BEA WebLogic Server™

BEA WebLogic Server 7.0 Upgrade Guide

Release 7.0
Document Date: June 2002
Revised: June 28, 2002

BEA WebLogic Server 7.0 Upgrade Guide

Part Number	Date	Software Version
N/A	June 28, 2002	BEA WebLogic Server Version 7.0

Contents

About This Document

This document provides procedures and other information you need to upgrade earlier versions of BEA WebLogic Server to WebLogic 7.0. It also provides information about moving applications from an earlier version of WegLogic Server to 7.0.

The document is organized as follows:

- Chapter 1, "Upgrading WebLogic Server 6.x to Version 7.0," describes how to upgrade to WebLogic Server 7.0 from WebLogic Server 6.x.
- Chapter 2, "Upgrading WebLogic Server 4.5 and 5.1 to Version 7.0," describes how to upgrade to WebLogic Server 7.0 from WebLogic Server 4.5 or 5.1.
- Appendix A, "The weblogic.properties Mapping Table," shows which `config.xml, web.xml, or weblogic.xml` attribute handles the function formerly performed by `weblogic.properties` properties.

Audience

This document is written for all users of WebLogic Server 4.5, 5.1, 6.0, and 6.1 who want to upgrade to WebLogic Server 7.0.

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 you enter from the keyboard.
	Examples:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

`monospace italic text`	Variables in code. *Example*: `String` *`CustomerName;`*
UPPERCASE TEXT	Device names, environment variables, and logical operators. *Examples*: 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.

CHAPTER

1 Upgrading WebLogic Server 6.x to Version 7.0

Upgrading WebLogic Server 6.x to version 7.0, under the simplest circumstances, involves changing your WebLogic Server start command scripts and environment settings. In some cases, it is necessary to move your domain directory. Sometimes, upgrading requires changes specific to the subsytem being upgraded.

The following sections contain information necessary to upgrade your system from WebLogic Server 6.x to WebLogic Server 7.0:

- "Upgrading Your WebLogic Server Configuration: Main Steps" on page 1-2
- "Modifying Startup Scripts" on page 1-3
- "Understanding the WebLogic Server 7.0 Directory Structure" on page 1-4
- "Porting an Application from WebLogic Server 6.x to WebLogic Server 7.0" on page 1-4
- "Upgrading Security" on page 1-6
- "Upgrading WebLogic Tuxedo Connector" on page 1-15
- "Additional Upgrade Procedures and Information" on page 1-21

For instructions on how to upgrade the Pet Store application from WebLogic Server 6.1 to WebLogic Server 7.0 and how to upgrade the WebLogic 6.0 and 6.1 Examples Servers to WebLogic Server 7.0, see "Upgrading the Pet Store Application and the Examples Server".

For information on upgrading to WebLogic Platform 7.0 (7.0.0.1), see the Upgrading section of the WebLogic Server FAQs.

Note: Throughout this document "upgrade" refers to upgrading to a later version of WebLogic Server and "port" refers to moving your applications from an earlier version of WebLogic Server to a later version.

Upgrading Your WebLogic Server Configuration: Main Steps

Take the following steps to upgrade from WebLogic Server 6.x to WebLogic Server 7.0:

1. Make a backup copy of your 6.x domain before you begin the upgrade procedure. After you start the server using WebLogic Server 7.0 classes, you cannot downgrade to 6.x.
2. Install WebLogic Server 7.0. See the the *Installation Guide*.

   Note: The installer will prevent you from installing the new version directly over the old version. You must select a new directory location.
3. Modify your 6.x startup scripts to work with WebLogic Server 7.0. See "Modifying Startup Scripts" on page 1-3.
4. Ensure that you have considered differences in the WebLogic Server 7.0 directory structure that may require you to make file location changes before startup. See "Understanding the WebLogic Server 7.0 Directory Structure" on page 1-4.
5. Port your applications to WebLogic Server 7.0. See "Porting an Application from WebLogic Server 6.x to WebLogic Server 7.0" on page 1-4.
6. If necessary, perform other upgrade procedures as described in "Upgrading Security" on page 1-6, "Upgrading WebLogic Tuxedo Connector" on page 1-15, and "Additional Upgrade Procedures and Information" on page 1-21.

To upgrade a cluster of servers, follow the above steps for each server and then follow the steps outlined in *Setting Up WebLogic Clusters* in *Using WebLogic Server Clusters*. In cases where you invoke an application by using RMI/T3 or RMI/IIOP, WebLogic Server 6.1 and 7.0 are interoperable. Within a domain, however, all servers must be of the same version.

For information on upgrading WebLogic Server license files, see *Upgrading Licenses from Previous WebLogic Server Releases* in the *Installation Guide*.

Modifying Startup Scripts

If you used WebLogic Server startup scripts with a previous version of the product, modify them to work with WebLogic Server 7.0.

Modify the startup scripts as described here. For another example of how to modify the startup scripts, see "Upgrading the Pet Store Application and the Examples Server".

Modify the startup scripts as described here.

1. Modify `bea.home` property

   to point to your BEA home directory containing the `license.bea` file for WebLogic Server 7.0. For example:

   ```
   -Dbea.home=C:\bea700
   ```

2. Modify `WL_HOME`

   must point to your WebLogic Server 7.0 installation directory. For example:

   ```
   WL_HOME=c:\bea700\weblogic700
   ```

3. Modify `PATH`

   so that it includes your `%WL_HOME%` 7.0 home. For example:

   ```
   PATH=%WL_HOME%\bin;%PATH%
   ```

4. Modify `CLASSPATH`

   so that it points to the WebLogic Server 7.0 classes. For example:

   ```
   CLASSPATH=%WL_HOME%\lib\weblogic_sp.jar;%WL_HOME%\lib\weblog
   ic.jar
   ```

5. Modify or eliminate any WebLogic Server 6.x startup script directory structure tests. For example, if your script tries to verify a relative path, either fix the directory structure test or remove it.

WebLogic Server 7.0 installs the JVM, JDK 1.3.1_02, with the server installation. The `setenv.sh` scripts provided with the server all point to the JVM. The latest information regarding certified JVMs is available at the *Certifications Page*.

Understanding the WebLogic Server 7.0 Directory Structure

The directory structure in WebLogic Server 7.0 is different from that of 6.x. For complete information on the updated directory structure see *Understanding the WebLogic Server Directory Structure* in *Performing Post-Installation Tasks* in the *Installation Guide*.

If you are booting your WebLogic Server 6.x domain with the WebLogic Server 7.0 environment, the new directory structure is created automatically. However, if you have custom tools or scripts that rely on the WebLogic Server 6.x domain directory structure, you need to update those tools relative to the new directory structure. Similarly, if you have a scripted tool for creating domains in the WebLogic Server 6.x environment, you will have to change those scripts. It is best to use the *Configuration Wizard* which can be scripted.

Porting an Application from WebLogic Server 6.x to WebLogic Server 7.0

Note: Throughout this document, the directory of the new WebLogic Server 7.0 domain that is created is referred to as `domain`.

Use the following steps to port WebLogic 6.x applications on WebLogic Server 7.0:

1. If you have not already installed WebLogic Server 7.0, do so now. See the *Installation Guide* for more information.

   Note: Installing the new version in the exact location of the old version is explicitly prohibited by the installer.

2. Each 6.x and 7.0 domain must have its own separate directory. It is not possible to have multiple `config.xml` files in the same directory.

   a. For each 6.x configuration domain that you wish to port to WebLogic Server 7.0, copy the `/config/`*`domain`* directory to a directory location of your choice. Exclude any directories that begin with a dot ("**.**"), which are files or directories that WebLogic Server has created for internal use.

      This directory is the location of your new domain and will contain all of your configuration information for that domain. If your 6.x config directory is not located in the WebLogic Server 6.x distribution, you may re-use your WebLogic 6.x configuration in WebLogic Server 7.0.

   b. Identify deployment descriptor files (`web.xml` and `weblogic.xml`), because those files may contain file paths to items such as the Java compiler or external files. WebLogic Server configurations rely on a number of files that may be stored on the file system. Typically, these files are persistence repositories (log files, file-based repositories, etc.) or utilities (Java compiler). These files can be configured using fully qualified or relative paths.

      If all external files are defined using relative paths and are located in or below the domain directory, skip the reamainder of this step.

      For external files that are defined using relative paths that are located outside the domain directory, re-create the directory structure relative to the new config directory and copy the associated files into the new directories. For external files that are defined using fully qualified paths, determine whether it is appropriate to re-use these files in the WebLogic Server 7.0 deployment.

      For example, log files and persistence stores can be re-used; however, you may want to update utilities such as the Java compiler to use the latest version. For files that should be updated, use the WebLogic Server 6.x Administration Console to configure the appropriate attribute to use the new file or utility before proceeding to the next step.

3. If you have not already edited the server start scripts, do so now. See "Modifying Startup Scripts" on page 1-3 for instructions.

4. When you deploy applications to WebLogic Server 7.0, use the Administration Console or `weblogic.Deployer` utility to deploy using the new two-phase deployment protocol. The older, WebLogic Server 6.x deployment protocol, utilities, and API are deprecated in WebLogic Server 7.0

Note: WebLogic Server 7.0 will not deploy an application that has errors in its deployment descriptor. Previous versions of WebLogic Server would deploy an application that had errors in its deployment descriptor.

Upgrading Security

WebLogic Server 7.0 has a new security architecture. For specific questions and answers, see the security section of the *Introduction to WebLogic Security*.

WebLogic Server 7.0 detects whether you are upgrading from an earlier WebLogic Server version or whether you are a new customer starting with 7.0. If you are upgrading from WebLogic Server 6.x, WebLogic Server 7.0 runs in Compatibility security, meaning that it allows you to keep your 6.x configuration of users and groups.

However, because some key 6.x security functionality is being deprecated - and because WebLogic Server 7.0 offers improved and expanded security features - users are encouraged to upgrade their security configuration. See the following list of issues and procedures:

- "Booting WebLogic Server in Compatibility Security" on page 1-7
- "ACLs on MBeans" on page 1-7
- "Upgrading from Compatibility Security to WebLogic Server 7.0 Security" on page 1-8
- "Security Realms" on page 1-9
- "Guest User" on page 1-11
- "password.ini File" on page 1-11
- "Upgrading the SSL Protocol" on page 1-12

Note: The WebLogic Server 7.0 examples and PetStore are configured to use the default security configuration. It is not possible to run the WebLogic Server 7.0 examples and PetStore in Compatibility security.

Booting WebLogic Server in Compatibility Security

In previous releases of WebLogic Server, the File realm was configured by default. Therefore, WebLogic Server could use the File Realm to boot even if there was no security realm defined in the `config.xml` file. However, in order to run WebLogic Server in Compatibility security, you need to have either a File realm or an alternative security realm defined in your `config.xml` file. Otherwise, your server may not boot.

If you are unable to boot WebLogic Server in Compatibility security, copy `SerializedSystemini.dat` to your new domain folder and then do one of the following:

- Boot your configuration under WebLogic Server 6.x, letting the server save the `config.xml` file, and then port the saved `config.xml` file to WebLogic Server 7.0.
- Edit your 6.x `config.xml` file to include the following definitions:

```
<Security Name=mydomain Realm=mysecurity/>
<Realm Name=mysecurity FileRealm=myrealm/>
<FileRealm Name=myrealm/>
```

ACLs on MBeans

ACLs on MBeans are not supported in WebLogic Server 7.0. For information on protecting MBeans in WebLogic Server 7.0, see *Protecting System Administration Operations* in the *Administration Guide*.

Upgrading from Compatibility Security to WebLogic Server 7.0 Security

If you want to leverage the new security features in WebLogic Server 7.0, you need to upgrade your existing security realm to a WebLogic Server 7.0 security realm. You upgrade by populating the security providers in WebLogic Server 7.0 with your existing user and group information and defining security policies on resources that reflect the ACLs.

During successful booting of your WebLogic Server 6.x configuration, the Compatibility realm is created as the default security realm. The Compatibility realm contains all your 6.x security data. In addition, a default WebLogic Server 7.0 security realm called myrealm is also created. To upgrade, you need to replace the Compatibility realm with myrealm. From within the WebLogic Server Administration Console:

1. Click on the Realms node.

   The Realms table appears with two security realms configured. The two security realms are the CompatibilityRealm and myrealm. The CompatibilityRealm will have the default attribute set to `true`.

2. Click on the myrealm node.

3. Click on the Providers tab to see the security providers configured for myrealm. By default, the WebLogic security providers are configured in myrealm.

4. Add a user that can boot WebLogic Server to the `Administrators` group. This user replaces the `system` user. To add a user to the `Administrators` group:

   a. Click on the Security node.

   b. Click on the Realms node.

   c. Click on the name of the realm you are configuring (for example, myrealm).

   d. Click on Groups.

      The Groups tab appears. This tab displays the names of all groups defined in the default Authentication provider.

   e. Click on the Administrators group on the Groups tab.

   f. Click the Membership tab to add the user who can boot WebLogic Server to the `Administrators` group.

   g. Click the Apply button to save your changes.

5. Add the users and groups that you had configured in the 6.x security realm to an Authentication provider.
6. Optionally, define roles for your 6.x users and groups. See *Securing WebLogic Resources*.
7. Express 6.x ACLs as security policies. See *Securing WebLogic Resources*.
8. Set myrealm as the default security realm. See *Setting the Default Security Realm* in *Managing WebLogic Security.*
9. Reboot WebLogic Server.

   Each time WebLogic Server is booted and the server is deployed, the roles and security policies are applied. Subsequent access to the server and its methods are constrained by these roles and security policies until they are changed.

Security Realms

The scope of security realms changed in WebLogic Server 7.0. In WebLogic Server 6.x, security realms provided authentication and authorization services. You chose from the File realm or a set of alternative security realms including the Lightweight Data Access Protocol (LDAP), Windows NT, UNIX or RDBMS realms. In addition, you could write a custom security realm.

In WebLogic Server 7.0, security realms act as a scoping mechanism. Each realm consists of a set of configured security providers, users, groups, roles, and security policies. Authentication and Authorization providers within a security realm offer authentication and authorization services.

You have the following choices when upgrading a 6.x security realm to WebLogic Server 7.0:

- Use Compatibility security to access the users, groups, and ACLs configured in an LDAP, Windows NT, UNIX, RDBMS, or custom security realm. The Realm Adapter Authentication provider in the Compatibility realm can access users, groups, and ACLs stored in a 6.x security realm.

For information about using Compatibility security, see "Booting WebLogic Server in Compatibility Security" on page 1-7.

- Use the Realm Adapter Authentication provider with the WebLogic Server 7.0 security providers. This option allows you to use the roles and security policies available in WebLogic Server 7.0 while accessing users and groups stored in an LDAP, Windows NT, UNIX or RDBMS security realm. You also have the option of configuring multiple Authentication providers so you can use your existing 6.x security realm while upgrading users and groups to an Authentication provider in WebLogic Server 7.0.

Note: You will not be able to access ACLs stored in the RDBMS realm, if you configure the Realm Adapter Authentication provider as an authentication provider in a 7.0 security realm. You will have to protect your application resources with roles and security policies.

To use the Realm Adapter Authentication provider in a WebLogic 7.0 security realm:

1. Boot Compatibility security.
2. Ensure that the Realm Adapter Authentication provider in the Compatibility realm is populated with users and groups from your 6.x security realm (check that existing users and groups appear in the Users and Groups table). The user and group information is copied into a `filerealm.properties` file.
3. Click on security realm in which you want to use the Realm Adapter Authentication provider (for example, myrealm).
4. Click Providers-->Authentication providers.
5. Configure a Realm Adapter Authentication provider in the security realm. Give the Realm Adapter Authentication provider the same name it had in the Compatibility realm.
6. Set the Control Flag attribute on the Realm Adapter Authentication provider to OPTIONAL.
7. Set the Control Flag attribute on the WebLogic Authentication provider (referred to as the Default Authenticator in the Administration Console) to SUFFICIENT.
8. Add a user that can boot WebLogic Server to the `Administrators` group. This user replaces the `system` user. For more information, see "Upgrading from Compatibility Security to WebLogic Server 7.0 Security" on page 1-8.

9. Reboot WebLogic Server.
10. Expand the Domains-->Security nodes.
11. Select the General tab.
12. Set the security realm in which you configured the Realm Adapter Authentication provider as the default security realm.
13. Click Apply.

Note: ACLs cannot be upgraded to WebLogic Server 7.0.

Guest User

The `guest` user is no longer supplied by default in WebLogic Server 7.0. To use the `guest` user, you must run in Compatibility security or define the `guest` user as a user in the default Authentication provider for your security realm. For information about defining users, see "Creating Users" in *Securing WebLogic Resources*.

In WebLogic Server 6.x, the `guest` user identified any unauthenticated user (anonymous user) as a guest user and allowed the guest user access to WebLogic Server resources. In 7.0, WebLogic Server distinguishes between the guest user and an anonymous user. To use the guest user as you did in WebLogic Server 6.x, add the guest user to the default Authentication provider and set the following property when starting WebLogic Server:

```
-Dweblogic.security.anonymousUserName=guest
```

Without this command line property, the anonymous user will have the name of `<anonymous>`.

password.ini File

Previous releases of WebLogic Server supported the use of a `password.ini` file for determining the administrative identity of a WebLogic Server deployment. The `password.ini` file is no longer supported in WebLogic Server 7.0. It is replaced by the `boot.properties` file, which contains your username and password in an encrypted format. For more information about using the `boot.properties` file, see

Creating a Boot Identity File in the *Administration Guide*. There is no direct upgrade of the old `password.ini` file because it contained a clear text password and no username.

Upgrading the SSL Protocol

This section contains information on how to upgrade the SSL protocol including instructions for creating a trusted CA Keystore, creating a private key Keystore, and using a CertAuthenticator in Compatibility security.

Creating a Trusted CA Keystore

By default in WebLogic Server 7.0, clients check the server's trusted certificate authority. This check is done whenever a client and server connect using SSL, including when WebLogic Server is acting an a client. For example, when a client is using the SSL protocol to connect to an Apache HTTP Server, the client checks the trusted certificate authorities presented by the server. The client rejects the server's trusted certificate authority if the certificate authority is not trusted by the client. Previous versions of WebLogic Server did not perform this trust validation.

Make the following changes to allow an existing 6.x WebLogic client to use SSL protocol to communicate with a server:

1. Specify the following command-line argument for the client:

   ```
   -Dweblogic.security.SSL.trustedCAKeyStore=absoluteFilename
   ```

   where *`absoluteFilename`* is the name of the keystore that contains the trusted certificate authority

Note: The file format is a keystore NOT a certificate file. The trusted certificate authority must be loaded into the keystore.

2. Load the server's trusted certificate authority into the client keystore. To list trusted certificate authorites in the keystore or to load new trusted certificate authorities into the keystore, use the JDK `keytool` utility.

   To add a trusted certificate authority to a keystore, enter the following at a command prompt:

`keytool -import -trustcacerts -`*`alias`* <some alias name> -*`file`* <the file that contains the trusted CA> -*`keystore`* <the trusted CA keystore> -*`storepass`* <your trusted CA Keystore password>

The trusted certificate authority shipped with WebLogic Server is located in *`WL_HOME`*`/server/lib/cacerts`. Use the following command to add the trusted certificate authority that is shipped with WebLogic Server to a keystore:

`keytool -import -trustcacerts -`*`alias`* <some alias name> -*`file`* <the file that contains the trusted CA> -*`keystore`* `WL_HOME/server/lib/cacerts` -*`storepass`* `changeit`

For more information about `keytool`, see SUN's website at http://java.sun.com/products/jdk/1.2/docs/tooldocs/solaris/keytool.html.

The `trustedCAKeyStore` command-line argument defaults to the JDK's `jre/lib/security/cacerts` keystore for clients. You can add your CAs to the JDK's trusted CA keystore and not specify the command-line argument, or you can create your own trusted CA keystore and point to it with the argument.

For two-way SSL or mutual authentication, in addition to performing the previous two steps on the client side, do either of the following steps on the server side:

- Add `-Dweblogic.security.SSL.trustedCAKeyStore=`*`absoluteFilename`* to the server command line.

  where *`absoluteFilename`* is the name of the trusted CA Keystore

  OR

- Set the `RootCAKeyStoreLocation` attribute when configuring a Keystore provider.

If you do not load the trusted CA certificate into the trusted CA Keystore, you may have problems using the secure port.

Using CertAuthenticator in Compatibility Security

In WebLogic Server 7.0, the CertAuthenticator is called first, before any username/password authentication. Because this is a change in behavior from WebLogic Server 6.x, a CertAuthenticator written for WebLogic Server 6.x may need to change if clients used both two-way SSL and they supplied a username and password for security credentials.

Using a CertAuthenticator that needs to be changed may result in access being denied in WebLogic Server 7.0, but allowed in WebLogic Server 6.x. To change the CertAuthenticator, have it return `NULL` for unrecognized certs, or for all certs, if certs are only being used to make an SSL connection.

In WebLogic Server 7.0, X.509 Identity Assertion is turned off by default. If your WebLogic Server 6.x `config.xml` file used a CertAuthenticator you need to manually configure the X.509 identity assertion when using Compatibility security. You enable X.509 identity assertion by configuring options on the Realm Adapter Authentication provider. In the WebLogic Server Administration Console while running in Compatibility security:

1. Click on the Security node.
2. Click on the Realms node.
3. Click on the CompatibilityRealm node.
4. Click on the Providers node.
5. Click on the Authentication Providers node.
6. Click on the RealmAdapterAuthenticator node.

   The General tab appears.
7. Enter `X.509` in the Active Types box.
8. Click Apply to save your change.
9. Reboot the WebLogic Server.

Cipher Suites

If you have trouble with your Certicom cipher suite, see the information about issue 073360 in the *Release Notes*.

Upgrading WebLogic Tuxedo Connector

You must make the following application and configuration changes to use WebLogic Tuxedo Connector in WebLogic Server 7.0:

- "Start the WebLogic Tuxedo Connector" on page 1-15
- "Convert WebLogic Tuxedo Connector XML Configuration Files" on page 1-16
- "Update Inbound RMI-IIOP Applications" on page 1-17
- "Authenticate Remote Users" on page 1-19
- "Set WebLogic Tuxedo Connector Properties" on page 1-20

Start the WebLogic Tuxedo Connector

Note: For more information on how to configure WebLogic Tuxedo Connector, see *Configuring WebLogic Tuxedo Connector Using the Administration Console* at http://e-docs.bea.com/wls/docs70/wtc_admin/Install.html#WTCuseCLI.

