SSL protocol support for servers in the domain.
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`log4jConfig.xml`	Configures Avitek Medical Records Log4j implementation including the `MedRecApp.log` file.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.

Table 13 Base Domain After Applying the Avitek Medical Records Sample Extension Template (Continued)

Directory	File	Description
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the Administration Server. For more information, see Domain Configuration Files in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."

Table 13 Base Domain After Applying the Avitek Medical Records Sample Extension Template (Continued)

Directory	File	Description
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).
	`MedRecWLDF.xml`	Diagnostic descriptor information for the Avitek Medical Records diagnostics instrumentation.
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`MedRecGlobalDataSource-jdbc.xml`	Global non-XA JDBC Data Source module for the Avitek Medical Records domain.
	`MedRecGlobalDataSourceXA-jdbc.xml`	Global XA JDBC Data Source module for the Avitek Medical Records domain.
`config\jms\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
	`MedRec-jms.xml`	Global JMS module for the Avitek Medical Records domain.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.

Table 13 Base Domain After Applying the Avitek Medical Records Sample Extension Template (Continued)

Directory	File	Description
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`incoming\`	`StJohnHospital.xml`	Location where XML files containing fictitious patient names are uploaded by the Administration application of the Avitek Medical Records sample application.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.

Table 13 Base Domain After Applying the Avitek Medical Records Sample Extension Template (Continued)

Directory	File	Description
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.
	`log4j.jar` `wllog4j.jar`	Libraries used by the Avitek Medical Records domain.
`medrecWseeFileStore\`		File store used by the Avitek Medical Records application.
`physicianFileStore\`		File store used by the Physician application.
`rmfilestore\`		File store used by system resources.
`security\`	`DefaultAuthenticatorInit.ldift` `DefaultAuthorizerInit.ldift` `DefaultRoleMapperInit.ldift` `XACMLAuthorizerInit.ldift` `XACMLRoleMapperInit.ldift`	Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information. **Note:** WebLogic domains created with this release use the XACML providers, by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111.
	`MedRecDBMSPlugin.jar`	A `CustomDBMSAuthenticatorPlugin` used to validate a user/password against a DBMS for the Avitek Medical Records sample application.
	`SerializedSystemIni.dat`	File containing encrypted security information.

Table 13 Base Domain After Applying the Avitek Medical Records Sample Extension Template (Continued)

Directory	File	Description
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the Avitek Medical Records Sample extension template.

Table 14 Resources Configured in an Avitek Medical Records Domain

Resource Type	Name	Extension Result
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `MedRecServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.
Application Deployments	`InitEAR`	Adds the `InitEAR` Web application and targets it to the Administration Server, `AdminServer`.
	`MedRecEAR`	Adds the `MedRecEAR` Web application and targets it to the Administration Server, `AdminServer`.
	`PhysicianEAR`	Adds the `PhysicianEAR` Web application and targets it to the Administration Server, `AdminServer`.
	`StartBrowserEAR`	Adds the `StartBrowserEAR` Web application and targets it to the Administration Server, `AdminServer`.

Table 14 Resources Configured in an Avitek Medical Records Domain (Continued)

Resource Type	Name	Extension Result
File Stores	`FileStore`	Adds the file store and targets the store to the Administration Server, `AdminServer`.
	`MedRecWseeFileStore`	Adds the file store to be used as the persistent store for the JMS server, `MedRecWseeJMSServer`, and targets the store to the Administration Server, `AdminServer`.
	`PhysicianFileStore`	Adds the file store and targets the store to the Administration Server, `AdminServer`.
JDBC Data Sources	`MedRecGlobalDataSource`	Identifies the JDBC data source as a `MedRecGlobalDataSource` system resource.
	`MedRecGlobalDataSourceXA`	Identifies the JDBC data source as a `MedRecGlobalDataSourceXA` system resource.
JDBC Store	`MedRecJMSJDBCStore`	Adds the JDBC store to be used with the JDBC data source, `MedRecGlobalDataSource`, and as the persistent store for the JMS server, `MedRecJMSServer`, and targets the store to the Administration Server, `AdminServer`.
JDBC System Resources	`MedRecGlobalDataSource` `MedRecGlobalDataSourceXA`	Identifies the JDBC data source and connection pool setups to be used for non-XA and XA JDBC system resources, and targets the resources to the Administration Server, `AdminServer`.
JMS Queues	`weblogic.wsee.reliability.wseeMedRecDestinationQueue`	Adds the JMS queue to the JMS server, `MedRecWseeJMSServer`.

Table 14 Resources Configured in an Avitek Medical Records Domain (Continued)

Resource Type	Name	Extension Result
JMS Servers	`MedRecJMSServer`	Adds the JMS server as a `MedRec-jms` system resource and targets it to the Administration Server, `AdminServer`.
	`MedRecWseeJMSServer`	Adds the JMS server as a `MedRec-jms` system resource and targets it to the Administration Server, `AdminServer`.
JMS System Resources	`MedRec-jms`	Adds the JMS servers, connection factories, and queues to be used as JMS system resources, and targets the resources to the Administration Server, `AdminServer`.
Mail Session	`mail/MedRecMailSession`	Adds the mail session.
SAF Agent	`MedRecSAFAgent`	Adds this store-and-forward agent, which uses the file store, `MedRecWseeFileStore`, and targets it to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided in the base WebLogic Server domain.
WLDF System Resource	`MedRecWLDF`	Adds the WLDF system resource and defined WLDF instrumentation monitors for dye injection, and targets them to the Administration Server, `AdminServer`.

BEA Workshop for WebLogic Extension Template

Using the Configuration Wizard or WLST, you can easily extend a base WebLogic Server domain to include the resources required for using BEA Workshop for WebLogic Platform. You accomplish this by adding the resources and services provided in the BEA Workshop for WebLogic Platform template to a base WebLogic Server domain.

Note: Using the Configuration Wizard in graphical mode, you can easily create a new BEA Workshop for WebLogic Platform domain by checking the BEA Workshop for WebLogic Platform check box in the **Select Domain Source** window. The result is the same as creating a base WebLogic Server domain first and then extending that domain with both the BEA Workshop for WebLogic Platform extension template. For more information about the templates required to create a BEA Workshop for WebLogic Platform domain, see "Relationships Between Templates" on page 7.

Generated Domain Output

The following table defines the default directory structure and files generated after applying the BEA Workshop for WebLogic Platform template to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the *BEA_HOME*`\user_projects\domains\base_domain` directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 15 Base Domain After Applying the BEA Workshop for WebLogic Platform Template

Directory	File	Description
`user_projects\applications\base_domain\`		
	n.a.	Directory serving as a placeholder for any custom application files that you create.
`user_projects\domains\base_domain\`		
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`URLs.dat`	File containing the URL for the JDBC database.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.

Table 15 Base Domain After Applying the BEA Workshop for WebLogic Platform Template (Continued)

Directory	File	Description
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the Administration Server. For more information, see Domain Configuration Files in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).

Table 15 Base Domain After Applying the BEA Workshop for WebLogic Platform Template (Continued)

Directory	File	Description
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`cgDataSource-jdbc.xml`	Global XA JDBC Data Source module for the domain configured for advanced Web services.
	`cgDataSource-nonXA-jdbc.xml`	Global non-XA JDBC Data Source module for the domain configured for advanced Web services.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.

Table 15 Base Domain After Applying the BEA Workshop for WebLogic Platform Template (Continued)

Directory	File	Description
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 15 Base Domain After Applying the BEA Workshop for WebLogic Platform Template (Continued)

Directory	File	Description

`security\`	`DefaultAuthenticatorInit.ldift`
	`DefaultRoleMapperInit.ldift`
	`XACMLRoleMapperInit.ldift`
		Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information.
		Note: WebLogic domains created with this release use the XACML providers, by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111

	`SerializedSystemIni.dat`	File containing encrypted security information.
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://edocs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the BEA Workshop for WebLogic Platform template.

Table 16 Resources Configured in a BEA Workshop for WebLogic Platform Domain

Resource Type	Name	Extension Result
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `cgServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.
JDBC Data Source	`cgDataSource`	Defines an XA JDBC data source including its associated jdbc connection pool. The data source is named `cgDataSource`.
	`cgDataSource-nonXA`	Includes the JDBC data source and connection pool setups defined as `cgDataSource` in the domain and targets them to the correct server(s).
JDBC Store	`cgJMSStore`	Uses the JDBC store provided by the BEA Workshop for WebLogic Platform extension template. The JDBC store is to be used with the JDBC data source, `cgDataSource-nonXA`, and the JMS server, `WseeJmsServer`, as a persistent store, and is targeted to the Administration Server, `AdminServer`.
JDBC System Resources	`cgDataSource` `cgDataSource-nonXA`	Identifies the JDBC data source and connection pool setups to be used for JDBC system.

