BEAWebLogic Server™

Deploying WebLogic Server Applications

Version 8.1
Revised: September 3, 2004

Contents

About This Document

1. Overview of WebLogic Server Deployment

2. Quickstart Guide to Deployment

3. Performing Common Deployment Tasks

4. Advanced Deployment Topics

5. Deployment Tools Reference

About This Document

This document describes how to deploy and redeploy applications and standalone modules on WebLogic Server in a production environment.

The document is organized as follows:

- Chapter 1, "Overview of WebLogic Server Deployment," provides an overview of the types of deployment units you can deploy to WebLogic Server.

- Chapter 2, "Quickstart Guide to Deployment," describes how to quickly deploy a new application or standalone module to WebLogic Server.

- Chapter 4, "Advanced Deployment Topics," provides detailed information about how WebLogic Server deploys and redeploys application modules.

- Chapter 3, "Performing Common Deployment Tasks," describes how to perform different deployment tasks using both the Administration Console and the `weblogic.Deployer` command-line utility.

- Chapter 5, "Deployment Tools Reference," provides a complete reference to the `weblogic.Deployer` command-line utility and describes other tools used to deploy application modules.

Audience

This document is written mainly for Administrators who want to deploy Java 2 Platform, Enterprise Edition (J2EE) applications or application modules to WebLogic Server. This document assumes that you are working in a production environment, which is generally

characterized by multiple WebLogic Server instances or clusters running on multiple machines. It also assumes that you have one or more application module archive files that have been tested and are ready to deploy on a production server.

If you are an engineer and need to deploy or package applications in a development environment, see *Developing WebLogic Server Applications*.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation.

How to Print the Document

You can print a copy of this document from a Web browser, one main topic at a time, by using the File—Print option on your Web browser.

A PDF version of this document is available on the WebLogic Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

Adobe Acrobat Reader is available at no charge from the Adobe Web site at http://www.adobe.com.

Contact Us!

Your feedback on BEA documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the documentation.

In your e-mail message, please indicate the software name and version you are using, as well as the title and document date of your documentation. If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number

- Your company name and company address

- Your machine type and authorization codes

- The name and version of the product you are using

- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Usage
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis and book titles.

`monospace text`	Code samples, commands and their options, Java classes, data types, directories, and file names and their extensions. Monospace text also indicates text that the user is told to enter from the keyboard. *Examples*:
	 `import java.util.Enumeration;`
	 `chmod u+w *`
	 `config/examples/applications`
	 `.java`
	 `config.xml`
	 `float`

`monospace italic text`	Placeholders. *Example*: `String `*`CustomerName`*`;`

`UPPERCASE MONOSPACE TEXT`	Device names, environment variables, and logical operators. *Example*s:
	 `LPT1`
	 `BEA_HOME`
	 `OR`

{ }	A set of choices in a syntax line.

Convention	Usage
[ ]	Optional items in a syntax line. *Example*: ``` java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send] ```
|	Separates mutually exclusive choices in a syntax line. *Example*: ``` java weblogic.deploy [list|deploy|undeploy|update] password {application} {source} ```
...	Indicates one of the following in a command line: • An argument can be repeated several times in the command line. • The statement omits additional optional arguments. • You can enter additional parameters, values, or other information
. . .	Indicates the omission of items from a code example or from a syntax line.

Overview of WebLogic Server Deployment

The following sections provide a basic overview of key BEA WebLogic Server™ deployment topics:

- "Deployment Units" on page 1-2

- "Deployment Files" on page 1-6

- "Deployment Targets" on page 1-8

- "Staging Modes" on page 1-9

- "Deployment Names" on page 1-12

- "Deployment Tools" on page 1-12

- "Security Roles Required for Deployment" on page 1-14

Document Scope and Audience

This document is written mainly for Administrators who want to deploy Java 2 Platform, Enterprise Edition (J2EE) applications or application modules to WebLogic Server. This document assumes that you are working in a production environment, which is generally characterized by multiple WebLogic Server instances or clusters running on multiple machines. It also assumes that you have one or more application module archive files that have been tested and are ready to deploy on a production server.

If you are an engineer and need to deploy or package applications in a development environment, see *Developing WebLogic Server Applications*. Also see "Deployment Topics for Developers" on page 4-12 for information about deployment features that apply to development environments.

Deployment Units

A *deployment unit* refers to a J2EE application (an Enterprise Application or Web Application) or a standalone J2EE module (an EJB or Resource Adapter) that has been organized according to the J2EE specification and can be deployed to WebLogic Server.

For each type of deployment unit, the J2EE specification defines both the required files and their location in the directory structure of the application or module. Deployment units may include Java classes for EJBs and servlets, resource adapters, Web pages and supporting files, XML-formatted deployment descriptors, and even other modules.

J2EE does not specify *how* a deployment unit is deployed on the target server—only how standard applications and modules are organized. WebLogic Server supports deployments that are packaged either as archive files using the jar utility, or as exploded archive directories.

Standalone Modules

J2EE provides specifications for creating standalone modules, which include Enterprise JavaBeans and resource adapter modules. WebLogic Server also supports deploying Web Services modules, which are not part of the J2EE specification.

Standalone modules generally provide parts of a larger, distributed application, but do not necessarily provide a direct user interface. For more information on a particular module type, refer to the J2EE 1.3 specification at: http://java.sun.com/j2ee/download.html#platformspec.

Enterprise JavaBean

Enterprise JavaBeans (EJBs) are reusable Java components that implement business logic and enable you to develop component-based distributed business applications. EJB modules are packaged as archive files having a .jar extension, but are generally deployed as exploded archive directories. The archive file or exploded archive directory for an EJB contains the compiled EJB classes, optional generated classes, and XML deployment descriptors for the EJB. See *Programming WebLogic Server Enterprise JavaBeans* for more information on the different types of EJBs.

Resource Adapter

A Resource Adapter (also referred to as a connector) adds Enterprise Information System (EIS) integration to the J2EE platform. Connectors provide a system-level software driver that WebLogic Server can use to connect to an EIS. Connectors contain both the Java classes, and if necessary, the native components required to interact with the EIS. See *Programming WebLogic Server J2EE Connectors* for more information.

Note: Resource Adapters cannot be deployed as exploded archive directories.

Web Services

Web Services are a type of service that can be shared by and used as components of distributed Web-based applications. They commonly interface with existing back-end applications, such as customer relationship management systems, order-processing systems, and so on.

A Web Service module is a virtual module that includes both a Web Application and a Web Service deployment descriptor at minimum. A Web Service module may also include EJBs that implement the Web Service. See *Programming WebLogic Web Services* for more information.

Deployable Applications

J2EE also defines several different types of deployable applications: Web Applications, Enterprise Applications, and client applications. Applications generally include one or more standalone J2EE modules to provide a logical service, and may also provide a direct user interface to the applications' contents.

Web Application

A Web Application always includes the following files:

- A servlet or JSP page, along with any helper classes.

- A web.xml deployment descriptor, a J2EE standard XML document that configures the contents of a WAR file.

Web Applications may also contain JSP tag libraries, static .html and image files, supporting classes and .jar files, and a weblogic.xml deployment descriptor, which configures WebLogic Server-specific elements for Web Applications. See *Developing Web Applications for WebLogic Server* for more information.

Enterprise Application

An Enterprise Application consists of one or more of the following J2EE applications or modules:

- Web Applications

- Enterprise Java Beans (EJB) modules

- Resource Adapter modules

An Enterprise Application is packaged as a JAR file with an `.ear` extension, but is generally deployed as an exploded EAR directory. An exploded EAR directory contains all of the JAR, WAR, and RAR modules (also in exploded format) for an application as well as the XML descriptor files for the Enterprise Application and its bundled applications and modules. See *Developing WebLogic Server Applications* for more information.

Client Application Archives

The J2EE specification provides the ability to include a client application archive file within an Enterprise Application. A J2EE client application module contains the Java classes that execute in the client JVM (Java Virtual Machine) and deployment descriptors that describe EJBs (Enterprise JavaBeans) and other WebLogic Server resources used by the client. This enables both the server-side and client-side components to be distributed as a single unit. Client modules in an EAR are defined using the J2EE standard `application-client.xml` deployment descriptor and an optional WebLogic Server descriptor, `client-application.runtime.xml`.

Startup and Shutdown Classes

Startup and shutdown classes are Java classes that you can configure to load and execute when you start up or gracefully shut down a server. The deployment process for startup and shutdown classes differs from other deployable units, because you do not specify any actual deployment files. Instead, you identify the servers that will execute the class, and specify the classpath and optional arguments to use for execution. See Startup and Shutdown Classes in the *Administration Console Online Help* for more information.

XML Deployment Descriptors for Deployment Units

A key file that defines each type of deployment unit is its XML deployment descriptor. The deployment descriptor is an XML document that defines certain runtime characteristics for the application or module. By editing the XML deployment descriptor, you modify runtime behavior for the deployment unit without having to recompile code or rebuild the deployment unit itself;

the new behavior is read from the descriptor file and implemented when you deploy the application or module.

For example, the WebLogic deployment descriptors for an EJB allow you to specify the number of beans to keep in the cache when you deploy the EJB to a server. If you need to change the number of cached beans for performance reasons, you simply change the deployment descriptor.

J2EE defines the organization and content of required XML deployment descriptors for each deployment unit. In addition, you can specify optional WebLogic Server XML deployment descriptors to configure WebLogic-specific behavior for handling the deployment. WebLogic Server deployment descriptors enable you to maintain the portability of the original J2EE module while utilizing features only available in WebLogic Server.

Table 1-1 lists the types of deployment units and their associated J2EE-standard and WebLogic-specific deployment descriptors.

Table 1-1 J2EE and WebLogic Deployment Descriptors

Deployment Unit	Scope	Deployment Descriptors
Web Application	J2EE	`WEB-INF/web.xml`
	WebLogic	`WEB-INF/weblogic.xml`
Enterprise Bean	J2EE	`META-INF/ejb-jar.xml`
	WebLogic	`META-INF/weblogic-ejb-jar.xml` `META-INF/weblogic-cmp-rdbms-jar.xml`
Connector	J2EE	`META-INF/ra.xml`
	WebLogic	`META-INF/weblogic-ra.xml`
Web Service	J2EE	Web Services have no J2EE descriptor. However, a Web Service module is typically packaged as an Enterprise Application that contains both a Web Application and EJB modules. For this reason, Web Services modules generally require J2EE descriptors required for those modules.
	WebLogic	`WEB-INF/web-services.xml`
Enterprise Application	J2EE	`META-INF/application.xml`
	WebLogic	`META-INF/weblogic-application.xml`

Table 1-1 J2EE and WebLogic Deployment Descriptors

Deployment Unit	Scope	Deployment Descriptors
Client Application	J2EE	`META-INF/application-client.xml`
	WebLogic	`client-application.runtime.xml`

You create deployment descriptors manually, or generate them with WebLogic Server Java-based utilities. When you receive a J2EE-compliant JAR file from a developer, it should already contain the J2EE-defined and WebLogic Server deployment descriptors. The Administration Console also enables you to modify key deployment descriptor elements of a deployed application, which can help you tune the application's run-time behavior and performance.

Deployment Files

WebLogic Server allows you to store deployment units either as a single archive file, or as an exploded directory that contains the same contents of the archive file.

Archive Files

In most production environments, you receive a deployment unit as an archive file. An archive file is a single file that contains all of an application's or module's classes, static files, directories, and deployment descriptor files. Archive files are typically created by using the `jar` utility or Ant's `jar` tool.

Deployment units that are packaged using the `jar` utility have a specific file extension depending on the type:

- EJBs are packaged as `.jar` files.

- Web Applications are packaged as `.war` files.

- Resource Adapters are packaged as `.rar` files.

- Enterprise Applications are packaged as `.ear` files, and can contain any combination of EJBs, Web Applications, and Resource Adapters.

- Web Services can be packaged either as `.ear` files or as `.war` files.

Note: Startup and shutdown classes are not packaged—you simply specify the classpath to use when assigning target servers.

Although you can deploy an archive file directly to WebLogic Server, deploying Exploded Archive Directories provides some additional redeployment capabilities.

Exploded Archive Directories

An exploded archive directory contains the same files and directories as a `jar` archive. However, the files and directories reside directly in your file system and are not packaged into a single archive file with the `jar` utility.

A deployment unit should be deployed as an exploded archive directory, rather than a single archive file, in the following circumstances:

- You want to perform partial updates to a deployed application without redeploying the entire application.