Previous releases of the connector used a WebLogic Server Startup class to start a WebLogic Tuxedo Connector session and a WebLogic Server Shutdown class to end a session. In WebLogic Server 7.0, WebLogic Tuxedo Connector does not use a Startup or a Shutdown class. WebLogic Tuxedo Connector sessions are managed using a WTCServer MBean.

- You start a WebLogic Tuxedo Connector session when you assign a configured WTCServer MBean to a selected server.
- You end a WebLogic Tuxedo Connector session by removing a WTCServer MBean from WebLogic Server or when you shut down WebLogic Server.

For more information on starting and ending a WebLogic Tuxedo Connector session, see *Assign a WTCServer to a Server* at http://e-docs.bea.com/wls/docs70/ConsoleHelp/wtc.html#AssignWTCServer.

Convert WebLogic Tuxedo Connector XML Configuration Files

WebLogic Tuxedo Connector is implemented as a service and no longer utilizes a separate XML configuration file. The `WTCMigrateCF` tool migrates XML configuration file information into the `config.xml` file of an active Administration server. Use the following steps to convert your WebLogic Tuxedo Connector XML configuration file:

1. Set up a WebLogic Server development shell as described in Setting Up your environment.
2. Start an instance of WebLogic Server.
3. Open a new shell window.
4. Start the `WTCMigrateCF` tool. Enter the following command:

```
java -Dweblogic.wtc.migrateDebug weblogic.wtc.gwt.WTCMigrateCF
-url URL -username USERNAME -password PASSWORD -infile CONFIGWTC
[-server SERVERNAME] [-domain DOMAIN] [-protocol PROTOCOL]
[-deploy]
```

The arguments for this command are defined as follows:

Argument	Description
`-Dweblogic.wtc.migrateDebug`	WebLogic property used to turn on `wtc.migrateDebug` mode.
`URL`	URL passed to your server. Example: `\\myServer:7001`
`USERNAME`	User Name passed to your server. Example: `system`
`PASSWORD`	Password passed to your server. Example: `mypasswd`

Argument	Description
`CONFIGWTC`	Fully qualified path and name of the WebLogic Tuxedo Connector XML configuration file to migrate to the `config.xml` file. Example: `d:\bea\weblogic700\server\samples\examples\wtc\atmi\simpapp\bdmconfig.xml`
`SERVERNAME`	Optional. The name of the administration or managed server that you want the new `WTCServer MBean` assigned to. Default: the current active administration server.
`DOMAIN`	Optional. The name of the WebLogic Server domain that you want the new `WTCServer MBean` assigned to. Default: the current active domain.
`PROTOCOL`	Optional. The protocol to use with `URL`. Default: `t3:`
`-deploy`	Optional. Use to target the `WTCServer MBean` to a selected server. If this flag is set, a WebLogic Tuxedo Connector session is immediately started to provide the services specified by the `WTCServer MBean` is immediately started.

When the migration is complete, the migration utility displays:

```
The WTC configuration file migration is done!
No error found!!!
```

The information from the specified XML configuration file is migrated to a WTCServer Mbean and placed in the `config.xml` file of the server specified in step 2.

Update Inbound RMI-IIOP Applications

For more information on how to use inbound RMI-IIOP with the WebLogic Tuxedo Connector, see *Using WebLogic Tuxedo Connector for RMI-IIOP* at http://bernal/stage/docs70/wtc_atmi/CORBA.html.

If you use inbound RMI-IIOP, you must modify the reference object that connects WebLogic Tuxedo Connector instances to Tuxedo CORBA applications. Tuxedo CORBA objects now use the server name to reference remote WebLogic Tuxedo Connector objects. Earlier releases used the `DOMAINID`.

1. Modify the `corbaloc:tgiop` or `corbaname:tgiop` object reference in your `ior.txt` file.

   Example: `corbaloc:tgiop:`*`servername`*`/NameService`

   where *`servername`* is your server name

2. Register the WebLogic Server (WLS) Naming Service by entering the following command:

   `cnsbind -o ior.txt` *your_bind_name*

   where *`your_bind_name`* is the object name from your Tuxedo application.

3. Modify the `*DM_REMOTE_SERVICES` section of your Tuxedo domain configuration file. Replace your WebLogic Server service name, formerly the `DOMAINID`, with the name of your WebLogic Server.

Listing 1-1 Domain Configuration File

```
*DM_RESOURCES
     VERSION=U22

*DM_LOCAL_DOMAINS
     TDOM1 GWGRP=SYS_GRP
     TYPE=TDOMAIN
     DOMAINID="TDOM1"
     BLOCKTIME=20
     MAXDATALEN=56
     MAXRDOM=89

*DM_REMOTE_DOMAINS
     TDOM2 TYPE=TDOMAIN
     DOMAINID="TDOM2"

*DM_TDOMAIN
     TDOM1 NWADDR="<network address of Tuxedo domain:port>"
     TDOM2 NWADDR="<network address of WTC domain:port>"

*DM_REMOTE_SERVICES
"//servername"
```

where *`servername`* is the name of your WebLogic Server.

4. Load your modified domain configuration file using `dmloadcf`.

You are now ready to start your applications.

Authenticate Remote Users

For more information, see *User Authentication* at http://e-docs.bea.com/wls/docs70/wtc_admin/BDCONFIG.html#WTCuserAuth.

WebLogic Tuxedo Connector uses Access Control Lists (ACLs) limit the access to local services within a local domain by restricting the remote domains that can execute these services. The valid values for this parameter are:

- LOCAL
- GLOBAL

ACL Policy is LOCAL

If the WebLogic Tuxedo Connector ACL Policy is set to `Local`, the Tuxedo remote domain `DOMAINID` must be authenticated as a local user. To allow WebLogic Tuxedo Connector to authenticate a `DOMAINID` as a local user, use the WebLogic Server Console to complete the following steps:

1. Click on the Security node.
2. Click on Realms.
3. Select your default security Realm.
4. Click on Users.
5. Click the Configure a new User text link.
6. Click DefaultAuthenticator
7. In the General tab, do the following:
    a. Add the Tuxedo `DOMAINID` in the Name field.
    b. Enter and validate a password.
    c. Click Apply.

ACL Policy is Global

If the WebLogic Tuxedo Connector ACL Policy is `GLOBAL`, the user's security token is passed. No administration changes are required.

Set WebLogic Tuxedo Connector Properties

Note: For more information on how to set WebLogic Tuxedo Connector properties, see *How to Set WebLogic Tuxedo Connector Properties* at http://e-docs.bea.com/wls/docs70/wtc_admin/Install.html#WTCprops.

TraceLevel and PasswordKey are now WebLogic Server properties.

To monitor the WebLogic Tuxedo Connector using the WebLogic Server log file, you must set the tracing level using the WebLogic Server TraceLevel property. For more information, see *Monitoring the WebLogic Tuxedo Connector* at http://e-docs.bea.com/wls/docs70/wtc_admin/troubleshooting.html#1104694.

- `-Dweblogic.wtc.TraceLevel=`*tracelevel*

  where *tracelevel* is a number between 10,000 and 100,000 that specifies the level of WebLogic Tuxedo Connector tracing.

To generate passwords using the weblogic.wtc.gwt.genpasswd utility, you must set a password key using the WebLogic Server PasswordKey property. For more information on how to generate passwords, see *Configuring a WTCPassword MBean* at http://e-docs.bea.com/wls/docs70/wtc_admin/BDCONFIG.html#1111404.

- `-Dweblogic.wtc.PasswordKey=`*mykey*

  where *mykey* is the key value.

Additional Upgrade Procedures and Information

The following sections provide additional information that may be useful about deprecated features, upgrades, and the important changes that have been made in WebLogic Server 7.0.

Note: WebLogic Server 7.0 uses PointBase 4.2 as a sample database and does not bundle the Cloudscape database.

- "ant.jar" on page 1-22
- "Apache Xalan XML Transformer" on page 1-22
- "Apache Xerces XML Parser" on page 1-23
- "Applications Directory" on page 1-23
- "Deployment" on page 1-24
- "EJB 2.0" on page 1-25
- "jCOM" on page 1-27
- "JDBC" on page 1-27
- "JMS" on page 1-28
- "JMX" on page 1-28
- "Jolt Java Client" on page 1-28
- "JSP" on page 1-29
- "Managed Servers" on page 1-29
- "MBean API Change" on page 1-30
- "Security" on page 30
- "Servlets" on page 1-31
- "Thread Pool Size" on page 1-31

- "Web Applications" on page 1-32
- "WebLogic Server Clusters on Solaris" on page 1-33
- "Web Services" on page 1-33
- "Writable config.xml File" on page 1-33
- "Deprecated APIs and Features" on page 1-34
- "Removed APIs and Features" on page 1-36

ant.jar

If you are going to develop new ant tasks, you must manually add the `server/lib/ant.jar` file to your classpath.

Apache Xalan XML Transformer

The built-in transformer in WebLogic Server 7.0 is based on the Apache Xalan 2.2 transformer.

Use of the Xalan APIs directly has been deprecated. If you are still using those APIs and encounter difficulties, you should use the *Java API for XML Processing* (JAXP) to use XSLT.

Changes were made to Apache's Xalan code to enable Xerces and Xalan to work together. You may encounter problems if you use Xalan from Apache, because it will not include these changes.

In general, it is best to use JAXP and to port any vendor-specific code to a neutral API such as JAXP for SAX, DOM, and XSL processing.

Previous versions of WebLogic Server included the unmodified versions of the Xerces parser and Xalan transformer from www.apache.org in the *`WL_HOME`*`\server\ext\xmlx.zip` file. The ZIP file no longer includes these classes and interfaces. Download the unmodified Xerces parser and Xalan transformer directly from the Apache Web site.

Apache Xerces XML Parser

The built-in XML parser for WebLogic Server 7.0 is based on the Apache Xerces 1.4.4 parser. The parser implements version 2 of the SAX and DOM interfaces. Users who used older parsers that were shipped in previous versions may receive deprecation messages.

WebLogic Server 7.0 also includes the WebLogic FastParser, a high-performance XML parser specifically designed for processing small to medium size documents, such as SOAP and WSDL files associated with WebLogic Web services. Configure WebLogic Server to use FastParser if your application handles mostly small to medium size (up to 10,000 elements) XML documents.

Previous versions of WebLogic Server included the unmodified versions of the Xerces parser and Xalan transformer from www.apache.org in the `WL_HOME\server\ext\xmlx.zip` file. The ZIP file no longer includes these classes and interfaces. Download the unmodified Xerces parser and Xalan transformer directly from the Apache Web site.

Applications Directory

In WebLogic Server 6.1 and 7.0 there is a division between runtime modes. The two modes are "development" and "production." The runtime mode is selected using a command line parameter when starting the Weblogic Server (`-Dweblogic.ProductionModeEnabled=true | false`). If this parameter is not set, the server runs in development mode. In development mode the server behavior is consistent with WebLogic Server 6.0. In production mode, however, the auto-deployment feature is disabled. Deployment units in the applications directory that are not explicitly deployed in the configuration repository (`config.xml`) will not be automatically deployed. Note that in WebLogic Server 6.1 and 7.0 the default domain (mydomain) and Pet Store configurations are shipped in production mode. The examples configuration is shipped in development mode.

Deployment

WebLogic Server 7.0 provides a new two-phase deployment model. For more information on this deployment model and other 7.0 deployment features, see *WebLogic Server Deployment*. By default, statically configured applications use the 6.x deployment model. Deployments initiated through the console or the new `weblogic.Deployer` command line utility use the new two-phase deployment model.

WebLogic Server 7.0 will not deploy an application that has any errors in its deployment descriptor. Previous versions of WebLogic Server would deploy an application that had errors in its deployment descriptor. For example, if your 6.x application was missing a reference description stanza in the deployment descriptor, the application will not deploy in the 7.0 server until you add that stanza. A typical stanza looks like:

```
<ejb-reference-description>
<ejb-ref-name>ejb/acc/Acc</ejb-ref-name>
<jndi-name>estore/account</jndi-name>
</ejb-reference-description>
```

Using WebLogic Server 7.0, you can no longer deploy through the console using the 6.x protocol. As a result, you must use the new deployment APIs. If your application is previously deployed in 6.x and you're just starting your server, the applications will get deployed with one-phase deployment. The `weblogic.deploy` and `weblogic.refresh` command line utilities and the `weblogic.management.tools.WebAppComponentRefreshTool` are deprecated in 7.0.

See "Deprecated APIs and Features" on page 1-34 for information on deprecated MBean attributes and operations.

The applications in the applications directory in development mode on the Administration Server are now staged. In 6.x, they were not. For more information see *Application Staging* in the section called *Two-PhaseDeployment* in *Developing WebLogic Server Applications*.

EJB 2.0

The EJB 2.0 specification has changed substantially between WebLogic Server 6.0 and WebLogic Server 7.0, and somewhat between WebLogic Server 6.1 and WebLogic Server 7.0.

Some of the prominent changes are listed here. To see a complete listing of the specification changes from WebLogic Server 6.0 to WebLogic Server 7.0, you can view and download the *EJB 2.0 final specification* at http://java.sun.com/products/ejb/2.0.html.

For more information about the changes between WebLogic Server 6.0 and WebLogic Server 6.1, see *EJB Enhancements in WebLogic Server* in *Introducing WebLogic Server Enterprise JavaBeans* in the WebLogic Server 6.1 documentation. EJB 1.1 beans that worked in WebLogic Server 6.x should work just as well in WebLogic Server 7.0 with no alteration.

You may have to make the following changes to EJB 2.0 beans:

- If your deployment descriptor contains a 6.0 element that has a different name in 7.0, you have to manually change the name in your deployment descriptor. The following are some examples of element names that you may need to change in 7.0:
  - In 7.0, the name of the element that is used to identify a particular EJB that participates in a relationship is `relationship-role-source`. In 6.0, the element name was `role-source`.
  - In 7.0, the name of the element that specifies whether the destination is a queue or a topic is `destination-type`. In 6.0, the element name was `jms-destination-type`.
  - In 7.0, the name of the element that specifies whether the destination is a queue or a topic is `run-as`. In 6.0, the element name was `run-as-specified-identity`.
- In 7.0, `EJB-QL` queries require a `SELECT` clause.
- All EJB 2.0 CMP beans must have an `abstract-schema-name` element specified in their `ejb-jar.xml` in WebLogic Server 7.0.

Other major changes that resulted from the EJB 2.0 specification changes are as follows:

- There were no local interfaces in 6.0, but they exist in 6.1 and 7.0.
- Remote relationships do not exist in 6.1 and 7.0.
- The new implementation of read-only beans in WebLogic 7.0 does not require `NOT_SUPPORTED` for a transaction attribute. It also doesn't do exclusive locking, but gives each transaction its own instance of the read-only bean to use.
- In pre-2.0 final versions of our EJB implementation you could refer to a CMP or CMR field in a `EJB QL` query without using a qualified path. However, as of the final specification, all `EJB QL` queries must have a qualified path.
- The new implementation of read-only beans in WebLogic 7.0 does not require `NOT_SUPPORTED` for a transaction attribute. The new implementation also does not do exclusive locking, but gives each transaction its own instance of the read-only bean to use.

If you have trouble with a servlet within the scope of application deployment see "Deployment" on page 1-24.

weblogic.management.configuration.EJBComponentMBean Changes

Beginning in Weblogic Server 6.1 and continuing in WebLogic Server 7.0, the interface `weblogic.management.configuration.EJBComponentMBean` changed so that it extends both `ComponentMBean` and `EJBContainerMBean`. Any methods that implement `EJBComponentMBean` (for example, `getVerboseEJBDeploymentEnabled`) must be changed to support `EJBContainerMBean` when you port from WebLogic Server 6.0 to 7.0.

max-beans-in-cache Parameter

In WebLogic Server 7.0 the `max-beans-in-cache` parameter controls the maximum number of beans in the cache for Database concurrency. In earlier WebLogic Server versions, `max-beans-in-cache` was ignored; the size of the cache was unlimited. You may need to increase the size of this parameter.

Fully Qualified Path Expressions

In an EJB QL Query on WebLogic Server 7.0, all path-expressions must be fully qualified. This is a change from WebLogic Server 6.x. If you see an ejbc error while running ejbc on WebLogic Server 7.0 using a 6.x EJB, you need to correct either your `ejb-ql` elements in your `ejb-jar.xml` file or your `weblogic-ql` elements in your `weblogic-cmp-jar.xml` file. For example:

- WebLogic Server 6.x would allow the following query to compile:

  ```
  SELECT address FROM CustomerBean AS c WHERE zip = ?1
  ```

- For WebLogic Server 7.0 to allow the same query to be compiled, the address and zip fields must be qualified:

  ```
  SELECT c.address FROM CustomerBean AS c WHERE c.zip = ?1
  ```

jCOM

For information about upgrading from WebLogic jCOM 6.1 to WebLogic jCOM 7.0 see *Upgrading Considerations* in *Programming WebLogic jCOM*.

JDBC

The minimum capacity increment for a JDBC connection pool has changed from 0, in WebLogic Server 6.1, to 1, in version 7.0. See JDBCConnectionPool in the WebLogic Server Configuration Reference.

The default value for `PreparedStatementCacheSize` and `XAPreparedStatementCacheSize` in the JDBC Connection Pool MBean has increased from 0 in WebLogic Server 6.1 to 5 in WebLogic Server 7.0SP2 and later releases. For more information about the Prepared Statement Cache and its limitations, see “Increasing Performance with the Prepared Statement Cache” in the *Administration Guide*.

JMS

WebLogic Server 7.0 supports the *JavaSoft JMS specification version 1.0.2*.

All WebLogic JMS 6.x applications are supported in WebLogic JMS 7.0. However, if you want your applications to take advantage of the new highly available JMS features, you will need to configure your existing physical destinations (queues and topics) to be part of a single distributed destination set. For more information on using JMS distributed destinations, see *Using Distributed Destinations* in *Programming WebLogic JMS*.

For more information on porting your WebLogic JMS applications, see *Porting WebLogic JMS Applications* in *Programming WebLogic JMS*.

JMX

All public WebLogic Server 6.x MBeans and attributes are supported in WebLogic Server 7.0. However, if you are employing internal MBeans or attributes, you may encounter porting issues.

See "Deprecated APIs and Features" on page 1-34 for information on deprecated MBean attributes and operations.

Jolt Java Client

Jolt users will need to upgrade to Jolt Java Client 8.0.1 to enable BEA Tuxedo services for the Web using BEA WebLogic Server 7.0. The Jolt Java Client 8.0.1 is available from the *BEA Product Download Center* at http://commerce.bea.com/downloads/products.jsp. Follow the link to BEA WebLogic Server 7.0 and select the Jolt Java Client 8.0.1 from the Modules for WebLogic Server list.

JSP

Due to a change in the JSP specification, null request attributes now return the string “null” instead of an empty string. WebLogic Server versions since 6.1 contain a new flag in `weblogic.xml` called `printNulls` which is true by default, meaning that "null" will be the default. Setting this to false ensures that expressions with "null" results are printed as an empty string, not the string “null.”

An example of configuring the `printNulls` element in `weblogic.xml` :

```
<weblogic-web-app>
   <jsp-param>
     <param-name>printNulls</param-name>
     <param-value>false</param-value>
   </jsp-param>
</weblogic-web-app>
```

Load Order for Startup Classes

The behavior of `LoadBeforeAppDeployments` in `StartupClassMbean` has changed between versions 6.x and 7.0 Service Pack 2.

`LoadBeforeAppDeployments` still exists in version 7.0 Service Pack 2, but its behavior has changed. In 6.x, setting `LoadBeforeAppDeployments` to true caused startup classes to be invoked after the datasources were created and before the applications were activated. In 7.0 Service Pack 2, it determines whether a startup class is loaded and run before the server activates JMS and JDBC services or deploys applications and EJBs.

Managed Servers

A Managed Server running WebLogic Server 6.x cannot obtain its configuration and boot using an Administration Server running WebLogic Server 7.0. Make sure that you do not copy the `running-managed-servers.xml` file from your WebLogic Server 6.x installation directory to your WebLogic Server 7.0 installation directory.

MBean API Change

Previous versions of this document and various other sample documents erroneously described using `weblogic.management.Admin.getInstance().getAdminMBeanHome()` as a way to look up the `MBeanHome` interface on the Administration Server.

However, the `weblogic.management.Admin` class is not public. Instead of using this non-public class, use JNDI to retrieve `MBeanHome`. See *Determining the Active Domain and Servers* in *Programming WebLogic Server JMX Services*.

Security

Guest and <Anonymous> Users

In WebLogic Server 6.x, any unauthenticated user (anonymous user) was identified as a user called `guest`. WebLogic Server allowed the `guest` user access to WebLogic resources. However, this functionality presented a potential security risk so the functionality was modified.

In this version of WebLogic Server, the `guest` user is no longer supplied by default. WebLogic Server now distinguishes between the `guest` user and an anonymous user, by assigning an anonymous user the name `<anonymous>`.

If you want to use the `guest` user as you did in WebLogic Server 6.x, do one of the following:

- Use Compatibility security. (For more information, see "Using Compatibility Security" in *Managing WebLogic Security*.)
- Define the `guest` user as a user in the WebLogic Authentication provider. (The WebLogic Authentication provider is already configured in the default security realm.) You do this by setting the following argument when starting a WebLogic Server instance:

  ```
  -Dweblogic.security.anonymousUserName=guest
  ```

Caution: This argument was added to assist existing WebLogic Server customers to upgrade their security functionality. You should take great caution when using the `guest` user in a production environment. For more information about upgrading, see "Upgrading Security" in the *Upgrade Guide for BEA WebLogic Server 7.0*.

Servlets

Update your web.xml file so that it uses the following new classes:

`weblogic.servlet.proxy.HttpClusterServlet`

instead of

`weblogic.servlet.internal.HttpClusterServlet`

and

`weblogic.servlet.proxy.HttpProxyServlet`

instead of

`weblogic.t3.srvr.HttpProxyServlet`

If you have trouble with a servlet within the scope of application deployment see "Deployment" on page 1-24.

Thread Pool Size

In WebLogic Server 6.0, the number of worker threads was specified via the `ThreadPoolSize` parameter on the server MBean. Starting in WebLogic Server 6.1, the number of worker threads is defined via an `ExecuteQueue` on the Server MBean.