Table 16 Resources Configured in a BEA Workshop for WebLogic Platform Domain (Continued)

Resource Type	Name	Extension Result
JMS Server	`WseeJmsServer`	Uses the JMS server provided by the Workshop for WebLogic Platform extension template. Identifies the JMS server as a system resource and targets it to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.
Commons-Logging Bridge	`wls-commonslogging-bridge#1.0@1.0`	Hooks commons-logging into the WLS logging mechanism.
Libraries Deployed	`beehive-netui-1.0#1.0@1.0`	Adds the Apache Beehive NetUI Version 1.0 libraries. These libraries support pageflow development, and depend upon the libraries contained in `struts-1.1.war` and `weblogic-beehive-1.0.ear`.
	`jstl#1.1@1.1.2`	Adds the Java standard tagging (JSTL) Version 1.1 libraries.
	`jsf-ri#1.1@1.1.1`	Adds the Java Server Faces Reference Implementation libraries.
	`jsf-myfaces#1.1@1.1.1`	Adds the Apache MyFaces libraries.
	`struts-1.1#1.1@1.0`	Adds the Apache Struts Version 1.1 libraries.
	`struts-1.2#1.2@1.0`	Adds the Apache Struts Version 1.2 libraries.

Table 16 Resources Configured in a BEA Workshop for WebLogic Platform Domain (Continued)

Resource Type	Name	Extension Result
	`weblogic-controls-10.0#10.0@10.0`	Adds the BEA Workshop for WebLogic controls extensions, including additional system controls (such as service control and timer control) as well as support for adding transactions, security, and message buffering to existing controls. Packaged for EARs.
	`weblogic-controls-10.0-war#10.0@10.0`	Adds the BEA Workshop for WebLogic Platform controls extensions including additional system controls (such as service control) as well as support for adding transactions, security, and message buffering to existing controls. Excludes those features which require EAR support such as timer control. Packaged for WARs.
	`beehive-controls-1.0#1.0@1.0`	Adds the Apache Beehive Controls 1.0.1 libraries to the domain. This includes the control runtime as well as the Beehive system controls - JdbcControl, JMSControl, and EJBControl.

Workshop for WebLogic 10.2 Extension Template

Using the Configuration Wizard or WLST, you can easily extend a base WebLogic Server domain to include the resources required for using Workshop for WebLogic 10.2. You accomplish this by adding the resources and services provided in the Workshop for WebLogic 10.2 template to a base WebLogic Server domain.

Note: Using the Configuration Wizard in graphical mode, you can easily create a new BEA Workshop for WebLogic Platform domain by checking the Workshop for WebLogic 10.2 check box in the **Select Domain Source** window. The result is the same as creating a base WebLogic Server domain first and then extending that domain with both the Workshop for WebLogic 10.2 extension template. For more information about the templates required to create a Workshop for WebLogic 10.2 domain, see "Relationships Between Templates" on page 7.

Generated Domain Output

The following table defines the default directory structure and files generated after applying the Workshop for WebLogic 10.2 template to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the *`BEA_HOME`*`\user_projects\domains\base_domain` directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 17 Base Domain After Applying the Workshop for WebLogic 10.2 Template

Directory	File	Description
`user_projects\applications\base_domain\`		
	n.a.	Directory serving as a placeholder for any custom application files that you create.
`user_projects\domains\base_domain\`		
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`URLs.dat`	File containing the URL for the JDBC database.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.

Table 17 Base Domain After Applying the Workshop for WebLogic 10.2 Template (Continued)

Directory	File	Description
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the Administration Server. For more information, see Domain Configuration Files in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).

Table 17 Base Domain After Applying the Workshop for WebLogic 10.2 Template (Continued)

Directory	File	Description
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`cgDataSource-jdbc.xml`	Global XA JDBC Data Source module for the domain configured for advanced web services.
	`cgDataSource-nonXA-jdbc.xml`	Global non-XA JDBC Data Source module for the domain configured for advanced web services.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.

Table 17 Base Domain After Applying the Workshop for WebLogic 10.2 Template (Continued)

Directory	File	Description
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 17 Base Domain After Applying the Workshop for WebLogic 10.2 Template (Continued)

Directory	File	Description

`security\`	`DefaultAuthenticatorInit.ldift`
	`DefaultRoleMapperInit.ldift`
	`XACMLRoleMapperInit.ldift`
		Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information.
		Note: WebLogic domains created with this release use the XACML providers, by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111

	`SerializedSystemIni.dat`	File containing encrypted security information.
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the Workshop for WebLogic 10.2 template.

Table 18 Resources Configured in a Workshop for WebLogic 10.2 template

Resource Type	Name	Extension Result
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `cgServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.
JDBC Data Source	`cgDataSource`	Defines an XA JDBC data source including its associated jdbc connection pool. The data source is named `cgDataSource`.
	`cgDataSource-nonXA`	Includes the JDBC data source and connection pool setups defined as `cgDataSource` in the domain and targets them to the correct server(s).
JDBC Store	`cgJMSStore`	Uses the JDBC store provided by the Workshop for WebLogic 10.2 extension template. The JDBC store is to be used with the JDBC data source, `cgDataSource-nonXA`, and the JMS server, `WseeJmsServer`, as a persistent store, and is targeted to the Administration Server, `AdminServer`.
JDBC System Resources	`cgDataSource` `cgDataSource-nonXA`	Identifies the JDBC data source and connection pool setups to be used for JDBC system.

Table 18 Resources Configured in a Workshop for WebLogic 10.2 template

Resource Type	Name	Extension Result
JMS Server	`WseeJmsServer`	Uses the JMS server provided by the Workshop for WebLogic 10.2 extension template. Identifies the JMS server as a system resource and targets it to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.
Commons-Logging Bridge	`wls-commonslogging-bridge#1.0@1.0`	Hooks commons-logging into the WLS logging mechanism.
Libraries Deployed	`beehive-netui-1.0#1.0@1.0`	Adds the Apache Beehive NetUI Version 1.0 libraries. These libraries support pageflow development, and depend upon the libraries contained in `struts-1.1.war` and `weblogic-beehive-1.0.ear`.
	`jstl#1.1@1.1.2`	Adds the Java standard tagging (JSTL) Version 1.1 libraries.
	`jsf-ri#1.1@1.1.1`	Adds the Java Server Faces Reference Implementation libraries.
	`jsf-myfaces#1.1@1.1.1`	Adds the Apache MyFaces libraries.
	`struts-1.1#1.1@1.0`	Adds the Apache Struts Version 1.1 libraries.
	`struts-1.2#1.2@1.0`	Adds the Apache Struts Version 1.2 libraries.

Table 18 Resources Configured in a Workshop for WebLogic 10.2 template

Resource Type	Name	Extension Result
	`weblogic-controls-10.0#10.0@10.0`	Adds the Workshop for WebLogic 10.2 controls extensions, including additional system controls (such as service control and timer control) as well as support for adding transactions, security, and message buffering to existing controls. Packaged for EARs.
	`weblogic-controls-10.0-war#10.0@10.0`	Adds the Workshop for WebLogic 10.2 controls extensions including additional system controls (such as service control) as well as support for adding transactions, security, and message buffering to existing controls. Excludes those features which require EAR support such as timer control. Packaged for WARs.
	`beehive-controls-1.0#1.0@1.0`	Adds the Apache Beehive Controls 1.0.1 libraries to the domain. This includes the control runtime as well as the Beehive system controls - JdbcControl, JMSControl, and EJBControl.

WebLogic Server Default Domain Extension Template

Using the Configuration Wizard or WLST, you can easily extend a base WebLogic Server domain to include resources required for a default WebLogic Server domain. You accomplish this by adding the resources and services provided in the WebLogic Server Default Domain extension template to a base WebLogic Server domain.

Note: Applying the WebLogic Server Default Domain extension template to a base WebLogic domain is a prerequisite to using the WebLogic Server Examples extension template. For information about the relationship between templates, see "Relationships Between Templates" on page 7.

For more information about the samples that are supported in the WebLogic Server Examples domain, see *Sample Application Examples and Tutorials for BEA WebLogic Server 10.0*.

Generated Domain Output

The following table defines the default directory structure and files generated after applying the WebLogic Server Default Domain extension template to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the *`BEA_HOME`*`\user_projects\domains\base_domain` directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 19 Base Domain After Applying the WebLogic Server Default Domain Extension Template

Directory	File	Description
`user_projects\applications\base_domain\`		
`server\docs\`	Various	Includes sub-directories containing style sheet and graphics files to support the online documentation.
`server\examples\build\`	Various	Includes WebLogic Server examples deployments.
`server\examples\src\`	Various	Includes source code and instructions for WebLogic Server examples.
`user_projects\domains\base_domain\`		
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`setExamplesEnv.cmd` `setExamplesEnv.sh`	Scripts that set up the environment to use the WebLogic Server Examples on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`startWebLogicEx.cmd` `startWebLogicEx.sh`	Scripts used to start the Administration Server for the WebLogic Server Examples domain on Windows and UNIX systems, respectively.

Table 19 Base Domain After Applying the WebLogic Server Default Domain Extension Template (Continued)

Directory	File	Description
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the Administration Server. For more information, see Domain Configuration Files in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."

Table 19 Base Domain After Applying the WebLogic Server Default Domain Extension Template (Continued)

Directory	File	Description
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`examples-demo-jdbc.xml`	Global non-XA JDBC Data Source module for the WebLogic Server default domain.
	`examples-demoXA-jdbc.xml`	Global XA JDBC Data Source module for the WebLogic Server default domain.
`config\jms\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.