- You want to use the Administration Console to dynamically edit and persist selected deployment descriptor values for the deployment. (You cannot edit deployment descriptor values in the console for deployments from archive files.) See "Dynamically Updating Descriptors for a Deployment Unit" on page 3-16 for more information.

- You are deploying a Web Application that performs direct file system I/O through the application context (for example, a Web Application that tries to dynamically edit or update parts of the Web Application itself). In this case, the modules that perform the I/O operations should have a physical filesystem directory in which to work; you cannot obtain a file when the application is deployed as an archive, as per the specification.

- You are deploying a Web Application or Enterprise Application that contains static files that you will periodically update. In this case, it is more convenient to deploy the application as an exploded directory, because you can update and refresh the static files without re-creating the archive.

Unpacking an Archive File

If you have an archive file that you want to deploy as an exploded archive directory, use the `jar` utility to unpack the archive file in a dedicated directory. For example:

```
mkdir /myapp

cd /myapp

jar xvf /dist/myapp.ear
```

If you are unpacking an archive file that contains other module archive files (for example, an Enterprise Application or Web Service that includes JAR or WAR files) and you want to perform

partial updates of those modules, you must expand the embedded archive files as well. Make sure that you unpack each module into a subdirectory having the same name as the archive file. For example, unpack a module named `myejb.jar` into a `/myejb.jar` subdirectory of the exploded Enterprise Application directory.

Note: If you want to use different subdirectory names for the archived modules in an exploded EAR file, you must modify any references to those modules in the application itself. For example, you must update the URI values specified in `application.xml` and `CLASSPATH` entries in the `manifest.mf` file.

Location of Files

To deploy a new archive or exploded archive directory, the file(s) must be accessible by the Administration Server for your domain. This means they must reside on the Administration Server machine, or they must be available via a remote, network-mounted directory.

When using the Administration Console to deploy a module for the first time, you have the option to upload files to the Administration Server machine if they are not otherwise available.

Deployment Targets

Deployment targets are the servers to which you deploy an application or standalone module. Valid deployment targets include WebLogic Server instances, clusters, and virtual hosts. During the deployment process, you select the list of targets from the available servers, clusters, and virtual hosts configured in your domain. You can also change the target list at any time after you have deployed a module.

The following sections describe target types in more detail.

WebLogic Server Instances

You can deploy applications and standalone modules to one or more Managed Servers in a domain, or to the Administration Server in a single-server domain. Although you can deploy to the Administration Server in a multiple-server domain, this practice is not recommended. The Administration Server in a multiple-server domain should be used only for administration purposes.

Clusters

If you are deploying to a cluster of WebLogic Server instances, by default the deployment targets all server instances in the cluster. This corresponds to homogenous module deployment, which is

recommended in most clusters. If you want to deploy a module only to a single server in the cluster (that is, "pin" a module to a servers), you can also select the individual server name. This type of deployment is less common, and should be used only in special circumstances where pinned services are required. See Understanding Cluster Configuration and Application Deployment for more information.

Note: Pinning a deployment to a subset of server instances in a cluster (rather than a single server) is not recommended and will generate a warning message.

Virtual Hosts

Virtual hosting allows you to define host names to which servers or clusters respond. When you use virtual hosting, you use DNS naming to specify one or more host names that map to the IP address of a WebLogic Server or cluster, and you specify which applications are served by each virtual host.

You select a configured virtual host name as you would a WebLogic Server instance or cluster name during deployment. See Configuring Virtual Hosting in *Configuring and Managing WebLogic Server* for more information.

Staging Modes

The deployment staging mode determines how deployment files are made available to target servers that must deploy an application or standalone module. WebLogic Server provides three different options for staging files: stage mode, nostage mode, and external_stage mode. The following table describes the behavior and best practices for using the different deployment staging modes.

Table 1-2 Application Deployment Staging Modes

Deployment Staging Mode	Behavior	When to Use
stage	The Administration Server first copies the deployment unit source files to the staging directories of target servers. (The staging directory is named stage by default, and it resides under the target server's root directory.) The target servers then deploy using their local copy of the deployment files.	• Deploying small or moderate-sized applications to multiple WebLogic Server instances. • Deploying small or moderate-sized applications to a cluster.

Deployment Staging Mode	Behavior	When to Use
nostage	The Administration Server does not copy deployment unit files. Instead, all servers deploy using the same physical copy of the deployment files, which must be directly accessible by the Administration Server and target servers. With nostage deployments of exploded archive directories, WebLogic Server automatically detects changes to a deployment's JSPs or Servlets and refreshes the deployment. (This behavior can be disabled if necessary.)	• Deploying to a single-server domain. • Deploying to a cluster on a multi-homed machine. • Deploying very large applications to multiple targets or to a cluster where deployment files are available on a shared directory. • Deploying exploded archive directories that you want to periodically redeploy after changing content. • Deployments that require dynamic update of selected Deployment Descriptors via the Administration Console.
external_stage	The Administration Server does not copy deployment files. Instead, the Administrator must ensure that deployment files are distributed to the correct staging directory location before deployment (for example, by manually copying files prior to deployment). With external_stage deployments, the Administration Server requires a copy of the deployment files for validation purposes. Copies of the deployment files that reside in target servers' staging directories are not validated before deployment.	• Deployments where you want to manually control the distribution of deployment files to target servers. • Deploying to domains where third-party applications or scripts manage the copying of deployment files to the correct staging directories. • Deployments that do not require dynamic update of selected Deployment Descriptors via the Administration Console (not supported in external_stage mode). • Deployments that do not require partial redeployment of application components.

Most deployments use either stage or nostage modes, and the Administration Console automatically suggests the appropriate mode when you deploy an application or module. For

more information about staging modes, see Deployment Staging Modes and Staging Directories in "Advanced Deployment Topics" on page 4-1.

Deployment Names

When you first deploy an application or standalone module to one or more WebLogic Server instances, you specify a deployment name to describe collectively the deployment files, target servers, and other configuration options you selected. You can later redeploy or stop the deployment unit on all target servers by simply using the deployment name. The deployment name saves you the trouble of re-identifying the deployment files and target servers when you want to work with the deployment unit across servers in a domain.

If you do not specify a deployment name at deployment time, the deployment tool selects a default name. For exploded archive directories, the default deployment name is the name of the top-level directory you deploy. For archive files, the deployment name is the name of the archive file without the file extension (`myear` for the file `myear.ear`).

Deployment Tools

WebLogic Server provides several tools for deploying applications and standalone modules:

- Administration Console—provides a series of web-based Deployment Assistants that guide you through the deployment process. The Administration Console also provides controls for changing and monitoring the deployment status, and changing selected deployment descriptor values while the deployment unit is up and running.

  Use the Administration Console when you need to perform basic deployment functions interactively and you have access to a supported browser.

- `weblogic.Deployer`—provides a command-line based interface for performing both basic and advanced deployment tasks.

  Use `weblogic.Deployer` when you do not have access to a browser or graphical user interface environment, and you want to perform deployment tasks. You can also use `weblogic.Deployer` when you want to automate deployment tasks through shell scripts or batch processes, rather than through Ant tasks.

- `wldeploy` is an Ant task version of the `weblogic.Deployer` utility. You can automate deployment tasks by placing `wldeploy` commands in an Ant `build.xml` file and running Ant to execute the commands.

- The deployment API allows you to perform deployment tasks programmatically using Java classes.

- The `applications` directory of a server allows you to deploy an application quickly for evaluation or testing in a development environment.

Best Practices for Deployment

BEA recommends the following best practices when deploying applications:

- Package deployment files in an archive format (`.ear`, `.jar`, `.war`, and so forth) when distributing files to different users or environments.

- Check the scenarios described under "Exploded Archive Directories" on page 1-7 before deploying. In many cases it is more convenient to deploy applications in exploded format rather than archive format.

- The Administration Console, `weblogic.Deployer` tool, and `wldeploy` Ant task all provide similar functionality for deploying applications:

  - Use the Administration Console for interactive deployment sessions where you do not know the exact location of deployment files, or the exact names of target servers or deployed applications.

  - Use `weblogic.Deployer` to integrate deployment commands with existing administrative shell scripts or automated batch processes.

  - Use `wldeploy` in conjunction with the split development directory for developing and deploying new applications. `wldeploy` can also be used in place of `weblogic.Deployer` in administration environments that use Ant, rather than shell scripts.

- Use `wldeploy`, rather than the `applications` directory, to deploy your own applications during development. The `applications` directory is best used for quickly running new applications in a test or temporary. For example, if you download a sample application and want to evaluate its functionality, the `applications` directory provides a quick way to deploy the application into a development server.

Deployment Operations

The deployment tools provide support for performing these common deployment operations:

- **Deploy**—making deployment source files available to target servers and loading classes into classloaders so that applications are available to clients.

- **Distribute**—for stage mode deployments, only copy the deployment files to the target servers (but do not start the deployment).

- **Redeploy**—updating an deployment unit or part of a deployment unit (for example, an EAR, a module within an EAR, or a static file in a Web Application) that is currently deployed and available to clients. When redeploying an entire application, all of the application's modules must redeploy successfully or the entire application is stopped.

Note: An application becomes unavailable to clients during redeployment. WebLogic Server doesn't guarantee the operation of the application and deployment task if there is an access from the client at this time. For this reason, redeployment is not recommended for use in a production environment.

- **Stop**—unloading an application's classes and making an application unavailable to clients. Stopping still leaves the deployment files and deployment name available to target servers for subsequent redeployment or starting.

- **Start**—reloading an application's classes into classloaders and making the application available to clients. Starting requires that the deployment files be available on target servers as a result of an earlier deployment.

- **Undeploy**—stopping a deployment unit and then removing its deployment files and deployment name from target servers.

Note: An application becomes unavailable to clients during undeployment. WebLogic Server doesn't guarantee the operation of the application and deployment task if there is an access from the client at this time.

Security Roles Required for Deployment

The built-in security roles for "Admin" and "Deployer" users allow you to perform deployment tasks using the WebLogic Server Administration Console. See Starting the Administration Console in the Administration Console Online Help for more information about the built-in security roles.

Quickstart Guide to Deployment

The following sections explain how to deploy an application or standalone module quickly in a BEA WebLogic Server™ domain:

Step 1: Unpack the Archive File (if Necessary)

In some production environments, you will deploy an archive file (`.ear`, `.war`, `.jar`, or `.rar` extension) unchanged. See "Exploded Archive Directories" on page 1-7 to determine if you should unpack the archive file before deploying.

If you want to deploy from an exploded archive directory, use the `jar` utility to unpack the archive file. Start by creating an empty directory in which you will store the files and moving to the new directory. Name the directory according to the application or standalone module you are deploying. For example:

```
mkdir mywebapp
cd mywebapp
```

Use the `jar` utility to unpack the archive:

```
jar xvf c:\production\mywebapp.war
```

If you are unpacking an EAR archive, the archive may contain additional archive files (`.jar` and `.war` extensions). In this case, also unpack the embedded archives in the exploded directory. Make sure the module subdirectories have the same name as the archive files, or you will need to update references to those modules elsewhere in the application. See "Exploded Archive Directories" on page 1-7.

Step 2: Start the Deployment Assistant

Access the Administration Console for the domain by loading its URL in your browser (for example, http://myhost:7001/console) and entering the administrator username and password.

Expand the Deployments Node in the left pane of the console, and select the type of deployment unit you want to deploy. The available deployment units are:

- Applications—Enterprise Applications or Web Services packaged as `.ear` files or directories
- EJB Modules—Enterprise JavaBean modules
- Web Application Modules—Web Applications or Web Services packaged as `.war` files or directories
- Connector Modules—Resource Adapters

The right pane of the Administration Console displays current deployments of the selected type. Click the Deploy a New *deployment unit* link in the right pane to start the associated Deployment Assistant.

Step 3: Upload Deployment Files (if Necessary)

When you use the Administration Console to deploy, the deployment files must be available on the Administration Server machine. If they are not yet available, use the "upload your file(s)" link on the first page of the Deployment Assistant to browse the directories on the local machine and locate an archive file to upload. Click the upload button to upload the archive file to the Administration Server machine.

Step 4: Select the Deployment Files

In the first page of the Deployment Assistant, use the links in the Location field to browse directories on the Administration Server machine and locate the deployment file(s). (If you uploaded the deployment files in the previous step, the Administration Server's upload directory is automatically selected so you can choose the uploaded file.)

When the assistant detects a valid archive file or exploded archive directory in the current directory, it lists the archive or directory name as a selection beneath the Location field. Select the name of the archive or directory you want to deploy.

