



BEA WebLogic Server™

BEA WebLogic Server 8.1 Upgrade Guide

Copyright

Copyright © 2004 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Table of Contents

About This Document

1. Common Upgrade Issues

- Classpath, System Path, and Environment Variables 1-2
 - WebLogic Server 8.1 Dependencies on JDK 1.4.1 1-2
- Data Sources 1-2
- Deployment 1-3
- JRockit 1-3
- JSP Parsing 1-3
- Security 1-4
- Server Configuration 1-4
- XML Parsing 1-4

2. Upgrading WebLogic Server 7.0 to Version 8.1

- Understanding the WebLogic Server 8.1 Directory Structure. 2-2
- Contents of a Domain Directory 2-2
- Upgrading Your WebLogic Server 7.0 Domain to Version 8.1 2-2
 - Shut Down WebLogic 7.0 Server Instances 2-3
 - Create a WebLogic Server 8.1 Domain Directory 2-3
 - Copy Contents of Your 7.0 Domain into the 8.1 Directory 2-4
 - Delete WebLogic Server 7.0 Settings from the 8.1 Domain 2-4
 - Modify Server Start Scripts 2-4
 - Upgrade Applications 2-5
 - Start 8.1 Administration and Managed Servers 2-5
 - Configure and Deploy Your Applications 2-6
- Modifying Start Scripts 2-6

Upgrading the Pet Store Application from WebLogic Server 7.0 to WebLogic Server 8.1.2-8	
Additional Upgrade Procedures and Information	2-10
Built-in Transformer Based on Apache Xalan XML 2.2 Transformer	2-11
Apache Xerces XML Parser	2-11
Connectors	2-12
Last-Resource Commit Optimization	2-12
Application Container Managed Security Log Messages.	2-12
Classloader and Packaging Issues.	2-13
Applications with Errors Will Not Deploy	2-13
Upgrading jCOM	2-13
JDBC	2-13
Removed Interfaces	2-13
Removed Classes	2-13
Removed Drivers	2-14
Interoperability Limitation	2-14
Statement Caches	2-14
Upgrading Your JDK	2-15
Stricter JSP Parsing in JDK 1.3.1_09	2-15
Reference JDK 1.4 When Compiling in 8.1	2-16
JMS Configuration Checking Is Stricter	2-17
Upgrading Your JVM to JRockit	2-17
Load Order Behavior for Startup Classes	2-18
Network Channels Include Network Access Point Features	2-19
Changes to Timeouts for Pool, JTS/JTA, and RMI Connections.	2-19
Prepared Statement Cache Algorithm	2-19
Default Names for Execute Queues Have Changed	2-20
Security	2-20
Troubleshooting Problems with Certificates.	2-20

Security Data Is Now Written to the config.xml File	2-21
Web Resource Is Replaced by URL Resource	2-21
Keystores Are Supported	2-21
Custom Keystore Providers Are No Longer Supported	2-22
The WebLogic Keystore Provider Is Deprecated	2-22
Upgrading to WebLogic Server 8.1 Keystores	2-22
Upgrading to the WebLogic Server 8.1 Trust Mechanism for Java Clients . . .	2-23
URIs	2-24
Web Applications	2-24
Web Services	2-25
EJB 2.0	2-25
Run DDConverter Before Using New Features	2-25
New Default Values for EJB Deployment Descriptor Elements	2-25
Inserting a New CMP Bean at Commit Time	2-26
Writable config.xml File	2-27

3. Upgrading WebLogic Server 6.x to Version 8.1

Understanding the WebLogic Server 8.1 Directory Structure.	3-2
Contents of a Domain Directory	3-2
Upgrading Your WebLogic Server 6.x Domain to Version 8.1	3-3
Shut Down WebLogic 6.x Server Instances	3-3
Create a WebLogic Server 8.1 Domain Directory	3-4
Copy Contents of Your 6.x Domain into the 8.1 Directory	3-4
Delete WebLogic Server 6.x Settings from the 8.1 Domain	3-4
Modify Server Start Scripts	3-4
Upgrade Applications	3-5
Start 8.1 Administration and Managed Servers	3-6
Configure and Deploy Your Applications	3-6

Modifying Start Scripts	3-6
Upgrading the Pet Store Application from WebLogic Server 6.1 Service Pack 4 to WebLogic Server 8.1	3-8
Repair Pet Store	3-8
Fix Erroneous <ejb-ref-name>	3-9
Add Missing <ejb-ref-name>	3-10
Fix Encoding Error	3-11
Clean fileRealm.properties	3-11
Fix JSP Parsing Errors	3-11
Rebuild Petstore	3-14
Upgrade the Pet Store Domain.	3-14
Security Realms in WebLogic Server 8.1 Versus 6.x.	3-17
Upgrading Your WebLogic Server 6.x Security Realms to 8.1	3-18
Upgrading from an LDAP V2 Security Realm.	3-19
Upgrading from a Windows NT Security Realm	3-24
Upgrading from a UNIX Security Realm	3-29
Upgrading from an RDBMS Security Realm	3-35
Upgrading from a Custom Security Realm.	3-42
Access Control Lists (ACLs)	3-43
ACLs on MBeans	3-43
ACLs That Protected WebLogic Resources	3-43
Upgrading from Compatibility Security to WebLogic Server 8.1 Security.	3-47
Guest and <Anonymous> Users	3-48
password.ini File	3-49
Upgrading SSL	3-49
Upgrading SSL Identity and Trust	3-49
New Options for Storing SSL Client Trust	3-50
Using Host Name Verification	3-51

Using a Certificate Authenticator	3-52
Upgrading Private Keys and Digital Certificates	3-52
Additional Upgrade Procedures and Information	3-53
Apache Xalan XML Transformer	3-54
Apache Xerces XML Parser	3-54
Runtime Modes	3-54
Deployment	3-55
Manual Changes to Deployment Descriptors for EJB 2.0	3-55
weblogic.management.configuration.EJBComponentMBean Changes	3-57
max-beans-in-cache Parameter	3-57
Fully Qualified Path Expressions	3-57
jCOM	3-57
JDBC	3-58
Upgrading Your JDK	3-58
Stricter JSP Parsing in JDK 1.3.1_09	3-58
WebLogic Server 8.1 Dependencies on JDK 1.4	3-59
JMS	3-59
JMX	3-60
Jolt Java Client	3-60
Upgrading Your JVM to JRockit	3-60
JSP	3-61
New Behavior of Load Order Methods for Startup Classes	3-62
6.1 Managed Servers Cannot Boot from an 8.1 Administrative Server	3-62
Default Queue Names Have Changed	3-63
MBean API Change	3-63
Update Your web.xml file with New Classes	3-63
ThreadPoolSize Is Now ThreadCount	3-64
404 Message for URIs with Extra Spaces	3-65