WebLogic Server 7.0 provides a porting path for this parameter, so that if it is specified in the `config.xml` file, or if it is passed to the client or server on the command line (`-Dweblogic.ThreadPoolSize=<xx>`), WebLogic Server ports your `ThreadPoolSize` to the `ThreadCount` setting automatically.

Setting the number of worker threads:

In WebLogic Server 6.x:

```
<Server
Name="myserver"
ThreadPoolSize="23"
```

```
...
/Server>
```

Starting in WebLogic Server 7.0:

```
<Server
Name="myserver"
... >
<ExecuteQueue
Name="default"
ThreadCount="23" />
/Server>
```

To change the thread count value via the Console:

1. In the console, select Servers > myServer > Monitoring > .
2. Click on Monitor all Active Queues.
3. Click on "default" queue (a list of threads and what they are doing appears).
4. Click on Configure Execute Queues (at the top of the page).
5. Click on "default" queue.
6. Enter the number of threads associated with this server.
7. Restart the server to make changes take effect.

Web Applications

The `weblogic.management.runtime.ServletRuntimeMBean.getName()` API (in WebLogic Server 6.0) has changed to `weblogic.management.runtime.ServletRuntimeMBean.getServletName()` in WebLogic Server 6.1 and 7.0. You will have to update your source code and recompile if you are using this interface.

With Java Servlet Specification 2.3, authorization-on-forward is no longer default behavior. To obtain authorization when you forward to a secure resource, add `<check-auth-on-forward>` to the `weblogic.xml` file.

Servlet Request and Response objects have a new API. Some serializable, lightweight implementations of these may no longer compile without implementing the new API. It is strongly recommended that you use the new Servlet 2.3 model and substitute your implementations of Servlet Request and Response objects. If you did this in WebLogic Server 6.0, you were probably relying on the undocumented, internal implementations of these objects. WebLogic Server 7.0 supports Servlet 2.3, so you should be able to take advantage of the new ServletRequest/ResponseWrapper objects.

WebLogic Server Clusters on Solaris

Certain applications (heavy EJB apps) deployed in a WebLogic Server cluster on Solaris will perform better using the client JVM rather than the server JVM. This is especially true under heavy loads.

Web Services

Due to changes in the Web service runtime system and architecture between versions 6.1 and 7.0 of WebLogic Server, you must upgrade Web services created in version 6.1 to run on version 7.0.

The WebLogic Web services client API included in WebLogic Server 6.1 of has been deprecated and you cannot use it to invoke 7.0 Web services. WebLogic Server 7.0 includes a new client API, based on the Java API for XML-based RPC (JAX-RPC).

For detailed information on upgrading a 6.1 WebLogic Web service to 7.0, see *Upgrading 6.1 WebLogic Web Services to 7.0* at http://e-docs.bea.com/wls/docs70/webServices/migrate.html.

For examples of using JAX-RPC to invoke WebLogic Web services, see *Invoking Web Services* at http://e-docs.bea.com/wls/docs70/webServices/client.html.

For general information on the differences between 6.1 and 7.0 Web services, see *Overview of WebLogic Web Services* at http://e-docs.bea.com/wls/docs70/webServices/overview.html.

Writable config.xml File

WebLogic Server 7.0 automatically updates configuration information read from the 6.x `config.xml` file to include WebLogic Server 7.0 information. In order for these changes to be retained between invocations of the server, the `config.xml` file must be writable. To allow the file to be writable, make a backup copy of your `config.xml` file from your 6.x configuration and change the file attributes.

Deprecated APIs and Features

- WebLogic Time Services is deprecated and should be replaced by JMX Timer Service. For documentation of JMX Timer Service, see *Interface TimerMBean* and *Class Timer*.
- WebLogic Workspaces
- Zero Administration Client (ZAC) is deprecated in this release of WebLogic Server 7.0. It is replaced by JavaWebStart.
- WebLogic Enterprise Connectivity (WLEC) is deprecated in this release of WebLogic Server 7.0. Tuxedo CORBA applications using WLEC should migrate to WebLogic Tuxedo Connector. For more information, see *WebLogic Tuxedo Connector*.
- `weblogic.deploy` is deprecated in this release of WebLogic Server 7.0 and has been replaced by `weblogic.Deployer`. For more information, see *Deployment Tools and Procedures* in *WebLogic Server Deployment*.
- This version of WebLogic Server deprecates deploying multiple J2EE modules without an `application.xml` file. If you select a directory for deployment, the directory must contain either one standalone J2EE module (an EJB, Web Application, or Resource Adapter), or multiple modules with an associated `application.xml` file (an Enterprise Application).
- This release provides a standard method for resource adapter deployers to plug in their specified authorization/authentication mechanism through secure password credential storage. This WebLogic Server storage mechanism has replaced the Security Principal Mapping mechanism provided with the `weblogic-ra.xml` deployment descriptor within the resource adapter archive. As a result, the element has been deprecated.
- The Password Converter Tool provided with the previous release of WebLogic Server has been deprecated.
- The `<connection-cleanup-frequency>` and `<connection-duration-time>` elements in the `weblogic-ra.xml` deployment descriptor have been deprecated in favor of a new connection leak detection mechanism.
- WebLogic Server 7.0 deprecates the single-phase deployment model used in WebLogic Server 6.x. All WebLogic 6.x deployment protocol APIs and tools are now deprecated and should not be used for deploying production applications.

Instead, use the new two-phase deployment protocol APIs and tools described in the WebLogic Server 7.0 documentation.

- The WebLogic jDriver for Microsoft SQL Server is deprecated and will be removed from a future release of WebLogic Server. BEA recommends that you use the JDBC driver available from Microsoft to connect to a Microsoft SQL Server database. See "Installing and Using the SQL Server 2000 Driver for JDBC from Microsoft" in Programming WebLogic JDBC at http://e-docs.bea.com/wls/docs70/jdbc/thirdparty.html#sqlserver.
- The WebLogic Keystore provider is deprecated in WebLogic Server 7.0 SP1.
- `weblogic.management.tools.WebAppComponentRefreshTool` and `weblogic.refresh` are both deprecated in this release of WebLogic Server 7.0. They have been replaced by `weblogic.Deployer`.
- WebLogic JDBC t3 Driver. See Deprecation of WebLogic File Services at http://bt04/stage/wls/docs70/file/filesrvc.html#1028649.
- If you did programmatic deployment or used the `weblogic.Admin` command in order to create application and component MBeans, set the attributes on the MBeans, and invoke operations on those MBeans to cause them to get deployed in the system, the following MBean attributes and operations have been deprecated:

  Deprecated `ApplicationMBean` operations:

  `deploy`

  `load`

  `undeploy`

  Deprecated `ApplicationMBean` attributes:

  `LastModified`

  `LoadError`

  `isDeployed`

  Please refer to the WebLogic Server 7.0 *javadocs* for the `ApplicationMBean` interface to see what has replaced these attributes and operations.

Removed APIs and Features

WebLogic Enterprise Connectivity (WLEC) examples have been removed.

CHAPTER

2 Upgrading WebLogic Server 4.5 and 5.1 to Version 7.0

Upgrading WebLogic Server 4.5 and 5.1 to version 7.0 is a multi-step process that involves defining Web applications and converting your `weblogic.properties` file(s) into a new XML file format. There are also several specification changes that affect the upgrade process. This chapter addresses the majority of upgrade issues, but may omit issues that are unique to a specific environment.

The following sections provide procedures and other information you need to upgrade your system from WebLogic Server 4.5 or 5.1 to WebLogic Server 7.0; to port your applications from WebLogic Server 4.5 or 5.1 to WebLogic Server 7.0; and deploy these applications. Instructions apply to upgrades from both WebLogic Server 4.5 and 5.1 to WebLogic Server 7.0.

- "Porting and Converting Enterprise JavaBeans Applications" on page 2-16
- "Creating an Enterprise Application" on page 2-23
- "Understanding J2EE Client Applications" on page 2-24
- "Upgrading JMS" on page 2-25
- "Upgrading Oracle" on page 2-25
- "Additional Porting and Deployment Considerations" on page 2-26

Note: Throughout this document the word, upgrade, is used to refer to upgrading to a later version of WebLogic Server and the word, port, is used to refer to moving your applications from an earlier version of WebLogic Server to a later version.

Upgrading Your WebLogic Server Configuration: Main Steps

Take the following steps to upgrade from WebLogic Server 4.5 or 5.1 to WebLogic Server 7.0:

1. Make a back-up copy of your 4.5 or 5.1 domain before you begin the upgrade procedure. After you start the server using WebLogic Server 7.0 classes, you will be unable to downgrade to a previous version.

2. Install WebLogic Server 7.0. See the *Installation Guide*.

   Prior to WebLogic Server 6.0, a separate download was required if you wanted to get 128-bit encryption instead of 56-bit encryption. WebLogic Server 7.0 has a single download for both 56-bit encryption and 128-bit encryption. For details of how to enable 128-bit encryption see *Enabling 128-Bit Encryption* in the *Installation Guide*.

   Note: Installing the new version in the exact location of the old version is explicitly prohibited by the installer.

3. Upgrade your server license files. For instructions on how to convert your licenses to the new format, see "Upgrading WebLogic Server License Files" on page 2-4.
4. Convert your `weblogic.properties` file. For instructions on how to convert your `weblogic.properties` file, see "Converting the weblogic.properties File to XML Files" on page 2-5 and the *Console Help* documentation.
5. Enter a name for you new Administration Server in the provided window in the Console. This name is used as the Server Name for the Administration Server. If the name is not specified, the name will be `myserver` by default.
6. Enter a directory location for the resulting conversion output configuration. All files and subdirectories created as a result of the conversion of your original domain will be placed in this directory location.
7. Add classes to your Java system `CLASSPATH`. For more information see "Classloading in WebLogic Server 7.0" on page 2-8.
8. Modify your existing startup scripts to work with WebLogic 7.0. See "Modifying Startup Scripts" on page 2-8.
9. Package and port your WebLogic server-side business object implementations (referred to as Web applications beginning with WebLogic Server 6.0) to run on WebLogic 7.0. See "Converting and Porting Your Existing Applications into Web Applications" on page 2-9.
10. If you need to port EJBs, see "Porting and Converting Enterprise JavaBeans Applications" on page 2-16.
11. Upgrade JMS. Many new configuration attributes have been added to JMS since WebLogic Server 4.5. For more information, see "Upgrading JMS" on page 2-25.

To upgrade a cluster of servers, follow the above steps for each server and then follow the steps outlined in *Setting up WebLogic Clusters* in *Using WebLogic Clusters*. All servers in the cluster should be running 7.0 since there is no interoperability between 4.5 and 7.0 or between 5.1 and 7.0.

To upgrade from WebLogic Server 7.0 to WebLogic Server 7.0.0.1, see *Updating WebLogic Server 7.0 GA (7.0.0.0) to 7.0.0.1* in the *Installation Guide*.

Note: The directory structure in WebLogic Server 7.0 is different from that of 4.5 and 5.1. For complete information on the updated directory structure see *Understanding the WebLogic Server Directory Structure* in *Performing Post-Installation Tasks*.

Upgrading WebLogic Server License Files

The Java format license file (`WebLogicLicense.class`) and the XML-format license file (`WebLogicLicense.XML`) are no longer supported. These files were used with earlier releases of WebLogic Server and must be converted to a new format. The new license file is called `license.bea`.

Converting a WebLogicLicense.class License

If a `WebLogicLicense.class` license file is used in your existing WebLogic Server installation, perform the following tasks before you install WebLogic Server 7.x:

1. Convert the `WebLogicLicense.class` license file to a `WebLogicLicense.XML` file using the `licenseConverter` utility at `http://www.weblogic.com/docs51/techstart/utils.html#licenseConverter`.
2. Convert the `WebLogicLicense.XML` file as described in Converting a WebLogicLicense.XML License.

Converting a WebLogicLicense.XML License

To convert a `WebLogicLicense.XML` file to a `license.bea` file (compatible with WebLogic Server 6.x and 7.x), complete the following steps. Be sure the `WebLogicLicense.XML` license file is available on the machine on which you perform this procedure.

1. Log in to the BEA Customer Support Web site at `http://websupport.beasys.com/custsupp`.

2. Click the link to update a WebLogic Server license. You may need to scroll down to see the link.
3. Browse and select the pathname for the directory containing the license file to be converted, or enter the pathname in the box provided. Then click Submit License.
4. You will receive the converted `license_wls`*`xx`*`.bea` file through e-mail. To update the `license.bea` file on your system, see Updating Your license.bea File in the *Installation Guide* at `http://e-docs.bea.com/wls/docs70/install/instlic.html#update_license`.

Converting the weblogic.properties File to XML Files

Note: Throughout this document, the WebLogic Server 7.0 directory domain that you create is referred to as `domain`.

Prior to WebLogic Server 6.0, WebLogic Server releases used a `weblogic.properties` file to configure applications. In WebLogic Server 7.0, configuration of applications is handled through XML descriptor files and the Administration Console. Converting a `weblogic.properties` file to the `config.xml` file creates a new domain for your applications and adds XML files that define how your applications are set up. BEA Systems recommends that you change the name of the default domain name that is created during the conversion so that the name is relevant to your work.

The `config.xml` file is an XML document that describes the configuration of an entire Weblogic Server domain. The `config.xml` file consists of a series of XML elements. The `domain` element is the top-level element, and all elements in the domain are children of the domain element. The `domain` element includes child elements, such as the `server`, `cluster`, and `application` elements. These child elements may have children themselves. Each element has one or more configurable attributes.

The `weblogic.xml` file contains WebLogic-specific attributes for a Web application. You define the following attributes in this file: HTTP session parameters, HTTP cookie parameters, JSP parameters, resource references, security role assignments, character set mappings, and container attributes.

The deployment descriptor `web.xml` file is defined by the servlet 2.3 specification from Sun Microsystems. The `web.xml` file defines each servlet and JSP page and enumerates enterprise beans referenced in the Web application. This deployment descriptor can be used to deploy a Web Application on any J2EE-compliant application server.

Convert your existing `weblogic.properties` file to the appropriate XML file by following these steps:

1. Start the WebLogic Server 7.0 examples server. For information on starting the WebLogic Server 7.0 examples server, see *Starting the Examples, Pet Store, and Workshop Examples Servers* in *Performing Post-Installation Tasks*.

   You will be prompted for a user name and password.

2. At the home page for the WebLogic Administration Console (for example: `http://localhost:7001/console/index.jsp`) click on the "Convert `weblogic.properties`" link under the heading Helpful Tools.

3. Use the Console's links to navigate the server's file system and find the root directory of your previous version of WebLogic Server (for example: `C:\weblogic`). When you have found the correct directory, click on the icon next to it to select it.

4. If you have additional per server `weblogic.properties` files or clustering `weblogic.properties` files in other directories, select them using the provided windows. If you have chosen the correct root directory of your previous version of WebLogic Server, your global `weblogic.properties` file will be converted regardless of any additional properties files that you select.

5. Enter a name for your new domain in the provided window in the Console. Click Convert.

When you convert your properties file, the `web.xml and weblogic.xml` files for the default web application are created for you and placed inside a `domain\applications\DefaultWebApp_myserver\WEB-INF` directory. The process of converting your `weblogic.properties` file also creates the `config.xml` file located in `domain`. This file contains configuration information specific to your domain.

Note: The conversion utility described above specifies the Java home location in the `weblogic.xml` file. It reads this location using the `System.getProperty(java.home)`, which means that it will specify the Java home location on which WebLogic Server was started for the conversion.

- It is strongly recommended that you not edit the `config.xml` file directly. Access the configuration through the Administration Console, a command line utility, or programmatically through the configuration API. For details on configuring WebLogic Server Web Components, see *Configuring WebLogic Server Web Components* in the *Administration Guide*.
- Security properties are stored in the `fileRealm.properties` file located in *`domain`*. For information about using a security realms in WebLogic Server 7.0, see Customizing the Default Security Configuration.
- The `weblogic.common.ConfigServicesDef` API, which provided methods to get properties out of the `weblogic.properties` file, has been removed from this version.

For more procedures for converting your `weblogic.properties` file, see the *Console Help* documentation.

For a list of which `config.xml, web.xml, or weblogic.xml` attribute handles the function formerly performed by `weblogic.properties` properties, see "The weblogic.properties Mapping Table".

The startup scripts, which are generated when a `weblogic.properties` file is converted, are named:

- `startdomainName.cmd` (for Windows users)
- `startdomainName.sh` (for UNIX users)

where *`domainName`* is the name of your *`domain`* directory.

These scripts exist under the *`domain`* directory in your WebLogic Server 7.0 distribution and start the administration server in the new domain.

See *Starting and Stopping WebLogic Servers* in the *Administration Guide* for more information on scripts and starting servers.

Classloading in WebLogic Server 7.0

Earlier versions of WebLogic Server used the WebLogic classpath property (`weblogic.class.path`) to facilitate dynamic classloading. In WebLogic 6.0 and later, the `weblogic.class.path` is no longer needed. You can now load classes from the Java system classpath.

To include the classes that were formerly specified in `weblogic.class.path` in the standard Java system classpath, set the `CLASSPATH` environment variable, or use the `-classpath` option on the command line as in the following example:

```
java -classpath %CLASSPATH%;%MyOldClassspath% weblogic.Server
```

where `%MyOldClasspath%` contains only the directories that point to your old applications.

Modifying Startup Scripts

If you used WebLogic Server startup scripts with a previous version of the product, modify them to work with 7.0.

- Modify the startup scripts as described in *Setting the Classpath* in the *Administration Guide*. The WebLogic classpath is no longer used; use the Java system classpath as described in the preceding section, "Classloading in WebLogic Server 7.0" on page 2-8.
- WebLogic Server 7.0 is started from the domain directory. Make sure that your startup script starts the server from the domain directory.
- It is no longer necessary to include the license file in the classpath.
- With the new management system, there is a distinction between an Administration Server and Managed Servers. Consequently, scripts that start servers must be rewritten according to how you plan to administer your servers. For the new commands and their required arguments, see *Starting and Stopping WebLogic Servers* in the *Administration Guide*.

WebLogic Server 7.0 J2EE Application Types

Applications on J2EE-compliant servers such as WebLogic Server 7.0 are created and deployed as one of the following four types: Web Applications, Enterprise JavaBeans, Enterprise Archives, and client applications. To port your existing components to WebLogic Server 7.0, create the appropriate J2EE deployment units. For more information on J2EE deployment units, see *Deploying Web Applications as Part of an Enterprise Application* in *Assembling and Configuring Web Applications*. Web Applications are usually a collection of servlets, JSPs, and HTML files, packaged as WAR files. Enterprise JavaBeans (packaged as JAR files) are server-side Java components written according to the EJB specification. Enterprise Archives (EAR files) contain all of the JAR and WAR component archive files for an application and an XML descriptor that describes the bundled components. Client Applications are Java classes that connect to WebLogic Server through Remote Method Invocation (RMI). Later sections discuss the aforementioned J2EE deployment units in greater detail.

Converting and Porting Your Existing Applications into Web Applications

In order to convert an application to a Web Application and then port it into a Web Application deployed on WebLogic Server 7.0, the application's files must be placed within a directory structure that follows a specific pattern. For development, these files can be left in an exploded directory format. However, for production situations, it is highly recommended that you bundle your applications into a WAR file as a single Web Application. For more information on Web Applications see *Understanding WebLogic Server J2EE Applications* and *Assembling and Configuring Web Applications*.

The following sections provide information you need to know about porting and deploying Web Applications, including a procedure for porting a simple servlet from WebLogic Server 5.1 to WebLogic Server 7.0:

- "Web Applications Directory Structure" on page 2-10

Web Applications Directory Structure

Web Applications are organized in a specified directory structure so that they can be archived and deployed on WebLogic Server. All servlets, classes, static files, and other resources belonging to a Web Application are organized under a directory hierarchy. The root of this hierarchy defines the document root of your Web Application. All files under this root directory can be served to the client, except for files under the special directories `WEB-INF` and `META-INF` located in the root directory. The root directory should be named with the name of your Web Application.

The following diagram illustrates the directory structure of any Web Application.

```
WebApplicationRoot\(Publically available files such as
                   |  .jsp, .html, .jpg, .gif)
                   |
                   +WEB-INF\-+
                             |
                             + classes\(directory containing
                             |         Java classes including
                             |         servlets used by the
                             |         Web Application)
                             |
                             + lib\(directory containing
                             |       JAR files used by the
                             |       Web Application)
                             |
                             + web.xml
                             |
                             + weblogic.xml
```

When you convert your `weblogic.properties` file, the appropriate `web.xml` and `weblogic.xml` files are created for you under the directory `domain\applications\DefaultWebApp_myserver\WEB-INF`. Follow the preceding directory structure and place the XML files in the `domain\applications\webAppName\WEB-INF` directory that you create. For information on deploying web applications, see *Developing WebLogic Server Applications*.

XML Deployment Descriptors

The Web Application Deployment Descriptor (`web.xml`) file is a standard J2EE descriptor used to register your servlets, define servlet initialization parameters, register JSP tag libraries, define security constraints, and define other Web Application parameters. For detailed instructions on creating the deployment descriptor, see *Writing the web.xml Deployment Descriptor* in *Assembling and Configuring Web Applications*.

There is also a WebLogic-specific Deployment Descriptor (`weblogic.xml`). In this file you define JSP properties, JNDI mappings, security role mappings, and HTTP session parameters. The WebLogic-specific deployment descriptor also defines how named resources in the `web.xml` file are mapped to resources residing elsewhere in WebLogic Server. For detailed instructions on creating the WebLogic-specific deployment descriptor, see *Writing the WebLogic-Specific Deployment Descriptor*. This file may not be required if you do not need the preceding properties, mappings, or parameters.

Use the `web.xml` and `weblogic.xml` files, in conjunction with the Administration console, to configure your applications. The XML files can be viewed through any text editor. To edit them, simply make your changes and save the file as `web.xml` or `weblogic.xml` with the appropriate path as specified by the prescribed directory structure under "Web Applications Directory Structure" on page 2-10. See *Assembling and Configuring Web Applications* for more information. If you do not want to deploy your applications together as a single Web Application, you need to split up the XML files that have been created for you, creating the appropriate XML files specific to each Web Application. Each Web Application needs a `weblogic.xml` file and a `web.xml` file as well as whichever files you choose to put in it.