Table 19 Base Domain After Applying the WebLogic Server Default Domain Extension Template (Continued)

Directory	File	Description
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 19 Base Domain After Applying the WebLogic Server Default Domain Extension Template (Continued)

Directory	File	Description

`security\`	`DefaultAuthenticatorInit.ldift`
	`DefaultAuthorizerInit.ldift`
	`DefaultRoleMapperInit.ldift`
	`XACMLAuthorizerInit.ldift`
	`XACMLRoleMapperInit.ldift`
		Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information.
		Note: WebLogic domains created with this release use the XACML providers by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111

	`SerializedSystemIni.dat`	File containing encrypted security information.
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the WebLogic Server Default Domain extension template.

Table 20 Resources Configured in a WebLogic Server Default Domain

Resource Type	Name	Extension Result
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `examplesServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.
Application Deployments	`ejb20BeanMgedEar`	Adds the application and targets it to the Administration Server, `AdminServer`.
	`examplesWebApp`	Adds the application and targets it to the Administration Server, `AdminServer`.
	`jdbcRowSetsEar`	Adds the application and targets it to the Administration Server, `AdminServer`.
	`jspSimpleTagEar`	Adds the application and targets it to the Administration Server, `AdminServer`.
	`mainWebApp`	Adds the application and targets it to the Administration Server, `AdminServer`.
	`webappCachingEar`	Adds the application and targets it to the Administration Server, `AdminServer`.
	`webservicesJwsSimpleEar`	Adds the application and targets it to the Administration Server, `AdminServer`.
	`xmlBeanEar`	Adds the application and targets it to the Administration Server, `AdminServer`.

Table 20 Resources Configured in a WebLogic Server Default Domain (Continued)

Resource Type	Name	Extension Result
JDBC Data Sources	`examples-demo`	Identifies the JDBC data source as an `examples-demo` system resource.
	`examples-demoXA`	Identifies the JDBC data source as an `examples-demoXA` system resource.
JDBC System Resources	`examples-demo` `examples-demoXA`	Identifies the JDBC data source and connection pool setups to be used for non-XA and XA JDBC system resources and targets them to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.

WebLogic Server Examples Extension Template

Using the Configuration Wizard or WLST, you can easily extend a base WebLogic Server domain to create a WebLogic Server Examples domain. You accomplish this by adding the resources and services provided in both the WebLogic Server Default and WebLogic Server Examples extension templates to a base WebLogic Server domain.

For more information about the samples that are supported in the WebLogic Server Examples domain, see *Sample Application Examples and Tutorials for BEA WebLogic Server 10.0*.

Generated Domain Output

The WebLogic Server Examples domain contains a collection of examples that illustrate best practices for coding individual J2EE APIs, and a set of scripts to run those examples. Once the WebLogic Server Default extension template has been applied to a base domain, applying the WebLogic Server Examples extension template allows you to create the WebLogic Server Examples domain. See "Relationships Between Templates" on page 7 for more details.

Table 21 Base Domain After Applying the WebLogic Server Default and WebLogic Server Examples Extension Templates

Directory	File	Description
`user_projects\applications\base_domain\`		
`server\`	`wls_samples_overview.html`	File that opens the WebLogic Server examples online documentation viewer.
`server\docs\`	Various	Directory and files supporting the WebLogic Server examples online documentation viewer.
`server\examples\build\`	Various	Includes sub-directories containing various Java and XML files used to build and work with WebLogic Server examples.
`server\examples\src\`	Various	Includes sub-directories containing various Java, XML, and HTML files used to work with WebLogic Server examples.
`user_projects\domains\base_domain\`		
	`client2certs.pem` `clientkey.pem`	Demo certificate and keystore files.
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`setExamplesEnv.cmd` `setExamplesEnv.sh`	Scripts that set up the environment to use the WebLogic Server Examples on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`startWebLogicEx.cmd` `startWebLogicEx.sh`	Scripts used to start the Administration Server for the WebLogic Server Examples domain on Windows and UNIX systems, respectively.

Table 21 Base Domain After Applying the WebLogic Server Default and WebLogic Server Examples Extension Templates (Continued)

Directory	File	Description
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the Administration Server. For more information, see Domain Configuration Files in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."

Table 21 Base Domain After Applying the WebLogic Server Default and WebLogic Server Examples Extension Templates (Continued)

Directory	File	Description
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`examples-demo-jdbc.xml`	Global non-XA JDBC Data Source module for the WebLogic Server Examples domain.

	`examples-demoXA-2-jdbc.xml`
	`examples-demoXA-jdbc.xml`
	`examples-multiDataSource-demoXAPool-jdbc.xml`
	`examples-oracleXA-jdbc.xml`
		Global XA JDBC Data Source modules for the WebLogic Server Examples domain.

`config\jms\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
	`examples-jms.xml`	Global JMS module for the WebLogic Server Examples domain.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java Virtual Machine starts.

Table 21 Base Domain After Applying the WebLogic Server Default and WebLogic Server Examples Extension Templates (Continued)

Directory	File	Description
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.

Table 21 Base Domain After Applying the WebLogic Server Default and WebLogic Server Examples Extension Templates (Continued)

Directory	File	Description
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

`security\`	`DefaultAuthenticatorInit.ldift`
	`DefaultAuthorizerInit.ldift`
	`DefaultRoleMapperInit.ldift`
	`XACMLAuthorizerInit.ldift`
	`XACMLRoleMapperInit.ldift`
		Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information.
		Note: WebLogic domains created with this release use the XACML providers, by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111

	`SerializedSystemIni.dat`	File containing encrypted security information.

Table 21 Base Domain After Applying the WebLogic Server Default and WebLogic Server Examples Extension Templates (Continued)

Directory	File	Description
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.
`WseeFileStore\`		Directory to be used for the file store for system resources.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the WebLogic Server Examples extension template.

Table 22 Resources Configured in a WebLogic Server Examples Domain

Resource Type	Name	Extension Result
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `examplesServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.

Table 22 Resources Configured in a WebLogic Server Examples Domain (Continued)

Resource Type	Name	Extension Result
Application Deployments	`ejb20BeanMgedEar`	Uses the application provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.
	`examplesWebApp`	Uses the application provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.
	`jdbcRowSetsEar`	Uses the application provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.
	`jspSimpleTagEar`	Uses the application provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.
	`mainWebApp`	Uses the application provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.
	`SamplesSearchWebApp`	Adds the application and targets it to the Administration Server, `AdminServer`.
	`webappCachingEar`	Uses the application provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.
	`webservicesJwsSimpleEar`	Uses the application provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.
	`xmlBeanEar`	Uses the application provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.

Table 22 Resources Configured in a WebLogic Server Examples Domain (Continued)

Resource Type	Name	Extension Result
File Store	`WseeFileStore`	Adds the file store to be used as the persistent store for the JMS server, `WseeJMSServer`, and the SAF Agent, `ReliableWseeSAFAgent`, and targets the store to the Administration Server, `AdminServer`.
JDBC Data Sources	`examples-demo` `examples-demoXA`	Uses the non-XA and XA JDBC data sources provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.
	`examples-oracleXA`	Adds the XA JDBC data source and targets it to the Administration Server, `AdminServer`.
	`examples-demoXA-2`	Adds the XA JDBC data source and targets it to the Administration Server, `AdminServer`.
	`examples-multiDataSource-demoXAPool`	Adds the XA JDBC multi data source and targets it to the Administration Server, `AdminServer`. Maps to `examples-demoXA` and `examples-demoXA-2` data sources.
JDBC Store	`exampleJDBCStore`	Adds the JDBC store to be used as the persistent store for the JDBC data source, `examples-demo`, and the JMS server, `examplesJMSServer`, and targets the store to the Administration Server, `AdminServer`.

Table 22 Resources Configured in a WebLogic Server Examples Domain (Continued)

Resource Type	Name	Extension Result
JDBC System Resources	`examples-demo` `examples-demoXA`	Uses the JDBC data source and connection pool setups provided by the WebLogic Server Default extension template applied to the base WebLogic Server domain.

	`examples-demoXA-2`
	`examples-oracleXA`
	`examples-multiDataSource-demoXAPool`
		Adds the JDBC data source and connection pool setups and targets them to the Administration Server, `AdminServer`.

JMS System Resources	`examples-jms`	Identifies the JMS servers, connection factories, queues, and topics to be used for JMS system resources.

JMS Connection Factories	`exampleTopic`
	`exampleTrader`
	`weblogic.examples.jms.QueueConnectionFactory`
		Adds the JMS connection factories as `examples-jms` system resources and targets them to the Administration Server, `AdminServer`.

JMS Servers	`examplesJMSServer`	Adds the JMS server as an `examples-jms` system resource and targets it to the Administration Server, `AdminServer`.
	`WseeJMSServer`	Adds the JMS server as an `examples-jms` system resource and targets it to the Administration Server, `AdminServer`.
JMS Queues	`exampleQueue`	Adds the JMS queue to the JMS server, `examplesJMSServer`.
	`jms/MULTIDATASOURCE_MDB_QUEUE`	Adds the JMS queue to the JMS server, `examplesJMSServer`.
	`weblogic.wsee.wseeExamplesDestinationQueue`	Adds the JMS queue to the JMS server, `WseeJMSServer`.