Web Application API Changes from 6.0	3-65
Clusters on Solaris Perform Better with Client JVM.	3-65
Change WebLogic Tuxedo Connector Configuration	3-66
Start the WebLogic Tuxedo Connector.	3-66
Convert WebLogic Tuxedo Connector XML Configuration Files	3-66
Update Inbound RMI-IIOP Applications	3-68
Authenticate Remote Users	3-70
Set WebLogic Tuxedo Connector Properties	3-71
Using the Runtime WTC ORB.	3-71
Upgrade Web Services.	3-71
Deprecated APIs and Features	3-72
Removed APIs and Features	3-73

4. Upgrading WebLogic Server 5.1 to Version 8.1

Understanding the WebLogic Server 8.1 Directory Structure	4-2
Contents of a Domain Directory	4-2
Upgrading WebLogic Server 5.1 to Version 8.1: Main Steps	4-3
Upgrading WebLogic Server License Files	4-4
Converting a WebLogicLicense.class License	4-4
Converting a WebLogicLicense.XML License	4-5
Converting the weblogic.properties File to config.xml	4-5
Classloading in WebLogic Server 8.1.	4-7
Modifying Startup Scripts.	4-8
Converting Applications to Web Applications	4-8
XML Deployment Descriptors.	4-9
WAR Files	4-9
Session Porting.	4-9
JavaServer Pages (JSPs) and Servlets	4-10

Upgrading a Simple Servlet from WebLogic Server 5.1 to WebLogic Server 8.1 . . .	4-11
Upgrading Enterprise Java Beans Applications	4-12
EJB Upgrade Considerations	4-12
Steps for Upgrading a 1.1 EJB from WebLogic Server 5.1 to WebLogic Server 8.1	4-14-15
Steps for Converting an EJB 1.1 to an EJB 2.0	4-15
Porting EJBs from Other J2EE Application Servers	4-16
Upgrading JMS	4-16
Upgrading Oracle.	4-17
Upgrading the Banking Application from WebLogic Server 5.1 to WebLogic Server 8.1 . . .	4-17
Set Up the Banking Application on WebLogic Server 5.1.	4-17
Convert the weblogic.properties File	4-18
Configure the Banking Application for WebLogic Server 8.1.	4-20
Edit the startmigration Script	4-20
Copy Banking Application Files to the Output Directory.	4-21
Deploy and Run the Banking Application	4-21
Additional Upgrade and Deployment Considerations.	4-21
Applications and Managed Servers	4-22
Reset Default Mime Type in weblogic.xml	4-23
Two-Phase Deployment Is the Default Deployment Model.	4-23
FileServlet Behavior Has Changed	4-23
Changes to Internationalization (I18N) Log Files	4-23
8.1 Classes Must Be Built Under JDK 1.4	4-24
Support for Java Transaction API (JTA)	4-24
Changes to Java Database Connectivity (JDBC)	4-24
Upgrading Your JVM to JRockit	4-25
Changes to JSPs	4-26
Error Handling.	4-26

Null Requests	4-26
JVM	4-27
Plug-ins Support SSL Communication	4-27
Default Queue Names Have Changed	4-27
Tips for Using RMI	4-27
Security	4-28
Upgrading Security Realms from WebLogic Server 5.1 to 6.1	4-28
Upgrading WebLogic Server 6.1 Security to Version 8.1	4-30
Standalone HTML and JSPs	4-30
URIs with Extra Spaces Result in a 404	4-30
Web Components	4-30
Define MIME Types for Wireless Application Protocol Applications in web.xml	4-31
XML 8.1 Parser and Transformer Are Updated	4-31
Deprecated APIs and Features	4-32
Removed APIs and Features	4-33

5. Upgrading WebLogic Server 8.1 Service Pack 2 to Version 8.1 Service Pack 3

New SDKs Bundled with WebLogic Platform 8.1	5-1
---	-----

A. Mapping weblogic.properties to 6.x config.xml Configuration Attributes

About This Document

This document provides procedures and other information you need to upgrade earlier versions of BEA WebLogic Server to WebLogic 8.1. It also provides information about moving applications from an earlier version of WebLogic Server to 8.1.

The document is organized as follows:

- [“Common Upgrade Issues” on page 1-1](#) describes the most common issues encountered during upgrades, and provides solutions to them.
- [Chapter 2, “Upgrading WebLogic Server 7.0 to Version 8.1,”](#) describes how to upgrade to WebLogic Server 8.1 from WebLogic Server 7.0.
- [Chapter 3, “Upgrading WebLogic Server 6.x to Version 8.1,”](#) describes how to upgrade to WebLogic Server 8.1 from WebLogic Server 6.x.
- [Chapter 4, “Upgrading WebLogic Server 5.1 to Version 8.1,”](#) describes how to upgrade to WebLogic Server 8.1 from WebLogic Server 5.1.
- [Appendix A, “Mapping weblogic.properties to 6.x config.xml Configuration Attributes,”](#) 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 5.1, 6.0, 6.1, and 7.0 who want to upgrade to WebLogic Server 8.1.

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.
<i>italics</i>	Emphasis and book titles.

About This Document

Convention	Usage
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. <i>Examples:</i> <pre>import java.util.Enumeration; chmod u+w * config/examples/applications .java config.xml float</pre>
<i>monospace italic text</i>	Variables in code. <i>Example:</i> <pre>String CustomerName;</pre>
UPPERCASE TEXT	Device names, environment variables, and logical operators. <i>Examples:</i> <pre>LPT1 BEA_HOME OR</pre>
{ }	A set of choices in a syntax line.
[]	Optional items in a syntax line. <i>Example:</i> <pre>java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]</pre>
	Separates mutually exclusive choices in a syntax line. <i>Example:</i> <pre>java weblogic.deploy [list deploy undeploy update] password {application} {source}</pre>

Convention	Usage
...	Indicates one of the following in a command line: <ul style="list-style-type: none">• 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.
.	
.	