WAR Files

A WAR file is a Web Application archive. If you have correctly followed the prescribed directory structure of a Web Application and created the appropriate `web.xml` and `weblogic.xml` files, it is strongly recommended that in production environments your applications be bundled together in a Web Application deployed as a WAR file. Once you have bundled your applications into a WAR file, it is important to remove the previously existing directory structure so that WebLogic Server only has one instance of each application.

Use the following command line from the root directory containing your Web Application to create a WAR file, replacing '*webAppName*' with the specific name you have chosen for your Web Application:

```
jar cvf webAppName.war *
```

You now have created a WAR file that contains all the files and configuration information for your Web Application.

Deploying Web Applications

To configure and deploy a web application using the WebLogic Server Administration Console:

1. Start the WebLogic Server Administration Console.
2. Select the Domain in which you will be working.
3. In the left pane of the Console, click Deployments.
4. In the left pane of the Console, click the Applications. A table is displayed in the right pane of the Console showing all the deployed Applications.
5. Select the Configure a new Application option.
6. Locate the application archive, or the directory containing the exploded application. Note that WebLogic Server will deploy all components it finds in and below the specified directory.
7. Click the icon to the left of a directory or file to choose it and proceed to the next step.

8. Enter a name for the application or component in the provided field and click Create.

9. Enter the following information:

   Staging Mode—specify the staging mode. The options include server, nostage, and stage.

   Deployed—using the provided checkbox, indicate whether the .ear, .war, .jar, or .rar file should be deployed upon creation.

10. To configure components for the application, click the Configure Components in this Application.

11. The Components table is displayed. Click a component to configure.

12. Using the available tabs, enter the following information:

    Configuration—Edit the staging mode and enter the deployment order.

    Targets—Indicate the Targets-Server for this configured application by moving the application from the Available list to the Chosen list.

    Deploy—Deploy the application to all of the selected targets or undeploy it from all targets.

    Monitoring—View monitoring information related to the application.

    Notes—Enter notes related to the application.

13. Click Apply.

For more information on setting attributes in the console, see the Web Applications section of the *Console Help*.

Session Porting

WebLogic Server 7.0 does not recognize cookies from previous versions because cookie format changed with WebLogic Server 6.0. WebLogic Server will ignore cookies with the old format and create new sessions. Be aware that new sessions are created automatically.

The default name for cookies has changed from 5.1, when it was `WebLogicSession`. In WebLogic Server 7.0, cookies are named `JSESSIONID` by default.

See *weblogic.xml Deployment Descriptor Elements* in *Assembling and Configuring Web Applications*.

JavaServer Pages (JSPs) and Servlets

This section contains information specific to JSPs and servlets that may be pertinent to your applications.

- Some changes will be necessary in code (both Java and HTML) where the code refers to URLs that may be different when servlets and JSPs are deployed in a Web Application other than the default Web Application. See *Administration and Configuration* in *Programming WebLogic Server HTTP Servlets* for more information. If relative URLs are used and all components are contained in the same Web Application, these changes are not necessary.
- Only serializable objects may be stored in a session if your application is intended to be distributable.
- You must convert your `weblogic.properties` file to `XML` attributes in `web.xml` and\or `weblogic.xml`. For additional information on this process, see the conversion section of the *Console Help*.
- Access control by an ACL has been replaced with security-constraint based access control in the web application deployment descriptor.
- Server-side-includes are not supported. You must use JSP to achieve this functionality.
- WebLogic Server 7.0 is fully compliant with the Servlet 2.3 specification.
- Update your web.xml file so that it uses the following new classes:

  `weblogic.servlet.proxy.HttpClusterServlet`

  instead of

  `weblogic.servlet.internal.HttpClusterServlet`

  and

  `weblogic.servlet.proxy.HttpProxyServlet`

  instead of

  `weblogic.t3.srvr.HttpProxyServlet`

Porting a Simple Servlet from WebLogic Server 5.1 to WebLogic Server 7.0

The following procedure ports the simple Hello World Servlet that was provided with WebLogic 5.1 Server to WebLogic Server 7.0.

1. In WebLogic Server 7.0, create the correct directory structure, as described in *Administration and Configuration* in *Programming WebLogic Server HTTP Servlets*. This involves creating a root application directory, such as `C:\hello`, as well as a `C:\hello\WEB-INF` directory and a `C:\hello\WEB-INF\classes` directory. Place the `HelloWorld.Servlet.java` file inside the `C:\hello\WEB-INF\classes` directory.

2. Create a `web.xml` file for this servlet. If you have converted your `weblogic.properties` file, a `web.xml` file has already been created for you. If you registered HelloWorldServlet in your `weblogic.properties` file before you converted it, the servlet will be properly configured in your new `web.xml` file. An XML file can be created with any text editor. The following is an example of a basic `web.xml` file that could be used with the HelloWorldServlet.

   ```
   <!DOCTYPE web-app (View Source for full doctype...)>
   - <web-app>
   - <servlet>
   <servlet-name>HelloWorldServlet</servlet-name>
   <servlet-class>examples.servlets.HelloWorldServlet</servlet-cla
   ss>
   </servlet>
   - <servlet-mapping>
   <servlet-name>HelloWorldServlet</servlet-name>
   <url-pattern>/hello/*</url-pattern>
   </servlet-mapping>
   </web-app>
   ```

   For more information on `web.xml` files, see *Writing Web Application Deployment Descriptors* in *Assembling and Configuring Web Applications*. A `weblogic.xml` file is not necessary with such a simple, stand-alone servlet as HelloWorld.

   For more information on `weblogic.xml` files, see *Writing the WebLogic-Specific Deployment Descriptor* in *Assembling and Configuring Web Applications*.

3. Move the `web.xml` file from `domain\applications\DefaultWebApp_myserver\WEB-INF` to `C:\hello\WEB-INF\`.

4. Set up your development environment (see *Establishing a Development Environment* in *Developing WebLogic Server Applications* for more information) and compile the HelloWorldServlet with a command like the following:

   ```
   C:\hello\WEB-INF\classes>javac -d  . HelloWorldServlet.java
   ```

   This should compile the file and create the correct package structure.

5. The servlet can now be bundled into an archive WAR file with the following command:

   ```
   jar cvf hello.war *
   ```

   This command will create a `hello.war` file and place it inside the `C:\hello` directory.

6. To install this Web Application, start your server and open the Administration Console. Under the Getting Started menu, choose Install Applications. Browse to the newly created WAR file and click Upload.

   The servlet should now be deployed and appear under the Web Applications node under Deployments, in the left-hand pane of the console.

7. To call the servlet, type the following in your browser URL window: `http://localhost:7001/hello/hello`.

In this case `/hello/` is the context path of the servlet. This is determined by the naming of the WAR file, in this case `hello.war`. The second `/hello` was mapped in the servlet mapping tags inside the `web.xml` file.

Porting and Converting Enterprise JavaBeans Applications

The following sections describe Enterprise Java Beans porting and conversion procedures.

EJB Porting Considerations

Consider the following when porting Enterprise JavaBeans to WebLogic Server 7.0.

- WebLogic Server Version 7.0 supports the Enterprise JavaBeans 1.1 and 2.0 specifications.
- The XML parser is stricter with XML deployment descriptors in WebLogic 7.0 than it was in WebLogic 5.1. Some errors allowed in earlier versions are no longer permitted. This is described in *Introducing WebLogic Server Enterprise Java Beans*.
- EJB 1.1 beans are deployable in WebLogic Server 7.0. However, if you are developing new beans, it is recommended that you use EJB 2.0. EJB 1.1 beans can be converted to 2.0 using the DDConverter utility. For more information, see the *DDConverter documentation* in *Programming WebLogic Enterprise JavaBeans*.
- You can upgrade EJB 1.0 deployment descriptors to EJB 2.0 using the DDConverter utility, but first those descriptors must be upgraded to 1.1. WebLogic Server 5.1 deployment descriptors can be upgraded to 7.0 to take advantage of new features in WebLogic Server 7.0. Details on the DDConverter utility are provided in the *WebLogic Server EJB Utilities* section of *Programming WebLogic Enterprise JavaBeans*.
- The finder expressions feature of EJB 1.1 is no longer supported. This is the only non-supported feature of EJB 1.1.
- Deploying beans is described in the *Packaging EJBs for the WebLogic Server Container* section of *Programming WebLogic Enterprise JavaBeans*.
- If `ejbc` has not been run on an EJB, WebLogic Server 7.0 will run `ejbc` automatically when the bean is deployed. You do not need to compile beans with `ejbc` before deploying. If you wish to run `ejbc` during startup, you may do so. See details in *Packaging EJBs for the WebLogic Server Container* in *Programming WebLogic Enterprise JavaBeans*.
- An EJB deployment includes a standard deployment descriptor in the `ejb-jar.xml` file. The `ejb-jar.xml` must conform to either the EJB 1.1 DTD (document type definition) or the EJB 2.0 DTD.
- An EJB deployment needs the `weblogic-ejb-jar.xml` file, a WebLogic Server-specific deployment descriptor that includes configuration information

for the WebLogic Server EJB container. This file must conform to the WebLogic Server 5.1 DTD or the WebLogic Server 7.0 DTD.

- In order to specify the mappings to the database, container-managed persistence entity beans require a CMP deployment descriptor that conforms to either the WebLogic Server 5.1 CMP DTD, the WebLogic Server 7.0 EJB 1.1 DTD, or the WebLogic Server 7.0 EJB 2.0 DTD.
- In WebLogic Server 7.0 the `max-beans-in-cache` parameter controls the maximum number of beans in the cache for Database concurrency. In earlier WebLogic Server versions, `max-beans-in-cache` was ignored; the size of the cache was unlimited. You may need to increase the size of this parameter.

EJB Porting Recommendations

- Use `TxDataSource`.

  EJBs should always get their database connections from a `TxDataSource`. This allows the EJB container's transaction management to interface with the JDBC connection, and it also supports XA transactions.

  The WebLogic Server 7.0 CMP Deployment Descriptor supports `TxDataSource`s and should be used instead of the WebLogic Server 5.1 CMP Deployment Descriptor which only specifies a connection pool.

- Use a fast compiler: `ejbc`.

  The WebLogic Server EJB compiler (`weblogic.ejbc`) generates Java code that is then compiled by the Java compiler. By default, WebLogic Server uses the `javac` compiler included with the bundled JDK. The EJB compiler runs much faster when a faster Java compiler is used. Use the `-compiler` option to specify an alternate compiler as in the following example:

  ```
  java weblogic.ejbc -compiler sj pre_AccountEJB.jar
  AccountEJB.jar
  ```

- Correct errors before deploying the EJB on WebLogic Server 7.0.

  The WebLogic Server 7.0 EJB compiler (`ejbc`) includes additional verification that was missing from earlier WebLogic Server releases. It is possible that an EJB deployed in a previous WebLogic Server version without error, but WebLogic Server 7.0 finds and complains about the error. These errors must be corrected before the EJB is deployed in WebLogic Server 7.0.

For instance, WebLogic Server 7.0 ensures that a method exists if a transaction attribute is set for that method name. This helps identify a common set of errors where transaction attributes were mistakenly set on non-existent methods.

- Look at the WebLogic Server 7.0 examples

  The WebLogic Server distribution contains several EJB examples including their deployment descriptors. The EJB examples can be found in the `samples\server\src\examples\ejb11` and `samples\server\src\examples\ejb20` directories of the WebLogic Server 7.0 distribution.

  The following table shows the descriptor combinations supported by WebLogic Server 7.0.

Table 2-1

EJB Version	**WebLogic Server Version**	**The CMP Version**
Any existing WebLogic Server 5.1 deployment uses the following combination and can be deployed without changing descriptors or code in WebLogic Server 7.0.		
1.1	5.1	5.1
The below combinations include a WebLogic Server 7.0 CMP deployment descriptor. The WebLogic Server 7.0 EJB 1.1 CMP deployment descriptor allows multiple EJBs to be specified within a single EJB JAR file, and it supports using a `TxDataSource` which is required when an EJB is enlisted in a two-phase /XA transaction.		
1.1	5.1	7.0
1.1	7.0	7.0
EJB 2.0 beans always use the WebLogic Server 6.x or 7.0 deployment descriptors.		
2.0	6.x	7.0
2.0	7.0	7.0

For more information on Enterprise JavaBeans, see *Enterprise JavaBean Components* and *Programming WebLogic Enterprise Java Beans*.

Steps for Porting a 1.0 EJB from WebLogic Server 4.5.x to WebLogic Server 7.0

WebLogic Server 3.1.x, 4.0.x, and 4.5.x supported the EJB 1.0 specification. To port a 1.0 EJB from WebLogic Server 4.5 to WebLogic Server 7.0:

1. Convert the EJB 1.0 deployment descriptor to either the EJB 1.1 or the EJB 2.0 XML deployment descriptor. You can do this automatically using the *DDCreator* tool.
2. Package the deployment descriptor in a JAR file which includes the deployment descriptor's output from step one above and the bean classes.
3. Run the WebLogic Server EJB compiler (ejbc) to compile the JAR file. The `ejbc` tool ensures that when the EJB compiles, it conforms to either the EJB 1.1 or EJB 2.0 specifications.
4. Correct any compliance errors before deploying the EJB in the EJB container.

To ensure EJB 1.1 or 2.0 compliance, make the following changes to the EJB 1.0 beans:

- EJB 1.0 beans referred to the `SessionContext` or `EntityContext` as transient. When EJB 1.1 or 2.0 beans are deployed, the reference cannot be transient. For example:

  ```
  private transient SessionContext ctx;
  ```

  should be:

  ```
  private SessionContext ctx;
  ```

- The `ejbCreate` method for EJB 1.0 CMP entity beans had a void return type. When EJB 1.1 or 2.0 beans are deployed, the return type must be the primary key class which allows you to write a bean-managed persistent entity bean and then sub-class it with a CMP implementation. For example:

  ```
  public void ejbCreate (String name) {
  firstName = name;
  }
  ```

  should be:

  ```
  public AccountPK ejbCreate (String name) {
  firstName = name;
  ```

```
return null; // required by the EJB specification
}
```

- In EJB 1.1 or 2.0, entity beans cannot use bean-managed transactions. Instead, they must run with a container-managed transaction attribute (for example, Required, Mandatory, etc.).

Steps for Porting a 1.1 EJB from WebLogic Server 5.1 to WebLogic Server 7.0

The WebLogic Server 5.1 deployment descriptor only allows the exclusive or read-only concurrency options. The database concurrency option is available when upgrading to the WebLogic Server 7.0 `weblogic-ejb-jar.xml` file. For more information about this option, see information on database concurrency in *weblogic-ejb-jar.xml Document Type Definitions* in *Programming WebLogic Enterprise JavaBeans*.

The WebLogic Server 7.0 CMP deployment descriptor allows multiple EJBs to be specified and it supports using a `TxDataSource` instead of a connection pool. Using a `TxDataSource` is required when XA is being used with EJB 1.1 CMP.

To port a 1.1 EJB from WebLogic Server 5.1 to WebLogic Server 7.0:

1. Open the Administration Console. From the home page, click on Install Applications under the Getting Started heading.

2. Locate the JAR file you wish to port by clicking the Browse button, then click Open and then Upload. Your bean should now be automatically deployed on WebLogic Server 7.0.

3. Run a `setEnv` script in a client window and set your development environment. (For more information, see *Establishing a Development Environment* in *Developing WebLogic Server Applications*.)

4. Compile all the needed client classes. For example, using the Stateless Session Bean sample that was provided with WebLogic Server 7.0, you would use the following command:

   ```
   javac -d %CLIENTCLASSES% Trader.java TraderHome.java
   TradeResult.java Client.java
   ```

5. To run the client, enter this command:

```
java -classpath %CLIENTCLASSES%;%CLASSPATH%
examples.ejb.basic.statelessSession.Client
```

This command ensures that the EJB interfaces are referenced in your client's classpath.

Steps for Converting an EJB 1.1 to an EJB 2.0

To convert an EJB 1.1 bean to an EJB 2.0 bean, you can use the WebLogic Server DDConverter utility.

BEA Systems recommends that you develop EJB 2.0 beans in conjunction with WebLogic Server 7.0. For 1.1 beans already used in production, it is not necessary to convert them to 2.0 beans. EJB 1.1 beans are deployable with WebLogic Server 7.0. If you do wish to convert 1.1 beans to 2.0 beans, see the *DDConverter* documentation in *Programming WebLogic Enterprise JavaBeans* for information on how to do this conversion.

The basic steps required to convert a simple CMP 1.1 bean to a 2.0 bean are as follows:

1. Make the bean class abstract. EJB 1.1 beans declare CMP fields in the bean. CMP 2.0 beans use abstract `getXXX` and `setXXX` methods for each field. For instance, 1.1 Beans will use `public String name`. 2.0 Beans should use `public abstract String getName()` and `public abstract void setName(String n)`. With this modification, the bean class should now read the container-managed fields with the `getName` method and update them with the `setName` method.

2. Any CMP 1.1 finder that used `java.util.Enumeration` should now use `java.util.Collection`. CMP 2.0 finders cannot return `java.util.Enumeration`. Change your code to reflect this:

   ```
   public Enumeration findAllBeans()
       Throws FinderException, RemoteException;
   ```

   becomes:

   ```
   public Collection findAllBeans()
       Throws FinderException, RemoteException;
   ```

Porting EJBs from Other J2EE Application Servers

Any EJB that complies with the EJB 1.1 or EJB 2.0 specifications may be deployed in the WebLogic Server 7.0 EJB container. Each EJB JAR file requires an `ejb-jar.xml` file, a `weblogic-ejb-jar.xml` deployment descriptor, and a CMP deployment descriptor if CMP entity beans are used. The WebLogic Server EJB examples located in `samples\examples\ejb11` and `samples\examples\ejb20` of the WebLogic Server distribution include sample weblogic deployment descriptors.

Creating an Enterprise Application

An Enterprise Application is a JAR file with an EAR extension. An EAR file contains all of the JAR and WAR component archive files for an application and an XML descriptor that describes the bundled components. The `META-INF\application.xml` deployment descriptor contains an entry for each Web and EJB module, and additional entries to describe security roles and application resources such as databases.

```
EnterpriseApplicationStagingDirectory\
                                     |
                                     +  .jar files
                                     |
                                     +  .war files
                                     |
                                     +META-INF\-+
                                                |
                                                +  application.xml
```

To create an EAR file:

1. Assemble all of the WAR and JAR files for your application.
2. Copy the WAR and EJB JAR files into the staging directory and then create a `META-INF\application.xml` deployment descriptor for the application. Follow the directory structure depicted above.

   The `application.xml` file contains a descriptor for each component in the application, using a DTD supplied by Sun Microsystems. For more information on the application.xml file, see *Application Deployment Descriptor Elements* in

Developing WebLogic Server Applications. Note that if you are using JSPs and want them to compile at run time you must have the home and remote interfaces of the bean included in the classes directory of your WAR file.

3. Create the Enterprise Archive by executing a jar command like the following in the staging directory:

   ```
   jar cvf myApp.ear *
   ```

4. Click on the Install Applications link under the Getting Started heading in the home page of the console and place the EAR file in the `domain\applications` directory. For more information on Enterprise Applications, see *Packaging Enterprise Applications* in *Developing WebLogic Server Applications*.

Understanding J2EE Client Applications

WebLogic Server supports J2EE client applications, packaged in a JAR file with a standard XML deployment descriptor. Client applications in this context are clients that are not Web browsers. They are Java classes that connect to WebLogic Server using Remote Method Invocation (RMI). A Java client can access Enterprise JavaBeans, JDBC connections, messaging, and other services using RMI. Client applications range from simple command line utilities that use standard I/O to highly interactive GUI applications built using the Java Swing/AWT classes.

To execute a WebLogic Server Java client, the client computer needs the `weblogic_sp.jar` file, the `weblogic.jar` file, the remote interfaces for any RMI classes and Enterprise Beans that are on WebLogic Server, as well as the client application classes. To simplify maintenance and deployment, it is a good idea to package a client-side application in a JAR file that can be added to the client's classpath along with the `weblogic.jar` and `weblogic_sp.jar` files. The `weblogic.ClientDeployer` command line utility is executed on the client computer to run a client application packaged to this specification. For more information about J2EE client applications, see *Packaging Client Applications* in *Developing WebLogic Server Applications*.

Upgrading JMS

WebLogic Server 7.0 supports the *JavaSoft JMS specification version 1.0.2*.

- Weblogic Server 4.5.1 — Porting is supported *only* for SP15. Customers running all service packs should contact BEA Support.
- Weblogic Server 5.1 — Customers running SP07 or SP08 should contact BEA Support before porting existing JDBC stores to version 7.0.
  - In order to port object messages, the object classes need to be in the Weblogic Server 7.0 server CLASSPATH.
  - For destinations that are not configured in Weblogic Server 7.0, the ported messages will be dropped and the event will be logged.

For more information on porting your WebLogic JMS applications, see *Porting WebLogic JMS Applications* in *Programming WebLogic JMS*. Note that WebLogic Events are deprecated and are replaced by JMS messages with NO_ACKNOWLEDGE or MULTICAST_NO_ACKNOWLEDGE delivery modes. Each of these delivery modes is described in *WebLogic JMS Fundamentals* in *Programming WebLogic JMS*.

Upgrading Oracle

BEA Systems, mirroring Oracle's support policy, supports the Oracle releases listed in the *Platform Support for WebLogic jDriver JDBC Drivers* on the *WebLogic Server Certifications* page. BEA no longer supports the following Oracle client versions: 7.3.4, 8.0.4, 8.0.5, and 8.1.5.

To use the Oracle Client Version 7.3.4, use the backward compatible oci816_7 shared library. As stated above, BEA no longer supports this configuration.

To upgrade to Oracle Client Version 9i, or read detailed documentation on the WebLogic jDriver and Oracle databases, see *Configuring WebLogic jDriver for Oracle* in *Installing and Using WebLogic jDriver for Oracle*.

For supported platforms, as well as DBMS and client libraries, see the BEA *Certifications Page*. The most current certification information will always be posted on the Certifications page.

Additional Porting and Deployment Considerations

The following sections provide additional information that may be useful when you deploy applications on WebLogic Server 7.0. Deprecated features, upgrades, and the important changes that have been made in WebLogic Server 7.0 are noted.