If you are deploying an Enterprise Application and your domain contains multiple WebLogic Server instances, you have the option to deploy all modules in the application to a single server, or target different modules of the application to different server instances. Click the Target Application or Target Each Module button, respectively. For standalone modules, click Continue to select target servers.

Step 5: Select the Target Servers

In the second page of the Deployment Assistant, use the check boxes to select target servers on which you will deploy the application or standalone module. To deploy to individual servers, select one or more server instances from the Independent Servers list and click Continue.

To deploy to a cluster of servers, select the name of the cluster from the Clusters list. By default, the assistant deploys to all server instances in the cluster (the All servers in the cluster option). If you want to deploy only to a subset of the servers in a cluster, select Parts of the cluster and then select the individual server instances to which you want to deploy the application or module. In most cases, you should deploy to all members of the cluster; see Deploy Homogeneously to Avoid Cluster-Level JNDI Conflicts in *Using WebLogic Server Clusters*.

If you are targeting individual modules of an Enterprise Application, your selections apply only to the module name displayed in the header of the console page (for example, Step 2 - Select targets for module "*module_name*"). If the application has additional modules to deploy, the console re-displays the Select Targets page after you click Continue, allowing you to target the next module to different server instances.

Click Continue when you have finished selecting target servers for the deployment.

Step 6: Deploy the Files

On the final page of the deployment assistant, you review your selection of target servers, choose a deployment staging mode, and define a deployment name. Review the entries under the Deployment Targets heading. If you need to change a target, click your browser's Back button.

The Source accessibility header displays the selected staging mode for the deployment source files:

- **Copy this application onto every target for me**—This option is selected by default if you targeted the deployment to a cluster or to multiple server instances. This corresponds to "stage" mode where the Administration Server copies the application files to each targeted server; the targeted servers then deploy the application using their copy of the source files.

- **I will make the application accessible from the following location**—This option is selected by default if you targeted the deployment to a single server instance. This corresponds to "nostage" mode where the server deploys from a single directory using the same physical deployment files; all targeted servers must be able to access the directory to deploy the application. Select this option if you are deploying to a cluster that resides on a single physical machine.

In the Identity header, the Name field specifies a unique name to refer to this deployment in the Administration Console. Accept the default name or enter a new name to describe the deployment unit.

Click Deploy to accept the values on this page and deploy to the listed server instances.

Performing Common Deployment Tasks

The following sections describes how to perform common deployment tasks using the Administration Console and `weblogic.Deployer` utility:

The following sections describe advanced deployment tasks:

- "Changing a Server Staging Mode or Staging Directory" on page 3-15
- "Dynamically Updating Descriptors for a Deployment Unit" on page 3-16
- "Redeploying Static Files in a Web Application" on page 3-17
- "Removing Files from a Web Application Deployment" on page 3-18
- "Managing Deployment Tasks" on page 3-18

Uploading Deployment Files to the Administration Server

In order to deploy an application or standalone module to servers in a domain, the deployment file(s) must be accessible to the domain's Administration Server. If the files do not reside on the Administration Server machine or are not available to the Administration Server machine via a network mounted directory, use the instructions below to upload files.

When you upload files to the Administration Server machine, the archive file is automatically placed in the server's upload directory. You can configure the path of this directory using the instructions in "Changing a Server Staging Mode or Staging Directory" on page 3-15.

Administration Console Tasks

Note: The Administration Console upload functionality helps you upload a single archive file to the Administration Server machine for deployment. If you need to upload an exploded archive directory, use the Weblogic.Deployer Tasks.

To upload an archive file to the Administration Server machine using the Administration Console:

1. Start the Administration Console for your domain.

2. Expand the Deployments node in the left pane to display the different deployment units.

3. In the left pane, select the type of deployment unit that you want to deploy. The available deployment units are:

   – Applications—Enterprise Applications or Web Services packaged as `.ear` files or directories

   – EJB Modules—Enterprise JavaBeans

   – Web Application Modules—Web Applications or Web Services packaged as `.war` files or directories

   – Connector Modules—Resource Adapters

– Startup & Shutdown—Startup classes and shutdown classes

4. In the right pane, select the Deploy a new *deployment unit* link, where *deployment unit* is the type of application or standalone module you want to deploy. This initiates the Deployment Assistant for the deployment unit.

5. The right pane provides a link to upload the deployment files. Click the link to display the upload page.

6. Use the Browse button to locate the archive file on the local file system that you want to upload to the Administration Server. Select the file and click Open. (You cannot select an exploded archive directory using the Administration Console.)

7. Click the Upload tab to upload the selected file to the Administration Server's upload directory. The Administration Console automatically opens the upload directory so that you can select the uploaded file for deployment.

8. Continue deploying using the uploaded archive file.

Weblogic.Deployer Tasks

When using `weblogic.Deployer` to deploy a new application or standalone module, add the `-upload` option to upload the deployment files. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name myapp -targets myserver -upload
   -deploy c:\localfiles\myapp.ear
```

`weblogic.Deployer` copies the deployment files to the Administration Server's upload directory before deploying it. To upload an exploded archive directory, specify the directory name instead of an archive filename (for example `c:\localfiles\myappEar`).

Deploying an Application or Standalone Module to Servers on the Same Machine

When you deploy a deployment unit to servers that reside on the same machine, you can use the nostage deployment mode. With nostage mode, the Administration Server and all target servers deploy using the same deployment files (deployment files are not copied to servers' stage directories).

Note: Nostage mode is the default when you deploy an application or standalone module to the Administration Server; you do not need to specify nostage mode in this case.

Administration Console Tasks

The Administration Console automatically defaults to nostage mode when you deploy to the Administration Server, or to a standalone server. If you target more than one server and the servers reside on the same machine, you must manually specify nostage mode.

To deploy to servers on one machine:

1. Start the Administration Console for your domain.

2. Expand the Deployments node in the left pane to display the different deployment units.

3. In the left pane, select the type of deployment unit that you want to deploy. The available deployment units are:

   – Applications—Enterprise Applications or Web Services packaged as `.ear` files or directories

   – EJB Modules—Enterprise JavaBeans packaged as `.jar` files or directories

   – Web Application Modules—Web Applications or Web Services packaged as `.war` files or directories

   – Connector Modules—Resource Adapters packaged as `.rar` files or directories

   – Startup & Shutdown—Startup classes and shutdown classes

4. In the right pane, select the Deploy a new *deployment unit* link, where *deployment unit* is the type of application or standalone module you want to deploy. This initiates the Deployment Assistant for the deployment unit.

5. In the first step of the Deployment Assistant, select the archive file or exploded archive directory that you want to deploy. See Deploying Applications and Modules in the *Administration Console Online Help* if you need further instructions for using the Deployment Assistants.

6. In the second step of the Deployment Assistant, select the servers and cluster to which you want to deploy the application.

7. In the third step of the Deployment Assistant, review your choices and specify a deployment name. In the Source Accessibility header, make sure that the following option is selected:

   – I will make the application accessible from the following location

   This option corresponds to nostage mode, where all targets deploy the files from the specified location.

8. Click Deploy to deploy the files.

Weblogic.Deployer Tasks

When using the `weblogic.Deployertool` utility, specify the nostage mode explicitly when you deploy to servers on the same machine. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
    -password weblogic -name mydeploymentname
    -targets myserver1,myserver2,myserver3 -nostage
    -deploy c:\localfiles\myapp.ear
```

Deploying an Application or Standalone Module to Servers on Multiple Machines

When deploying a deployment unit to servers on different machines, you generally use the stage deployment mode, which copies deployment files from the Administration Server to each target server before deploying. If all servers can access the deployment files from a shared directory, you can also use nostage mode, which does not copy files to the target servers. In rare circumstances you may also use external_stage mode, in which you must ensure that deployment files are copied to each server's staging directory before deployment. See "Staging Modes" on page 1-9.

Note: Stage mode is the default when you deploy an application or standalone module to a Managed Server; you do not need to specify stage mode in this case.

Administration Console Tasks

The Administration Console allows you to select between nostage and stage mode when deploying files. If you need to deploy in external_stage mode, use the `weblogic.Deployer` instructions in the next section or in "Deploying an Enterprise Application Using external_stage Mode" on page 3-13.

To deploy a deployment unit to multiple servers on different machines:

1. Start the Administration Console for your domain.

2. Expand the Deployments node in the left pane to display the different deployment units.

3. In the left pane, select the type of deployment unit that you want to deploy. The available deployment units are:

- Applications—Enterprise Applications or Web Services packaged as `.ear` files or directories

- EJB Modules—Enterprise JavaBeans packaged as `.jar` files or directories

- Web Application Modules—Web Applications or Web Services packaged as `.war` files or directories

- Connector Modules—Resource Adapters packaged as `.jar` files or directories

- Startup & Shutdown—Startup classes and shutdown classes

4. In the right pane, select the Deploy a new *deployment unit* link, where *deployment unit* is the type of application or standalone module you want to deploy. This initiates the Deployment Assistant for the deployment unit.

5. In the first step of the Deployment Assistant, select the archive file or exploded archive directory that you want to deploy. See Deploying Applications and Modules in the *Administration Console Online Help* if you need further instructions about using the Deployment Assistants.

6. In the second step of the Deployment Assistant, select the servers and cluster to which you want to deploy the application.

7. In the third step of the Deployment Assistant, review your choices and specify a deployment name. In the Source Accessibility header, make sure that the following option is selected:

- Copy this application onto every target for me

This option corresponds to stage mode, where the Administration Server copies the deployment files to each target server before the servers deploy the application or module.

8. Click Deploy to deploy the files.

Weblogic.Deployer Tasks

When deploying to multiple machines using the `weblogic.Deployer` utility, specify the nostage or external_stage mode explicitly at the command line. The deployment uses stage mode by default if no mode is specified. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name mydeploymentname
   -targets myserver1,myserver2,myserver3 -stage
   -deploy c:\localfiles\myapp.ear
```

Changing the Target List for an Existing Deployment

After you deploy an application or standalone module in a WebLogic Server domain, you can change the target server list to add new WebLogic Server instances or to remove existing server instances. If you remove a target server, only the target list itself is updated—the deployment unit remains deployed to the removed server until you explicitly undeploy it. Similarly, if you add a new target server, you must explicitly deploy the deployment unit on the new server before it is active on that server.

Administration Console Tasks

Note: After adding a new target list using the Administration Console, you must redeploy the deployment unit on all existing servers in order to deploy it on the newly-added target server.

1. Start the Administration Console for your domain.

2. Expand the Deployments node in the left pane to display the different deployment units.

3. In the left pane, select the type of deployment unit whose target server list you want to change. The available deployment units are:

   – Applications—Enterprise Applications or Web Services packaged as `.ear` files or directories

   – EJB Modules—Enterprise JavaBeans

   – Web Application Modules—Web Applications or Web Services packaged as `.war` files or directories

   – Connector Modules—Resource Adapters

   – Startup & Shutdown—Startup classes and shutdown classes

4. In the right pane, select the deployment name of the actual deployment unit whose target server list you want to modify. This displays a table with the deployment status on all target servers.

5. Click the Targets tab in the right pane to show the current target servers for the deployment. If you selected an Enterprise Application, the pane displays a target list for each module that makes up the application.

6. Use the checkboxes next to each server name to add or remove servers from the target list. Click Apply to apply your changes.

7. Select the Deploy tab in the right pane.

8. Click Deploy this Application; the Administration Console deploys the application on newly-added servers, and redeploys the application on existing target servers.

Weblogic.Deployer Tasks

Note: You cannot use `weblogic.Deployer` to remove a server from a deployment unit's target list if the application or module is already deployed.

To add a new server to the target list using `weblogic.Deployer`, simply specify the new list of target servers with the `deploy` command. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name mydeploymentname -deploy
   -targets server1, newserver
```

Redeploying or Stopping a Deployment Unit

To redeploy or stop a deployment unit that is already deployed in the domain, you reference the deployment name rather than the actual archive file or exploded directory. When redeploying a deployment unit, target servers use the available deployment files (local copies of the files, if the unit was deployed in stage mode, or the original deployment files for nostage mode). During redeployment, the application may be unavailable to clients for a short time while classes are reloaded. WebLogic Server doesn't guarantee the operation of the application and deployment task if there is an access from the client at this time. For this reason, redeployment is not recommended in production environments.

Note that when you stop a deployment unit, it is not available to clients. However, you can later redeploy a stopped deployment unit using the available source files and deployment name; you do not need to reselect the deployment files, as they remain associated with the deployment name in the domain.