About This Document

Common Upgrade Issues

The following sections briefly describe the issues most likely to complicate your upgrade to WebLogic Server 8.1, and provide solutions and links to further information:

- [“Classpath, System Path, and Environment Variables” on page 1-2](#)
- [“Data Sources” on page 1-2](#)
- [“Deployment” on page 1-3](#)
- [“JRockit” on page 1-3](#)
- [“JSP Parsing” on page 1-3](#)
- [“Security” on page 1-4](#)
- [“Server Configuration” on page 1-4](#)
- [“XML Parsing” on page 1-4](#)

For complete documentation on upgrading to WebLogic Server 8.1, see the chapter for your upgrade path:

- [“Upgrading WebLogic Server 7.0 to Version 8.1” on page 2-1](#)
- [“Upgrading WebLogic Server 6.x to Version 8.1” on page 3-1](#)
- [“Upgrading WebLogic Server 5.1 to Version 8.1” on page 4-1](#)

Classpath, System Path, and Environment Variables

Properly setting up your runtime environment is likely to be your most challenging upgrade task. WebLogic Server and your application depend on accurate classpath, system path, and other environmental variable settings. The classpath should be updated to contain WebLogic Server 8.1 paths and JAR files, such as `weblogic.jar`, while continuing to point to resources specific to your application.

The following issues involve the configuration of environment variables.

Issue: Classpath pointing to the wrong JDK or compiler.

Symptom: `ejbc` fails when it loads a JAR file.

Solution: Make sure the classpath points to the new JDK shipped with WebLogic Server 8.1, which is JDK 1.4.1. If you have configured a specific compiler in your `config.xml` file, make sure that the 1.4.1 compiler is the one specified.

WebLogic Server 8.1 Dependencies on JDK 1.4.1

JAXP, the JAAS package, and the Principal Authenticator object were all included in previous versions of `weblogic.jar`, but were left out of the WebLogic Server 8.1 `weblogic.jar` because they are included in JDK 1.4.1.

Issue: Items no longer included in `weblogic.jar` are now in JDK 1.4.1.

Symptom: Build failure with error messages about JAXP, the JAAS package, or the Principal Authenticator object.

Solution: Make sure that your environment references JDK 1.4.1 when you build your classes. The JDK reference is usually in a `setenv` script. For example:

```
set JAVA_HOME=C:\bea\weblogic81\jdk141_03
```

Data Sources

The following data source issue is very commonly encountered.

Issue: No connection with data source.

Symptom: An error message indicating a problem creating a connection, such as:

```
<May 20, 2003 2:40:14 PM PDT> <Warning> <JDBC> <BEA-001129> <Received exception while creating connection for pool "petstorePool": Database 'petStore' not found.>
```

```
<May 20, 2003 2:40:15 PM PDT> <Error> <JDBC> <BEA-001150> <Connection
Pool

"petstorePool" deployment failed with the following error: 0:Could not
create pool connection. The DBMS driver exception was: Database
'petStore' not found..><May 20, 2003 2:40:15 PM PDT> <Error> <JDBC>
<BEA-001151> <Data Source "InventoryDB" deployment failed with the
following error: DataSource(jdbc.InventoryDB) can't be created with
non-existent Pool (connection or multi) (petstorePool).>
```

Solution: Check that your domain configuration and start scripts point to the correct datasource, and that you are using the correct drivers for the datasource.

Deployment

Issue: Deployment descriptors have changed to meet new J2EE standards.

Symptom: An EJB will not load in WebLogic Server 8.1.

Solution: The load failure message from the WebLogic Server EJB container should provide detailed resolution information. If you are upgrading from version 6.x, for example, you may need to repair mismatched resource references. See “[Fix Erroneous <ejb-ref-name>”](#) on page 3-9.

JRockit

See also the [Troubleshooting](#) section of the appropriate version of the JRockit documentation.

Issue: Start scripts not using the correct Java Virtual Machine for JRockit.

JRockit does not support all the same command-line parameters that the Sun JDK does (-client, for example).

Symptom: Weblogic Server 8.1 will not start.

Solution: Verify that the command-line parameters on your start command line are all supported by JRockit.

JSP Parsing

Issue: Minor JSP errors occur under JDK 1.4.

Symptom: The application loads but you cannot browse to a desired page. Or, the build fails with a JSP error message.

Solution: Use the build or runtime error message to find and fix the errors. For an example, see [“Fix JSP Parsing Errors” on page 3-11](#), where JSP errors in Pet Store are repaired.

Security

Issue: Pre-7.0 security model not upgraded correctly.

Symptom: Server fails to start, security error message.

Solution: Set up Compatibility security. See [Upgrading WebLogic Server 6.x Security to Version 8.1](#) and [Running Compatibility Security: Main Steps](#) in *Managing WebLogic Security*.

Issue: Digital security certificate fails.

Symptom: `SSLListenThread` error, failure to connect to localhost.

Solution: Either disable the Hostname Verifier (recommended only if you are operating in development mode), or obtain a new digital certificate for the client. See the [Security FAQ](#).

Server Configuration

Issue: All servers in a domain must be on the same version of WebLogic Server.

For example, a Managed Server running WebLogic Server 6.x cannot obtain its configuration and boot using an Administration Server running WebLogic Server 8.1.

Symptom: Both an Administration server and a Managed server are running WebLogic Server 8.1. The Administration server boots, but the Managed server fails to boot properly.

Solution: Make sure that you have not copied the `running-managed-servers.xml` file from your WebLogic Server 6.x installation directory to your WebLogic Server 8.1 installation directory.

XML Parsing

Issue: The new XML parser rejects some XML documents that were accepted by the previous parser.

The built-in XML parser for WebLogic Server 8.1 is based on the Apache Xerces 1.4.4 parser, which is stricter than its predecessors.

Symptom: An EAR file’s deployment descriptors are all correct, but the EAR still cannot be loaded.

Solution: Verify that the other XML documents in your deployment comply with Version 2 SAX and DOM interfaces. Track down and repair the XML. For 6.x see [“Apache Xerces XML Parser” on page 3-54](#); for 7.1 see [“Apache Xerces XML Parser” on page 2-11](#). For an example, see [“Fix Encoding Error” on page 3-11](#), where a minor encoding error is fixed.

Common Upgrade Issues

Upgrading WebLogic Server 7.0 to Version 8.1

Upgrading WebLogic Server 7.0 to version 8.1 involves copying your domain to a new directory and changing your WebLogic Server start command scripts and environment settings.