Note: WebLogic Server 7.0 uses PointBase 4.2 as a sample database and does not bundle the Cloudscape database.

- "Applications and Managed Servers" on page 2-27
- "Deployment" on page 2-27
- "Plug-ins" on page 2-27
- "Internationalization (I18N)" on page 2-28
- "Java Transaction API (JTA)" on page 2-28
- "Java Database Connectivity (JDBC)" on page 2-29
- "JSP" on page 29
- "JVM" on page 2-30
- "RMI" on page 2-30
- "Security" on page 2-31
- "Session Porting" on page 2-33
- "Standalone HTML and JSPs" on page 2-33
- "Web Components" on page 2-34

Applications and Managed Servers

By default, applications are deployed to the Administration Server. However, in most cases, this is not good practice. You should use the Administration Server only for administrative purposes. Use the Administration Console to define new managed servers and associate the applications with those servers. For more information, see *Using WebLogic Server Clusters* and *Overview of WebLogic System Administration* in the *Administration Guide*.

Deployment

By default, WebLogic Server version 7.0 uses the two-phase deployment model. For more information on this deployment model and other 7.0 deployment features, see *WebLogic Server Deployment* in *Developing WebLogic Server Applications*. Therefore, if you deploy a 4.5 or 5.1 application in your 7.0 server, the deployment model is unspecified and, thus, uses a two-phase deployment. For more information, see the *Release Notes*.

Plug-ins

The communication between the plug-in and WebLogic Server 4.5 and 5.1 is clear text. The plug-ins in WebLogic Server 7.0 support SSL communication between the plug-in and the back-end WebLogic Server.

To upgrade the plug-in, copy the new plug-in over the old one and restart the IIS, Apache, or iPlanet web server.

FileServlet

In WebLogic Server 6.1 Service Pack 2 and later, the behavior of FileServlet, which is the default servlet for a Web Application, has changed. FileServlet now includes the SERVLET_PATH when determining the source filename. This means that it is possible to explicitly only serve files from specific directories by mapping the FileServlet to `/dir/*` etc.

See Setting Up a Default Servlet.

Internationalization (I18N)

Several internationalization and localization changes have been made in this version:

- Changes to the log file format affect the way that messages are localized. The new message format also has additions to the first line: *begin marker*, *machine name*, *server name*, *thread id*, *user id*, *tran id*, and *message id*.
- A new internationalized logging API enables users to log messages in the server and clients.
- Clients log to their own logfiles, which are in the same format as the server logfiles, with the exception of the *servername* and *threadid* fields.
- `LogServicesDef` is deprecated. Instead, use the internationalized API or `weblogic.logging.NonCatalogLogger` (when internationalization is not required).

For details on internationalization in this version, see the *Internationalization Guide*.

Java Transaction API (JTA)

JTA has changed as follows:

- WebLogic Server 7.0 supports the JTA 1.0.1 specification. Updated JTA documentation is provided in *Programming WebLogic JTA*.

- Based on the inclusion of support for JTA, the JTS JDBC driver (with properties in `weblogic.jts.*` and URL `jdbc:weblogic:jts`:..) has been replaced by a JTA JDBC/XA driver. Existing properties are available for backward compatibility, but you should change the class name and properties to reflect the JTS to JTA name change.

Java Database Connectivity (JDBC)

The following changes have been made to JDBC:

- The WebLogic T3 API was deprecated in WebLogic Server 6.1; use the RMI JDBC driver in its place. This also applies to users porting from WebLogic Server 4.5.x.
- The `weblogic.jdbc20.*` packages are being replaced with `weblogic.jdbc.*` packages. All WebLogic JDBC drivers are now compliant with JDBC 2.0.
- If you have a current connection and are using a `preparedStatement`, and the stored procedure gets dropped in the DBMS, use a new name to create the stored procedure. If you recreate the stored procedure with the same name, the `preparedStatement` will not know how to access the newly created stored procedure—it is essentially a different object with the same name.

JSP

Error Handling

The behavior of the JSP include directive has changed between WebLogic Server 5.1 and the current version. In versions through WebLogic Server 5.1, the JSP include directive logged a Warning-level message if it included a non-existent page. In WebLogic Server 6.0 and later, it reports 500 Internal Server Error in that case.You can avert the error by placing an empty file at the referenced location.

Null Attributes

Due to a change in the JSP specification, null request attributes now return the string "null" instead of an empty string. WebLogic Server versions since 6.1 contain a new flag in `weblogic.xml` called `printNulls` which is true by default, meaning that returning "null" is the default. Setting `printNulls` to false ensures that expressions with "null" results are printed as an empty string, not the string "null."

An example of configuring the `printNulls` element in `weblogic.xml`:

```
<weblogic-web-app>

<jsp-param>

<param-name>printNulls</param-name>

<param-value>false</param-value>

</jsp-param>

</weblogic-web-app>
```

JVM

WebLogic Server 7.0 installs the Java Virtual Machine (JVM), JDK 1.3.1_02, with the server installation. The `setenv.sh` scripts provided with the server all point to the JVM. The latest information regarding certified JVMs is available at the *Certifications Page*.

RMI

The following tips are for users porting to WebLogic Server 7.0 who used RMI in their previous version of WebLogic Server:

- Re-run the WebLogic RMI compiler, `weblogic.rmic`, on any existing code to regenerate the wrapper classes so they are compatible with WebLogic Server 7.0.
- Use `java.rmi.Remote` to tag interfaces as remote. Do not use `weblogic.rmi.Remote`.
- Use `java.rmi.*Exception` (e.g., `import java.rmiRemoteException;`). Do not use `weblogic.rmi.*Exception`.

- Use JNDI instead of `*.rmi.Naming`.
- Use `weblogic.rmic` to generate dynamic proxies and bytecode; with the exception of RMI IIOP, stubs and skeletons classes are no longer generated.

Note: For more information, see *WebLogic RMI Fea tures and Guidelines* in *Programming WebLogic RMI*.

- Use `weblogic.rmi.server.UnicastRemoteObject.exportObject()` to get a stub instance.
- The RMI examples have not currently been updated to use java.rmi.* and JNDI. The examples will be revised to reflect `java.rmi.*` and JNDI in a future release.

Security

Upgrading to the New Security Architecture

WebLogic Server 7.0 has a new security architecture. Upgrading WebLogic Server 4.5 or 5.1 to the security functionality in WebLogic Server 7.0 is a two-step process.

1. Upgrade your security configuration to WebLogic Server 6.x.

   For instructions on how to upgrade security configurations from WebLogic Server 4.5 or 5.1 to WebLogic Server 6.x see the Security section under *Additional Migration and Deployment Considerations* in *Migrating WebLogic Server 4.5 and 5.1 Applications to 6.x*.

2. Upgrade your 6.x security configuration to WebLogic Server 7.0.

   See Upgrading Security in *Upgrading WebLogic Server 6.x to Version 7.0* for details.

For specific information about the new security architecture in WebLogic Server 7.0, see the Security section of the WebLogic Server 7.0 documentation.

Digital Certificates Generated by the Certificate Servlet

Digital certificates obtained through a CSR created by the Certificate Request Generator servlet in WebLogic Server 5.1 cannot be used with this release of WebLogic Server.

When creating a CSR using the Certificate Request Generator servlet in WebLogic Server 5.1, the servlet does not make you specify a password for the private key. The password is required in order to use the private key and associated digital certificate with this release of WebLogic Server.

Use the JDK keytool utility to define a password for the digital certificate's private key. The digital certificate can then be used with this release of WebLogic Server. Before using keytool to define the password for the private key, you may need to delete extra characters at the end of each line in the private key.

Private Keys and Digital Certificates

In this release of WebLogic Server, more stringent checks are performed on private keys and digital certificates. In order to use an existing private key and digital certificate, you must perform the following upgrade steps:

1. If the private key is encrypted, convert the key to PEM format using the `java utils der2pem` command and modify the header as follows:

   ```
   ----------BEGIN ENCRYPTED PRIVATE KEY----------
   ...
   -----------END RSA PRIVATE KEY---------------------
   ```

   If the private key is not in PEM format, you receive the following exception:

   ```
   java.lang.Exception:Cannot read private key from file
   C:\bea700sp5\user_proects\mydomain\privatkey.der
   Make sure password specified in environnment property
   weblogic.management.pkpassword is valid.
   ```

   If the private key is unencrypted, use the `java utils der.2pem` command and modify the header as follows:

   ```
   ----------BEGIN RSA PRIVATE KEY----------
   ...
   ----------END RSA PRIVATE KEY----------
   ```

2. Check to see if the digital certificate has an extra line at the end of the file. The following should be the last line of the certificate file:

```
----------END CERTIFICATE----------
```

Remove any extra lines.

If the existing private key is not password protected, you do not need to specify the `weblogic.management.pkpassword` argument when starting the server.

When configuring the SSL protocol in the WebLogic Server Administration Console, note that the Key Encrypted attribute is not used to dictate whether or not the private key is password encrypted. The attribute is irrelevant if a password is not used for the private key passphrase.

If you want to import the converted private key and digital certificate into a keytore, use `java utils.ImportPrivateKey`.

Session Porting

WebLogic Server 6.0 and later does not recognize cookies from previous versions because cookie format changed with 6.0. WebLogic Server ignores cookies with the old format, and creates new sessions.

The default name for cookies has changed from 5.1, when it was `WebLogicSession`. Beginning in WebLogic 6.0, cookies are named `JSESSIONID` by default.

See *weblogic.xml Deployment Descriptor Elements* in *Assembling and Configuring Web Applications* for more information.

Standalone HTML and JSPs

In the original domain provided with WebLogic Server 7.0, as well as in any domains that have been created using the `weblogic.properties` file converter, `domain\applications\DefaultWebApp_myserver` directory is created. This directory contains files made available by your Web server. You can place HTML and JSP files here and make them available, separate from any applications you install. If necessary, you can create subdirectories within the `DefaultWebApp_myserver` directory to handle relative links, such as image files.

Web Components

The following tips are for users porting to WebLogic Server 7.0 who used Web components in their previous version of WebLogic Server:

- All Web components in WebLogic Server now use Web Applications as the mechanism for defining how WebLogic Server serves up JSPs, servlets, and static HTML pages. In a new installation of WebLogic Server, the server will configure a default Web Application. Customers upgrading to WebLogic Server 7.0 should not need to perform any registrations because this default Web Application closely approximates the document root, the JSPServlet, and servlet registrations performed using the `weblogic.properties` file contained in earlier versions.
- Convert your existing `weblogic.properties` file to XML files using the Administration Console. See the *Console Help* for more details.
- SSI is no longer supported.
- URL ACLs are deprecated. Use Servlet 2.3 features instead.
- Some information has moved from `web.xml` to `weblogic.xml`. This reorganization allows a third-party Web application based strictly on Servlet 2.3 to be deployed without modifications to its J2EE standard deployment descriptor (`web.xml`). WebLogic Server 5.1 style settings made in the `web.xml` file using `<context-param>` elements are supported for backward compatibility, but you should adopt the new way of deploying. The following sets of parameters previously defined in `web.xml` are now defined in `weblogic.xml`:

  JSP Parameters (`keepgenerated, precompile compileCommand, verbose, packagePrefix, pageCheckSeconds, encoding`)

  HTTP sessionParameters (`CookieDomain, CookieComment, CookieMaxAgeSecs, CookieName, CookiePath, CookiesEnabled, InvalidationIntervalSecs, PersistentStoreDir, PersistentStorePool, PersistentStoreType, SwapIntervalSecs, IDLength, CacheSize, TimeoutSecs, JDBConnectionTimeoutSecs, URLRewritingEnabled`)
- For more information, see *Writing Web Application Deployment Descriptors* in *Assembling and Configuring Web Applications*.

Wireless Application Protocol Applications

To run a Wireless Application Protocol (WAP) application on WebLogic Server 7.0, you must now specify the MIME types associated with WAP in the `web.xml` file of the web application. In WebLogic Server 5.1, the default mime-type can be set using `weblogic.httpd.defaultMimeType` in `weblogic.properties` where its default value is "text/plain". WebLogic Server 6.0, WebLogic Server 6.1, and WebLogic Server 7.0 do not have a default `mime-type`. You must explicitly specify `mime-type` for each extension in the `web.xml` file.. For information on required MIME types see *Programming WebLogic Server for Wireless Services*. For information on creating and editing a `web.xml` file, see *Writing Web Application Deployment Descriptors* in *Assembling and Configuring Web Applications*.

An example configuration of the `mime-types` in the `web.xml` file:

```
<web-app>
  <mime-mapping>
    <extension>tiff</extension>
    <mime-type>image/tiff</extension>
  </mime-mapping>
  <mime-mapping>
    <extension>tif</extension>
    <mime-type>image/tiff</extension>
  </mime-mapping>
</web-app>
```

Writable config.xml File

WebLogic Server 7.0 automatically updates configuration information read from the 6.x `config.xml` file to include version 7.0 information. In order for these changes to be retained between invocations of the server, the `config.xml` file must be writable. To allow the file to be writable, make a back-up copy of your `config.xml` file from your 6.x configuration and change the file attributes.

XML 7.0 Parser and Transformer

The built-in parser and transformer in WebLogic Server 7.0 have been updated to Xerces 1.4.4 and Xalan 2.2, respectively. If you used the APIs that correspond to older parsers and transformers that were shipped in previous versions of WebLogic Server, and if you used classes, interfaces, or methods that have been deprecated, you might receive deprecation messages in your applications .

WebLogic Server 7.0 also includes the WebLogic FastParser, a high-performance XML parser specifically designed for processing small to medium size documents, such as SOAP and WSDL files associated with WebLogic Web services. Configure WebLogic Server to use FastParser if your application handles mostly small to medium size (up to 10,000 elements) XML documents.

The WebLogic Server 7.0 distribution no longer includes the unmodified Xerces parser and Xalan transformer in the `WL_HOME\server\ext\xmlx.zip` file.

Deprecated APIs and Features

The following APIs and features are deprecated in anticipation of future removal from the product:

- WebLogic Events

  WebLogic Events are deprecated and should be replaced by JMS messages with NO_ACKNOWLEDGE or MULTICAST_NO_ACKNOWLEDGE delivery modes. See *Non-transacted session* in *Programming WebLogic JMS* for more information.

- WebLogic HTMLKona
- WebLogic JDBC t3 Driver. See Deprecation of WebLogic File Services at http://bt04/stage/wls/docs70/file/filesrvc.html#1028649.
- WebLogic Enterprise Connectivity
- WebLogic Time Services is deprecated and should be replaced by JMX Timer Service. For documentation of JMX Timer Service, see *Interface TimerMBean* and *Class Timer*.
- WebLogic Workspaces

- Zero Administration Client (ZAC) is deprecated and should be replaced by JavaWebStart.
- `-Dweblogic.management.host`
- `weblogic.deploy` is deprecated in this release of WebLogic Server 7.0 and is replaced by `weblogic.Deployer`. For more information, see *Deployment Tools and Procedures* in *Developing WebLogic Server Applications*.
- `weblogic.management.tools.WebAppComponentRefreshTool` and `weblogic.refresh` are both deprecated in this release of WebLogic Server 7.0. They have been replaced by `weblogic.Deployer`.

Removed APIs and Features

The following APIs and features have been removed:

- The old administrative console GUI
- The Deployer Tool
- WebLogic Beans
- WebLogic jHTML
- WebLogic Remote
- WorkSpaces
- WebLogic Server Tour
- T3Client
- Jview support
- SSI
- Weblogic Bean Bar
- RemoteT3
- Jview support
- Weblogic COM

This feature relied on the Microsoft JVM (Jview), which is no longer supported.

A The weblogic.properties Mapping Table

The weblogic.properties mapping table shows which `config.xml, web.xml, or weblogic.xml` attribute handles the function formerly performed by `weblogic.properties` properties.

weblogic.properties file Property	**.xml Configuration Attribute**	**Console Label**
`weblogic.administrator.email`	`config.xml:` `EmailAddress` (`Administrator` element)	
`weblogic.administrator.location`	`config.xml:` `Notes (freeform, optional)` (`Administrator` element)	
`weblogic.administrator.name`	`config.xml:` `Name` (`Administrator` element)	
`weblogic.administrator.phone`	`config.xml:` `PhoneNumber` (`Administrator` element)	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.cluster.defaultLoadAlgorithm`	`config.xml: DefaultLoadAlgorithm` (`Cluster` element)	Clusters: *clustername*: Configuration: General: Default Load Algorithm
`weblogic.cluster.multicastAddress`	`config.xml: MulticastAddress` (`Cluster` element)	Clusters: *clustername*: Configuration: Multicast: Multicast Address
`weblogic.cluster.multicastTTL`	`config.xml: MulticastTTL` (`Cluster` element)	Clusters: *clustername*: Configuration: Multicast: Multicast TTL
`weblogic.cluster.name`	`config.xml Cluster Address` (`Cluster` element)	Clusters: *clustername*: Configuration: General: Cluster Address
`weblogic.httpd.authRealmName`	`config.xml: AuthRealmName` (`WebAppComponent` element)	Deployments: Web Applications: *applicationname*: Configuration: Other: Auth Realm Name
`weblogic.httpd.charsets`	`config.xml: Charsets` (`WebServer` element)	
`weblogic.httpd.clustering.enable`	`config.xml: ClusteringEnabled` (`WebServer` element)	
`weblogic.httpd.defaultServerName`	`config.xml DefaultServerName` (`WebServer` element)	Servers: *servername*: Configuration: HTTP: Default Server Name

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.httpd.defaultServlet`	`web.xml`: define a servlet-mapping with the URL pattern of / `<servlet-mapping>` element	
`weblogic.httpd.defaultWebApp`	`config.xml: DefaultWebApp` (`WebServer` element)	
`weblogic.httpd.enable`	`config.xml: HttpdEnabled` (`Server` element)	
`weblogic.httpd.enableLogFile`	`config.xml: LoggingEnabled` (`WebServer` element)	
`weblogic.httpd.http.keepAliveSecs`	`config.xml: KeepAliveSecs` (`WebServer` element)	
`weblogic.httpd.https.keepAliveSecs`	`config.xml: HttpsKeepAliveSecs` (`WebServer` element)	
`weblogic.httpd.indexDirectories`	`config.xml: IndexDirectoryEnabled` (`WebAppComponent` element)	Deployments: Web Applications: *`applicationname`*: Configuration: Files: Index Directories
`weblogic.httpd.keepAlive.enable`	`config.xml: KeepAliveEnabled` (`WebServer` element)	Servers: *`servername`*: Configuration: HTTP: Enable Keep Alives
`weblogic.httpd.logFileBufferKBytes`	`config.xml: LogFileBufferKBytes` (`WebServer` element)	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.httpd.logFileFlushSecs`	`config.xml:` `LogFileFlushSecs` (`WebServer` element)	
`weblogic.httpd.logFileFormat`	`config.xml:` `LogFileFormat` (`WebServer` element)	Services: Virtual Host: Log File Format
`weblogic.httpd.logFileName`	`config.xml:` `LogFileName` (`WebServer` element)	Services: Virtual Host: Log File Name
`weblogic.httpd.logRotationPeriodMins`	`config.xml:` `LogRotationTimeBegin` (`WebServer` element)	
`weblogic.httpd.logRotationPeriodMins`	`config.xml:` `LogRotationPeriodMins` (`WebServer` element)	
`weblogic.httpd.logRotationType`	`config.xml:` `LogRotationType` (`WebServer` element)	Servers: *`servername:`* Logging: HTTP: Rotation Type
`weblogic.httpd.maxLogFileSizeKBytes`	`config.xml:` `MaxLogFileSizeKBytes` (`WebServer` element)	Servers: *`servername`*: Logging: HTTP: Max Log File Size Kbytes
`weblogic.httpd.mimeType`	`web.xml:` `mime-type` (`<mime-mapping>` element)	
`weblogic.httpd.postTimeoutSecs`	`config.xml:` `PostTimeoutSecs` (`WebServer` element)	Servers: *`servername`*: Configuration: HTTP: Post Timeout Secs