Administration Console Tasks

To redeploy or stop a deployment unit using the Administration Console:

1. Start the Administration Console for your domain.

2. Expand the Deployments node in the left pane to display the different deployment units.

3. In the left pane, select the type of deployment unit that you want to stop or redeploy. The available deployment units are:

    – Applications—Enterprise Applications or Web Services packaged as `.ear` files or directories

    – EJB Modules—Enterprise JavaBeans packaged as `.jar` files or directories

    – Web Application Modules—Web Applications or Web Services packaged as `.war` files or directories

    – Connector Modules—Resource Adapters packaged as `.rar` files or directories

    – Startup & Shutdown—Startup classes and shutdown classes

4. In the right pane, select the name of the deployment unit you want to redeploy or stop. A table displays the status of the deployment on all target servers.

5. To redeploy or stop a deployment on a single target server, click the Redeploy or Stop button to the right of the target server name. To redeploy or stop a deployment on all target servers, click the Redeploy All or Stop All button at the bottom of the deployment status table.

Weblogic.Deployer Tasks

To redeploy a deployment unit using the `weblogic.Deployer` utility, use the redeploy command and specify the deployment name. Specifying only the deployment name redeploys an application on all existing target server instances, as in:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name myenterpriseapp -redeploy
```

Note: For applications deployed in a cluster, redeployment occurs on all target server instances in the cluster. If the application was previously deployed to all servers in the cluster, you cannot subsequently redeploy the application on a subset of servers in the cluster.

If an application was previously deployed to multiple, non-clustered server instances, you can specify a target list to redeploy the application on a subset of the target servers, as in:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name mydeploymentname -redeploy
   -targets myserver1,myserver2
```

The `weblogic.Deployer` utility uses a different command form if you want to redeploy individual modules of a deployed Enterprise Application. To redeploy a subset of the modules of

an Enterprise Application, specify *modulename@servername* in the target server list. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name myenterpriseapp -redeploy
   -targets mymodule1@myserver1,mymodule2@myserver2
```

Note: If the application was previously deployed to a cluster, you must redeploy the module to the entire cluster, rather than a subset of servers. If you specify a subset of servers in the cluster, `weblogic.Deployer` responds with the error:

```
An attempt to add server target target_name to module module_name has
been rejected . This is because its parent cluster, cluster_name, is
aso targeted by the module.
```

Undeploying a Deployment Unit

After you deploy a new application or standalone module to servers in a domain, the deployment name remains associated with the deployment files you selected. Even after you stop the deployment on all servers, the files remain available for redeployment using either the Administration Console or `weblogic.Deployer` utility.

If you want to remove a deployment name and its associated deployment files from the domain, you must explicitly undeploy the application or standalone module. If you need to redeploy a deployment unit after deleting it, you must identify the deployment files, staging mode, and deployment name using the instructions in "Deploying an Application or Standalone Module to Servers on the Same Machine" on page 3-3 or "Deploying an Application or Standalone Module to Servers on Multiple Machines" on page 3-5. Undeploying a deployment unit does not remove the original source files used for deployment. It only removes the deployment's configuration from the domain, as well as any deployment files created by the system (for example, files copied with stage deployment mode).

Note: An application becomes unavailable to clients during undeployment. WebLogic Server doesn't guarantee the operation of the application and deployment task if there is an access from the client at this time.

Administration Console Tasks

To undeploy a deployment unit from the domain:

1. Expand the Deployments node in the left pane to display the different deployment units.

2. In the left pane, select the type of deployment unit that you want to undeploy.

3. In the left pane, right-click the name of the deployment you want to remove, and select Delete *deployment_name...* from the menu.

4. In the right pane, select Yes to remove the application or module.

Weblogic.Deployer Tasks

To undeploy a deployment unit from the domain using `weblogic.Deployer`, specify the `undeploy` option. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
    -password weblogic -name mymodule -undeploy
```

Deploying Modules of an Enterprise Application to Different WebLogic Server Instances

An Enterprise Application (EAR file) differs from other deployment units because the `.ear` file can contain other module types (WAR and JAR archives). When you deploy an Enterprise Application using the Administration Console, you can target all of the archive's modules together, or distribute the application by targeting different modules to different servers in the domain.

Administration Console Tasks

1. Start the Administration Console for your domain.

2. Expand the Deployments node in the left pane to display the different deployment units.

3. In the left pane, select the Applications node to display the currently-deployed Enterprise Application in your domain.

4. In the right pane, select the Deploy a new Application link. This initiates the Enterprise Application Deployment Assistant.

5. In the first step of the Deployment Assistant, select the EAR file or exploded Enterprise Application directory that you want to deploy. See Deploying Applications and Modules in the *Administration Console Online Help* if you need further instructions about using the Deployment Assistants.

    After you have selected the deployment files, click the Target Each Module button to begin targeting the individual JAR and WAR modules included in the Enterprise Application.

6. In the second page of the Deployment Assistant, select the servers and clusters to which you want to deploy the current module. Note that your selection of target servers applies only to the module name displayed at the top of the page.

    Click Continue to select target servers for the next module in the Enterprise Application.

7. Repeat step 6 for each module included in the Enterprise Application. When you have finished targeting all available modules, continue with the next step.

8. After you have finished targeting each module, you can review your choices. The Deployment Targets section of the page displays each module in the Enterprise Application and its associated targets. If you need to change the target servers for one or more modules, click your browser's back button and reselect the targets.

9. Select one of the following options in the Source accessibility section to set the deployment staging mode for the application:

    – **Copy this application onto every target for me**—This option corresponds to stage mode, where the Administration Server copies the deployment files to each target server before the servers deploy the application.

    – **I will make the application accessible from the following location**—This option corresponds to nostage mode, where all targets deploy the files from the specified location.

10. Accept the default name for the Enterprise Application, or enter a new name to use to refer to this application in the Administration Console.

11. Click Deploy to deploy all modules.

Weblogic.Deployer Tasks

When deploying modules to different Weblogic Server instances using the `weblogic.Deployer` utility, specify individual modules using the *module_name*@*target_name* syntax. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name myenterpriseapp
   -targets module1@myserver1,module2@myserver2,module3@myserver3
   -stage -deploy c:\localfiles\myapp.ear
```

When you specify Web Application modules that are part of an .ear file, you must use the Web Application's context root name as the module name. For example, if the application.xml file for a file, myapp.ear, defines:

```
<module>
  <web>
    <web-uri>myweb.war</web-uri>
    <context-root>/welcome</context-root>
  </web>
</module>
```

you could deploy only the Web Application by using a command similar to:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
  -password weblogic -name mywebapplication -targets welcome@myserver1
  -stage -deploy c:\localfiles\myapp.ear
```

Deploying an Enterprise Application Using external_stage Mode

Use external_stage mode when you want to manually copy very large files to target servers before deployment to reduce the time required to deploy the application. You can also use external_stage mode if you have a third-party application or an automated script that copies files for you. You cannot use the Administration Console to deploy in external_stage mode.

To deploy an application using external_stage mode:

1. Make sure that the deployment files are accessible to the Administration Server.

2. On each target server for the deployment, create a subdirectory in the staging directory that has the same name as the deployment name. For example, if you will specify the name myEARExternal for the deployment name, create a myEARExternal subdirectory in the staging directories for each target server.

   Note: If you do not specify a deployment name at deployment time, WebLogic Server selects a default name. See "Deployment Names" on page 1-12 for more information.

3. Copy the deployment files into the staging subdirectories you created in Step 2 above.

4. Deploy the application or module using the weblogic.Deployer utility. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -name weblogic
  -password weblogic -external_stage -name myEARExternal
  -deploy c:\myapps\myear
```

Deploying an Enterprise Application with an Alternate (External) Application Descriptor

Alternate deployment descriptor files change the run-time configuration of an application without changing the application's contents. See "Deploying Enterprise Applications With Alternate Deployment Descriptors" on page 4-11 for more information.

To specify alternate deployment descriptors (external to the deployed EAR file) when deploying an Enterprise Application, specify the file names with the weblogic.Deployer command. Note that you cannot specify alternate descriptor files when deploying with the Administration Console.

weblogic.Deployer Tasks

To specify an external application.xml and weblogic-application.xml file when deploying an Enterprise Application, use the command:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name myapplication
   -targets myserver1,myserver2,myserver3 -stage -deploy
   -altappdd c:\myfiles\myextapplication.xml
   -altwlsappdd c:\myfiles\myextwlsapplication.xml c:\localfiles\myapp.ear
```

Deploying a Module Newly Added to an EAR

If you add the module newmodule.war to a deployed application named myapp.ear, and you update the module in the application.xml file, you can deploy newmodule.war by using the weblogic.Deployer command:

```
java weblogic.Deployer -username myname -password mypassword
   -name myapp.ear -deploy -targets newmodule.war@myserver
   -source /myapp/myapp.ear
```

Note: This command deploys the new module without redeploying the other modules in the application. You must specify the correct file extension (.war in the above example) for archived modules in an EAR file.

Changing the Order of Deployment for a Deployment Unit

You can change the deployment order for a deployed application or *standalone* module by setting the ApplicationMBean LoadOrder attribute in the Administration Console. The LoadOrder attribute controls the load order of deployments relative to one another—modules with lower

LoadOrder values deploy before those with higher values. In all cases, applications and standalone modules are deployed after the WebLogic Server instance has initialized dependent subsystems.

Notes: Modules *within* a deployed Enterprise Application are loaded in the order specified by the application.xml file; you cannot change this ordering using the Administration Console.

You cannot change the load order of applications and standalone modules using the weblogic.Deployer utility.

Modules deployed in versions of WebLogic Server prior to 7.0 specify the load order value in their deployment descriptor files; you cannot change this load order using the Administration Console.

Administration Console Tasks

Follow these steps to view or change the deployment order of deployments in the WebLogic Server domain:

1. Select the Deployments node in the left pane. The right pane displays all applications and standalone modules configured for deployment in the domain, listed in their current deployment order.

2. Click the Change button next to a deployment name to display the Change Deployment Order page.

3. Enter a new value in the Load Order field, and click Apply to apply your changes. The right pane again displays the complete list of deployment units configured for the domain.

Changing a Server Staging Mode or Staging Directory

The server staging mode specifies the default deployment mode for a server if none is specified at deployment time. For example, the server staging mode is used if you deploy an application or standalone module using weblogic.Deployer and you do not specify a staging mode. See "Deployment Staging Modes and Staging Directories" on page 4-5 for help on when to use staging modes.

Notes: You can only change the server staging mode by using the Administration Console or by directly changing the `ServerMBean` via JMX.

Changing the server staging mode does not affect existing applications. If you want to change the staging mode for an existing application, you must undeploy the application deployment and then redeploy it with the new staging mode.

Administration Console Tasks

To set the server staging mode:

1. Expand the Servers node in the left pane.

2. Select the name of the server instance that you want to configure.

3. Select the Configuration->Deployment tab in the right pane to display the current staging mode.

4. Select stage, nostage, or external_stage from the Staging Mode menu. These modes correspond to the staging modes described in "Deployment Staging Modes and Staging Directories" on page 4-5, and apply only to the selected server instance.

5. Enter a path in the Staging Directory Name attribute to store staged deployment files. The path is relative to the root directory of the selected server.

6. If you are configuring the staging mode for the Administration Server, also specify an Upload Directory Name, relative to the server's root directory. This is the directory where the Administration Server stores uploaded files for deployment to servers and clusters in the domain.

7. Click Apply to change the staging mode and directory.

Dynamically Updating Descriptors for a Deployment Unit

When you deploy an application or standalone module using an exploded archive directory, the Administration Console allows you to directly edit selected deployment descriptors in a production environment. The changes you make are automatically persisted to the module's deployment descriptor files, and the changes are dynamically applied to all target servers that host the module.

Note: You cannot use the Administration Console to edit deployment descriptors for applications and modules that are deployed from archive files.

To edit deployment descriptors for a deployed Enterprise Application:

1. Deploy the Enterprise Application as an exploded archive directory. If the application contains multiple EJB, Web Application, or Resource Adapter modules, make sure that those modules also reside as exploded directories having the correct subdirectory names. (Module subdirectories must match the URI values specified in the Enterprise Application's `application.xml` descriptor).

2. In the Administration Console, expand the Deployments node in the left pane.

3. Expand the node in the left pane that corresponds to the deployed Enterprise Application. This displays all of the modules contained in the enterprise application.

4. In the left pane, select the name of the EJB, Web Application, or Resource Adapter module whose descriptors you want to edit.