The following sections of this document contain information about upgrading your domain from WebLogic Server 7.0 to WebLogic Server 8.1:

- [“Understanding the WebLogic Server 8.1 Directory Structure”](#) on page 2-2
- [“Contents of a Domain Directory”](#) on page 2-2
- [“Upgrading Your WebLogic Server 7.0 Domain to Version 8.1”](#) on page 2-2
- [“Modifying Start Scripts”](#) on page 2-6
- [“Upgrading the Pet Store Application from WebLogic Server 7.0 to WebLogic Server 8.1”](#) on page 2-8
- [“Additional Upgrade Procedures and Information”](#) on page 2-10

To see an example of a domain being upgraded from WebLogic Server 7.0 to WebLogic Server 8.1, see [“Upgrading the Pet Store Application from WebLogic Server 7.0 to WebLogic Server 8.1”](#) on page 2-8.

For answers to specific questions on upgrading to WebLogic Platform 8.1, see the [Upgrading](#) section of the [WebLogic Server FAQs](#).

For information on upgrading WebLogic Server license files, see [“Installing and Updating WebLogic Platform License Files”](#) in *Installing WebLogic Platform*.

For information on upgrading WebLogic Server security, see [“Security”](#) on page 2-20.

Understanding the WebLogic Server 8.1 Directory Structure

For complete information on the updated directory structure see [Understanding the WebLogic Server Directory Structure](#) in *Performing Post-Installation Tasks* in the *Installation Guide*, and [Domain Restrictions](#) in *Overview of WebLogic Server Domains*. BEA WebLogic recommends that you locate domain directories outside the WebLogic Server installation directory. Starting with WebLogic Server 7.0, domain directories can be in any location that can access the WebLogic Server installation and the JDK.

If you change the location of your domain directory, remember to update any custom tools or scripts relative to the new directory structure. Similarly, if you use a scripted tool for creating domains, change its scripts. The scriptable [Configuration Wizard](#) is the recommended tool for creating domains.

Contents of a Domain Directory

When you upgrade from WebLogic Server 7.0 to 8.1, you are upgrading a WebLogic Server domain. For a full description of WebLogic Server domains see [Overview of WebLogic Server Domains](#) in *Configuring and Managing WebLogic Server*.

Domain configuration settings are stored in the `config.xml` file of the domain directory. `config.xml` stores the name of the domain and the configuration parameter settings for each server instance, cluster, resource, and service in the domain.

The domain directory also contains the server start script files that you can use to start the Administration Server and Managed Servers in the domain.

The domain directory structure has a root directory with the same name as the domain, such as `mydomain` or `petstore`. This directory contains the following:

- The configuration file (usually `config.xml`) for the domain.
- Any scripts you use to start server instances and establish your environment.
- Optionally, a subdirectory for storing the domain's application files.

For more information about WebLogic Server domains, see [WebLogic Server Domains](#) in *Configuring and Managing WebLogic Server*.

Upgrading Your WebLogic Server 7.0 Domain to Version 8.1

To upgrade a domain with multiple server instances, upgrade as you would a single-server domain. In a nutshell, you upgrade a domain by installing WebLogic Server 8.1 alongside your

7.0 installation, copying the contents of your 7.0 domain into a new 8.1 domain directory, and changing the domain scripts and environment settings to point to the new 8.1 domain and server instances.

If the domain includes a cluster and you want to take advantage of new clustering features in WebLogic Server 8.1, refer to [Setting Up WebLogic Clusters](#) in *Using WebLogic Server Clusters* for WebLogic Server 8.1 cluster configuration guidelines.

As a prerequisite to performing the upgrade procedures in this section, install WebLogic Server 8.1 on all the machines that contain 7.0 server instances whose domain you will upgrade. See [Installing WebLogic Platform](#) for installation instructions for WebLogic Server 8.1.

- [“Shut Down WebLogic 7.0 Server Instances” on page 2-3](#)
- [“Create a WebLogic Server 8.1 Domain Directory” on page 2-3](#)
- [“Copy Contents of Your 7.0 Domain into the 8.1 Directory” on page 2-4](#)
- [“Delete WebLogic Server 7.0 Settings from the 8.1 Domain” on page 2-4](#)
- [“Modify Server Start Scripts” on page 2-4](#)
- [“Upgrade Applications” on page 2-5](#)
- [“Start 8.1 Administration and Managed Servers” on page 2-5](#)
- [“Configure and Deploy Your Applications” on page 2-6](#)

Shut Down WebLogic 7.0 Server Instances

Shut down all server instances in the WebLogic Server 7.0 domain you are upgrading.

Shutting down the servers ensures that any recent changes to the domain or its applications are persisted. See [Starting and Stopping Servers: Quick Reference](#).

Create a WebLogic Server 8.1 Domain Directory

Create a new directory where your 8.1 domain will reside.

Consider that after you move the contents of your 7.0 domain directory to this new 8.1 domain directory, any references from within the 7.0 domain to resources outside the domain will need to change unless the new domain directory is in the same relative location to the external resources.

Copy Contents of Your 7.0 Domain into the 8.1 Directory

Copy the contents of your WebLogic Server 7.0 domain directory to the new 8.1 domain directory, including the server start scripts and configuration settings (see [“Contents of a Domain Directory” on page 2-2](#)).

Once you have upgraded the domain to WebLogic Server 8.1, you may not be able to convert it back to WebLogic Server 7.0, so it is a good idea to leave the original domain as a backup.

BEA Systems recommends that domains be located outside of the WebLogic Server install directory.

Note: At this point, application paths in the newly created domain still point to the applications that were deployed in the 7.0 domain. Do not try to start server instances in either domain until you have completed the domain conversion—if you simultaneously run server instances that deploy the same applications under different versions of WebLogic Server, problems are likely to occur.

Delete WebLogic Server 7.0 Settings from the 8.1 Domain

In the new domain directory, delete the contents of all `.internal` and `.wlnotdelete` directories. These directories may contain version-specific generated settings that can cause problems in the 8.1 domain.

The location of these directories depends on your domain configuration. In the Petstore upgrade example (see [“Upgrading the Pet Store Application from WebLogic Server 7.0 to WebLogic Server 8.1” on page 2-8](#)), these directories are located at

```
C:\petstorefrom70to81\server\config\petstore\petstoreServer\.internal and  
C:\petstorefrom70to81\server\config\petstore\petstoreServer\.internal\.wln  
otdelete.
```

Modify Server Start Scripts

Modify the server start scripts in the 8.1 domain directory to start WebLogic Server 8.1 server instances instead of 7.0 server instances. Do this for all Administration Servers and Managed Servers in the upgraded domain. The names of default start scripts created with new WebLogic Server domains are `startWebLogic.cmd` (or `.sh`) (for Administration Servers) and `startManagedWebLogic.cmd` (or `.sh`) (for Managed Servers).