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.httpd.servlet.extensionCaseSensitive`	`config.xml:` `ServletExtensionCaseSensitive` (`WebAppComponent` element)	Deployments: Web Applications: *applicationname*: Configuration: Files: Case Sensitive Extensions
`weblogic.httpd.servlet.reloadCheckSecs`	`config.xml:` `ServletReloadCheckSecs` (`WebAppComponent` element)	Deployments: Web Applications: *applicationname*: Configuration: Files: Reload Period
`weblogic.httpd.servlet.SingleThreadedModelPoolSize`	`config.xml:` `SingleThreadedServletPoolSize` (`WebAppComponent` element)	Deployments: Web Applications: *applicationname*: Configuration: Files: Single Threaded Servlet Pool Size
`weblogic.httpd.session.cacheEntries`	`weblogic.xml:` `CacheSize` `<param-name>/<param-value>` element pair	Servers: *servername*: Configuration: SSL: Certificate Cache Size
`weblogic.httpd.session.cookie.comment`	`weblogic.xml:` `CookieComment` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.cookie.domain`	`weblogic.xml:` `CookieDomain` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.cookie.maxAgeSecs`	`weblogic.xml:` `CookieMaxAgeSecs` `<param-name>/<param-value>` element pair	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.httpd.session.cookie.name`	`weblogic.xml: CookieName` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.cookie.path`	`weblogic.xml: CookiePath` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.cookies.enable`	`weblogic.xml: CookiesEnabled` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.debug`	`weblogic.xml: SessionDebuggable` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.enable`	`weblogic.xml: TrackingEnabled` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.invalidationintervalSecs`	`weblogic.xml: InvalidationIntervalSecs` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.jdbc.connTimeoutSecs`	`weblogic.xml: JDBCConnectionTimeoutSecs` `<param-name>/<param-value>` element pair	

weblogic.properties file Property	**.xml Configuration Attribute**	**Console Label**
`weblogic.httpd.session.persistentStoreDir`	`weblogic.xml: PersistentStoreDir` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.persistentStorePool`	`weblogic.xml: PersistentStorePool` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.persistentStoreShared`	`weblogic.xml: SessionPersistentStoreShared` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.persistentStoreType`	`weblogic.xml: PersistentStoreType` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.sessionIDLength`	`weblogic.xml: IDLength` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.swapintervalSecs`	`weblogic.xml: SwapIntervalSecs` `<param-name>/<param-value>` element pair	
`weblogic.httpd.session.timeoutSecs`	`weblogic.xml: TimeoutSecs` `<param-name>/<param-value>` element pair	Servers: *`servername`*: Configuration: HTTP: Post Timeout Secs
`weblogic.httpd.session.URLRewriting.enable`	`weblogic.xml: URLRewritingEnabled` `<param-name>/<param-value>` element pair	

weblogic.properties file Property	**.xml Configuration Attribute**	**Console Label**
`weblogic.httpd.tunneling.clientPingSecs`	`config.xml: TunnelingClientPingSecs` (`Server` element)	Servers: *`servername:`* Configuration: Tuning: Tunneling Client Ping
`weblogic.httpd.tunneling.clientTimeoutSecs`	`config.xml: TunnelingClientTimeoutSecs` (`Server` element)	Servers: *`servername:`* Configuration: Tuning: Tunneling Client Timeout
weblogic.httpd.tunnelingenabled	config.xml TunnelingEnabled (`Server` element)	Servers: *`servername:`* Configuration: Tuning: Enable Tunneling
`weblogic.httpd.URLResource`	`config.xml: URLResource` (`WebServer` element)	
`weblogic.iiop.password`	`config.xml: DefaultIIOPPassword` (`Server` element)	Servers: *`servername:`* Configuration: Protocols: Default IIOP Password
`weblogic.iiop.user`	`config.xml: DefaultIIOPUser` (`Server` element)	Servers: *`servername:`* Configuration: Protocols: Default IIOP User

weblogic.properties file Property	**.xml Configuration Attribute**	**Console Label**
`weblogic.jdbc.connectionPool`		
■ `url=`*URL for JDBC Driver*	`URL`	
■ `driver=`*full package name for JDBC driver*	`DriveName`	
■ `loginDelaySecs=`*seconds between connections*	`LoginDelaySeconds`	
■ `initialCapacity=`*initial number of JDBC connections*	`InitialCapacity`	
■ `maxCapacity=`*maximum number of JDBC connections*	`MaxCapacity`	
■ `capacityIncrement=`*increment interval*	`CapacityIncrement`	
■ `allowShrinking=`*true to allow shrinking*	`AllowShrinking`	
■ `shrinkPeriodMins=`*interval before shrinking*	`ShrinkPeriodMinutes`	
■ `testTable=`*name of table for autorefresh test*	`TestTableName`	
■ `refreshTestMinutes=`*interval for autorefresh test*	`RefreshMinutes`	
■ `testConnsOnReserve=`*true to test connection at reserve*	`TestConnectionsOnReserve`	
■ `testConnsOnRelease=`*true to test connection at release*	`TestConnectionsOnRelease`	
■ `props=`*props for JDBC connection*	`Properties`	
	`JDBCConnectionPool Element`	
	`ConnLeakProfilingEnabled`	
	`ACLName`	
	`CapacityEnabled`	
	`SupportsLocalTransaction`	
	`KeepLogicalConnOpenOnRelease`	
	`Password`	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
weblogic.jdbc.enableLogFile	config.xml: JDBCLoggingEnabled (Server element)	
weblogic.jdbc.logFileName	config.xml: JDBCLogFileName (Server element)	
weblogic.jms.ConnectionConsumer	config.xml JMSConnectionConsumer element MessagesMaximum Selector Destination	
weblogic.jms.connectionFactoryArgs.<<factoryName>> ■ ClientID ■ DeliveryMode ■ TransactionTimeout	config.xml: JMSConnectionFactory element ClientID DefaultDeliveryMode TransactionTimeout UserTransactionsEnabled AllowCloseInOnMessage	
weblogic.jms.connectionFactoryName	config.xml: JMSConnectionFactory element JNDIName	
weblogic.jms.connectionPool	ConnectionPool (JMSJDBCStore element)	
weblogic.jms.queue	config.xml: JNDIName StoreEnabled (JMSDestination element)	

weblogic.properties file Property	**.xml Configuration Attribute**	**Console Label**
`weblogic.jms.queueSessionPool`	`config.xml:` `ConnectionConsumer` `ConnectionFactory` `ListenerClass` `AcknowledgeMode` `SessionsMaximum` `Transacted` (`JMSSessionPool` element)	
`weblogic.jms.tableNamePrefix`	`config.xml:` `PrefixName`	
`weblogic.jms.topic`	`config.xml` `JNDIName` `StoreEnabled` (`JMSDestination` element)	Services: JMS: Connection Factories: JNDI Name
`weblogic.jms.topicSessionPool`	`config.xml:` `ConnectionConsumer` `ConnectionFactory` `ListenerClass` `AcknowledgeMode` `SessionsMaximum` `Transacted` (`JMSSessionPool` element)	
`weblogic.jndi.transportableObjectFactories`	`config.xml:` `JNDITransportableObjectFactoryList` (`Server` element)	Servers: *`servername:`*
`weblogic.login.readTimeoutMillisSSL`	`config.xml` `LoginTimeoutMillis` (`SSL` element)	Servers: *`servername:`*
`weblogic.security.audit.provider`	`config.xml` `AuditProviderClassName` (`Security` element)	Security: General: Audit Provider Class

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.security.certificate.authority`	`config.xml` `ServerCertificateChainFileName` (`SSL` element)	Servers: *servername*: Configuration: SSL: Server Certificate Chain File Name
`weblogic.security.certificate.server`	`config.xml:` `ServerCertificateFileName` (`SSL` element)	Servers: *servername*: Configuration: SSL: Server Certificate File Name
`weblogic.security.certificateCacheSize`	`config.xml:` CertificateCacheSize (`SSL` element)	Servers: *servername*: Configuration: SSL: Certificate Cache Size
`weblogic.security.clientRootCA`	`config.xml:` `TrustedCAFileName` (`SSL` element)	Servers: *servername*: Configuration: SSL: Trusted CA File Name
`weblogic.security.disableGuest`	`config.xml:` `GuestDisabled` (`Security` element)	Security: General: Guest Disabled
`weblogic.security.enforceClientCert`	`config.xml:` ClientCertificateEnforced (`SSL` element)	Servers: *servername*: Configuration: SSL: Client Certificate Enforced
`weblogic.security.key.export.lifespan`	`config.xml:` ExportKeyLifespan (`SSL` element)	Servers: *servername*: Configuration: SSL: Export Key Lifespan
`weblogic.security.key.server`	`config.xml:` `ServerKeyFileName` (`SSL` element)	Servers: *servername*: Configuration: SSL: Server Key File Name
`weblogic.security.ldaprealm.authentication`	`config.xml:` `AuthProtocol` (`LDAPRealm` element)	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.security.ldaprealm.credential`	`config.xml: Credential` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.factory`	`config.xml LdapProvider` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.groupDN`	`config.xml: GroupDN` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.groupIsContext`	`config.xml: GroupIsContext` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.groupNameAttribute`	`config.xml: GroupNameAttribute` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.groupUsernameAttribute`	`config.xml: GroupUsernameAttribute` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.principal`	`config.xml: Principal` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.ssl`	`config.xml: SSLEnable` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.url`	`config.xml: LDAPURL` (`LDAPRealm` element)	
`weblogic.security.ldaprealm.userAuthentication`	`config.xml: UserAuthentication` (`LDAPRealm` element)	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
weblogic.security.ldaprealm.userDN	config.xml: UserDN (LDAPRealm element)	
weblogic.security.ldaprealm.userNameAttribute	config.xml: UserNameAttribute (LDAPRealm element)	
weblogic.security.ldaprealm.userPasswordAttribute	config.xml: UserPasswordAttribute (LDAPRealm element)	
weblogic.security.net.connectionFilter	config.xml: ConnectionFilter (Security element)	
weblogic.security.ntrealm.domain	config.xml: PrimaryDomain (NTRealm element)	
weblogic.security.realm.cache.acl.enable	config.xml: ACLCacheEnable (CachingRealm element)	
weblogic.security.realm.cache.acl.size	config.xml: ACLCacheSize (CachingRealm element)	
weblogic.security.realm.cache.acl.ttl.negative	config.xml: ACLCacheTTLNegative (CachingRealm element)	
weblogic.security.realm.cache.acl.ttl.positive	config.xml: ACLCacheTTLPositive (CachingRealm element)	
weblogic.security.realm.cache.auth.enable	config.xml: AuthenticationCacheEnable (CachingRealm element)	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.security.realm.cache.auth.size`	`config.xml: AuthenticationCacheSize` (`CachingRealm` element)	
`weblogic.security.realm.cache.auth.ttl.negative`	`config.xml: AuthenticationCacheTTLNegative` (`CachingRealm` element)	
`weblogic.security.realm.cache.auth.ttl.positive`	`config.xml: AuthenticationCacheTTLPositive` (`CachingRealm` element)	
`weblogic.security.realm.cache.caseSensitive`	`config.xml: CacheCaseSensitive` (`CachingRealm` element)	
`weblogic.security.realm.cache.group.enable`	`config.xml: GroupCacheEnable` (`CachingRealm` element)	
`weblogic.security.realm.cache.group.size`	`config.xml: GroupCacheSize` (`CachingRealm` element)	
`weblogic.security.realm.cache.group.ttl.negative`	`config.xml: GroupCacheTTLNegative` (`CachingRealm` element)	
`weblogic.security.realm.cache.group.ttl.positive`	`config.xml: GroupCacheTTLPositive` (`CachingRealm` element)	
`weblogic.security.realm.cache.perm.enable`	`config.xml: PermissionCacheEnable` (`CachingRealm` element)	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.security.realm.cache.perm.size`	`config.xml:` `PermissionCacheSize` (`CachingRealm` element)	
`weblogic.security.realm.cache.perm.ttl.negative`	`config.xml:` `PermissionCacheTTLNegative` (`CachingRealm` element)	
`weblogic.security.realm.cache.perm.ttl.positive`	`config.xml:` `PermissionCacheTTLPositive` (`CachingRealm` element)	
`weblogic.security.realm.cache.user.enable`	`config.xml:` `UserCacheEnable` (`CachingRealm` element)	
`weblogic.security.realm.cache.user.size`	`config.xml:` `UserCacheSize` (`CachingRealm` element)	
`weblogic.security.realm.cache.user.ttl.negative`	`config.xml:` `UserCacheTTLNegative` (`CachingRealm` element)	
`weblogic.security.realm.cache.user.ttl.positive`	`config.xml:` `UserCacheTTLPositive` (`CachingRealm` element)	
`weblogic.security.realm.certAuthenticator`	`config.xml:` `CertAuthenticator` (`SSL` element)	Servers: *servername*: Configuration: SSL: Cert Authenticator
`weblogic.security.SSL.ciphersuite`	`config.xml` `Ciphersuites` (`SSL` element)	

weblogic.properties file Property	.xml Configuration Attribute	Console Label
`weblogic.security.ssl.enable`	`config.xml:` `Enabled` (`SSL` element)	Servers: *servername*: Configuration: SSL: Enabled
`weblogic.security.SSL.hostnameVerifier`	`config.xml` `HostnameVerifier` (`SSL` element)	Servers: *servername*: Configuration: SSL: Hostname Verifier
`weblogic.security.SSL.ignoreHostnameVerification`	`config.xml` `HostNameVerificationIgnored` (`SSL` element)	
`weblogic.security.SSLHandler.enable`	`config.xml:` `HandlerEnabled` (`SSL` element)	Servers: *servername*: Configuration: SSL: Handler Enabled
`weblogic.security.unixrealm.authProgram`	`config.xml:` `AuthProgram` (`UnixRealm` element)	
`weblogic.system.AdministrationPort`	`config.xml` `AdministrationPort` (`Server` element)	Servers: *servername:* Configuration: General: Administration Port
`weblogic.system.bindAddr`	`config.xml:` `ListenAddress` (`Server` element)	
`weblogic.system.defaultProtocol`	`config.xml:` `DefaultProtocol` (`Server` element)	Servers: *servername:* Configuration: Protocols: Default Protocol
`weblogic.system.defaultSecureProtocol`	`config.xml:` `DefaultSecureProtocol` (`Server` element)	Servers: *servername:* Configuration: Protocols: Default Secure Protocol

weblogic.properties file Property	.xml Configuration Attribute	Console Label
weblogic.system.enableConsole	config.xml: StdoutEnabled (Kernel element)	Servers: *servername:* Logging: General: Log to Stdout
weblogic.system.enableIIOP	config.xml: IIOPEnabled (Server element)	
weblogic.system.enableReverseDNS Lookups	config.xml: ReverseDNSAllowed (Server element)	
weblogic.system.enableSetGID,	config.xml: PostBindGID	
weblogic.system.enableSetUID,	config.xml: PostBindUIDEnabled	
weblogic.system.enableTGIOP	config.xml TGIOPEnabled (Server element)	Servers: *servername:*
weblogic.system.helpPageURL	config.xml HelpPageURL (Server element)	Servers: *servername:*
weblogic.system.home	config.xml: RootDirectory (Server element)	
weblogic.system.ListenPort	config.xml ListenPort (Server element)	Servers: *servername*: Configuration: SSL: Listen Port
weblogic.system.logFile	config.xml: FileName (Log element)	

weblogic.properties file Property	**.xml Configuration Attribute**	**Console Label**
`weblogic.system.MagicThreadBackToSocket`	`config.xml: MagicThreadDumpBackToSocket` (`ServerDebug` element)	
`weblogic.system.MagicThreadDumpFile`	`config.xml: MagicThreadDumpFile` (`ServerDebug` element)	
`weblogic.system.MagicThreadDumpHost`	`config.xml: MagicThreadDumpHost` (`ServerDebug` element)	
`weblogic.system.magicThreadDumps`	`config.xml: MagicThreadDumpEnabled` (`ServerDebug` element)	
`weblogic.system.maxLogFileSize`	`config.xml: FileMinxSize` (`Log` element)	
`weblogic.system.nativeIO.enable`	`config.xml: NativeIOEnabled` (`Server` element)	Servers: *`servername:`* Configuration: Tuning: Enable Native IO
`weblogic.system.nonPrivGroup`	`config.xml PostBindGID` (`UnixMachine` element)	
`weblogic.system.nonPrivUser`	`config.xml PostBindUID` (`UnixMachine` element)	
`weblogic.system.percentSocketReaders`	`config.xml: ThreadPoolPercentSocketReaders` (`Kernel` element)	Servers: *`servername:`* Configuration: Tuning: Socket Readers

weblogic.properties file Property	**.xml Configuration Attribute**	**Console Label**
`weblogic.system.readTimeoutMillis`	`config.xml: LoginTimeoutMillis` (`Server` element)	Servers: *`servername:`*
`weblogic.system.SSL.useJava`	`config.xml: UseJava` (`SSL` element)	Servers: *`servername`*: Configuration: SSL: Use Java
`weblogic.system.SSLListenPort`	`config.xml: ListenPort` (`SSL` element)	Servers: *`servername`*: Configuration: SSL: Listen Port
`weblogic.system.startupFailureIsFatal`	`config.xml FailureIsFatal` (`StartupClass` element)	
`weblogic.system.user`	`config.xml: SystemUser` (`Security` element)	
`weblogic.system.weight`	`config.xml ClusterWeight` (`Server` element)	Servers: *`servername:`* Configuration: Cluster: Cluster Weight

B Upgrading the Pet Store Application and the Examples Server

This appendix presents examples of the following processes:

- Upgrading the Pet Store Application From WebLogic 6.1 Service Pack 3 to WebLogic Server 7.0
- Upgrading the WebLogic 6.0 Service Pack 2 Examples Server to WebLogic Server 7.0
- Upgrading the WebLogic 6.1 Service Pack 2 Examples Server to WebLogic Server 7.0

Note: The WebLogic Server 7.0 examples and PetStore are configured to use the default security configuration. It is not possible to run the WebLogic Server 7.0 examples and PetStore in Compatibility security.

Terms Used in This Document

Where all three versions are being discussed, the instructions use version-specific terms for WebLogic home directories. These conventions are used in this document to make explaining how to port your domain configurations easier.

In this document `WL_HOME` is defined to be the home of WebLogic Server 6.x and 7.0.

For 6.0 `WL_HOME=D:\WLS_6.0\wlserver6.0`

For 6.1 `WL_HOME=D:\WLS_6.1\wlserver6.1`

For 7.0 `WL_HOME=D:\WLS_7.0\weblogic700`

Upgrading the Pet Store Application From WebLogic 6.1 Service Pack 3 to WebLogic Server 7.0

It is not necessary to upgrade the WebLogic 6.1 Service Pack 3 Pet Store application to WebLogic Server 7.0; this section just provides the steps to do so as an example of how to upgrade an application from 6.1 to 7.0. To upgrade the WebLogic 6.1 Service Pack 3 Pet Store application for use on WebLogic Server 7.0:

1. Install WebLogic Server 7.0.
2. Set Up the WebLogic Server 7.0 Environment with Your 6.1 Service Pack 3 Domain Configuration.
3. Start the Pet Store Application on WebLogic Server 7.0.

Install WebLogic Server 7.0

Install WebLogic Server 7.0. See the *Installation Guide*.

Note: Installing the new version in the exact location of the old version is explicitly prohibited by the installer.

Set Up the WebLogic Server 7.0 Environment with Your 6.1 Service Pack 3 Domain Configuration

To upgrade the Pet Store application from WebLogic Server 6.1 Service Pack 3 to WebLogic Server 7.0:

1. Repair tag library errors in Pet Store that were accepted under WebLogic Server 6.x but are rejected by the parser in WebLogic Server 7.0.
2. Copy the `WL_HOME/config/petstore` directory from WebLogic Server 6.1 to a location in your WebLogic Server 7.0 installation. BEA recommends not copying the directory to `WL_HOME` in order to avoid having to move the domain the next time you upgrade.

Note: If the directory in WebLogic Server 7.0 to which you copy Pet Store is the same as Pet Store's directory location in WebLogic Server 6.1, you will not need to edit the `config.xml` file to reflect a new directory location. If you copy Pet Store to a different directory location, you will need to find all of the fully qualified file and directory paths in your `config.xml` file and change them to relative paths.

3. Edit the `startPetstore.cmd` script to reflect the new WebLogic Server 7.0 installation as well as the new directory location (if you have one).
4. Edit the `config.xml` file to reflect the new WebLogic Server 7.0 installation as well as the new directory location (if you have one).

This section contains:

- Instructions for fixing the Pet Store errors: "Fix JSP Parsing Errors" on page 4.
- An example of the "startPetstore.cmd script used to boot WebLogic Server 6.1 Service Pack 3" on page 7.
- An example of the "above-listed startPetstore.cmd script modified to boot WebLogic Server 7.0" on page 9.
- An example of the "config.xml file used for WebLogic Server 6.1 Service Pack 3" on page 12.

- An example of the above-listed config.xml file used for WebLogic Server 7.0 which includes explanations for how to change this script to upgrade the Examples Server toWebLogic Server 7.0.

Note: To upgrade the Pet Store application to WebLogic Server 7.0, it is not necessary to update the DTDs in the weblogic.xml and web.xml files. See weblogic-ejb-jar.xml Document Type Definitions in *Programming WebLogic Enterprise JavaBeans* for information on WebLogic Server 7.0 DTDs

Fix JSP Parsing Errors

Minor errors that were parsable in earlier versions of WebLogic Server cause errors in WebLogic Server 8.1 because JDK 1.4 does not accept them. The errors corrected in this section are property settings for which the method and setter properties do not agree.

Correcting the errors requires making changes to these source files:

```
ListTag.java
CartListTag.java
MyListTag.java
ProductItemListTag.java
ProductListTag.java
SearchListTag.java
```

All of these files are located in the `WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\petstore\taglib\list` directory (where `WL_HOME` is the WebLogic Server installation directory)

Use these steps to make the replacement in `ListTag.java`:

1. In a command console, navigate to `WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\petstore\taglib\list`. For example:

   C:\> cd `WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\petstore\taglib\list`

2. Open `ListTag.java` in a text editor. For example:

   ```
   WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprin
   ts\petstore\taglib\list>notepad ListTag.java.
   ```

3. Change:

   ```
   public void setNumItems(String numItemsStr) {
      numItems = Integer.parseInt(numItemsStr);
      }
   ```

   to:

   ```
   public void setNumItems(int numItemsIn) {
      numItems = numItemsIn;
      }
   ```

4. Change:

   ```
   public void setStartIndex(String startIndexStr) {
      startIndex = Integer.parseInt(startIndexStr);
      }
   ```

   to:

   ```
   public void setStartIndex(int startIndexIn) {
      startIndex = startIndexIn;
      }
   ```

5. Save and close `ListTag.java`.

Make the replacements in the rest of the files as follows:

1. In the command console, navigate to `WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\petstore\taglib\list`. For example:

   ```
   C:\>cd
   WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprin
   ts\petstore\taglib\list
   ```

2. Open `CartListTag.java` in a text editor. For example:

   ```
   WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprin
   ts\petstore\taglib\list>notepad CartListTag.java.
   ```

3. Remove the following lines from `CartListTag.java`:

   ```
   public void setNumItems(String numItemsStr) {
      super.setNumItems(numItemsStr);
   ```

```
   }
  public void setStartIndex(String startIndexStr) {
   super.setNumItems(startIndexStr);
   }
```

4. Replace with the following:

```
public void setNumItems(int numItems) {
   super.setNumItems(numItems);
   }


public void setStartIndex(int startIndex) {
   super.setNumItems(startIndex);
   }
```

5. Save and close `ProductListTag.java`.
6. Repeat steps 1 through 5 for the remaining files:

   `MyListTag.java`

   `ProductItemListTag.java`

   `ProductListTag.java`

   `SearchListTag.java`

Rebuild Pet Store

After making the corrections to Pet Store, rebuild the application.