Table 22 Resources Configured in a WebLogic Server Examples Domain (Continued)

Resource Type	Name	Extension Result
JMS Topics	`exampleTopic`	Adds the JMS topic to the JMS server, `examplesJMSServer`.
	`quotes`	Adds the JMS topic to the JMS server, `examplesJMSServer`.
SAF Agent	`ReliableWseeSAFAgent`	Adds the SAF agent and targets it to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.

WebLogic Integration BPM Extension Template

Using the Configuration Wizard, you can extend a base WebLogic Server Domain to create a WebLogic Integration Domain. You accomplish this by adding the resources and services provided in the WebLogic Integration.

Generated Domain output

The following table defines the default directory structure and files generated after applying the WebLogic Integration Extension template to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the *BEA_HOME*`\user_projects\domains\base_domain` directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 23 Output Generated from WebLogic Integration BPM Extension Template

Directory	File	Description
`user_projects\domains\base_domain\`		
		Directory serving as a placeholder for any custom application files that you create
`user_projects\domains\base_domain\`		

Table 23 Output Generated from WebLogic Integration BPM Extension Template (Continued)

	`fileRealm.properties`	File containing ACL, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`apacheLog4jCfg.xml`	File specifying the WebLogic Integration BPM logging configuration and logging levels for WebLogic Server
	`wli-config.properties`	File containing domain-specific parameters that are used by Business Processes.
	`workshop.properties`	File containing domain-specific parameters for compatibility with 8.1 Web Services Stack.
	`jws-config.properties`	File containing Business Process Management-related JMS information used for compatibility with 8.1 Web Services Stack.
	`URLs.dat`	File containing the URL for the Pointbase database
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.

Table 23 Output Generated from WebLogic Integration BPM Extension Template (Continued)

	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the Administration Server. For more information, see Domain Configuration Files in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is then used for staging an application when the staging mode of the application is staged.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is then used for staging an application when the staging mode of the application is staged.
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`cgDataSource-jdbc.xml`	Global XA JDBC Data Source module for the domain configured for advanced Web services and Business Process Management
	`cgDataSource-nonXA-jdbc.xml`	Global non-XA JDBC Data Source module for the domain configured for Business Process Management.
	`bpmArchDataSource-jdbc.xml`	Global XA JDBC Data Source module for the domain configured for Business Process management archiving.

Table 23 Output Generated from WebLogic Integration BPM Extension Template (Continued)

`config\jms\`	`conversational-jms.xml`	Global JMS module for the domain configured for Business Process Management.
	`wseejmsmodule-jms.xml`	Global JMS module for the domain configured for advanced Web Services.
`config\lib\`	`readme.txt`	
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.

Table 23 Output Generated from WebLogic Integration BPM Extension Template (Continued)

`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.
`security\`	`DefaultAuthorizerInit.ldift` `DefaultAuthenticatorInit.ldift` `DefaultRoleMapperInit.ldift` `XACMLRoleMapperInit.ldift` `XAMLAuthorizerInit.ldift`	Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information. **Note:** WebLogic domains created with this release use the XACML providers, by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111.
	`SerializedSystemIni.dat`	File containing encrypted security information.
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.
`WseeFileStore\`		Directory to be used for the file store for system resources.

Table 23 Output Generated from WebLogic Integration BPM Extension Template (Continued)

`wliconfig\`		Directory to be used for the file that stores Event Generator and Application Integration information.
	`AIConfiguration.xml`	File containing information about the Application views created in the domain.
	`EmailEventGen.xml`	File containing information about the email Event Generators created in the domain.
	`FileEventGen.xml`	File containing information about the file Event Generators created in the domain.
	`HttpEventGen.xml`	File containing information about the http Event Generators created in the domain.
	`JMSEventGen.xml`	File containing information about the jms Event Generators created in the domain.
	`MQEventGen.xml`	File containing information about the mq Event Generators created in the domain.
	`RDBMSEventGen.xml`	File containing information about the rdbms Event Generators created in the domain.
	`TibRVEventGen.xml`	File containing information about the tibcoRv Event Generators created in the domain.
	`TimerEventGen.xml`	File containing information about the timerEvent Generators created in the domain.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the WebLogic Integration BPM extension template. Prior to extending the domain, the WebLogic Integration

BPM extension Template requires the WebLogic Advanced Web Services (`wls_webservices`) and Workshop for WebLogic extension (`workshop_wl`) template to be applied on the domain.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

Resource Type	Name	Extension Result
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 1-18
Application Deployments	`WLI System EJBs`	WLI System EAR adds the following EJBs Module: • `wliadmin`: Adds WLI Admin EJB module • `adminhelper` :Adds WLI Admin Helper EJB module • `tracking`: Adds WLI Process Tracking EJB module • `proxydispatcher`: Adds WLI Process Proxy Dispatcher EJB module • `wlai-processors-ejb.jar`: Adds WLI AI Message Processors EJB module • `wlai-rarupload-ejb.jar`: Adds WLI AI RAR Upload EJB module (upload utilities for AI) • `rosettanet`: Adds WLI RosettaNet EJB module • `ebxml` : Adds WLI ebXML EJB module • `message-tracking` : Adds WLI Message Tracking EJB module • `transport/responsehandler`: Adds Sync/Async Transport EJB module • Adds the following Web applications: – `sync2AsyncIM`: Adds the Sync Async Web Application with module name transport/http – `TransportServlet` : Adds B2B HTTP Transport Web Application with module name b2btransport-webapp
	`WLI Console`	Adds the WLI BPM Console Web application.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

	`wlai-designtime`	Defines the Application View design console needed to define new Application Views and publish them to a Workshop application.
	`B2BDefaultWebAppApplication`	Adds a web application for B2B messages to work, targets to AdminServer and Cluster.
	`JWSQueueTransport`	Adds the KNEX.bean.QueueTransport Message Driven Bean used by the Business Process to support JMS Transport
	`rdbmsEG_ear`	Adds WLI RDBMSEG EAR for creating and managing rdbms Event Generators.
	`tibRVEG_ear`	Adds WLI TIBCORVEG EAR for creating and managing tibcoRV Event Generators.
	`httpEG_ear`	Adds WLI HTTPEG EAR for creating and managing http Event Generators.
	`mqEG_ear`	Adds WLI MQEG EAR for creating and managing MQ Event Generators.
`FileStore`	`WseeFileStore`	Adds the file store to be used as the persistent store for the JMS server, `WseeJMSServer`, and the SAF Agent, `ReliableWseeSAFAgent`, and targets the store to the Administration Server, `AdminServer`.
	`conversational-jms`	Uses the JMS system resources provided by the WebLogic Integration BPM extension template.
`JDBC Data Source`	`cgDataSource`	Uses the XA JDBC data source provided by the Workshop for WebLogic extension template. Identifies the XA JDBC data source as a `cgDataSource` system resource.
	`cgDataSource-nonXA`	Uses the non-XA JDBC data source provided by the Workshop For WebLogic extension template. Identifies the non-XA JDBC data source as a `cgDataSource-nonXA` system resource.
`JDBC Store`	`cgJMSStore`	Uses the JDBC store provided by the WebLogic Integration BPM extension template. The JDBC store is to be used with the JDBC data source, `cgDataSource-nonXA`, and the JMS server, `cgJMSServer`, as a persistent store, and is targeted to the Administration Server, `AdminServer`.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

JDBC System Resources	cgDataSource cgDataSource-nonXA	Uses the JDBC data source and connection pool setups provided by the Workshop For WebLogic extension template. These JDBC system resources are targeted to the Administration Server, `AdminServer`.
	samplesDataSource	Uses the JDBC data source and connection pool setups provided by the Workshop For WebLogic extension template. These JDBC system resources are targeted to the Administration Server, `AdminServer`.
SAF Agent	ReliableWseeSAFAgent	Adds the SAF agent and targets it to the Administration Server, `AdminServer`.
JMS Connection Factory	cgQueue	Uses the JMS connection factory provided by the WebLogic Integration BPM extension template with JNDI Name `weblogic.jws.jms.QueueConnectionFactory` Identifies the JMS connection factory as a `conversational-jms` system resource and targets it to the Administration Server, `AdminServer`.
	com.bea.wli.b2b.server.TopicConnectionFactory	JMS connection factory for B2B events, targeted to AdminServer.
	wli.internal.egrdbms.XAQueueConnectionFactory	Uses the JMS connection factory provided by the WebLogic Integration BPM extension template with JNDI Name `wli.internal.egrdbms.XAQueueConnectionFactory`. Identifies the JMS connection factory as a conversational-jms system resource and targets it to the AdminServer
WS Reliable Delivery Policy	RMDefaultPolicy	Adds the default Reliable Message Policy provided by the WebLogic Integration BPM extension.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