The server start scripts in both the 7.0 and the 8.1 domains reference a server start script `startWLS.cmd` in the `WL_HOME\server\bin` directory, where `WL_HOME` is the WebLogic Server installation.

Edit server start scripts that call the `startWLS.cmd` script in your WebLogic Server 7.0 `WL_HOME70\server\bin` directory to call instead the `startWLS.cmd` script in your WebLogic Server 8.1 `WL_HOME81\server\bin` directory.

Depending on your server start script, it is likely that you need to change the settings of several of its properties. See [“Modifying Start Scripts” on page 2-6](#).

Upgrade Applications

Upgrade your applications to WebLogic Server 8.1.

This may involve editing your application’s deployment descriptor files and the domain configuration file, `config.xml`.

1. Make sure that when your applications reference a file outside of your new 8.1 domain, they still reference the correct address of the external file.

References to external files may have been made inaccurate when you moved your application to the new 8.1 domain, if the 8.1 domain is in a location relative to the external files that is not the same as the location of the old 7.0 domain. Deployment descriptor files and `config.xml` must accurately reference external files (for example log files, file-based repositories, the Java compiler) relative to the new domain directory.

2. Optionally, upgrade references to utilities that you want to upgrade.

For example, update the Java compiler to a newer version by using the WebLogic Server 8.1 Administration Console to update the appropriate attribute for the new version of the utility. For example, to change the Java compiler in the Administration Console, select the server in the left hand pane, go to Configuration | General in the right hand pane, and change the value in the Java Compiler field. If you plan to recompile your applications using WebLogic Server 8.1, reference JDK 1.4 when you build (see [“Reference JDK 1.4 When Compiling in 8.1” on page 2-16](#)).

3. Your application may require additional upgrade steps to account for other factors (for example, new parsers, new or deprecated deployment descriptor elements, altered APIs). See [Additional Upgrade Procedures and Information](#) for information about these factors.

Start 8.1 Administration and Managed Servers

Start WebLogic Server 8.1 Administration and Managed servers. For information about starting WebLogic Server 8.1, see [Starting and Stopping Servers: Quick Reference](#).

Note: WebLogic Server 8.1 automatically updates configuration information read from the 7.0 `config.xml` file to include WebLogic Server 8.1 information. In order for these changes

to be retained between invocations of the server, the `config.xml` file must be writable. If you have made your `config.xml` read-only, access its file properties and change the attribute so that it is writable. For example, in Windows, right-click the file in Windows Explorer, select Properties, and make sure that the Read-Only attribute is unchecked.

Configure and Deploy Your Applications

For information about configuring and deploying your applications, see [Deployment](#).

Modifying Start Scripts

For a concrete example of a 7.0 start script being modified to 8.1, see “[Upgrading the Pet Store Application from WebLogic Server 7.0 to WebLogic Server 8.1](#)” on page 2-8.

The following general procedure shows how to modify start scripts from WebLogic Server 7.0 to work with WebLogic Server 8.1.

1. In the new 8.1 domain created in “[Upgrading Your WebLogic Server 7.0 Domain to Version 8.1](#)” on page 2-2, edit the WebLogic 7.0 start scripts which still point to the 7.0 `startWLS.cmd` (or `.sh`) script to point to the WebLogic Server 8.1 `startWLS.cmd` (or `.sh`) script.

Change all of your WebLogic 7.0 server start scripts to 8.1 server start scripts, including start scripts based on the scripts that the WebLogic Server 7.0 Domain Configuration Wizard generated for your WebLogic Server 7.0 domain Administrative and Managed Servers.

These scripts still point to the WebLogic Server 7.0 script located at `WL_HOME70\server\bin\startWLS.cmd` (or `.sh`); edit them to point to the new WebLogic Server 8.1 script.

For example, change:

```
call "WL_HOME70\server\bin\startWLS.cmd"
```

to:

```
call "WL_HOME81\server\bin\startWLS.cmd"
```

Where:

- `WL_HOME70` is the install directory for WebLogic Server 7.0, typically `C:\bea\70platform\weblogic700`
- `WL_HOME81` is the install directory for WebLogic Server 8.1, typically `C:\bea\weblogic81`

An alternative method for changing this value in a server start script is to define a `WL_HOME` variable at the start of the script, for example:

```
set WL_HOME=C:\bea\weblogic81
```

and then change the value from

```
call "C:\bea\70platform\weblogic700\server\bin\startWLS.sh"
```

to:

```
call "%WL_HOME%/server/bin/startWLS.sh"
```

Depending on what settings your start scripts contain, you may need to modify additional settings such as those described in the remaining steps of this section.

2. Modify `JAVA_HOME`

to point to the JDK that WebLogic Server 8.1 uses. For example, change:

```
set JAVA_HOME=WL_HOME\jdk131_03
```

to:

```
set JAVA_HOME=WL_HOME\jdk141
```

See [Java 2 Platform, Standard Edition Version 1.4.0 Compatibility with Previous Releases](#) for information if you receive error messages about JDK 1.4.1.

3. Modify `bea.home` property to point to your BEA home directory containing the `license.bea` file for WebLogic Server 8.1. For example:

```
-Dbea.home=C:\bea810
```

4. Modify `WL_HOME` to point to the WebLogic Server 8.1 installation directory. For example:

```
WL_HOME=c:\bea810\weblogic810
```

5. Modify `PATH` to include your `%WL_HOME%` 8.1 home. For example:

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

6. Modify `CLASSPATH` to point to the WebLogic Server 8.1 classes. For example:

```
CLASSPATH=%WL_HOME%\lib\weblogic.jar
```

7. Modify or eliminate any WebLogic Server 7.0 start script directory structure tests.

For example, if your script contains a test such as

```
if not exist lib\weblogic.jar goto wrongplace
```

and `lib\weblogic.jar` is not accurate from the location of the script, either change it to point to the relative location of `weblogic.jar`, or simply delete the line.

8. Consider upgrading your JVM to JRockit (see [“Upgrading Your JVM to JRockit” on page 2-17](#)).

WebLogic Server 8.1 installs the JVM, JDK 1.4.1, with the server installation. The `setenv.cmd` and `.sh` scripts provided with the server all point to the JVM. The latest information regarding certified JVMs is available at the [Certifications Page](#).