1. In a command console, change to theWebLogic Server 6.x `WL_HOME\config\examples` directory, and set your environment:

   `WL_HOME\config\examples> setexamplesenv.cmd` (or `.sh`)

2. In the same console, change to the `WL_HOME\samples\petStore\src\petstore\src` directory and rebuild:

   ```
   WL_HOME\samples\petStore\src\petstore\src> build
   ```

   The script builds `petstore.ear` to `WL_HOME\samples\petStore\src\petstore\build`.

startPetstore.cmd script used to boot WebLogic Server 6.1 Service Pack 3

```
@echo off

@rem This script can be used to start WebLogic Server for the
purpose
@rem of running the PetStore application. This script ensures
that the server is started
@rem using the config.xml file found in this directory and that
the CLASSPATH
@rem is set appropriately. This script contains the following
variables:
@rem
@rem JAVA_HOME     - Determines the version of Java used to start
@rem                 WebLogic Server. This variable must point
to the
@rem                 root directory of a JDK installation. and
will be set
@rem                 for you by the WebLogic Server installer.
Note that
@rem                 this script uses the hotspot VM to run
WebLogic Server.
@rem               If you choose to use a JDK other than the one
@rem               included in the disribution, make sure that
the JDK
@rem                 includes the hotspot VM. See the WebLogic
platform
@rem                 support page
(http://e-docs.bea.com/wls/platforms/index.html)
@rem                 for an up-to-date list of supported JVMs
on Windows NT.
@rem
@rem When setting these variables below, please use short file
names (8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server
(http://e-docs.bea.com/wls/docs61/install/index.html).
```

```
SETLOCAL

cd ..\..

@rem Set user-defined variables.
set JAVA_HOME=d:\610sp2\jdk131

@rem Check that script is being run from the appropriate
directory
if not exist lib\weblogic.jar goto wrongplace
goto checkJDK

@rem :wrongplace
@rem echo startPetStore.cmd must be run from the config\petStore
directory. 1>&2
@rem goto finish

:checkJDK
if exist "%JAVA_HOME%/bin/javac.exe" goto runWebLogic
echo.
echo Javac wasn't found in directory %JAVA_HOME%/bin.
echo Please edit the startPetStoreServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your JDK
installation.
goto finish

:runWebLogic
echo on
set PATH=.\bin;"%JAVA_HOME%\bin";%PATH%

set CLASSPATH=.;.\lib\weblogic_sp.jar;.\lib\weblogic.jar;.\
samples\eval\cloudscape\lib\cloudscape.jar;.\config\petStore\se
rverclasses
echo off

echo.
echo ***************************************************
echo *  To start WebLogic Server, use the password     *
echo *  assigned to the system user.  The system       *
echo *  username and password must also be used to     *
echo *  access the WebLogic Server console from a web  *
echo *  browser.                                       *
echo ***************************************************

@rem Set WLS_PW equal to your system password for no password
prompt server startup.
set WLS_PW=

@rem Set Production Mode.  When set to true, the server starts
up in production mode.
@rem When set to false, the server starts up in development mode.
```

```
The default is false.
set STARTMODE=true

echo on
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath
"%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="d:\610sp2" -Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="d:\610sp2\wlserver6.1/lib/weblogic.pol
icy" weblogic.Server
goto finish


:finish
cd config\petStore
ENDLOCAL
```

above-listed startPetstore.cmd script modified to boot WebLogic Server 7.0

```
@echo off

@rem This script can be used to start WebLogic Server for the
purpose
@rem of running the PetStore application. This script ensures
that
@rem the server is started using the config.xml file found in
@rem this directory and that the CLASSPATH is set appropriately.
@rem This script contains the following variables:
@rem
@rem JAVA_HOME     - Determines the version of Java used to start
@rem                  WebLogic Server. This variable must point
to the
@rem                  root directory of a JDK installation. and
will be set
@rem                  for you by the WebLogic Server installer.
Note that
@rem                  this script uses the hotspot VM to run
WebLogic Server.
@rem                If you choose to use a JDK other than the one
@rem                included in the disribution, make sure that
the JDK
@rem                  includes the hotspot VM. See the WebLogic
platform
@rem                  support page
(http://e-docs.bea.com/wls/platforms/index.html)
@rem                  for an up-to-date list of supported JVMs
```

```
on Windows NT.
@rem
@rem When setting these variables below, please use short file
names (8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server
(http://e-docs.bea.com/wls/docs61/install/index.html).

SETLOCAL

cd ..\..

@rem Set user-defined variables.
@rem 1.  SET THE NEW JAVA HOME APPROPRIATELY
set JAVA_HOME=D:\70bea\jdk131

@rem 2.  FOR SIMPLICITY, CREATE AND SET BEA_HOME AND WL_HOME70
set BEA_HOME=d:\wls70
set WL_HOME70=%BEA_HOME%\weblogic700

@rem 3.  REMOVE THIS ENTIRE CHECK AND ITS TAG SINCE
@rem NEITHER IS RELEVANT ANY LONGER
@rem Check that script is being run from the appropriate
directory
@rem if not exist lib\weblogic.jar goto wrongplace
@rem goto checkJDK

@rem :wrongplace
@rem echo startPetStore.cmd must be run from the config\petStore
directory. 1>&2
@rem goto finish

:checkJDK
if exist "%JAVA_HOME%/bin/javac.exe" goto runWebLogic echo.
echo Javac wasn't found in directory %JAVA_HOME%/bin.
echo Please edit the startPetStoreServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your JDK
installation.
goto finish
```

```
@rem 4.  SET THE PATH VARIABLE APPROPRIATELY USING WL_HOME70 YOU
DEFINED IN STEP 2 ABOVE
:runWebLogic
echo on
set PATH=%WL_HOME70%\server\bin;"%JAVA_HOME%\bin";%PATH%

@rem 5.  SET YOUR CLASSPATH SO THE NEW WLS70 CLASSES ARE USED
WHILE RETAINING ALL CLASS LOCATIONS
@rem RELEVANT TO YOUR APPLICATION.  TO DO THIS, USE WL_HOME70 YOU
SET IN STEP 2.
set CLASSPATH=.;%WL_HOME70%\server\lib\weblogic.jar;.\samples\
eval\cloudscape\lib\cloudscape.jar;.\config\petStore\serverclas
ses
echo off

echo.
echo ***************************************************
echo *  To start WebLogic Server, use the password     *
echo *  assigned to the system user.  The system       *
echo *  username and password must also be used to     *
echo *  access the WebLogic Server console from a web  *
echo *  browser.                                       *
echo ***************************************************

@rem Set WLS_PW equal to your system password for no password
prompt server startup.
set WLS_PW=

@rem Set Production Mode.  When set to true, the server starts
up in production mode.
@rem When set to false, the server starts up in development mode.
The default is false.
set STARTMODE=true

@rem 6.  SET THE -Dbea.home COMMAND LINE OPTION USING THE
BEA_HOME VARIABLE YOU SET IN STEP 2.
echo on
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath
"%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="%BEA_HOME%"
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="d:\610sp2\wlserver6.1/lib/weblogic.pol
icy" weblogic.Server
goto finish
```

```
:finish
cd config\petStore
ENDLOCAL
```

config.xml file used for WebLogic Server 6.1 Service Pack 3

```
<Domain
Name="petstore">
<JDBCTxDataSource
JNDIName="jdbc.EstoreDB"
Name="EstoreDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<JDBCTxDataSource
JNDIName="jdbc.InventoryDB"
Name="InventoryDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<JDBCTxDataSource
JNDIName="jdbc.SignOnDB"
Name="SignOnDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<Application
Deployed="true"
Name="tour"
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/tour.war">
<WebAppComponent
Name="tour"
Targets="petstoreServer"
URI="tour.war"/>
</Application>

<Application
Deployed="true"
Name="petstore"
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/petstore.ear">
<EJBComponent
Name="customerEjb"
Targets="petstoreServer"
URI="customerEjb.jar"/>

<EJBComponent
Name="inventoryEjb"
```

```
Targets="petstoreServer"
URI="inventoryEjb.jar"/>

<EJBComponent
Name="mailerEjb"
Targets="petstoreServer"
URI="mailerEjb.jar"/>
<EJBComponent
Name="personalizationEjb"
Targets="petstoreServer"
URI="personalizationEjb.jar"/>

<EJBComponent
Name="signonEjb"
Targets="petstoreServer"
URI="signonEjb.jar"/>

<EJBComponent
Name="shoppingcartEjb"
Targets="petstoreServer"
URI="shoppingcartEjb.jar"/>

<EJBComponent
Name="petstoreEjb"
Targets="petstoreServer"
URI="petstoreEjb.jar"/>

<WebAppComponent
Name="petstore"
Targets="petstoreServer"
URI="petstore.war"/>

</Application>

<Application
Deployed="true"
Name="petstoreAdmin"
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/petstoreAdmin.ear>

<EJBComponent
Name="petstoreAdminEjb"
Targets="petstoreServer"
URI="petstoreadminEjb.jar"/>

<WebAppComponent
Name="petstoreadmin"
Targets="petstoreServer"
URI="petstoreadmin.war"/>

</Application>
```

```
<Server
JavaCompiler="D:\WLS 6.1\jdk131/bin/javac"
ListenPort="7001"
Name="petstoreServer"
RootDirectory="D:\WLS 6.1\wlserver6.1"
ThreadPoolSize="15"
TransactionLogFilePrefix="config/petstore/logs/"
IIOPEnabled="false">

<WebServer
DefaultWebApp="tour"
LogFileName="./config/petstore/logs/access.log"
LoggingEnabled="true"
Name="petstoreServer"/>

<SSL
CertificateCacheSize="3"
Enabled="true"
ListenPort="7002"
ServerCertificateChainFileName="./config/petstore/ca.pem"
ServerCertificateFileName="./config/petstore/democert.pem"
ServerKeyFileName="./config/petstore/demokey.pem"
TrustedCAFileName="./config/petstore/ca.pem"

Ciphersuites="SSL_RSA_EXPORT_WITH_RC4_40_MD5,SSL_RSA_WITH_DES_C
BC_SHA,
SSL_RSA_EXPORT_WITH_DES_40_CBC_SHA,SSL_NULL_WITH_NULL_NULL"/>

<Log FileName="./config/petstore/logs/weblogic.log"/>

</Server>

<Log FileName="./config/petstore/logs/wl-domain.log"/>

<JDBCConnectionPool
CapacityIncrement="1"
DriverName="COM.cloudscape.core.JDBCDriver"
InitialCapacity="1"
MaxCapacity="1"
Name="petstorePool"
Properties="user=none;password=none;server=none"
Targets="petstoreServer"
URL="jdbc:cloudscape:petStore"/>

<FileRealm
Name="myFileRealm"/>

<Security
Realm="myRealm"/>

<Realm
FileRealm="myFileRealm"
Name="myRealm"/>
```

```
<MailSession
Name="mailSession"
Targets="petstoreServer"
JNDIName="mail.Session"

Properties="mail.from=orders@javapetstoredemo.com;mail.host=san
-francisco.beasys.com"/>

<StartupClass
Arguments="port=7001"
ClassName="com.bea.estore.startup.StartBrowser"
FailureIsFatal="false"
Name="StartBrowser"
Targets="petstoreServer"

Notes="On Windows, this class automatically starts a browser
after the server has finished booting."/>

</Domain>
```

above-listed config.xml file used for WebLogic Server 7.0

1. Change the path to point to your 7.0 `petstore.ear` file.

```
<Domain ConfigurationVersion="7.0.0.0"
Name="petstore"
Path="D:\700sp0\weblogic700\samples\server\config\petstore\appl
ications\
petstore.ear" StagedTargets="" TwoPhase="false">

<JDBCTxDataSource
JNDIName="jdbc.EstoreDB"
Name="EstoreDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<JDBCTxDataSource
JNDIName="jdbc.InventoryDB"
Name="InventoryDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<JDBCTxDataSource
JNDIName="jdbc.SignOnDB"
Name="SignOnDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<Application
Deployed="true"
Name="tour"
```

```
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/tour.war">
<WebAppComponent
Name="tour"
Targets="petstoreServer"
URI="tour.war"/>
</Application>

<Application
Deployed="true"
Name="petstore"
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/petstore.ear">
<EJBComponent
Name="customerEjb"
Targets="petstoreServer"
URI="customerEjb.jar"/>

<EJBComponent
Name="inventoryEjb"
Targets="petstoreServer"
URI="inventoryEjb.jar"/>

<EJBComponent
Name="mailerEjb"
Targets="petstoreServer"
URI="mailerEjb.jar"/>
<EJBComponent
Name="personalizationEjb"
Targets="petstoreServer"
URI="personalizationEjb.jar"/>

<EJBComponent
Name="signonEjb"
Targets="petstoreServer"
URI="signonEjb.jar"/>

<EJBComponent
Name="shoppingcartEjb"
Targets="petstoreServer"
URI="shoppingcartEjb.jar"/>

<EJBComponent
Name="petstoreEjb"
Targets="petstoreServer"
URI="petstoreEjb.jar"/>

<WebAppComponent
Name="petstore"
Targets="petstoreServer"
URI="petstore.war"/>
```

```
</Application>
```

2. Change the path to point to your `petstoreAdmin.ear` file.

```
<Application
Deployed="true"
Name="petstoreAdmin"
Path="D:\700sp0\weblogic700\samples\server\config\petstore\appl
ications\
petstoreAdmin.ear" StagedTargets="" TwoPhase="false">
<EJBComponent
Name="petstoreAdminEjb"
Targets="petstoreServer"
URI="petstoreadminEjb.jar"/>

<WebAppComponent
Name="petstoreadmin"
Targets="petstoreServer"
URI="petstoreadmin.war"/>

</Application>
```

3. Change the path so that it points to the location of your WebLogic Server 7.0 Java compiler.

```
<Server
JavaCompiler="D:\700sp0\jdk131_02/bin/javac"
ListenPort="7001"
Name="petstoreServer"
RootDirectory="D:\700sp0"
ThreadPoolSize="15"
TransactionLogFilePrefix="config/petstore/logs/"
IIOPEnabled="false">

<WebServer
DefaultWebApp="tour"
LogFileName="./config/petstore/logs/access.log"
LoggingEnabled="true"
Name="petstoreServer"/>

<SSL
CertificateCacheSize="3"
Enabled="true"
ListenPort="7002"
ServerCertificateChainFileName="./config/petstore/ca.pem"
ServerCertificateFileName="./config/petstore/democert.pem"
ServerKeyFileName="./config/petstore/demokey.pem"
TrustedCAFileName="./config/petstore/ca.pem"
```

```
Ciphersuites="SSL_RSA_EXPORT_WITH_RC4_40_MD5,SSL_RSA_WITH_DES_C
BC_SHA,
SSL_RSA_EXPORT_WITH_DES_40_CBC_SHA,SSL_NULL_WITH_NULL_NULL"/>

<Log FileName="./config/petstore/logs/weblogic.log"/>

</Server>

<Log FileName="./config/petstore/logs/wl-domain.log"/>

<JDBCConnectionPool
CapacityIncrement="1"
DriverName="COM.cloudscape.core.JDBCDriver"
InitialCapacity="1"
MaxCapacity="1"
Name="petstorePool"
Properties="user=none;password=none;server=none"
Targets="petstoreServer"
URL="jdbc:cloudscape:petStore"/>

<FileRealm
Name="myFileRealm"/>

<Security
Realm="myRealm"/>

<Realm
FileRealm="myFileRealm"
Name="myRealm"/>

<MailSession
Name="mailSession"
Targets="petstoreServer"
JNDIName="mail.Session"

Properties="mail.from=orders@javapetstoredemo.com;mail.host=san
-francisco.beasys.com"/>

<StartupClass
Arguments="port=7001"
ClassName="com.bea.estore.startup.StartBrowser"
FailureIsFatal="false"
Name="StartBrowser"
Targets="petstoreServer"

Notes="On Windows, this class automatically starts a browser
after the server has finished booting."/>

</Domain>
```

Start the Pet Store Application on WebLogic Server 7.0

To start the Pet Store application on WebLogic Server 7.0:

1. Open a new web browser window.
2. Go to http://localhost:7001/estore/index.html
3. Click **Enter the Store**.

Upgrading the WebLogic 6.0 Service Pack 2 Examples Server to WebLogic Server 7.0

It is not necessary to upgrade the WebLogic 6.0 Examples Server to WebLogic Server 7.0; this section just provides the steps to do so as an example of how to upgrade a server from 6.0 to 7.0. To upgrade the WebLogic 6.0 Examples Server domain configuration for use on WebLogic Server 7.0:

- Install WebLogic Server 7.0
- Set Up the WebLogic Server 7.0 Environment with Your 6.0 Service Pack 2 Domain Configuration
- Start the Examples Server on WebLogic Server 7.0

Install WebLogic Server 7.0

Install WebLogic Server 7.0. See the *Installation Guide*.

Note: Installing the new version in the exact location of the old version is explicitly prohibited by the installer.

Set Up the WebLogic Server 7.0 Environment with Your 6.0 Service Pack 2 Domain Configuration

It is important to keep the examples domain directory within the config directory when you copy your 6.0 `WL_HOME/config/examples` directory to your 7.0 directory. For example, you could use the following directory structure:

```
c:\my_application_domains\config\examples
```

To upgrade the WebLogic 6.0 Examples Server to WebLogic Server 7.0, you need to edit the following two scripts:

```
setExamplesEnv.cmd
```

```
startExamplesServer.cmd
```

These scripts are provided in both DOS and Unix versions, `.cmd` and `.sh`, respectively.

The `setExamplesEnv` script at `SAMPLES_HOME\server\config\examples` sets certain environment variables in your development shell, the command window from which you build and run the examples.

`setExamplesEnv` sets these variables:

- `CLASSPATH`

  contains all of the classes needed to build and run the examples

- `CLIENT_CLASSES`

  points to the directory that stores client classes

- `SERVER_CLASSES`

  points to the directory that stores server-side classes

- `EX_WEBAPP_CLASSES`

  points to the directory that stores classes used by the Examples Web Application

- `PATH`

  contains your system path, appended with the JDK and WebLogic Server `bin` directories

The `java` and `javac` commands use the `CLASSPATH` variable to locate the Java classes that are required to compile source files and run examples. `CLASSPATH` must contain the appropriate classes for compiling and running the example.

To upgrade the Examples Server from WebLogic Server 6.0 to WebLogic Server 7.0, you must edit the WebLogic Server 7.0 `setExamplesEnv.cmd` script so that you can access your WebLogic Server 6.0 classes, your WebLogic Server 7.0 classes, and any native libraries that you want to use in WebLogic Server 7.0.

The steps used to boot WebLogic Server 7.0 are no different from those used to boot WebLogic Server 6.0.

This section contains:

- An example of the setExamplesEnv.cmd script used to boot a WebLogic 6.0 Service Pack 2 Examples Server.
- An example of the above-listed setExamplesEnv.cmd script modified to boot a WebLogic 7.0 Examples Server which includes explanations for how to change this script to upgrade the Examples Server toWebLogic Server 7.0.
- An example of the startExamplesServer.cmd script used to boot a WebLogic 6.0 Service Pack 2 Examples Server.
- An example of the above-listed startExamplesServer.cmd script modified to boot a WebLogic 7.0 Examples Server which includes explanations for how to change this script to upgrade the Examples Server toWebLogic Server 7.0.

setExamplesEnv.cmd script used to boot a WebLogic 6.0 Service Pack 2 Examples Server

```
@echo on
@rem This script should be used to set up your environment for
@rem compiling and running the examples included with WebLogic
@rem Server. It contains the following variables:
@rem
@rem WL_HOME    - This must point to the root directory of your
WebLogic
@rem             installation.
@rem JAVA_HOME - Determines the version of Java used to compile
@rem           and run examples. This variable must point to the
@rem          root directory of a complete JDK installation. See
@rem             the WebLogic platform support page
@rem            (http://e-docs.bea.com/wls/platforms/index.html)
@rem             for an up-to-date list of supported JVMs on
@rem  Windows NT.
@rem
@rem When setting these variables below, please use short file
names(8.3).
```

```
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server (/install/index.html in your local documentation set
or on the
@rem Internet at
@rem http://e-docs.bea.com/wls/docs60/install/index.html).

@rem Set user-defined variables.
set WL_HOME=D:\WLS_6.0\wlserver6.0
set JAVA_HOME=D:\WLS_6.0\jdk130

@if exist %WL_HOME%\lib\weblogic.jar goto checkJava
@echo.
@echo The WebLogic Server wasn't found in directory %WL_HOME%.
@echo Please edit the setExamplesEnv.cmd script so that the
@echo WL_HOME
@echo variable points to the WebLogic Server installation
@echo directory.
@echo Your environment has not been set.
@goto finish

:checkJava

@if exist %JAVA_HOME%\bin\java.exe goto setEnv
@echo.
@echo The JDK wasn't found in directory %JAVA_HOME%.
@echo Please edit the setEnv.cmd script so that the JAVA_HOME
@echo variable points to the location of your JDK.
@echo Your environment has not been set.
@goto finish

:setEnv
set APPLICATIONS=%WL_HOME%\config\examples\applications
set CLIENT_CLASSES=%WL_HOME%\config\examples\clientclasses
set SERVER_CLASSES=%WL_HOME%\config\examples\serverclasses
set
EX_WEBAPP_CLASSES=%WL_HOME%\config\examples\applications\exampl
esWebApp\WEB-INF\classes

set
CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\lib\weblogic_sp.
```

```
jar;%WL_HOME%\lib\weblogic.jar;%WL_HOME%\lib\xmlx.jar;%WL_HOME%
\samples\eval\cloudscape\lib\cloudscape.jar;%CLIENT_CLASSES%;
%SERVER_CLASSES%;%EX_WEBAPP_CLASSES%;D:\WLS 6.0

set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%
@echo.
@echo Your environment has been set.

:finish
```

above-listed setExamplesEnv.cmd script modified to boot a WebLogic 7.0 Examples Server

```
@echo on
@rem This script should be used to set up your environment for
@rem compiling and running the examples included with WebLogic
@rem Server. It contains the following variables:
@rem
@rem WL_HOME   - This must point to the root directory of your
WebLogic
@rem             installation.
@rem JAVA_HOME - Determines the version of Java used to compile
@rem           and run examples. This variable must point to the
@rem          root directory of a complete JDK installation. See
@rem             the WebLogic platform support page
@rem           (http://e-docs.bea.com/wls/platforms/index.html)
@rem             for an up-to-date list of supported JVMs on
Windows NT.
@rem
@rem When setting these variables below, please use short file
@rem names(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server (/install/index.html in your local documentation set
or on the
@rem Internet at
@rem http://e-docs.bea.com/wls/docs60/install/index.html).
```

```
@rem Set user-defined variables.
@rem changed: set WL_HOME=C:\bea60sp2\wlserver6.0
```