JMS Queues	`cgJWSQueue`	Adds the JMS queue with JNDI name jws,queue provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wlwJWSBuffer`	Adds the JMS queue with JNDI name jws.bufferqueue provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wlwJWSErrors`	Adds the JMS queue with JNDI name jws.errorqueue provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`WSInternaljms.internal.queue.WSStoreForwardQueuecgServer`	Adds the JMS queue with local JNDI name `WSInternaljms.internal.queue.WSStoreForwardQueuecgServer` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `WSStoreForwardInternalJMSServercgServer`.
	`WSInternaljms.internal.queue.WSDupsEliminationHistoryQueuecgServer`	Adds the JMS queue with local JNDI name `WSInternaljms.internal.queue.WSDupsEliminationHistoryQueuecgServer` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `WSStoreForwardInternalJMSServercgServer`.
	`WSInternaljms.internal.queue.WSDupsEliminationMessageQueuecgServer`	Adds the JMS queue with local JNDI name `WSInternaljms.internal.queue.WSDupsEliminationMessageQueuecgServer` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, WSStoreForwardInternalJMSServercgServer.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

	`wli.internal.b2b.rosettanetencoder.queue`	Adds the JMS queue for outbound RosettaNet messages with local JNDI name `wli.internal.b2b.rosettanetencoder.queue` and targets it to `cgJMSServer`.
	`wli.b2b.mt.event.stream_error`	Adds the JMS queue for reporting (archiving) errors with local JNDI name `wli.b2b.mt.event.stream_error` and targets it to `cgJMSServer`.
	`wli.internal.msgtracking.queue`	Adds the internal JMS queue with JNDI name `wli.internal.msgtracking.queue` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.process.event.stream_error`	Adds the JMS queue with JNDI name `wli.process.event.stream_error` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.scheduling.queue`	Uses the JMS queue with JNDI name `wli.internal.scheduling.queue` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.scscheduling.queue_error`	Adds the internal JMS queue with JNDI name `wli.process.event.stream_error` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.ai.event_suspend`	A queue that holds any event that was accepted from an AI event generator, but was suspended because of an ApplicationView being suspended. This queue is drained when the ApplicationView is resumed.
	`wli.internal.egfile.queue`	Adds the internal JMS queue with JNDI name `wli.internal.egfile.queue` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a conversational-jms system resource and targets it to the JMS server, `cgJMSServer`. Used by file Event Generator for holding events.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

	`wli.b2b.failedmessage.queue`	Adds the JMS queue for failed B2B messages with JNDI name `wli.b2b.failedmessage.queue` and targets it to cgJMSServer.
	`wli.internal.egmq.queue`	Adds the internal JMS queue with JNDI name wli.internal.egmq.queue provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a conversational-jms system resource and targets it to the JMS server, `cgJMSServer`. Used by mq EG for holding events.
	`wli.internal.instance.info.buffer_error`	Adds the JMS queue with JNDI name `wli.internal.instance.info.buffer_error` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.b2b.ebxmlencoder.queue`	Adds the JMS queue for outbound EBXML messages with local JNDI name `wli.internal.b2b.ebxmlencoder.queue` and targets it to cgJMSServer.
	`wli.internal.tracking.buffer_error`	Adds the JMS queue with JNDI name `wli.internal.tracking.buffer_error` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.instance.info.buffer`	Adds the JMS queue with JNDI name `wli.internal.instance.info.buffer` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.egmail.queue`	Adds the internal JMS queue with JNDI name wli.internal.egmail.queue provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`. Used by email Event Generator for holding events.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

	`wli.internal.egrdbms.queue`	Adds the internal JMS queue with JNDI name `wli.internal.egrdbms.queue` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`. Used by rdbms Event Generator for holding events.
	`wli.b2b.mt.event.stream`	Adds the JMS queue for archiving of the B2B message tracking data with local JNDI name `wli.b2b.mt.event.stream` and targets it to `cgJMSServer`.
	`wli.internal.configfile.request.queue`	Adds the JMS queue with JNDI name `wli.internal.configfile.request.queue` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.process.event.stream`	Adds the JMS queue with JNDI name `wli.process.event.stream` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.sync2Async.soapResponse`	Adds the JMS queue with JNDI name `wli.internal.sync2Async.soapResponse` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.ai.async.response`	Holds async response messages for ApplicationView services that were invoked asynchronously from a client-side Java client. ApplicationView control clients will have their asynchronous responses dispatched directly to them.
	`wli.internal.egtimer.queue`	Adds the internal JMS queue with JNDI name `wli.internal.egtimer.queue` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a conversational-jms system resource and targets it to the JMS server, `cgJMSServer`. Used by timer EG for holding events.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

	`wli.internal.SQLStore.cleanup.documents`	Adds the JMS queue with JNDI name `wli.internal.SQLStore.cleanup.documents` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.ai.async.request`	A queue to hold asynchronous service requests on ApplicationViews. This queue is drained by the `WLI AI Message Processors` application (via AsyncRequestProcessorMDB)
	`wli.internal.tracking.buffer`	Adds the JMS queue with JNDI name `wli.internal.tracking.buffer` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.sample.egjms.queue`	Adds the JMS queue with JNDI name wli.sample.egjms.queue provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
	`wli.internal.b2b.events.topic`	Adds a JMS topic for B2B events with JNDI name `wli.internal.b2b.events.topic` and targets it to `cgJMSServer`.
	`wli.internal.ai.event`	A topic to distribute AI ApplicationView events to client-side Java clients. JPD clients receive events via MessageBroker channels, and not this topic.
	`wli.internal.configfile.update.topic`	Adds the JMS Topic with JNDI name `wli.internal.configfile.update.topic` provided by the WebLogic Integration BPM extension template. Identifies the JMS queue as a `conversational-jms` system resource and targets it to the JMS server, `cgJMSServer`.
JMS Server	`cgJMSServer`	Adds the JMS server provided by the WebLogic Integration BPM extension template. Identifies the JMS server as a `conversational-jms` system resource and targets it to the Administration Server, `AdminServer`.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

	`WSStoreForwardInternalJMSServercgServer`	Adds the JMS server provided by the WebLogic Integration BPM extension template. Identifies the JMS server as a `conversational-jms` system resource and targets it to the Administration Server, `AdminServer`.
	`WseeJmsServer`	Adds the JMS server provided by the WebLogic Advanced Web Services extension. Identifies the JMS Server as a `wseejmsmodule-jms` system resource and targets it to the Administration Server, `AdminServer`.
Work Manager	`weblogic.wsee.mdb.DispatchPolicy`	Adds an optional and configurable global Work Manager for internal WebLogic Server applications
Startup Classes	WLI Startup Class	Adds the WLI Startup class. The server instance runs it at startup after activating JMS and JDBC services and before activating applications and EJBs.
	WLI Post-Activation Startup Class	Adds the WLI Post-Activation Startup class. The server instance runs it at startup after activating JMS and JDBC services, EJBs, and applications.
Shutdown Classes	`WLI Shutdown Class`	Adds the WLI Shutdown Class. The server instance invokes it when it shuts down.
JMS System Resources	`conversational-jms`	Uses the JMS system resources provided by the WebLogic Integration BPM extension template.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.
Commons-Logging Bridge	`wls-commonslogging-bridge#1.0@1.0`	Hooks commons-logging into the WLS logging mechanism. Packaged for EARs.
	`wls-commonslogging-bridge-war#1.0@1.0`	Hooks commons-logging into the WLS logging mechanism. Packaged for WARs. This is not a WebLogic Integration Artifact.
Libraries Deployed	`beehive-netui-1.0#1.0@1.0`	Adds the Apache Beehive NetUI Version 1.0 libraries. These libraries support pageflow development, and depend upon the libraries contained in `struts-1.1.war` and `weblogic-beehive-1.0.ear`.

Table 24 Resources and Services Configured in WebLogic BPM Extension Template

	`beehive-netui-resources-1.0#1.0@1.0`	Adds the Apache Beehive NetUI ResourceVersion 1.0 libraries. This is a WebLogic Workshop artifact.
	`jstl#1.1@1.1.2`	Adds the Java standard tagging (JSTL) Version 1.1 libraries. (note: not a WLI artifact, WLW team forgot to mention it).
	`jsf-ri#1.1@1.1.1`	Adds the Java Server Faces Reference Implementation libraries.
	`jsf-myfaces#1.1@1.1.1`	Adds the Apache MyFaces libraries.
	`struts-1.1#1.1@1.0`	Adds the Apache Struts Version 1.1 libraries.
	`struts-1.2#1.2@1.0`	Adds the Apache Struts Version 1.2 libraries.
	`weblogic-controls-1.0#1.0@1.0`	Adds the BEA Workshop for WebLogic controls extensions including additional system controls (such as service control and timer control) as well as support for adding transactions, security, and message buffering to existing controls. Packaged for EARs.
	`weblogic-controls-1.0-war#1.0@1.0`	Adds the BEA Workshop for WebLogic Platform controls extensions including additional system controls (such as service control) as well as support for adding transactions, security, and message buffering to existing controls. Excludes those features which require EAR support, such as timer control. Packaged for WARs.
	`beehive-controls-1.0#1.0@1.0`	Adds the Apache Beehive Controls 1.0.1 libraries to the domain. This includes the control runtime as well as the Beehive system controls - JdbcControl, JMSControl, and EJBControl. Packaged for EARs.
	`beehive-controls-1.0-war#1.0@1.0`	Adds the Apache Beehive Controls 1.0.1 libraries to the domain. This includes the control runtime as well as the Beehive system controls - JdbcControl, JMSControl, and EJBControl.Packaged for WARs. This is not a WebLogic Integration Artifact.