Upgrading the Pet Store Application from WebLogic Server 7.0 to WebLogic Server 8.1

This section walks through an actual upgrade of Sun’s Pet Store application from WebLogic Server 7.0 to 8.1. This walkthrough uses the version of Pet Store that is included with WebLogic Server 7.0.

The prerequisite for the following procedures is WebLogic Server 7.0 and 8.1 installed on the same machine.

1. Create a new directory in which to upgrade the WebLogic Server 7.0 domain to a WebLogic Server 8.1 domain. In this walkthrough, the new directory is located at `C:\petstorefrom70to81`. This domain should reside outside the WebLogic Server 8.1 installation directory.
2. From the WebLogic Server 7.0 installation, copy the `WL_HOME\samples\server` directory and its contents to `C:\petstorefrom70to81`.
3. Open the `config.xml` file in `C:\petstorefrom70to81\server\config\petstore`. Make the following edits.
 - a. Edit the application paths to point to `C:/petstorefrom70to81`. Replace:

```
Path="WL_HOME/samples/server/stage/petstore/petstore.ear"  
TwoPhase="true">
```

with:

```
Path="c:/petstorefrom70to81/server/stage/petstore/petstore.ear"  
TwoPhase="true">
```

For `petstoreadmin.ear`, `opc.ear`, and `supplier.ear`, and `tour.war`, likewise replace the WebLogic Server 7.1 `WL_HOME` path with the `c:\petstorefrom70to81` path.

- b. Add `XAConnectionFactoryEnabled="true"` to the “Topic” `JMSConnectionFactory` configuration so that it is automatically enlisted in JTA transactions. Replace:

```
<JMSConnectionFactory JNDIName="jms/TopicConnectionFactory"
Name="Topic" Targets="petstoreServer" />
```

with:

```
<JMSConnectionFactory JNDIName="jms/TopicConnectionFactory"
Name="Topic" Targets="petstoreServer"
XAConnectionFactoryEnabled="true" />
```

- c. Optionally replace the `XAServerEnabled` attribute in the “Queue” `JMSConnectionFactory` with `XAConnectionFactoryEnabled`. Replace:

```
<JMSConnectionFactory JNDIName="jms/QueueConnectionFactory"
Name="Queue" Targets="petstoreServer" XAServerEnabled="true" />
```

with:

```
<JMSConnectionFactory JNDIName="jms/QueueConnectionFactory"
Name="Queue" Targets="petstoreServer"
XAConnectionFactoryEnabled="true" />
```

`XAServerEnabled` is deprecated in WebLogic Server 8.1, but is still supported for backward compatibility.

- d. Change the path to the Java compiler, if necessary.
4. Open the `startpetstore.cmd` (or `.sh`) script in `C:\petstorefrom70to81\server\config\petstore` and make the following changes:

- a. Change the `%JAVA_HOME%` alias from the JDK that your WebLogic Server 7.0 installation uses to the JDK that your WebLogic Server 8.1 installation uses. For example:

```
set JAVA_HOME=WL_HOME\jdk131_03
```

becomes:

```
set JAVA_HOME=WL_HOME\jdk141
```

- b. Change the `%SAMPLES_HOME%` alias from the WebLogic Server 7.0 `\samples` directory to the WebLogic Server 8.1 `\samples` directory. For example:

```
set SAMPLES_HOME=WL_HOME\samples
```

becomes:

```
set SAMPLES_HOME=C:\petstorefrom70to81
```

- c. Change the `startWLS.cmd` (or `.sh`) path from the WebLogic Server 7.0 installation to the WebLogic Server 8.1 installation:

```
call "C:\bea70spl\weblogic700\server\bin\startWLS.cmd"
```

- d. Under `JAVA_OPTIONS`, change the `cacerts` path from the WebLogic Server 7.1 installation to the WebLogic Server 8.1 installation.

```
-Dweblogic.security.SSL.trustedCAKeyStore=WL_HOME\server\lib\cacerts
```

5. Test the upgrade by using the `startpetstore.cmd` or `.sh` command to start Pet Store, and then browsing to `http://localhost:7001/petstore` to run the application.

Additional Upgrade Procedures and Information

This section contains additional information that you need to be aware of when upgrading from WebLogic Server 7.0 to 8.1.

- [“Built-in Transformer Based on Apache Xalan XML 2.2 Transformer”](#) on page 2-11
- [“Apache Xerces XML Parser”](#) on page 2-11
- [“Connectors”](#) on page 2-12
- [“Applications with Errors Will Not Deploy”](#) on page 2-13
- [“Upgrading jCOM”](#) on page 2-13
- [“JDBC”](#) on page 2-13
- [“Upgrading Your JDK”](#) on page 2-15
- [“JMS Configuration Checking Is Stricter”](#) on page 2-17
- [“Upgrading Your JVM to JRockit”](#) on page 17
- [“Changes to Timeouts for Pool, JTS/JTA, and RMI Connections”](#) on page 2-19
- [“Prepared Statement Cache Algorithm”](#) on page 2-19
- [“Default Names for Execute Queues Have Changed”](#) on page 2-20
- [“Security”](#) on page 2-20
- [“URIs”](#) on page 2-24
- [“Web Applications”](#) on page 2-24
- [“Web Services”](#) on page 2-25
- [“EJB 2.0”](#) on page 2-25

- “Writable config.xml File” on page 2-27

Built-in Transformer Based on Apache Xalan XML 2.2 Transformer

The built-in transformer in WebLogic Server 8.1 is based on the Apache Xalan 2.2 transformer.

Direct use of the Xalan APIs has been deprecated. If you use those APIs and encounter difficulties, use the *Java API for XML Processing (JAXP)* to use XSLT.

WebLogic Server’s built-in transformer contains changes to Apache’s Xalan code that 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.

Apache Xerces XML Parser

The Xerces XML parser used in Weblogic Server 8.1 is stricter than the one used in WebLogic Server 7.0, and may refuse to parse erroneous XML that was accepted by the WebLogic Server 7.0 parser.

The built-in XML parser for WebLogic Server 8.1 is based on the Apache Xerces 1.4.4 parser. The parser implements Version 2 of the SAX and DOM interfaces. Parsers from previous versions may trigger deprecation messages.

WebLogic Server 8.1 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. For more information, see [Administering WebLogic Server XML](#).

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.

Connectors

The following sections describe changes to resource adapter functionality in WebLogic Server 8.1.