5. In the right pane, select the Configuration->Descriptor tab.

6. Use the Configuration->Descriptor tab to make available changes to the module's deployment descriptors, and click Apply. Your changes are automatically persisted to the module's XML deployment descriptor file, and the changes are reflected in all WebLogic Server instances that host the deployed module.

Redeploying Static Files in a Web Application

In a production environment, you may occasionally need to refresh the static content of a Web Application—HTML files, image files, and so forth—without redeploying the entire application. If you deployed the Web Application as an exploded archive directory, you can use the `weblogic.Deployer` utility to notify the server that static files have changed. See Avoiding Unnecessary JSP Recompilation on dev2dev.com.

weblogic.Deployer Tasks

To redeploy a single file associated within a deployment unit, specify the file name at the end of the redeploy command. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name mywebapp -redeploy mywebapp/copyright.html
```

Always specify the pathname to updated files relative to the root directory of the exploded archive directory. In the above example, the Web Application is deployed as part of an Enterprise Application, so the module directory is specified (`mywebapp/copyright.html`). If the Web Application module had been deployed standalone, rather than as part of an Enterprise Application, the file would have been specified alone (`copyright.html`).

You can also redeploy an entire directory of files by specifying a directory name instead of a single file:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name mywebapp -redeploy mywebapp/myjsps
```

In the above example, all files and subdirectories located in the myjsps subdirectory of the Enterprise Application are redeployed.

Note: The refresh tools are deprecated and will be removed in a future WebLogic Server release. Use weblogic.Deployer instead.

Removing Files from a Web Application Deployment

If you deploy a Web Application using an exploded archive directory, you can update static contents of the Web Application either by refreshing the files (see "Redeploying Static Files in a Web Application" on page 3-17), or by deleting files from the deployment. To delete files, you must use the weblogic.Deployer utility with the delete_files option. For example:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name mywebapp -delete_files mywebapp/copyright.html
```

Always specify the pathname to updated files relative by starting at the top level of the exploded archive directory. In the above example, the Web Application resides in an exploded archive directory named mywebapp.

Managing Deployment Tasks

The WebLogic Server deployment system automatically assigns a unique ID to deployment tasks so that you can track and manage their progress.

Administration Console Tasks

The Administration Console allows you to monitor running and completed tasks, as well as cancel running tasks.

After deploying, redeploying, or stopping a deployment unit, the Deploy tab in the right pane shows the status of the deployment action in the deployment status tables. To view more details about an individual task, click a link in the Status of Last Action column of the table.

To view a complete list of ongoing and completed deployment tasks (as well as other domain configuration tasks) select the Tasks node in the left pane.

weblogic.Deployer Tasks

The `weblogic.Deployer` utility enables you to monitor and cancel running deployment tasks. `weblogic.Deployer` also enables you to assign your own task identification numbers for use with subsequent commands.

The following command assigns task ID 73 to a new deployment action:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -name mymodule -targets myserver -id redeployPatch2
   -nowait -deploy c:\localfiles\myapp.ear
```

You can later check the status of the task using the following command:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -id redeployPatch2 -list
```

You can later check the status of running tasks using the following command:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -listtask
```

If a task takes too long to complete you can cancel it using the command:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
   -password weblogic -id redeployPatch2 -cancel
```

Performing Common Deployment Tasks

Advanced Deployment Topics

The following sections describe more advanced deployment topics that might not be used in every WebLogic Server installation:

Two-Phase Deployment Protocol

WebLogic Server uses a two-phase deployment protocol when deploying applications and standalone modules in a domain. The protocol ensures the consistency of deployments to multiple WebLogic Servers in a domain, and also supports advanced deployment features such as application ordering and improved monitoring.

Each deployment in a WebLogic Server domain takes place in two phases: prepare and activate.

Phase 1: Prepare

The prepare phase ensures that the application and its modules are in a state in which they can be reliably deployed. During the prepare phase, WebLogic Server instances in the domain complete the following tasks:

1. If necessary, the Administration Server copies deployment files to the target servers' staging directories.

2. Each target server validates the deployment files and checks for possible errors to ensure that the files can actually be deployed.

Phase 2: Activate

In the activate phase, target servers make the application available for clients to use. The activate phase constitutes the actual deployment of the application or module.

The activation phase only takes place after the prepare phase successfully completes.

Benefits of Two-Phase Deployment

The primary goal of the two-phase deployment model is to ensure that a deployment to multiple targets (for example, deployments to a cluster of WebLogic Server instances) succeeds or fails as a logical unit on the available target servers. In a cluster, you can ensure that a deployment does not activate if even one of the available servers fails to verify the deployment files or detects an error in the deployment files.

Note: The WebLogic Server 6.x deployment protocol, which is deprecated in WebLogic Server 8.1, does not ensure consistent deployment to clusters. See "WebLogic Server 6.x Deployment Protocol (Deprecated)" on page 4-3 for more information.

The two-phase deployment protocol also enables the following new features for deployed applications:

- **Consistent deployment states for clusters**. If an application targeted to a cluster fails on any of the cluster members in the prepare phase and then in the activate phase, the application is not activated on any of the cluster members. This helps to ensure that the cluster is kept homogeneous. You can also ensure that a deployment fails if any member of the cluster is not available at the time of deployment; see "Enforcing Cluster Constraints with Two-Phase Deployment" on page 4-4 for more information.

- **Application ordering**. At server startup, you set the order of application activations. See "Deployment Order" on page 4-7.

- **Application-scoped configuration**. Certain resources can be configured and scoped for an application. These include connection pools, security realms and XML related resources. See Overview of Application Scoping in *Programming WebLogic XML*.

- **Improved redeployment**. You do not need to undeploy before redeploying. See "Redeploying or Stopping a Deployment Unit" on page 3-8.

- **Improved API**. A simple API separates configuration from the actual deployment operations. When a deployment is requested, this API creates the necessary configuration (MBeans) for you. Also, the deployment operations are not on the MBeans themselves, so you can change the configuration (such as the target lists) without affecting the deployed application, until a deployment request is initiated. See "Deployment Management API" on page 5-11, and see also the API documentation for `weblogic.management.deploy.`

- **Deployment status**. It is now easier to track the progress of a deployment especially when it has multiple targets. See the WebLogic Administration Console Online Help for Tasks.

WebLogic Server 6.x Deployment Protocol (Deprecated)

By default, the two-phase deployment protocol is used to deploy new applications by all available deployment tools. The Administration Server also uses the deprecated WebLogic Server 6.x deployment protocol when:

- Pre-configured applications do not specify the two-phase deployment protocol (`ApplicationMBean.TwoPhase=true`), or

- The application contains multiple modules, but no `application.xml` descriptor.

The 6.x deployment protocol is now deprecated. If you have applications that use the 6.x protocol, follow these instructions to use two-phase deployment:

1. Undeploy the application using `weblogic.Deployer`. Enter a command in the following form:

   ```
   java weblogic.Deployer -adminurl http://admin:7001 -username weblogic
       -name app -undeploy
   ```

2. If the earlier deployment did not have an `application.xml` file, create one in the application's `META-INF` subdirectory.

3. Deploy the application using `weblogic.Deployer`. Enter a command in the following form:

```
java weblogic.Deployer -adminurl http://admin:7001 -username weblogic
   -deploy -name ArchivedEarJar -source C:/MyApps/JarEar.ear
   -target server1,server2
```

This command redeploys the application with the new protocol.

Enforcing Cluster Constraints with Two-Phase Deployment

When you deploy to a WebLogic Server cluster, the default two-phase deployment behavior ensures that the deployment succeeds only on clustered server instances that are reachable by the Administration Server. If servers are unreachable at the time of deployment (for example, because of a network failure between the Administration Server and a Managed Server) those servers will not receive the deployment request until the network connection is restored. This can lead to a situation where some servers in the cluster use a new version of a deployment unit, while other, unreachable servers use an older version.

The `ClusterConstraintsEnabled` option enforces a strict two-phase deployment policy for all servers in a domain. `ClusterConstraintsEnabled` ensures that a deployment to a cluster succeeds only if all members of the cluster are reachable and can deploy the specified files.

You can set the `ClusterConstraintsEnabled` for the domain when you start the Administration Server by supplying a startup argument:

- `-DClusterConstraintsEnabled=true` enforces strict cluster deployment for servers in a domain.

- `-DClusterConstraintsEnabled=false` uses the default two-phase deployment behavior for servers in a domain; deployment is ensured only for clustered servers that are reachable by the Administration Server.

You can also set the `ClusterConstraintsEnabled` option using the Administration Console:

1. Log into the Administration Console for the domain.

2. Select the name of your domain at the top of the left pane.

3. Select the Configuration->General tab in the right pane.

4. Use the Enable Cluster Constraints box to enable or disable cluster constraints for the entire domain.

5. Click Apply and reboot the Administration Server to apply the change.

Deployment Staging Modes and Staging Directories

The deployment staging mode determines whether or not the Administration Server copies deployment files to target servers during the activate phase of a deployment. WebLogic Server provides three different options for staging archive files: stage mode, nostage mode, and external_stage mode, as described in "Staging Modes" on page 1-9.

A server's staging directory is the directory in which the Administration Server copies deployment files for stage mode deployments. It is also the directory in which deployment files must reside before deploying an application using external_stage mode.

Stage Mode

In stage mode, the Administration Server copies the deployment files from their original location on the Administration Server machine to the staging directories of each target server. For example, if you deploy a J2EE Application to three servers in a cluster using stage mode, the Administration Server copies the deployment files to directories on each of the three server machines. Each server then deploys the J2EE Application using its local copy of the archive files.

When copying files to the staging directory, the Administration Server creates a subdirectory with the same name as the deployment name. So if you deployed using the command:

```
java weblogic.Deployer -adminurl http://localhost:7001 -user weblogic
    -password weblogic -name mytestear -stage -targets mycluster
    -deploy c:\bea\weblogic81\samples\server\medrecd\dist\physicianEar
```

a new directory, `mytestear`, would be created in the staging directory of each server in `mycluster`. If you do not specify a deployment name, a default deployment name (and staging subdirectory) is used:

- For exploded archive deployments, the deployment name and staging subdirectory are the name of the directory you deployed (*physicianEar* in the example above).

- For archived deployments, the default deployment name is the name of the archive file without the extension. For example, if you deploy `physicianEar.ear`, the deployment name and staging subdirectory are `physicianEar`.

The Administration Console uses stage mode as the default mode when deploying to more than one WebLogic Server instance. `weblogic.Deployer` uses the target server's staging mode as the default, and managed servers use stage mode by default.

Stage mode ensures that each server has a local copy of the deployment files on hand, even if a network outage makes the Administration Server unreachable. However, if you are deploying very large applications to multiple servers or to a cluster, the time required to copy files to target

servers can be considerable. You may consider nostage mode to avoid the overhead of copying large files to multiple servers.

Nostage Mode

In nostage mode, the Administration Server does not copy the archive files from their source location. Instead, each target server must access the archive files from a single source directory for deployment. The staging directory of target servers is ignored for nostage deployments.

For example, if you deploy a J2EE Application to three servers in a cluster, each server must be able to access the same application archive files (from a shared or network-mounted directory) to deploy the application.

Note: The source for the deployment files in nostage mode is the path provided by the user at deployment time (as opposed to stage mode, where the source is the path in each server's staging directory). However, even in nostage mode, WebLogic Server copies out parts of the deployment to temporary directories. This enables users to update entire archived deployments or parts of archived deployments.

In nostage mode, the Web Application container automatically detects changes to JSPs and servlets. Nostage also allows you to later update only parts of an application by updating those parts in one file system location and then redeploying.

The Administration Console uses nostage mode as the default when deploying only to the Administration Server (for example, in a single-server domain). `weblogic.Deployer` uses the target server's staging mode, and Administration Servers use stage mode by default. You can also select nostage mode if you run a cluster of server instances on the same machine, or if you are deploying very large applications to multiple machines that have access to a shared directory. Deploying very large applications in nostage mode saves time during deployment because no files are copied.

External_stage mode

External_stage mode is similar to stage mode, in that target servers deploy using local copies of the deployment files. However, the Administration Server does not automatically copy the deployment files to targeted servers in external_stage mode; instead, you must copy the files to the staging directory of each target server before deployment. You can perform the copy manually or using automated scripts.

Within each target server's staging directory, deployment files must be stored in a subdirectory that reflects the deployment name. This can either be the name you type in for the deployment, or the default deployment name (the name of the exploded archive directory, or the name of the

archive file without its file extension). See "Deploying an Enterprise Application Using external_stage Mode" on page 3-13 for an example.

External_stage mode is the least common deployment staging mode. It is generally used only in environments that are managed by third-party tools that automate the required copying of files. You may also choose to use external_stage mode when you are deploying very large applications to multiple machines and you do not have a shared file system (and cannot use nostage mode). Using external stage in this scenario decreases the deployment time because files are not copied during deployment.

Server Staging Modes vs. Application Staging Modes

When you deploy an application or standalone module using the Administration Console, the staging mode is set at the application level. The application staging mode always overrides any deployment mode specified for the target server itself.

The server staging mode specifies the default deployment mode for a server if none is specified at deployment time. For example, the server staging mode is used if you deploy using `weblogic.Deployer` and you do not specify a staging mode. The default value for the server staging mode is nostage for the Administration Server and stage mode for Managed Servers.

Deployment Order

By default, WebLogic Server deploys server-level resources (JDBC followed by JMS) before deploying applications and standalone modules, followed by startup classes. (The order of startup class execution is configurable, as described in "Ordering Startup Class Execution and Deployment" on page 4-8.)

Note: WebLogic Server security services are always initialized before server resources, applications, and startup classes are deployed. For this reason, you cannot configure custom security providers using startup classes, nor can custom security provider implementations rely on deployed server resources such as JDBC.

The deployment order of deployment units relative to one another is determined by the Load Order attribute. By default, each deployment unit is configured with a Load Order value of 100. Deployments with a lower Load Order value are deployed before those with a higher value during startup. Deployments with the same Load Order value are deployed in alphabetical order using the deployment name. You can change the load order for a deployment unit by setting the Load Order attribute in the Administration Console or programmatically using the `ApplicationMBean`. See "Changing the Order of Deployment for a Deployment Unit" on page 3-14 for instructions on changing the Load Order using the Administration Console.

Deployment Order for Modules Within an Enterprise Application

If the application is an EAR, the individual modules are loaded in the order in which they are declared in the application.xml deployment descriptor. See Enterprise Application Deployment Descriptor Elements in *Developing WebLogic Server Applications*.

Ordering Startup Class Execution and Deployment

By default WebLogic Server startup classes are run after the server initializes JMS and JDBC services, and after applications and standalone modules have been deployed.

If you want to perform startup tasks after JMS and JDBC services are available, but before applications and modules have been activated, you can select the Run Before Application Deployments option in the Administration Console (or set the StartupClassMBean's LoadBeforeAppActivation attribute to "true").

If you want to perform startup tasks before JMS and JDBC services are available, you can select the Run Before Application Activations option in the Administration Console (or set the StartupClassMBean's LoadBeforeAppDeployments attribute to "true").

The following figure summarizes the time at which WebLogic Server executes startup classes.

Figure 4-1 Startup Class Execution
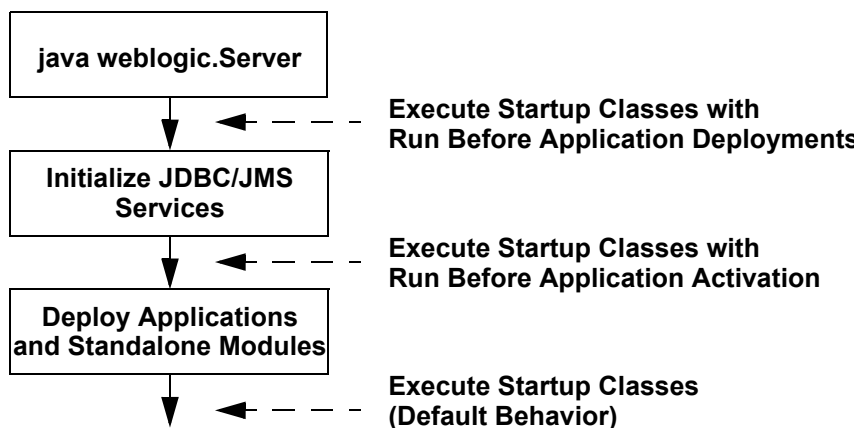
BEA provides the `examples.jms.startup` API code example which demonstrates how to establish a JMS message consumer from a WebLogic startup class.See the *Administration Console Online Help* or the full Javadocs for `StartupClassMBean` for more information.

Note: WebLogic Server 8.1optionally installs API code examples in `WL_HOME\samples\server\examples\src\examples`, where `WL_HOME` is the top-level directory of your WebLogic Server installation. You can start the examples server, and obtain information about the samples and how to run them.

Redeploying Applications and Modules

Once you deploy an application, you can redeploy the application itself or selected parts of the application. Redeploying an entire application involves unloading its classes and then deploying the application again with the changed deployment files. (For applications deployed using stage mode, redeployment also involves re-copying deployment files to target servers.)

Note: An application becomes unavailable to clients during redeployment. WebLogic Server doesn't guarantee the operation of the application and deployment task if there is an access from the client at this time. For this reason, redeployment is not recommended for use in a production environment.

Redeploying a part of an application, such as a Web Application or EJB module in an Enterprise Application, involves unloading only the classes in a particular classloader and reloading those classes using the updated class files. See WebLogic Server Application Classloading in *Developing WebLogic Server Applications* for more information on classloading behavior. Individual files in a Web Application (for instance, a servlet, JSP, or static HTML page) can also be redeployed independently of the rest of the application, without affecting the application classloaders.

See "Redeploying or Stopping a Deployment Unit" on page 3-8 and "Redeploying Static Files in a Web Application" on page 3-17 for example instructions on redeploying applications and modules.

Limitations of Redeploying Applications and Modules

When redeploying applications and modules, consider the following limitations:

- For an application or module that was originally deployed to a WebLogic Server cluster, subsequent redeployment must occur on all members of the cluster. For example, you cannot deploy an application to a WebLogic Server cluster and later redeploy the application or a part of the application on a subset of servers in the cluster.

- Redeploying entire, staged applications may have performance implications due to increased network traffic when deployment files are copied to the Managed Servers.

- By default, WebLogic Server destroys current user sessions when you redeploy a Web Application. If you want to preserve Web Application user sessions through a redeployment, set `save-sessions-enabled` to "true" in the `container-descriptor` stanza of the `weblogic.xml` deployment descriptor file.

- If you change an application's deployment descriptor files, the container redeploys the entire application.

- To redeploy an Enterprise Application, all of the application's modules must redeploy successfully.

- When redeploying only one module of an Enterprise Application, if the module affects other classes loaded in the same classloader, the redeployment is rejected with a message saying you must explicitly redeploy all the affected modules. See WebLogic Server Application Classloading in *Developing WebLogic Server Applications* for more information about application classloading behavior.

- If you redeploy a module of an Enterprise Application that affects other modules (because of classloader dependencies), you must explicitly redeploy the entire application. `weblogic.Deployer` displays the message:

```
Exception:weblogic.management.ApplicationException: [J2EE:160076] You
must include all of [module_list] in your files list to modify [module]
```

Partial Redeployment for Exploded Web Applications

Partial redeployment refers to changing or adding a part of a deployed Web Application, and using the redeploy action to have the change take place. WebLogic Server supports partial redeployment only for Web Applications that are deployed as exploded WAR files. The following kinds of partial redeployment are supported:

- Static files such as HTML files, graphics files, and JSPs that do not modify the session or update code can be redeployed on-the-fly without interrupting clients of the application.

- Classes in the `WEB-INF/classes` directory can be redeployed independently of the rest of the Web Application. You can also deploy only the updated classes (rather than the entire `WEB-INF/classes` directory) by setting the Reload Period for the Web Application. (See Deploying Web Applications in *Developing Web Applications for WebLogic Server* for more information about Reload Period.)

- JAR files in `WEB-INF/lib` cannot be redeployed independently of the rest of the Web Application. The container automatically redeploys the entire application, but maintains the state, when redeploying JAR files in `WEB-INF/lib`.

Deploying Enterprise Applications With Alternate Deployment Descriptors

WebLogic Server enables you to change the run-time deployment configuration of an Enterprise Application without having to modify and repackage the contents of the archive itself. You accomplish this by specifying an alternate deployment descriptor file to use when deploying the Enterprise Application.

An alternate deployment descriptor file can reside anywhere outside a packaged EAR file or exploded EAR directory. (You cannot store alternate Deployment Descriptor files within the archive or directory.) You identify the external EAR descriptor file at deployment time, and WebLogic Server uses the alternate file in place of the deployment's existing (packaged) deployment descriptor. If the files are deployed using stage mode, alternate deployment descriptor files are copied with the deployment files to the top level of the deployment's subdirectory in each target server's staging directory (for example, *domain_directory*/*servername*/stage/myapp). With other staging modes, no copying is performed.

You can specify an alternate deployment descriptor file to use in place of either the standard J2EE deployment descriptor (`application.xml`) or the WebLogic Server deployment descriptor (`weblogic-application.xml`).

Common Uses for Alternate Deployment Descriptors

Alternate deployment descriptors are generally used with archived Enterprise Applications (EAR files), because they enable you to change deployment parameters without repackaging the application itself. However, you can also specify alternate descriptors to use with exploded EAR directories.

Common uses for alternate deployment descriptors include:

- Changing deployment parameters for an encrypted EAR file that is cumbersome to repackage or explode.

- Changing the subset of modules that you choose to deploy from the EAR. For example, although an EAR may contain multiple, logical applications, you can use an alternate

deployment descriptor to deploy only a subset of available applications to a particular WebLogic Server instance.

Limitations of Alternate Deployment Descriptors

If you deploy an Enterprise Application using an alternate descriptor, you cannot change the alternate descriptor file's location when you redeploy the application. For example, you cannot use the `weblogic.Deployer -redeploy` command and specify a different filename or path to the alternate deployment descriptor file. Note, however, that changes to the original descriptor file are implemented during redeployment.

If you have deployed an application with a standard deployment descriptor, you cannot use an alternate descriptor with the deployed application using the `-redeploy` or `-deploy` commands. Instead, you must first undeploy the application, then deploy it with an alternate descriptor file

Command-Line Options for Specifying Descriptors

To use an alternate deployment descriptor, you simply use one or both of the following options to `weblogic.Deployer`:

- `-altappdd`—specifies the name of an alternate J2EE deployment descriptor (`application.xml`).

- `-altwlsappdd`—specifies the name of an alternate WebLogic Server deployment descriptor (`weblogic-application.xml`).

See "Deploying an Enterprise Application with an Alternate (External) Application Descriptor" on page 3-14 for an example of using alternate descriptor files.

Deployment Topics for Developers

This section contains topics for J2EE application developers who are developing and deploying applications to WebLogic Server:

- "Auto-Deployment" on page 4-12
- "Application Lifecycle Events" on page 4-15

Auto-Deployment

Notes: Auto-deployment is a method for quickly deploying an application to a standalone server (Administration Server) for evaluation or testing. It is recommended that this method be

used only in a single-server development environment. Use of auto-deployment in a production environment or for deployment on Managed Servers is not recommended.

BEA recommends that you use the WebLogic Server split development directory and `wldeploy` ant task, rather than auto-deployment, when developing an application. See Introducing the Split Development Directory Structure in *Developing WebLogic Server Applications*.

If auto-deployment is enabled, when an application is copied into the `\applications` directory of the Administration Server, the Administration Server detects the presence of the new application and deploys it automatically (if the Administration Server is running). If WebLogic Server is not running when you copy the application to the `\applications` directory, the application is deployed the next time the WebLogic Server Administration Server is started. Auto-deployment deploys only to the Administration Server.

Notes: Due to the file locking limitations of Windows NT, if an application is exploded, all the modules within the application must also be exploded. In other words, you cannot use auto-deployment with an exploded application or module that contains a JAR file.

Auto-deployment is intended for use with a single server target in a development environment. If you use other tools, such as the Administration Console, to add targets to an auto-deployed, exploded application, redeploying the application does not propagate changes to the new target servers.

Enabling and Disabling Auto-Deployment

You can run a WebLogic Server domain in two different modes: development and production.

Development mode enables a WebLogic Server instance to automatically deploy and update applications that are in the `domain_name/applications` directory (where `domain_name` is the name of a WebLogic Server domain). In other words, development mode lets you use auto-deploy. Production mode disables the auto-deployment feature.

By default, a WebLogic Server domain runs in development mode. To specify the mode for a domain, see Creating and Configuring Domains Using the Configuration Wizard in *Configuring and Managing WebLogic Server*.

Auto-Deploying, Redeploying, and Undeploying Archived Applications

To auto-deploy an archived application, copy its archive file to the `/applications` directory. WebLogic Server automatically sets the application's deployment mode to stage mode.

A deployment unit that was auto-deployed can be dynamically redeployed while the server is running. To dynamically redeploy, copy the new version of the archive file over the existing file in the /applications directory.

To undeploy an archived deployment unit that was auto-deployed, delete the application from the /applications directory. WebLogic Server stops the application and removes it from the configuration.

Auto-Deploying, Redeploying, and Undeploying Applications in Exploded Archive Format

To auto-deploy an application in exploded archive format, copy the entire exploded archive directory to the /applications directory. WebLogic Server automatically deploys exploded archive applications using the nostage deployment mode. If you deploy an exploded EAR directory that contains archived modules (JAR files), the JAR files are locked during the deployment.

When an application has been auto-deployed in exploded archive format, the Administration Server periodically looks for a file named REDEPLOY in the exploded application directory. If the timestamp on this file changes, the Administration Server redeploys the exploded directory.

To redeploy files in an exploded application directory:

1. When you first deploy the exploded application directory, create an empty file named REDEPLOY, and place it in the WEB-INF or META-INF directory, depending on the application type you are deploying:

    An exploded enterprise application has a META-INF top-level directory; this contains the application.xml file.

    An exploded Web application has a WEB-INF top-level directory; this contains the web.xml file.

    An exploded EJB application has a META-INF top-level directory; this contains the ejb-jar.xml file.

    An exploded connector has a META-INF top-level directory; this contains the ra.xml file.

    Note: The REDEPLOY file works only for an entire deployed application or standalone module. If you have deployed an exploded Enterprise Application, the REDEPLOY file controls redeployment for the entire application—not for individual modules (for example, a Web Application) within the Enterprise Application. If you deploy a Web Application by itself as an exploded archive directory, the REDEPLOY file controls redeployment for the entire Web Application.

2. To redeploy the exploded application, copy the updated files over the existing files in that directory.

3. After copying the new files, modify the REDEPLOY file in the exploded directory to alter its timestamp.

When the Administration Server detects the changed timestamp, it redeploys the contents of the exploded directory.

Note: You must touch the REDEPLOY file (alter its timestamp) any time you wish to trigger redeployment of an auto-deployed application. Even if you modify an application while a server is shut down, you must touch REDEPLOY to ensure that changes are applied when the server next starts up.

To undeploy an application that was auto-deployed in exploded format, use the weblogic.Deployer -undeploy command, or use the Administration Console to remove the deployment configuration. Then remove the application files from the /applications directory.

Application Lifecycle Events

Application lifecycle listener events provide handles on which developers can control behavior during deployment, undeployment, and redeployment. This section discusses how you can use the application lifecycle listener events.

Four application lifecycle events are provided with WebLogic Server, which can be used to extend listener, shutdown, and startup classes. These include:

*   Listeners—attachable to any event. Possible methods for Listeners are:

    ```
    public void preStart(ApplicationLifecycleEvent evt) {}
    ```

    – The preStart event is the beginning of the prepare phase, or the start of the application deployment process.)

    ```
    public void postStart(ApplicationLifecycleEvent evt) {}
    ```

    – The postStart event is the end of the activate phase, or the end of the application deployment process. The application is deployed.

    ```
    public void preStop(ApplicationLifecycleEvent evt) {}
    ```

    – The preStop event is the beginning of the deactivate phase, or the start of the application removal or undeployment process.

    ```
    public void postStop(ApplicationLifecycleEvent evt) {}
    ```

– The postStop event is the end of the remove phase, or the end of the application removal or undeployment process.

● Shutdown classes only get postStop events.

● Startup classes only get preStart events.

Note: For Startup and Shutdown classes, you only implement a `main{}` method. If you implement any of the methods provided for Listeners, they are ignored.

Note: No `remove{}` method is provided in the ApplicationLifecycleListener, because the events are only fired at startup time during deployment (prestart and poststart) and shutdown during undeployment (prestop and poststop).

Registering Events in weblogic-application.xml

In order to use these events, you must register them in the `weblogic-application.xml` deployment descriptor. See "Application Deployment Descriptor Elements." Define the following elements:

● `listener`—Used to register user defined application lifecycle listeners. These are classes that extend the abstract base class
`weblogic.application.ApplicationLifecycleListener`.

● `shutdown`—Used to register user-defined shutdown classes.

● `startup`—Used to register user-defined startup classes.

Basic Functionality

You create a listener by extending the abstract class (provided with WebLogic Server) `weblogic.application.ApplicationLifecycleListener`. The container then searches for your listener.

You override the following methods provided in the WebLogic Server `ApplicationLifecycleListener` abstract class to extend your application and add any required functionality:

● `preStart{}`

● `postStart{}`

● `preStop{}`

● `postStop{}`

Listing 4-1 illustrates how you override the `ApplicationLifecycleListener`. In this example, the public class `MyListener` extends `ApplicationLifecycleListener`.

Listing 4-1   MyListener