WebLogic Integration Worklist Extension Template

Generated Domain Output

Base Integration Domain resulting from a Base WebLogic Server domain extended with the WebLogic Integration Worklist (requires also extending with WebLogic Personalization extension template and Workshop for WebLogic Extension template).

Table 25 Output Generated from WebLogic Integration BPM Extension Template

Directory	File	Description
`user_projects\domains\base_domain\`	`bin\setDomainEnv.cmd`	Added `%WL_HOME%\integration\lib\worklist-system-required.jar` to the `POST_CLASSPATH` variable
	`security\XACMLAuthorizerInit.ldift`	Adds global Worklist system and task plan policies. This information is initialized upon the first start of the server. It can be loaded into a default (LDAP) authenticator, or a SQL Authenticator.
	`Security\XACMLAuthenticatorInit.ldift`	Adds some useful group and role definitions, for example, IntegrationMonitors, IntegrationDeployers, and so on.

	`jdbc/portalDataSource-jdbc.xml`
	`jdbc/p13nDataSource-jdbc.xml`
	`jdbc/cgDataSource-jdbc.xml`
		When portalDataSource and p13nDataSource use a non-XA driver (determined by checking against a list of known non-XA drivers), the Worklist extension template combines them with cgDataSource to avoid having multiple non-XA resources enlisted in a single XA transaction. The combine logic takes JNDI names from portalDataSource and/or p13nDataSource, adds them to cgDataSource, and deletes portalDataSource and/or p13nDataSource.

Resources and Services Configured

TheTable 26 identifies the resources and services configured in a domain extended with the WebLogic Integration Worklist Extension template. The WebLogic Integration Worklist Extension Template requires the Personalization Extension template (p13n) and Workshop for WebLogic Extension template.

Table 26 Resources and Services Configured

Resource Type	Name	Notes
Application Deployment	`Worklist_Console`	This application deploys the Worklist administrative console. `Cluster Targets: <admin server>`
	`worklist-admin`	Administrative initialization services used by all Worklist apps. `Cluster Targets: <admin server>`
	`worklist-ejbs-worksub`	WorkSubstituteManager EJB for use by all Worklist apps. `Cluster Targets: <cluster>`
	`calendar-ejbs`	Calendar services for use by all Worklist applications and timer event generators. Cluster Targets: <admin server>, <cluster> **Note:** This application is targeted at the admin server in a cluster to allow it to initialize in a one-time-only fashion, critical calendar resources in the runtime datastore. The actual file deployed in the server depends on the type of database being used. For non-Oracle databases, `calendar-ejbs-generic.ear` is used. For Oracle, `calendar-ejbs-oracle.ear` is used.
Libraries		

Table 26 Resources and Services Configured (Continued)

	`wli-calendar#10.0@10.0.0`	Optional package (no J2EE app resources) containing calendar API and implementation.
	`worklist-web-lib#10.0@10.0.0`	Worklist web library for Worklist user portal. Include this library in weblogic.xml to include a user portal in your web application.
	`worklist-client#10.0@10.0.0`	Optional package (no J2EE app resources) that defines the public API (and implementation) for the Worklist API.
	`wli-util#10.0@10.0.0`	Optional package containing WLI utility classes.
	`wli-util-datatype#10.0@10.0.0`	Optional package containing the WLI data type abstraction packages (used by Worklist runtime).
	`worklist-ejbs-lib#10.0@10.0.0`	Worklist application library. Include this library from weblogic-application.xml to include a Worklist system instance in your application. The actual file deployed in the server depends on the type of database being used. For non-Oracle databases, we deploy worklist-ejbs-generic.ear. For Oracle, we deploy worklist-ejbs-oracle.ear.
	`wli-worklist-common-console-1.0#1.0@1.0`	Common console utilities used by the Worklist management console.
	`wli-worklist-webapp-1.0#1.0@1.0`	
	`wlp-framework-struts-1.2-web-lib#10.2.0@10.2.0`	Portal framework library used by the Worklist user portal.
	`wlp-framework-common-web-lib#10.2.0@10.2.0`	Portal framework library used by the Worklist user portal.

Table 26 Resources and Services Configured (Continued)

	`wlp-light-web-lib#10.2.0@10.2.0`	Portal framework library used by the Worklist user portal.
	`wlp-lookandfeel-web-lib#10.2.0@10.2.0`	Portal framework library used by the Worklist user portal.
JMS Server	`cgJMSServer`	A JMS Server used by Worklist for timer and event message handling.
JDBC JMS Store	`cgJMSStore`	JMS store used by `cgJMSServer.`
JDBC DataSource		Depends on cgDataSource from Workshop template. See 'Generated Domain Output' section above for logic Worklist uses to 'combine' dataSources from p13n and portal products in the non-XA case.
Queue Connection Factory	`<app name>/WorklistQueueConnectionFactory`	Per-Worklist application queue connection factory, used to interact with the per-app Worklist queues.

JMS Resources	`<app name>/WorklistTimerQueue`
	`<app name>/WorklistEventQueue`
		Per-Worklist application queues used for timer and event, respectively, message delivery. Customers never use these queues directly.

WebLogic Integration Worklist (Compatibility) Extension Template

Generated Domain Output

Base Integration Domain resulting from a Basic WebLogic Server domain extended with the WebLogic Integration Worklist (Compatibility) template. The compatibility template also requires extending with WebLogic Integration Worklist extension template and all the template it depends on.

This template is used to add 8.1.x backward-compatibility to a 10.2 Worklist domain. By default, the applications and other resources needed to support backward compatibility are not deployed to a WebLogic

Integration domain. You can also apply the Worklist (Compatibility) extension to a domain to add this support. This template is named `wli_worklist81x.jar`.

Table 27 Generated Domain Output for Compatibility Extension Template

Directory	File	Description
`user_projects\domains\base_domain\`	`bin\setDomainEnv.cmd`	Adds `%WL_HOME%\integration\lib\worklist-system-required.jar` to the POST_CLASSPATH variable
	`security\XACMLAuthorizerInit.ldift`	Adds Worklist system policies for the worklist-ejbs-81x application and policies for the Compatibility 8.1.x task plan. This information is initialized upon the first start of the server. It can be loaded into a Default (LDAP) authenticator, or a SQL Authenticator.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the WebLogic Integration Worklist extension template. The WebLogic Integration Worklist extension Template requires the Personalization Extension template (p13n) and Workshop for WebLogic Extension template.

Table 28 Resources and Services Configured

Resource Type	Name	Notes
Application Deployment	`worklist-admin-81x`	Performs administrative initialization specific to Worklist backward compatibility. This is only deployed to a single-server, or the admin server in a cluster domain.

Table 28 Resources and Services Configured

	`worklist-ejbs-81x`	Defines the WorklistManager EJB and Worklist portal needed for backward compatibility. This is only deployed to a single-server or the cluster in a cluster domain.
	`calendar-ejbs-81x`	Defines the UserInfo EJB needed for backward compatibility. This is only deployed to a single-server or the cluster in a cluster domain.

WebLogic Portal Extension Template

Using the Configuration Wizard or WLST, you can extend a Basic WebLogic Server domain to include resources required for a WebLogic Portal Extension domain. You can accomplish this by adding the resources and services provided in the WebLogic Portal Extension template to a base WebLogic Server domain. The Basic WebLogic Server domain is extended with WebLogic Advanced Web Services and BEA Workshop for WebLogic Extension. The compatibility template also requires extending with WebLogic Personalization extension template and Workshop for WebLogic 10.2.

Note: Using the Configuration Wizard in graphical mode, you can easily create a new WebLogic Portal domain by checking the WebLogic Portal check box in the **Select Domain Source** window. The result is the same as creating a Basic WebLogic Server domain first and then extending that domain with extension templates for WebLogic Portal extension. For more information about the templates required to create a WebLogic Portal domain, see "Relationships Between Templates" on page 7.

Generated Domain Output

Table 29 defines the default directory structure and files generated after applying the WebLogic Portal Extension template to a Basic WebLogic Server domain. By default, the Configuration Wizard creates the domain in the `BEA_HOME\user_projects\domains\base_domain` directory. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 29 Generated Domain Output for WebLogic Portal Extension Template

Directory	File	Description
`user_projects\domains\base_domain\`		

Table 29 Generated Domain Output for WebLogic Portal Extension Template

	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`URLs.dat`	File containing the URL for the JDBC database.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.
`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.