Last-Resource Commit Optimization

Normally, for a client to perform operations on multiple resource adapter connections to participate in a global / XA transaction, the resource adapters involved are required to support XATransaction. However, resource adapters that only support local transactions may also be involved in a global / XA transaction, in a limited manner. This is due to the fact that they do not receive two-phase commit messages from the Transaction Manager.

In WebLogic Server 8.1, if the server detects a Local Transaction capable resource adapter connection in a global transaction, the transaction manager first issues prepare messages to the XAResources involved in the transaction. Then, after all XAResources have prepared successfully, the operation on the local transaction-capable resource adapter is performed. If the operation is successful, the global transaction is committed. If the operation fails, the global transaction is rolled back. This prevents the possibility of the Local Transaction resource adapter's commit failing after an XA resource has already been committed.

Notes: This optimization allows multiple XAResources but not more than one Local Transaction capable resource adapter to participate in a global transaction. Attempts to include more than one Local Transaction capable resource adapter in a global / XA transaction will fail with an exception.

Prior to WebLogic Server 8.1, there was no restriction on the number of Local Transaction capable resource adapters that could be involved in a global / XA transaction. Also, the ordering of the prepares and commits was not coordinated specially for Local Transaction capable resource adapters as it now is in WebLogic Server 8.1. It was previously possible for a call to commit on a Local Transaction capable resource adapter to fail after some of the XA Transaction resource adapters had already been successfully committed. This event caused some resources to be committed and some rolled back for a global / XA transaction. The optimization resolves this previous issue.

Application Container Managed Security Log Messages

Messages indicating whether the client is using application or container-managed security are no longer written to the server log.

Classloader and Packaging Issues

In WebLogic Server 8.1 each resource adapter has its own classloader to load its classes in the same manner as Web applications. With this change in effect, components like Web applications and EJBs that are packaged along with a resource adapter in an application archive (.ear file), do not have visibility into the resource adapter's classes. If visibility is required, place the classes of the resource adapter in `APP-INF/classes` or (packaged in a JAR file) in `APP-INF/lib` directory of the application archive.

Applications with Errors Will Not Deploy

Unlike previous versions, WebLogic Server 8.1 will not deploy an application that has any errors in its deployment descriptor. For example, if your WebLogic Server 7.0 application was missing a reference description stanza in the deployment descriptor, the application will not deploy in WebLogic Server 8.1 until you add that stanza. A typical stanza looks like this:

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

Upgrading jCOM

For information about upgrading from WebLogic jCOM 6.1 to WebLogic jCOM 8.1 see [Upgrading Considerations](#) in *Programming WebLogic jCOM*.

JDBC

This section outlines JDBC-related changes in WebLogic Server 8.1.

Removed Interfaces

Several interfaces that were previously marked deprecated have been removed:

- DB Kona
- JDBCService

Removed Classes

In WebLogic Server 8.1, the `weblogic.jdbc.pool` classes were removed, except for the `weblogic.jdbc.pool.Driver` class. These classes were removed because they were

incompatible with an internal change that enables and enhances support for JDBC extensions provided in JDBC drivers.

If your application uses these classes, to migrate your application to WebLogic Server 8.1, you must change the application to use vendor-specific JDBC driver classes. For example, if your application uses the `weblogic.jdbc.pool.CallableStatement` class, you should change it to use the class from the JDBC driver, such as `oracle.jdbc.OracleCallableStatement`:

Change this line:

```
weblogic.jdbc.pool.CallableStatement cStat =  
(weblogic.jdbc.pool.CallableStatement)connection.prepareCall(call);
```

To this:

```
oracle.jdbc.OracleCallableStatement cStat =  
(oracle.jdbc.OracleCallableStatement)connection.prepareCall(call);
```

Removed Drivers

The following JDBC drivers were removed from WebLogic Server 8.1:

- The WebLogic Sybase jDriver
- The WebLogic MSSQL jDriver

Interoperability Limitation

When a WebLogic Server 8.1 client interoperates with an older version of the server (accessing data sources, for instance), it is not possible to use Oracle JDBC extensions.

Statement Caches

Releases before WebLogic Server 8.1 had separate statement cache implementations for XA and non-XA JDBC connection pools. In WebLogic Server 8.1, one statement cache implementation serves both XA and non-XA connection pools. Connection pool attributes in the `JDBCConnectionPoolMBean` for configuring the statement cache are now deprecated.

In version 8.1, Weblogic Server enforces the following order of precedence for these MBean attributes:

`PreparedStatementCacheSize`

`XAPreparedStatementCacheSize`

`StatementCacheSize`

For example, if the `PreparedStatementCacheSize` for a JDBC connection pool is set to 5 and the `StatementCacheSize` is set to 10, the actual statement cache size for each connection in the connection pool will be 5 because `PreparedStatementCacheSize` takes precedence over `StatementCacheSize`.

Other changes:

- Global pools: applications cannot disable statement caching by setting `PreparedStatementCacheSize` to 0.
- Global pools: when upgraded from 7.0 SP3 or later, the `XAPreparedStatementCacheSize` value is ignored when you configure the cache. Instead, use `PreparedStatementCacheSize` to assign the cache size.
- Application-scoped pools: application-specified values for `XaParamsMBean.PreparedStatementCacheSize` are ignored when configuring the cache. Instead, `PreparedStatementMBean.CacheSize` is used.
- If you are upgrading an application from 7.0 SP3 or later that is using the XA statement cache, if you want to disable XA statement caching, in WebLogic Server 8.1 set the non-XA control (`PreparedStatementMBean.CacheSize`) to 0.

For more information, see [Configuring and Managing the Statement Cache for a Connection Pool](#), in *Configuring and Using WebLogic JDBC*.

Upgrading Your JDK

This section addresses issues you may encounter when upgrading to a later JDK.

Stricter JSP Parsing in JDK 1.3.1_09

Improper code that was acceptable under JDKs before 1.3.1_09 may cause errors under JDK 1.3.1_09 and later. This failure occurs for JSPs and all beans that are subject to the Introspector API. Specifically, disagreement between a property's setter and getter types may cause startup errors like the following after you migrate to JDK 1.3.1_09 or later:

```
Error in using tag library uri='/WEB-INF/tlds/taglib.tld' prefix='j2ee':
There is no setter method for property 'numItems', for Tag class
'com.sun.j2ee.blueprints.petstore.taglib.list.SearchListTag' probably
occurred due to an error in /template.jsp line 8: <%@ taglib
uri="/WEB-INF/tlds/taglib.tld" prefix="j2ee" %>
```

When a class fails conform to Java bean specifications about type agreement, the `java.beans.Introspector` API no longer returns read or write methods for the offending property. Correct such errors by ensuring that the setters and getters in your classes do not disagree in type. If the setter is an integer, the getter must be an integer also, and must not be a string.