```
import weblogic.application.ApplicationLifecycleListener;

import weblogic.application.ApplicationLifecycleEvent;

public class MyListener extends ApplicationLifecycleListener {

  public void preStart(ApplicationLifecycleEvent evt) {

     System.out.println

     ("MyListener(preStart) -- we should always see you..");

   } // preStart

  public void postStart(ApplicationLifecycleEvent evt) {

     System.out.println

     ("MyListener(postStart) -- we should always see you..");

   } // postStart

  public void preStop(ApplicationLifecycleEvent evt) {

     System.out.println

     ("MyListener(preStop) -- we should always see you..");

   } // preStop

  public void postStop(ApplicationLifecycleEvent evt) {

     System.out.println

     ("MyListener(postStop) -- we should always see you..");

   } // postStop

   public static void main(String[] args) {

     System.out.println

     ("MyListener(main): in main .. we should never see you..");

   } // main
```

```
}
```

Listing 4-2 illustrates how you implement the shutdown class. The shutdown class is attachable to preStop and postStop events. In this example, the public class MyShutdown extends ApplicationLifecycleListener.

Listing 4-2   MyShutdown

```
import weblogic.application.ApplicationLifecycleListener;

import weblogic.application.ApplicationLifecycleEvent;

public class MyShutdown extends ApplicationLifecycleListener {

   public static void main(String[] args) {

     System.out.println

     ("MyShutdown(main): in main .. should be for post-stop");

   } // main

}
```

Listing 4-3 illustrates how you implement the startup class. The startup class is attachable to preStart and postStart events. In this example, the public class MyStartup extends ApplicationLifecycleListener.

Listing 4-3   MyStartup

```
import weblogic.application.ApplicationLifecycleListener;

import weblogic.application.ApplicationLifecycleEvent;

public class MyStartup extends ApplicationLifecycleListener {

   public static void main(String[] args) {

     System.out.println

     ("MyStartup(main): in main .. should be for pre-start");
```

```
    } // main

}
```

Configuring Lifecycle Events: URI Parameter

The following examples illustrate how you configure application lifecycle events in the
`weblogic-application.xml` deployment descriptor file. The URI parameter is not required.
You can place classes anywhere in the application `$CLASSPATH`. However, you must ensure that
the class locations are defined in the `$CLASSPATH`. You can place listeners in
`APP-INF/classes` or `APP-INF/lib`, if these directories are present in the EAR. In this case,
they are automatically included in the `$CLASSPATH`.

The following example illustrates how you configure application lifecycle events using the URI
parameter. In this case, the archive `foo.jar` contains the classes and exists at the top level of the
EAR file. For example: `myEar/foo.jar`

Listing 4-4   Configuring Application Lifecycle Events Using the URI Parameter

```
<listener>

     <listener-class>MyListener</listener-class>

     <listener-uri>foo.jar</listener-uri>

</listener>

<startup>

     <startup-class>MyStartup</startup-class>

     <startup-uri>foo.jar</startup-uri>

</startup>

<shutdown>

     <shutdown-class>MyShutdown</shutdown-class>

     <shutdown-uri>foo.jar</shutdown-uri>

</shutdown>
```

The following example illustrates how you configure application lifecycle events without using the URI parameter.

Listing 4-5   Configuring Application Lifecycle Events without Using the URI Parameter

```
<listener>

     <listener-class>MyListener</listener-class>

</listener>

<startup>

     <startup-class>MyStartup</startup-class>

</startup>

<shutdown>

     <shutdown-class>MyShutdown</shutdown-class>

</shutdown>
```

Application Lifecycle Event Behavior During Re-deployment

Application lifecycle events are only triggered if a full re-deployment of the application occurs. During a full re-deployment of the application—provided the application lifecycle events have been registered—the application lifecycle first commences the shutdown sequence, next re-initializes its classes, and then performs the startup sequence.

For example, if your listener is registered for the full application lifecycle set of events (preStart, postStart, preStop, postStop), during a full re-deployment, you see the following sequence of events:

1. preStop{}

2. postStop{}

3. Initialization takes place. (Unless you have set debug flags, you do not see the initialization.)

4. preStart{}

5. postStart{}

Deployment Tools Reference

The following sections describe tools for deploying applications and standalone modules to
WebLogic Server:

- "weblogic.Deployer Utility" on page 5-1

- "wldeploy Ant Task" on page 5-7

- "WebLogic Builder" on page 5-11

- "Deployment Management API" on page 5-11

weblogic.Deployer Utility

The `weblogic.Deployer` utility replaces the earlier `weblogic.deploy` utility, which has been
deprecated. The `weblogic.Deployer` utility is a Java-based deployment tool that provides a
command-line interface to the WebLogic Server "Deployment Management API" on page 5-11.
This utility was developed for administrators and developers who need to initiate deployment
from the command line, a shell script, or any automated environment other than Java.

Using weblogic.Deployer Utility

To use the `weblogic.Deployer` utility:

1. Set up your local environment so that WebLogic Server classes are in your system
   `CLASSPATH` and the JDK is available. You can use the `setenv` script located in your server's
   `/bin` directory to set the `CLASSPATH`.

2. Use the following command syntax:

```
% java weblogic.Deployer [options] [actions] [file(s)]
```

You can also list specific files in the archive that are to be deployed (or redeployed, or undeployed). The file list can include file names and directories relative to the root of the application. If you specify a directory, its entire subtree is deployed or redeployed.

weblogic.Deployer Actions and Options

The following table lists `weblogic.Deployer` options.

Note: The `activate` and `deactivate` actions have been deprecated. Use `deploy` or `start` instead of `activate`. Use `undeploy` or `stop` instead of `deactivate`.

Table 5-1 weblogic.Deployer Options

Option	Description
-adminurl <protocol>://<server>:<port>	The URL of the Administration Server. Default is http://localhost:7001.
-username <user>	The Administrator username. If you supply the -username option but you do not supply a corresponding -password option, weblogic.Deployer prompts you for the password.
-password <password>	The password of the Administrator user. To avoid having the plain text password appear in scripts or in process utilities such as ps, first store the username and encrypted password in a configuration file using the STOREUSERCONFIG command with weblogic.Admin. Omit both the -username and -password options to weblogic.Deployer to use the values stored in the default configuration file. See STOREUSERCONFIG in the *weblogic.Admin Command-Line Reference* for more information on storing and encrypting passwords. If you want to use a specific configuration file and key file, rather than the default files, use the -userconfigfile and -userkeyfile options to weblogic.Deployer.
-name <deployment name>	Specifies the deployment name for the application or standalone module. This can be the name of an existing, configured application or the name to use when creating a new configuration.

Option	Description
-targets `<target list>`	A comma-separated list of the target servers and/or cluster names (`<server1>`,`<cluster1>`,...). Each target may be qualified with a J2EE module name (`<module1>@<server1>`). This enables different modules of the archive to be deployed on different servers. For an application that is currently deployed, `-targets` defaults to all current targets. For an application that you are deploying for the first time, `-targets` defaults to the Administration Server.

-delete_files
`<file list>`
	Used after the `-redeploy` action, this option removes static files from a server's staging directory. `delete_files` is valid only for unarchived deployments, and only for applications deployed using stage mode. You must specify target servers when using this option, as shown in the following example:
	
	`java weblogic.Deployer -adminurl`
	`http://myserver:7001 -username weblogic`
	`-password weblogic -name myapp`
	`-targets myapp@myserver -redeploy`
	`-delete_files myapp/tempindex.html`
	
	`delete_files` only removes files that WebLogic Server copied to the staging area during deployment. If you use the `delete_files` option with an application that was deployed using either nostage or external_stage mode, the command does not delete the files.
	
	`delete_files` can only be used in combination with the `redeploy` action.

-userconfigfile `<path to file>`	Specifies the location of a user configuration file to use for the administrative username and password. Use this option, instead of the `-user` and `-password` options, in automated scripts or in situations where you do not want to have the password shown on-screen or in process-level utilities such as `ps`. Before specifying the `-userconfigfile` option, you must first generate the file using the `weblogic.Admin STOREUSERCONFIG` command as described in STOREUSERCONFIG **in the** *weblogic.Admin Command-Line Reference*.

Option	Description
-userkeyfile <path to file>	Specifies the location of a user key file to use for encrypting and decrypting the username and password information stored in a user configuration file (the -userconfigfile option). Before specifying the -userkeyfile option, you must first generate the key file using the weblogic.Admin STOREUSERCONFIG command as described in STOREUSERCONFIG **in the** *weblogic.Admin Command-Line Reference*.
-verbose	Displays additional progress messages, including details about the prepare and activate phases of the deployment.
-output <raw>	Specify either raw or formatted to control the appearance of weblogic.Deployer output messages. Both output types contain the same information, but raw output does not contain embedded tabs. By default, weblogic.Deployer displays raw output.
-debug	Display debug messages in the standard output.
-help	Prints command-line help text for the most commonly-used weblogic.Deployer actions and options.
-advanced	Prints full command-line help text for all weblogic.Deployer actions and options.
-examples	Display example command lines for common tasks.
-upload	Transfers the specified source file(s) to the Administration Server. Use this option when you are on a different machine from the Administration Server and you cannot copy the deployment files by other means. The application files are uploaded to the WebLogic Server Administration Server's upload directory prior to deployment.
-remote	Indicates that weblogic.Deployer is not running on the same machine as the Administration Server, and that source paths specified in the command are valid for the Administration Server machine itself. If you do not use the -remote option, weblogic.Deployer assumes that all source paths are valid paths on the local machine.
-nostage	Does not copy the deployment files to target servers; instead, deploys them from the fixed location specified using the -source option. nostage is the default when you deploy to the Administration Server (for example, in a single-server domain).

Option	Description
-stage	Copies deployment files to target servers' staging directories before deployment. stage is used as the default when deploying to Managed Server targets.
-external_stage	Does not copy the deployment files to target servers; instead, you must ensure that deployment files have been copied to the correct subdirectory in the target servers' staging directories before deployment. You can manually copy the files or use a third-party tool or script.
-nowait	weblogic.Deployer prints the task ID and exits without waiting for the action to complete. This option is used to initiate multiple tasks and then monitor them later with the -list action.
-timeout <seconds>	Specifies the maximum time, in seconds, to wait for the deployment task to complete. After the time expires, weblogic.Deployer prints out the current status of the deployment and exits.
-source <archive file or exploded archive directory>	Specifies the location of the archive file or exploded archive directory to deploy.
-altappdd	Specifies the name of an alternate J2EE deployment descriptor (application.xml) to use for deployment.
-altwlsappdd	Specifies the name of an alternate WebLogic Server deployment descriptor (weblogic-application.xml) to use for deployment.
-id <task identifier>	Specifies the task identifier of a running deployment task. You can specify an identifier with the -deploy, -redeploy, or -undeploy actions, and use it later as an argument to the -cancel or -list actions. Make sure that the identifier is unique to all other running deployment tasks. The system automatically generates a unique identifier if you do not specify one.
-version	Prints version information.

Table 5-2 weblogic.Deployer Actions

Action	Description
-deploy	Deploys or redeploys an application.
-undeploy	Stops the deployment unit and removes staged files and deployment information from all target servers. (You may also use -remove as an alias.)
	Note: An application becomes unavailable to clients during an undeployment. WebLogic Server doesn't guarantee the operation of the application and deployment task if there is an access from the client at this time.
-redeploy	Replaces a running application or part of a running application.
	Note: An application becomes unavailable to clients during redeployment. WebLogic Server doesn't guarantee the operation of the application and deployment task if there is an access from the client at this time. For this reason, redeployment is not recommended for use in a production environment.
-stop	Makes an application inactive and unavailable to clients on target servers. The deployment information and staged files remain available on target servers for subsequent -start, -deploy, -redeploy, or -undeploy actions.
-start	Makes an inactive application available to clients on target servers. -start does not redistribute deployment files to target servers; the files must already be available via an earlier -deploy or -distribute action.
-distribute	Perform only the activate stage of a deployment by copying deployment files to target server staging directories (if necessary) and registering the deployment name. The application is ready for deployment, but is unavailable to clients on the target server. You can later use the -start action to make the application available to clients.
-cancel	Attempts to cancel the task identified by -id if it is not yet completed.
-list	Lists the status of the task identified by -id.

Action	Description
-listtask	Lists running deployment tasks in the domain.
-listapps	Lists all deployment names for applications and standalone modules deployed in the domain.

wldeploy Ant Task

The `wldeploy` Ant task enables you to perform `weblogic.Deployer` functions using attributes specified in an Ant XML file. You can use `wldeploy` along with other WebLogic Server Ant tasks to create a single Ant build script that:

- Builds your application from source, using `wlcompile`, `appc`, and the Web Services Ant tasks.

- Creates, starts, and configures a new WebLogic Server domain, using the `wlserver` and `wlconfig` Ant tasks.

- Deploys a compiled application to the newly-created domain, using the `wldeploy` Ant task.

See Using Ant Tasks to Configure a WebLogic Server Domain in the *WebLogic Server Command Reference* for more information about `wlserver` and `wlconfig`. See Programming Topics in *Developing WebLogic Server Applications* for information about `wlcompile`.

Basic Steps for Using wldeploy

To use the `wldeploy` Ant task:

1. Set your environment.

   On Windows NT, execute the `setWLSEnv.cmd` command, located in the directory `WL_HOME\server\bin`, where `WL_HOME` is the top-level directory of your WebLogic Server installation.

   On UNIX, execute the `setWLSEnv.sh` command, located in the directory `WL_HOME/server/bin`, where `WL_HOME` is the top-level directory of your WebLogic Server installation.

2. In the staging directory, create the Ant build file (`build.xml` by default). If you want to use an Ant installation that is different from the one installed with WebLogic Server, start by defining the `wldeploy` Ant task definition:

```
<taskdef name="wldeploy"
classname="weblogic.ant.taskdefs.management.WLDeploy"/>
```

3. If necessary, add task definitions and calls to the `wlserver` and `wlconfig` tasks in the build
   script to create and start a new WebLogic Server domain. See Using Ant Tasks to Configure
   a WebLogic Server Domain in the *WebLogic Server Command Reference* for information
   about `wlserver` and `wlconfig`.

4. Add a call to `wldeploy` to deploy your application to one or more WebLogic Server
   instances or clusters. See "Sample build.xml Files for wldeploy" on page 5-8 and "wldeploy
   Ant Task Reference" on page 5-9.

5. Execute the Ant task or tasks specified in the `build.xml` file by typing `ant` in the staging
   directory, optionally passing the command a target argument:

   ```
   prompt> ant
   ```

Sample build.xml Files for wldeploy

The following sample shows a `wldeploy` target that deploys an application to a single WebLogic
Server instance:

```
<target name="deploy">
   <wldeploy action="deploy"
      source="${build}/ejb11_basic_statelessSession.ear" name="ejbapp"
      user="a" password="a" verbose="true" adminurl="t3://localhost:7001"
      debug="true" targets="myserver"/>
</target>
```

The following sample shows a more advanced `wldeploy` target that redeploys a single file in an
exploded Web Application:

```
<target name="redeploy">
   <wldeploy action="redeploy"
      source="c:\myapps\myWebApp\" name="myWebApp"
      user="weblogic" password="weblogic" verbose="true"
      adminurl="t3://localhost:7001"
      debug="true" targets="examplesServer">
    <files file="index.jsp"/>
   </wldeploy>
</target>
```

wldeploy Ant Task Reference

The following table describes the attributes of the `wldeploy` Ant task. For more information about the definition of various terms, see "weblogic.Deployer Actions and Options" on page 5-2.

Table 5-3 Attributes of the wldeploy Ant Task

Attribute	Description	Data Type	Required?
action	The deployment action to perform. Valid values are `deploy`, `cancel`, `undeploy`, `redeploy`, `distribute`, `start`, and `stop`.	String	No
adminurl	The URL of the Administration Server.	String	No
altappdd	Specifies the name of an alternate J2EE deployment descriptor (`application.xml`) to use for deployment.	String	No
altwlsappdd	Specifies the name of an alternate WebLogic Server deployment descriptor (`weblogic-application.xml`) to use for deployment.	String	No
debug	Enable `wldeploy` debugging messages.	boolean	No
external_stage	Specifies whether the deployment uses external_stage deployment mode.	boolean	No
files	Specifies a file or list of files on which to perform a deployment action (for example, a list of JSPs to redeploy). See "Sample build.xml Files for wldeploy" on page 5-8.	FileSet	No
id	Identification used for obtaining status or cancelling the deployment.	String	No
name	The deployment name for the deployed application.	String	No
nostage	Specifies whether the deployment uses nostage deployment mode.	boolean	No
nowait	Specifies whether wldeploy returns immediately after making a deployment call (by deploying as a background task).	boolean	No

Table 5-3 Attributes of the wldeploy Ant Task

Attribute	Description	Data Type	Required?
password	The administrative password.	String	No
	To avoid having the plain text password appear in the build file or in process utilities such as ps, first store a valid username and encrypted password in a configuration file using the weblogic.Admin STOREUSERCONFIG command. Then omit both the username and password attributes in your Ant build file. When the attributes are omitted, wldeploy attempts to login using values obtained from the default configuration file.		
	If you want to obtain a username and password from a non-default configuration file and key file, use the userconfigfile and userkeyfile attributes with wldeploy.		
	See STOREUSERCONFIG in the *weblogic.Admin Command-Line Reference* for more information on storing and encrypting passwords.		
remote	Specifies whether the server is located on a different machine. This affects how filenames are transmitted.	boolean	No
source	The source file to deploy.	File	No
stage	Specifies whether the deployment uses stage deployment mode.	boolean	No
targets	The list of target servers to deploy to.	String	No
timeout	The maximum time to wait for a deployment to succeed.	int	No
upload	Specifies whether the source file(s) are copied to the Administration Server's upload directory prior to deployment.	boolean	No
user	The administrative username.	String	No

Table 5-3 Attributes of the wldeploy Ant Task

Attribute	Description	Data Type	Required?
userconfigfile	Specifies the location of a user configuration file to use for obtaining the administrative username and password. Use this option, instead of the user and password attributes, in your build file when you do not want to have the plain text password shown in-line or in process-level utilities such as ps. Before specifying the userconfigfile attribute, you must first generate the file using the weblogic.Admin STOREUSERCONFIG command as described in STOREUSERCONFIG in the *weblogic.Admin Command-Line Reference*.	String	No
userkeyfile	Specifies the location of a user key file to use for encrypting and decrypting the username and password information stored in a user configuration file (the userconfigfile attribute). Before specifying the userkeyfile attribute, you must first generate the key file using the weblogic.Admin STOREUSERCONFIG command as described in STOREUSERCONFIG in the *weblogic.Admin Command-Line Reference*.	String	No
verbose	Specifies whether wldeploy displays verbose output messages.	boolean	No
failonerror	This is a global attribute used by WebLogic Server Ant tasks. It specifies whether the task should fail if it encounters an error during the build. This attribute is set to true by default.	Boolean	No

WebLogic Builder

WebLogic Builder is a WebLogic Server tool for generating and editing deployment descriptors for J2EE applications. It can also deploy applications to single servers.

See the WebLogic Builder Online Help.

Deployment Management API

A deployment task is initiated through a DeployerRuntimeMBean—a singleton (an object for which only one instance exists) that resides on an Administration Server.

`DeployerRuntimeMBean` provides methods for activating, deactivating, and removing an application. These methods return a `DeploymentTaskRuntimeMBean` that encapsulates the request and provides the means for tracking its progress. `DeploymentTaskRuntimeMBean` provides ongoing status of the request through TargetStatus objects, one per target.

The WebLogic Server deployment management API is defined by the following WebLogic Server MBeans:

- `DeployerRuntimeMBean`—programmatic interface to deployment requests. Deployment requests provided through the `DeployerRuntimeMBean` manifest the configuration state into the application and appropriate component configuration MBeans. These MBeans persist the deployment state of applications in the WebLogic Server domain.

- `DeploymentTaskRuntimeMBean`—interface for encompassing deployment tasks.

The deployment management API is asynchronous. The client must poll the status or utilize `ApplicationMBean` notifications to determine when the task is complete.

For more information about WebLogic Server deployment management APIs, see the `weblogic.management.deploy` Javadoc.