Table 29 Generated Domain Output for WebLogic Portal Extension Template

`config\`	`config.xml`	File containing the configuration information used by the “Domain Configuration Files” Administration Server. For more information, see in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application’s staging mode is “staged.”
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).
`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`cgDataSource-jdbc.xml`	Global XA JDBC Data Source module for the domain configured for advanced Web services and Business Process Management
	`cgDataSource-nonXA-jdbc.xml`	Global non-XA JDBC Data Source module for the domain configured for Business Process Management.
`config\jms\`	readme.txt	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
	`wseejmsmodule-jms.xml`	Global JMS module for the domain configured for advanced Web Services.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server’s Java virtual machine starts.

Table 29 Generated Domain Output for WebLogic Portal Extension Template

`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.
`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 29 Generated Domain Output for WebLogic Portal Extension Template

`security\`	`DefaultAuthenticatorInit.ldift` `DefaultRoleMapperInit.ldift` `XACMLRoleMapperInit.ldift`	Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information. **Note:** WebLogic domains created with this release use the XACML providers, by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111
	`SerializedSystemIni.dat`	File containing encrypted security information.
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.
`WseeFileStore\`		Directory to be used for the file store for system resources.

Resources and Services Configured

The following table describes the resources and services configured in a domain that is extended with the WebLogic Portal Extension template.

Table 30 Resources and Services Configured

Resource Type	Name	Notes
Administration Server	`AdminServer`	Uses the Administration Server provided in the Basic WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `cgServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.
JDBC Data Source	`cgDataSource`	Defines an XA JDBC data source including its associated jdbc connection pool. The data source is named `cgDataSource`.
	`cgDataSource-nonXA`	Includes the JDBC data source and connection pool setups defined as `cgDataSource` in the domain and targets them to the correct server(s).

Table 30 Resources and Services Configured

JDBC Store	`cgJMSStore`	Uses the JDBC store provided by the BEA Workshop for WebLogic Platform extension template. The JDBC store is to be used with the JDBC data source, `cgDataSource-nonXA`, and the JMS server, `cgJMSServer`, as a persistent store, and is targeted to the Administration Server, `AdminServer`.
JDBC System Resources	`cgDataSource` `cgDataSource-nonXA`	Identifies the JDBC data source and connection pool setups to be used for JDBC system.
JMS Server	`cgJMSServer`	Uses the JMS server provided by the Workshop for WebLogic Platform extension template. Identifies the JMS server as a `conversational-jms` system resource and targets it to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.
Commons-Logging Bridge	`wls-commonslogging-bridge#1.0@1.0`	Hooks commons-logging into the WLS logging mechanism.

Table 30 Resources and Services Configured

Libraries Deployed	`beehive-netui-1.0#1.0@1.0`	Adds the Apache Beehive NetUI Version 1.0 libraries. These libraries support pageflow development, and depend upon the libraries contained in `struts-1.1.war` and `weblogic-beehive-1.0.ear`.
	`jstl-1.1#1.1@1.0`	Adds the Java standard tagging (JSTL) Version 1.1 libraries.
	`jsf-ri#1.1@1.1.1`	Adds the Java Server Faces Reference Implementation libraries.
	`jsf-myfaces#1.1@1.1.1`	Adds the Apache MyFaces libraries.
	`struts-1.1#1.1@1.0`	Adds the Apache Struts Version 1.1 libraries.
	`struts-1.2#1.2@1.0`	Adds the Apache Struts Version 1.2 libraries.

Table 30 Resources and Services Configured

	`weblogic-controls-1.0#1.0@1.0`	Adds the BEA Workshop for WebLogic controls extensions including additional system controls (such as service control and timer control) as well as support for adding transactions, security, and message buffering to existing controls. Packaged for EARs.
	`weblogic-controls-1.0-war#1.0@1.0`	Adds the BEA Workshop for WebLogic Platform controls extensions including additional system controls (such as service control) as well as support for adding transactions, security, and message buffering to existing controls. Excludes those features which require EAR support such as timer control. Packaged for WARs.
	`beehive-controls-1.0#1.0@1.0`	Adds the Apache Beehive Controls 1.0.1 libraries to the domain. This includes the control runtime as well as the Beehive system controls - JdbcControl, JMSControl, and EJBControl.

WebLogic Portal Collaboration Repository

Using the Configuration Wizard or WLST, you can extend a base WebLogic Server domain to include resources required for a WebLogic Portal Collaboration Repository domain. You can accomplish this by adding the resources and services provided in the WebLogic Portal Collaboration Repository template to a base WebLogic Server domain.

Generated Domain Output

WebLogic Portal Collaboration Repository results from a Base WebLogic Server domain extended with the WebLogic Portal Extension template. This template is named `wlp_groupspacedb.jar`.

The following table defines the default directory structure and files generated after applying the WebLogic Portal Collaboration Repository template to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the `BEA_HOME\user_projects\domains\base_domain directory`. If you modify the default configuration settings, the output directory structure may be different from the structure described here.

Table 31 Generated Domain Output for WebLogic Portal Collaboration Template

Directory	File	Description
`user_projects\domains\base_domain\`		
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`URLs.dat`	File containing the URL for the JDBC database.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.

Table 31 Generated Domain Output for WebLogic Portal Collaboration Template

`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the "Domain Configuration Files" Administration Server. For more information, see in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).

Table 31 Generated Domain Output for WebLogic Portal Collaboration Template

`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`cgDataSource-jdbc.xml`	Global XA JDBC Data Source module for the domain configured for advanced Web services and Business Process Management
	`cgDataSource-nonXA-jdbc.xml`	Global non-XA JDBC Data Source module for the domain configured for Business Process Management.
config\jms\	readme.txt	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
	`wseejmsmodule-jms.xml`	Global JMS module for the domain configured for advanced Web Services.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server’s Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain’s current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.

Table 31 Generated Domain Output for WebLogic Portal Collaboration Template

`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 31 Generated Domain Output for WebLogic Portal Collaboration Template

`security\`	`DefaultAuthenticatorInit.ldift` `DefaultRoleMapperInit.ldift` `XACMLRoleMapperInit.ldift`	Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information. **Note:** WebLogic domains created with this release use the XACML providers by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111
	`SerializedSystemIni.dat`	File containing encrypted security information.
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.
`WseeFileStore\`		Directory to be used for the file store for system resources.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the WebLogic Portal Collaboration Repository template.

Table 32 Resources and Services Configured

Resource Type	Name	Notes
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `cgServer`. For information about naming the Administration Server during domain creation, see “Resources and Services Configured for WebLogic Server Domain Template” on page 18.
JDBC Data Source	`cgDataSource`	Defines an XA JDBC data source including its associated jdbc connection pool. The data source is named `cgDataSource`.
	`cgDataSource-nonXA`	Includes the JDBC data source and connection pool setups defined as `cgDataSource` in the domain and targets them to the correct server(s).

Table 32 Resources and Services Configured

JDBC Store	`cgJMSStore`	Uses the JDBC store provided by the BEA Workshop for WebLogic Platform extension template. The JDBC store is to be used with the JDBC data source, `cgDataSource-nonXA`, and the JMS server, `cgJMSServer`, as a persistent store, and is targeted to the Administration Server, `AdminServer`.
JDBC System Resources	`cgDataSource` `cgDataSource-nonXA`	Identifies the JDBC data source and connection pool setups to be used for JDBC system.
JMS Server	`cgJMSServer`	Uses the JMS server provided by the Workshop for WebLogic Platform extension template. Identifies the JMS server as a `conversational-jms` system resource and targets it to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.
Commons-Logging Bridge	`wls-commonslogging-bridge#1.0@1.0`	Hooks commons-logging into the WLS logging mechanism.

Table 32 Resources and Services Configured

Libraries Deployed	`beehive-netui-1.0#1.0@1.0`	Adds the Apache Beehive NetUI Version 1.0 libraries. These libraries support pageflow development, and depend upon the libraries contained in `struts-1.1.war` and `weblogic-beehive-1.0.ear`.
	`jstl-1.1#1.1@1.0`	Adds the Java standard tagging (JSTL) Version 1.1 libraries.
	`jsf-ri#1.1@1.1.1`	Adds the Java Server Faces Reference Implementation libraries.
	`jsf-myfaces#1.1@1.1.1`	Adds the Apache MyFaces libraries.
	`struts-1.1#1.1@1.0`	Adds the Apache Struts Version 1.1 libraries.
	`struts-1.2#1.2@1.0`	Adds the Apache Struts Version 1.2 libraries.

Table 32 Resources and Services Configured

	`weblogic-controls-1.0#1.0@1.0`	Adds the BEA Workshop for WebLogic controls extensions including additional system controls (such as service control and timer control) as well as support for adding transactions, security, and message buffering to existing controls. Packaged for EARs.
	`weblogic-controls-1.0-war#1.0@1.0`	Adds the BEA Workshop for WebLogic Platform controls extensions including additional system controls (such as service control) as well as support for adding transactions, security, and message buffering to existing controls. Excludes those features which require EAR support such as timer control. Packaged for WARs.
	`beehive-controls-1.0#1.0@1.0`	Adds the Apache Beehive Controls 1.0.1 libraries to the domain. This includes the control runtime as well as the Beehive system controls - JdbcControl, JMSControl, and EJBControl.