1. Set the `WL60_HOME` variable so that you can access your WebLogic Server 6.0 classes.

   ```
   set WL60_HOME=C:\bea60sp2\wlserver6.0
   ```

2. Set the `WL_HOME` variable so that you can access your WebLogic Server 7.0 classes.

   ```
   set WL_HOME=C:\bea700\weblogic700
   ```

   ```
   @rem changed: set JAVA_HOME=C:\bea60sp2\jdk130
   ```

3. Point to the JDK on WebLogic Server 7.0.

   ```
   set JAVA_HOME=c:\bea700\jdk131
   ```

4. Point to application archives created when you build the examples.

   ```
   set APPLICATIONS=%WL60_HOME%\config\examples\applications
   ```

5. Point to the directory used to store client classes.

   ```
   set CLIENT_CLASSES=%WL60_HOME%\config\examples\clientclasses
   ```

6. Point to the directory used to store server-side classes.

   ```
   set SERVER_CLASSES=%WL60_HOME%\config\examples\serverclasses
   ```

7. Point to the directory used to store classes used by the Examples Web Application.

   ```
   set
   EX_WEBAPP_CLASSES=%WL60_HOME%\config\examples\applications\exam
   plesWebApp\WEB-INF\classes
   ```

8. Point to the WebLogic Server 6.0 and WebLogic Server 7.0 classes.

   ```
   set
   CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\lib\weblogic_sp.
   jar;%WL_HOME%\lib\weblogic.jar;%WL_HOME%\lib\xmlx.jar;%WL60_
   HOME%\samples\eval\cloudscape\lib\cloudscape.jar;%CLIENT_
   CLASSES%;%SERVER_CLASSES%;%EX_WEBAPP_CLASSES%
   ```

9. Point to your `%WL_HOME%` 7.0 home.

   ```
   set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%


   @echo.
   @echo Your environment has been set.
   ```

startExamplesServer.cmd script used to boot a WebLogic 6.0 Service Pack 2 Examples Server

```
@echo off

@rem This script can be used to start WebLogic Server for the
@rem purpose
@rem of running the examples. This script ensures that the server
is started
@rem using the config.xml file found in this directory and that
the CLASSPATH
@rem is set appropriately. This script contains the following
@rem variable:
@rem
@rem JAVA_HOME     - Determines the version of Java used to start
@rem                 WebLogic Server. This variable must point
to the
@rem                 root directory of a JDK installation and
will be set
@rem                 for you by the WebLogic Server installer.
Note that
@rem                 this script uses the hotspot VM to run
WebLogic Server.
@rem                If you choose to use a JDK other than the one
@rem                 included in the disribution, make sure that
the JDK
@rem                 includes the hotspot VM. See the WebLogic
platform
@rem                 support page
@rem (http://e-docs.bea.com/wls/platforms/index.html)
@rem                 for an up-to-date list of supported JVMs
@rem on Windows NT.
@rem
@rem When setting the variable below, please use short file names
(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
@rem libraries
@rem required for jDriver for Oracle have been installed in the
@rem proper
@rem location and that your system PATH variable has been set
@rem appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
```

```
@rem Server
@rem (http://e-docs.bea.com/wls/docs60/install/index.html).

SETLOCAL

cd ..\..

@rem Set user-defined variables.
set JAVA_HOME=D:\WLS 6.0\jdk130

if exist %JAVA_HOME%\lib\nul goto runWebLogic
echo.
echo The JRE wasn't found in directory %JAVA_HOME%.
echo Please edit the startExamplesServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your Java
installation.
goto finish

:runWebLogic
echo on
set PATH=.\bin;%PATH%

set
CLASSPATH=.;.\lib\weblogic_sp.jar;.\lib\weblogic.jar;.\samples
\eval\cloudscape\lib\cloudscape.jar;.\config\examples\server
classes

%JAVA_HOME%\bin\java -hotspot -ms64m -mx64m -classpath
%CLASSPATH% -Dweblogic.Domain=examples -Dweblogic.
Name=examplesServer -Dbea.home=D:\WLS 6.0
-Dcloudscape.system.home=./samples/eval/cloudscape/
data -Djava.security.policy==D:\WLS
6.0\wlserver6.0/lib/weblogic.policy weblogic.Server

goto finish

:finish
cd config\examples
ENDLOCAL
```

above-listed startExamplesServer.cmd script modified to boot a WebLogic 7.0 Examples Server

```
@echo off

SETLOCAL
```

```
@rem Set user-defined variables.
@rem original:set JAVA_HOME=C:\bea60sp2\jdk130
```

1. Set your JAVA_HOME to your new JDK in WebLogic 7.0.

   ```
   set JAVA_HOME=C:\bea700\jdk131
   ```

   ```
   @rem added:
   ```

2. Set `WL60_HOME` in order to be able to access your WebLogic Server 6.0 classes.

   ```
   set WL60_HOME=c:\bea60sp2\wlserver6.0
   ```

3. Set this in order to be able to access your WebLogic Server 7.0 classes.

   ```
   set WL_HOME=c:\bea700\weblogic700


   :checkJRE
   if exist %JAVA_HOME%\lib\nul goto runWebLogic
   echo.
   echo The JRE wasn't found in directory %JAVA_HOME%.
   echo Please edit the startExamplesServer.cmd script so that the
   JAVA_HOME
   echo variable points to the root directory of your Java
   installation.
   goto finish

   :runWebLogic
   echo on
   @rem original: set PATH=.\bin;%PATH%
   ```

4. Set the `PATH` to look in your %WL_HOME% 7.0 home. If this is not set, the server won't boot.

   ```
   set PATH=%WL_HOME%\bin;%PATH%

   @rem original: set
   @rem CLASSPATH=.;.\lib\weblogic_sp.jar;
   @rem .\lib\weblogic.jar;.\samples\eval\cloudscape\
   @rem lib\cloudscape.
   @rem jar;.\config\examples\serverclasses
   ```

5. Set the `CLASSPATH` to point to old classes and to new classes that you need.

   ```
   set
   CLASSPATH=%WL_HOME%\lib\weblogic_sp.jar;%WL_HOME%\lib\weblogic.
   jar;
   %WL60_HOME%\samples\eval\cloudscape\lib\
   cloudscape.jar;%WL60_HOME%\config\examples\serverclasses

   echo CLASSPATH=%CLASSPATH%
   ```

```
@rem original: %JAVA_HOME%\bin\java -hotspot -ms64m -mx64m
@rem -classpath %CLASSPATH% -Dweblogic.Domain=examples
@rem -Dweblogic.Name=examplesServer -Dbea.home=C:\bea60sp2
@rem -Dcloudscape.system.home
@rem =./samples/eval/cloudscape/data
@rem -Djava.security.policy==C:\bea60sp2\wlserver6.0
@rem /lib/weblogic.policy weblogic.Server

%JAVA_HOME%\bin\java -hotspot -ms64m -mx64m -classpath
%CLASSPATH% -Dweblogic.Name=examplesServer
-Dweblogic.ProductionModeEnabled=true -Dbea.home=C:\bea700
-Dcloudscape.system.home=%WL60_HOME%/
samples/eval/cloudscape/data
-Djava.security.policy==%WL60_HOME%/lib/weblogic.policy
weblogic.Server

goto finish


:finish
ENDLOCAL
```

Start the Examples Server on WebLogic Server 7.0

To start the Examples Server on WebLogic Server 7.0:

On Windows:

1. On the taskbar, click **Start**.
2. Choose **Programs**.
3. Choose **BEA WebLogic E-Business Platform**.
4. Choose **WebLogic Server 7.0**.
5. Choose **Examples**.
6. Choose **Start Examples Server**.
7. Watch for the **Out-of-the-Box Examples Index Page**.

Or ...

1. In Windows Explorer, go to the SAMPLES_HOME\server\config\examples directory.

2. Double-click the startExamplesServer icon.
3. Watch for the **Out-of-the-Box Examples Index Page**.

On UNIX Bourne shell:

1. cd `$SAMPLES_HOME/server/config/examples`
2. sh `startExamplesServer.sh`

Upgrading the WebLogic 6.1 Service Pack 2 Examples Server to WebLogic Server 7.0

It is not necessary to upgrade the WebLogic 6.1 Examples Server to WebLogic Server 7.0; this section just provides the steps to do so as an example of how to upgrade a server from 6.1 to 7.0. To upgrade the WebLogic 6.1 Examples Server domain configuration for use on WebLogic Server 7.0:

- Install WebLogic Server 7.0
- Set Up the WebLogic Server 7.0 Environment with Your 6.1 Service Pack 2 Domain Configuration
- Start the Examples Server on WebLogic Server 7.0

Install WebLogic Server 7.0

Install WebLogic Server 7.0. See the *Installation Guide*.

Note: Installing the new version in the exact location of the old version is explicitly prohibited by the installer.

Set Up the WebLogic Server 7.0 Environment with Your 6.1 Service Pack 2 Domain Configuration

It is important to keep the examples domain directory within the config directory when you copy your 6.1 `WL_HOME/config/examples` directory to a new location. For example, you could use the following directory structure:

```
c:\my_application_domains\config\examples
```

To upgrade the WebLogic 6.1 Examples Server to WebLogic Server 7.0, you need to edit the following two scripts:

```
setExamplesEnv.cmd
```

```
startExamplesServer.cmd
```

These scripts are provided in both DOS and Unix versions, `.cmd` and `.sh`, respectively.

The `setExamplesEnv` script at `SAMPLES_HOME\server\config\examples` sets certain environment variables in your development shell, the command window from which you build and run the examples.

`setExamplesEnv` sets these variables:

- `CLASSPATH`

  contains all of the classes needed to build and run the examples
- `CLIENT_CLASSES`

  points to the directory that stores client classes
- `SERVER_CLASSES`

  points to the directory that stores server-side classes
- `EX_WEBAPP_CLASSES`

  points to the directory that stores classes used by the Examples Web Application
- `PATH`

  contains your system path, appended with the JDK and WebLogic Server `bin` directories

The `java` and `javac` commands use the `CLASSPATH` variable to locate the Java classes that are required to compile source files and run examples. `CLASSPATH` must contain the appropriate classes for compiling and running the example.

To upgrade the Examples Server from WebLogic Server 6.1 to WebLogic Server 7.0, you must edit the WebLogic Server 7.0 `setExamplesEnv.cmd` script so that you can access your WebLogic Server 6.1 classes, your WebLogic Server 7.0 classes, and any native libraries that you want to use in WebLogic Server 7.0.

The steps used to boot WebLogic Server 7.0 are no different from those used to boot WebLogic Server 6.1.

This section contains:

- An example of the setExamplesEnv.cmd script used to boot a WebLogic 6.1 Service Pack 2 Examples Server.
- An example of the above-listed setExamplesEnv.cmd script modified to boot WebLogic Server 7.0 which includes explanations for how to change this script to upgrade the Examples Server toWebLogic Server 7.0.
- An example of the startExamplesServer.cmd script used to boot a WebLogic 6.1 Service Pack 2 Examples Server.
- An example of the above-listed startExamplesServer.cmd script modified to boot WebLogic Server 7.0 which includes explanations for how to change this script to upgrade the Examples Server toWebLogic Server 7.0.

setExamplesEnv.cmd script used to boot a WebLogic 6.1 Service Pack 2 Examples Server

```
@echo on
@rem This script should be used to set up your environment for
@rem compiling and running the examples included with WebLogic
@rem Server. It contains the following variables:
@rem
@rem WL_HOME   - This must point to the root directory of your
WebLogic
@rem             installation.
@rem JAVA_HOME - Determines the version of Java used to compile
@rem            and run examples. This variable must point to the
@rem           root directory of a complete JDK installation. See
@rem             the WebLogic platform support page
@rem            (http://e-docs.bea.com/wls/platforms/index.html)
@rem             for an up-to-date list of supported JVMs on
Windows NT.
@rem When setting these variables below, please use short file
names(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
```

```
@rem names with spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
@rem proper location and that your system PATH variable
@rem has been set appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server (/install/index.html in your local documentation set
or on the Internet at
@rem http://e-docs.bea.com/wls/docs61/install/index.html).

@rem Set user-defined variables.
set WL_HOME=D:\WLS 6.1\wlserver6.1
set JAVA_HOME=D:\WLS 6.1\jdk131

@dir %WL_HOME%\lib > nul
if errorlevel 0 goto checkJava
@echo.
@echo The WebLogic Server wasn't found in directory %WL_HOME%.
@echo Please edit the setExamplesEnv.cmd script so that the
@echo WL_HOME variable points to the WebLogic Server installation
@echo directory.
@echo Your environment has not been set.
@goto finish

:checkJava
@dir %JAVA_HOME%\jre\bin\java.exe > nul
if errorlevel 0 goto setEnv
@echo.
@echo The JDK wasn't found in directory %JAVA_HOME%.
@echo Please edit the setEnv.cmd script so that the JAVA_HOME
@echo variable points to the location of your JDK.
@echo Your environment has not been set.
@goto finish

:setEnv
set APPLICATIONS=%WL_HOME%\config\examples\applications
set CLIENT_CLASSES=%WL_HOME%\config\examples\clientclasses
set SERVER_CLASSES=%WL_HOME%\config\examples\serverclasses
set EX_WEBAPP_CLASSES=%WL_HOME%\config\examples\applications\
examplesWebApp\WEB-INF\classes

set CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\lib\
weblogic_sp.jar;%WL_HOME%\lib\weblogic.jar;%WL_HOME%\lib\
xmlx.jar;%WL_HOME%\samples\eval\cloudscape\lib\cloudscape.jar;
%CLIENT_CLASSES%;%SERVER_CLASSES%;%EX_WEBAPP_CLASSES%;
D:\WLS 6.1
```

```
set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%
@echo.
@echo Your environment has been set.

:finish
```

above-listed setExamplesEnv.cmd script modified to boot WebLogic Server 7.0

```
@echo on
@rem This script should be used to set up your environment for
@rem compiling and running the examples included with WebLogic
@rem Server. It contains the following variables:
@rem
@rem WL_HOME   - This must point to the root directory of your
WebLogic
@rem             installation.
@rem JAVA_HOME - Determines the version of Java used to compile
@rem           and run examples. This variable must point to the
@rem          root directory of a complete JDK installation. See
@rem             the WebLogic platform support page
@rem           (http://e-docs.bea.com/wls/platforms/index.html)
@rem             for an up-to-date list of supported JVMs on
Windows NT.
@rem
@rem When setting these variables below, please use short file
@rem names(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server (/install/index.html in your local documentation set
or on the
@rem Internet at
http://e-docs.bea.com/wls/docs61/install/index.html).

@rem Set user-defined variables.
@rem changed: set WL_HOME=C:\bea61sp2\wlserver6.1
```

1. Set the `WL61_HOME` variable so that you can access your WebLogic Server 6.1 classes.

```
set WL61_HOME=C:\bea61sp2\wlserver6.1
```

2. Set the `WL_HOME` variable so that you can access your WebLogic Server 7.0 classes.

   ```
   set WL_HOME=C:\bea700\weblogic700
   ```

3. Point to the JDK on WebLogic Server 7.0.

   ```
   @rem changed: set JAVA_HOME=C:\bea61sp2\jdk130
   set JAVA_HOME=c:\bea700\jdk131
   ```

4. Point to application archives created when you build the examples.

   ```
   set APPLICATIONS=%WL61_HOME%\config\examples\applications
   ```

5. Point to the directory used to store client classes.

   ```
   set CLIENT_CLASSES=%WL61_HOME%\config\examples\clientclasses
   ```

6. Point to the directory used to store server-side classes.

   ```
   set SERVER_CLASSES=%WL61_HOME%\config\examples\serverclasses
   ```

7. Point to the directory used to store classes used by the Examples Web Application.

   ```
   set
   EX_WEBAPP_CLASSES=%WL61_HOME%\config\examples\applications\exam
   plesWebApp\WEB-INF\classes
   ```

8. Point to the WebLogic Server 6.1 and WebLogic Server 7.0 classes.

   ```
   set
   CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\lib\weblogic_sp.
   jar;%WL_HOME%\lib\weblogic.jar;%WL_HOME%\lib\xmlx.jar;%WL61_
   HOME%\samples\eval\cloudscape\lib\cloudscape.jar;%CLIENT_
   CLASSES%;%SERVER_CLASSES%;%EX_WEBAPP_CLASSES%
   ```

9. Point to your `%WL_HOME%` 7.0 home.

   ```
   set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%


   @echo.
   @echo Your environment has been set.
   ```

startExamplesServer.cmd script used to boot a WebLogic 6.1 Service Pack 2 Examples Server

```
@echo off
```

```
@rem This script can be used to start WebLogic Server for the
purpose
@rem of running the examples. This script ensures that the server
is started
@rem using the config.xml file found in this directory and that
the CLASSPATH
@rem is set appropriately. This script contains the following
variable:
@rem
@rem JAVA_HOME     - Determines the version of Java used to start
@rem                 WebLogic Server. This variable must point
to the
@rem                 root directory of a JDK installation and
will be set
@rem                 for you by the WebLogic Server installer.
Note that
@rem                 this script uses the hotspot VM to run
WebLogic Server.
@rem               If you choose to use a JDK other than the one
@rem               included in the disribution, make sure that
the JDK
@rem                 includes the hotspot VM. See the WebLogic
platform
@rem                 support page
(http://e-docs.bea.com/wls/platforms/index.html)
@rem                 for an up-to-date list of supported JVMs
on Windows NT.
@rem

@rem When setting the variable below, please use short file names
(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server
(http://e-docs.bea.com/wls/docs61/install/index.html).

SETLOCAL

cd ..\..
```

```
@rem Set user-defined variables.
set JAVA_HOME=D:\WLS 6.1\jdk131

@rem Check that script is being run from the appropriate
directory
if not exist lib\weblogic.jar goto wrongplace
goto checkJDK

:wrongplace

echo startExamplesServer.cmd must be run from the
config\examples directory. 1>&2
goto finish

:checkJDK
if exist %JAVA_HOME%/bin/javac.exe goto runWebLogic
echo.
echo Javac wasn't found in directory %JAVA_HOME%/bin.
echo Please edit the startExamplesServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your JDK
installation.
goto finish

:runWebLogic
echo on
set PATH=.\bin;%PATH%

set
CLASSPATH=.;.\lib\weblogic_sp.jar;.\lib\weblogic.jar;.\samples
eval\cloudscape\lib\cloudscape.jar;.\config\examples\serverclas
ses
echo off

echo.
echo ***************************************************
echo *  To start WebLogic Server, use the password     *
echo *  assigned to the system user.  The system       *
echo *  username and password must also be used to     *
echo *  access the WebLogic Server console from a web  *
echo *  browser.                                       *
echo ***************************************************

@rem Set WLS_PW equal to your system password for no password
prompt server startup.
set WLS_PW=

echo on
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath
"%CLASSPATH%" -Dweblogic.Domain=examples
-Dweblogic.Name=examplesServer
-Dweblogic.management.password=%WLS_PW% -Dbea.home="D:\WLS 6.1"
```

```
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="D:\WLS
6.1\wlserver6.1/lib/weblogic.policy" weblogic.Server
goto finish

:finish
cd config\examples
ENDLOCAL
```

above-listed startExamplesServer.cmd script modified to boot WebLogic Server 7.0

```
@echo off

SETLOCAL


@rem Set user-defined variables.
@rem original:set JAVA_HOME=C:\bea61sp2\jdk130
```

1. Set your `JAVA_HOME` to your new JDK in WebLogic 7.0.

   ```
   set JAVA_HOME=C:\bea700\jdk131

   @rem added:
   ```

2. Set this in order to be able to access your WebLogic Server 6.1 classes.

   ```
   set WL61_HOME=c:\bea61sp2\wlserver6.1
   ```

3. Set this in order to be able to access your WebLogic Server 7.0 classes.

   ```
   set WL_HOME=c:\bea700\weblogic700
   ```

```
:checkJRE
if exist %JAVA_HOME%\lib\nul goto runWebLogic
echo.
echo The JRE wasn't found in directory %JAVA_HOME%.
echo Please edit the startExamplesServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your Java
installation.
goto finish
```

```
:runWebLogic
echo on
@rem original: set PATH=.\bin;%PATH%
```

4. Set the `PATH` to look in your `%WL_HOME%` 7.0 home. If this is not set, the server won't boot.

```
set PATH=%WL_HOME%\bin;%PATH%

@rem original: set
@rem CLASSPATH=.;.\lib\weblogic_sp.jar;
@rem .\lib\weblogic.jar;.\samples\eval\cloudscape\
@rem lib\cloudscape.
@rem jar;.\config\examples\serverclasses


set
CLASSPATH=%WL_HOME%\lib\weblogic_sp.jar;%WL_HOME%\lib\weblogic.
jar;
%WL61_HOME%\samples\eval\cloudscape\lib\cloudscape.jar;
%WL61_HOME%\config\examples\serverclasses
```

5. Set the `CLASSPATH` to point to old classes and to new classes that you need.

```
echo CLASSPATH=%CLASSPATH%

@rem original: %JAVA_HOME%\bin\java -hotspot -ms64m -mx64m
@rem -classpath %CLASSPATH% -Dweblogic.Domain=examples
@rem -Dweblogic.Name=examplesServer -Dbea.home=C:\bea61sp2
@rem -Dcloudscape.system.home
@rem =./samples/eval/cloudscape/data
@rem -Djava.security.policy==C:\bea61sp2\wlserver6.1
@rem /lib/weblogic.policy weblogic.Server

%JAVA_HOME%\bin\java -hotspot -ms64m -mx64m -classpath
%CLASSPATH% -Dweblogic.Name=examplesServer
-Dweblogic.ProductionModeEnabled=true -Dbea.home=C:\bea700
-Dcloudscape.system.home=%WL61_HOME%/
samples/eval/cloudscape/data
-Djava.security.policy==%WL61_HOME%/lib/weblogic.policy
weblogic.Server

goto finish


:finish
ENDLOCAL
```

Start the Examples Server on WebLogic Server 7.0

To start the Examples Server on WebLogic Server 7.0:

On Windows:

1. On the taskbar, click **Start**.
2. Choose **Programs**.
3. Choose **BEA WebLogic E-Business Platform**.
4. Choose **WebLogic Server 7.0**.
5. Choose **Examples**.
6. Choose **Start Examples Server**.
7. Watch for the **Out-of-the-Box Examples Index Page**.

Or ...

1. In Windows Explorer, go to the SAMPLES_HOME\server\config\examples directory.
2. Double-click the startExamplesServer icon.
3. Watch for the **Out-of-the-Box Examples Index Page**.

On UNIX Bourne shell:

1. cd `$SAMPLES_HOME/server/config/examples`
2. sh `startExamplesServer.sh`