WebLogic Portal GroupSpace Application

Using the Configuration Wizard or WLST, you can easily extend a base WebLogic Server domain to include resources required for a WebLogic Portal GroupSpace Application domain. You accomplish this by adding the resources and services provided in the WebLogic Portal GroupSpace Application template to a base WebLogic Server domain.

Generated Domain Output

The WebLogic Portal GroupSpace Application is created from a Base WebLogic Server domain extended with the WebLogic Portal Extension template and the WebLogic Portal Collaboration Repository template. This template is named: `wli_groupspace.jar`.

The following table defines the default directory structure and files generated after applying the WebLogic Portal GroupSpace Application template to a base WebLogic Server domain. Unless otherwise specified, by default, the Configuration Wizard creates the domain in the `BEA_HOME\user_projects\domains\base_domain directory`. If you modify the default configuration settings, the output directory structure may be different from the structure described here. .

Table 33 Generated Domain Output for WebLogic GroupSpace Application Template

Directory	File	Description
`user_projects\domains\base_domain\`		
	`fileRealm.properties`	File containing ACLs, users, and groups that can be used for the default security realm when Compatibility security is used.
	`pointbase.ini`	File containing initialization information for a PointBase JDBC database.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`URLs.dat`	File containing the URL for the JDBC database.
`autodeploy\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for automatic deployments.

Table 33 Generated Domain Output for WebLogic GroupSpace Application Template

`bin\`	`setDomainEnv.cmd` `setDomainEnv.sh`	Scripts used to set up the development environment on Windows and UNIX systems, respectively.
	`startManagedWebLogic.cmd` `startManagedWebLogic.sh`	Scripts used to start a Managed Server on Windows and UNIX systems, respectively.
	`startPointBaseConsole.cmd` `startPointBaseConsole.sh`	Scripts used to start the PointBase console on Windows and UNIX systems, respectively.
	`startWebLogic.cmd` `startWebLogic.sh`	Scripts used to start the Administration Server on Windows and UNIX systems, respectively.
	`stopManagedWebLogic.cmd` `stopManagedWebLogic.sh`	Scripts used to stop a Managed Server on Windows and UNIX systems, respectively.
	`stopWebLogic.cmd` `stopWebLogic.sh`	Scripts used to stop the Administration Server on Windows and UNIX systems, respectively.
`config\`	`config.xml`	File containing the configuration information used by the "Domain Configuration Files" Administration Server. For more information, see in *Understanding Domain Configuration*.
`config\deployments\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for staging an application when the application's staging mode is "staged."
`config\diagnostics\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing the system modules associated with instrumentation in the WebLogic Diagnostic Framework (WLDF).

Table 33 Generated Domain Output for WebLogic GroupSpace Application Template

`config\jdbc\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JDBC modules that can be configured directly from JMX (as opposed to JSR-88).
	`cgDataSource-jdbc.xml`	Global XA JDBC Data Source module for the domain configured for advanced Web services and Business Process Management
	`cgDataSource-nonXA-jdbc.xml`	Global non-XA JDBC Data Source module for the domain configured for Business Process Management.
config\jms\	readme.txt	File providing information about the directory, which initially serves as a placeholder, and is later used for storing global JMS modules that can be configured directly from JMX (as opposed to JSR-88).
	`wseejmsmodule-jms.xml`	Global JMS module for the domain configured for advanced Web Services.
`config\lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing JAR files that are added to the system classpath of the server when the server's Java virtual machine starts.
`config\nodemanager\`	`nm_password.properties`	File containing Node Manager password property values.
`config\security\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules for the security framework. The directory contains one security provider configuration extension for each type of security provider in the domain's current realm.
`config\startup\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder, and is later used for storing system modules that contain startup plans. Startup plans are used to generate shell scripts that can be used as part of server startup.

Table 33 Generated Domain Output for WebLogic GroupSpace Application Template

`console-ext\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for custom extensions to the WebLogic Server Administration Console.
`init-info\`	`domain-info.xml`	File used to identify domain creation and extension information. Such information includes the identity of the components in the domain, the location of the JDK and applications directory used by the domain, and the templates used to create and extend the domain.
	`security.xml`	File used for creating user groups and roles that establish identity and access to domain resources.
	`startscript.xml`	File used to create the `*.cmd` and `*.sh` files that are placed into the domain's root and `bin` directories.
	`tokenValue.properties`	File that contains the actual values to substitute for the tokens specified in the start scripts.
`lib\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for the domain's libraries. The JAR files in this directory are added dynamically to the end of the server classpath at server startup.

Table 33 Generated Domain Output for WebLogic GroupSpace Application Template

`security\`	`DefaultAuthenticatorInit.ldift` `DefaultRoleMapperInit.ldift` `XACMLRoleMapperInit.ldift`	Files used for bootstrapping tasks, including authentication (user and group), authorization, and role mapping. These files contain LDAP-specific information. **Note:** WebLogic domains created with this release use the XACML providers by default. These XACML security providers are compatible with policies and roles created using the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). For more information, see WebLogic Security Providers in Understanding WebLogic Security at http://e-docs.bea.com/wls/docs100/secintro/archtect.html#archtect_0111
	`SerializedSystemIni.dat`	File containing encrypted security information.
`servers\AdminServer\security\`	`boot.properties`	File containing server startup properties, including the user name and password required to boot the server (in encrypted format). It is generated only when you select development startup mode. This file enables you to bypass the prompt for user name and password during a server's startup cycle. For more information, see "Provide User Credentials to Start and Stop Servers" in Starting and Stopping Servers in Managing Server Startup and Shutdown at http://e-docs.bea.com/wls/docs100/server_start/overview.html.
`user_staged_config\`	`readme.txt`	File providing information about the directory, which initially serves as a placeholder for configuration information optionally staged by an administrator to be copied to managed servers in the domain.
`WseeFileStore\`		Directory to be used for the file store for system resources.

Resources and Services Configured

The following table identifies the resources and services configured in a domain extended with the WebLogic Portal GroupSpace Application template.

Table 34 Resources and Services Configured

Resource Type	Name	Notes
Administration Server	`AdminServer`	Uses the Administration Server provided in the base WebLogic Server domain. The default name is `AdminServer`, unless changed during domain creation. The Administration Server referenced in the extension template is `cgServer`. For information about naming the Administration Server during domain creation, see "Resources and Services Configured for WebLogic Server Domain Template" on page 18.
JDBC Data Source	`cgDataSource`	Defines an XA JDBC data source including its associated jdbc connection pool. The data source is named `cgDataSource`.
	`cgDataSource-nonXA`	Includes the JDBC data source and connection pool setups defined as `cgDataSource` in the domain and targets them to the correct server(s).

Table 34 Resources and Services Configured

JDBC Store	`cgJMSStore`	Uses the JDBC store provided by the BEA Workshop for WebLogic Platform extension template. The JDBC store is to be used with the JDBC data source, `cgDataSource-nonXA`, and the JMS server, `cgJMSServer`, as a persistent store, and is targeted to the Administration Server, `AdminServer`.
JDBC System Resources	`cgDataSource` `cgDataSource-nonXA`	Identifies the JDBC data source and connection pool setups to be used for JDBC system.
JMS Server	`cgJMSServer`	Uses the JMS server provided by the Workshop for WebLogic Platform extension template. Identifies the JMS server as a `conversational-jms` system resource and targets it to the Administration Server, `AdminServer`.
Security realm	`myrealm`	Uses the security realm provided by the base WebLogic Server domain.
Commons-Logging Bridge	`wls-commonslogging-bridge#1.0@1.0`	Hooks commons-logging into the WLS logging mechanism.

Table 34 Resources and Services Configured

Libraries Deployed	`beehive-netui-1.0#1.0@1.0`	Adds the Apache Beehive NetUI Version 1.0 libraries. These libraries support pageflow development, and depend upon the libraries contained in `struts-1.1.war` and `weblogic-beehive-1.0.ear`.
	`jstl-1.1#1.1@1.0`	Adds the Java standard tagging (JSTL) Version 1.1 libraries.
	`jsf-ri#1.1@1.1.1`	Adds the Java Server Faces Reference Implementation libraries.
	`jsf-myfaces#1.1@1.1.1`	Adds the Apache MyFaces libraries.
	`struts-1.1#1.1@1.0`	Adds the Apache Struts Version 1.1 libraries.
	`struts-1.2#1.2@1.0`	Adds the Apache Struts Version 1.2 libraries.

Table 34 Resources and Services Configured

	`weblogic-controls-1.0#1.0@1.0`	Adds the BEA Workshop for WebLogic controls extensions including additional system controls (such as service control and timer control) as well as support for adding transactions, security, and message buffering to existing controls. Packaged for EARs.
	`weblogic-controls-1.0-war#1.0@1.0`	Adds the BEA Workshop for WebLogic Platform controls extensions including additional system controls (such as service control) as well as support for adding transactions, security, and message buffering to existing controls. Excludes those features which require EAR support such as timer control. Packaged for WARs.
	`beehive-controls-1.0#1.0@1.0`	Adds the Apache Beehive Controls 1.0.1 libraries to the domain. This includes the control runtime as well as the Beehive system controls - JdbcControl, JMSControl, and EJBControl.