For example, the following snippet shows a valid setter and getter from which we can expect the Introspection API to return valid read and write method(s):

```
        private String foo;

    public void setFoo(String f) {
            foo = f;
        }

    public String getFoo() {
            return foo;
        }
}
```

The following snippet demonstrates bad code that does not conform to Java bean specifications:

```
private int foo; // note that foo is an int

public void setFoo(String f) {
        foo = Integer.parseInt(f);
}

public int getFoo() {
        return foo;
}
}
```

In the second case, the type disagreement between the setter and getter will cause an error under JDK 1.3.1_09 and later, because the newer JDKs adhere strictly to the Java bean specifications. */

Reference JDK 1.4 When Compiling in 8.1

If you compile your application in WebLogic Server 8.1, it is advisable to do subsequent builds referencing JDK 1.4 rather than earlier JDKs.

Some APIs and other items added in JDK 1.4 have been removed in the WebLogic Server 8.1 version of `weblogic.jar`. Such APIs and other items cease to be available to your application

in WebLogic Server 8.1 after you compile your application with the 8.1 `weblogic.jar` in place of the earlier version of `weblogic.jar` in your classpath.

JAXP, for example, was included in previous versions of `weblogic.jar`, but is absent from the WebLogic Server 8.1 `weblogic.jar`. JAXP is not in the JDK 1.3, so if your WebLogic Server 7.0 application uses JAXP and you compile it after replacing the 7.0 `weblogic.jar` with the 8.1 `weblogic.jar`, build your classes using the JDK 1.4.

Other WebLogic Server 8.1 dependencies on JDK 1.4 include the JAAS package and the Principal Authenticator object.

JMS Configuration Checking Is Stricter

JMS configuration checking is tighter in WebLogic Server 8.1 than it was in release 7.0, particularly in regards to naming JMS destinations and connection factories. In release 7.0, administrators could configure:

- Multiple JMS destinations within a cluster with the same replicated JNDI name (not including distributed destination members).
- A connection factory with the same JNDI name as the default connection factory.

In release 8.1, however, you cannot assign identical JNDI names in these cases. Therefore, a release 7.0 configuration that has either of these replicated JNDI name scenarios may fail to boot after it is upgraded to release 8.1.

Upgrading Your JVM to JRockit

When you upgrade a domain to WebLogic Server 8.1, consider upgrading your JVM to JRockit. WebLogic JRockit is a JVM designed for running server-side applications in Windows and Linux running on Intel architectures. For server-side applications, JRockit has these advantages over other virtual machines:

- It employs adaptive optimization, which detects and removes bottlenecks in the deployed application.
- It is designed specifically for the special requirements of server-side applications, which tend to be parallel and thread-intensive, to run for longer periods of time, and not to use graphical interfaces.
- You can monitor JRockit using the WebLogic Server Administration Console.

For JRockit platform and user information, see the appropriate version of the JRockit [User Guide](http://edocs.bea.com/wljrockit/docs81/userguide/index.html) at <http://edocs.bea.com/wljrockit/docs81/userguide/index.html> .

To upgrade a WebLogic Server domain to the JRockit JVM:

1. In the server start scripts, set `JAVA_HOME` (or equivalent) shell variables to point to the JRockit root directory. For example, change:

```
@rem Set user-defined variables.
```

```
set JAVA_HOME=WL_HOME\jdk131
```

where `WL_HOME` is the WebLogic Server 7.0 installation directory, to

```
@rem Set user-defined variables.
```

```
set JAVA_HOME=WL_HOME\jrockit81_141_02
```

where `WL_HOME` is the WebLogic Server 8.1 installation directory.

2. Change the domain's `config.xml` to use the JRockit `javac.exe`. For example, change

```
JavaCompiler="WL_HOME\jdk131\bin\javac"
```

where `WL_HOME` is the WebLogic Server 7.0 installation directory, to

```
JavaCompiler=WL_HOME\jrockit81_141_02\bin\javac"
```

where `WL_HOME` is the WebLogic Server 8.1 installation directory.

3. Remove from server start scripts any switches specific to the Sun JVM. For example, from the start command:

```
echo on "%JAVA_HOME%\bin\java" -hotspot .... weblogic.Server
```

```
delete "-hotspot".
```

4. Start and configure JRockit. See the [Starting and Configuring the WebLogic JRockit JVM](http://edocs.bea.com/wljrockit/docs81/userguide/config.html) at <http://edocs.bea.com/wljrockit/docs81/userguide/config.html> section for the appropriate version of the JRockit documentation.

Load Order Behavior for Startup Classes

The behavior of the load order methods in `StartupClassMbean` has changed between versions 7.0 Service Pack 1 and version 8.1.

In 7.0 Service Pack 1, setting `LoadBeforeAppDeployments` to true caused startup classes to be invoked after the datasources were created and before the applications were activated. In version 8.1, achieve the same load order by setting `LoadBeforeAppActivation` to true.

`LoadBeforeAppDeployments` still exists in version 8.1 but its behavior has changed since 7.0 Service Pack 1. Beginning with 7.0 Service Pack 2, `LoadBeforeAppDeployments` has determined whether a startup class is loaded and run before the server activates JMS and JDBC services or deploys applications and EJBs.

See details about these methods in WebLogic Server 8.1 at

<http://e-docs.bea.com/wls/docs81/javadocs/weblogic/management/configuration/StartupClassMBean.html>.

Network Channels Include Network Access Point Features

In WebLogic 8.1, network channels encompass the features that, in WebLogic Server 7.0, required both network channels and network access points. If you receive a network channel error message on starting up an upgraded 7.0 domain in WebLogic Server 8.1, reconfigure your network channels as described in [Configuring a Channel](#) in *Configuring and Managing WebLogic Server*.

Network access points are deprecated.

The use of `NetworkChannelMBean` is deprecated in favor of `NetworkAccessPointMBean`.

Changes to Timeouts for Pool, JTS/JTA, and RMI Connections

In past releases, pool connections waited for 5 seconds before timing out. JTS/JTA connections waited 10 seconds before timing out. RMI Datasource connections were non-blocking. In the WebLogic Server 8.1 release, all connections block for up to 10 seconds before timing out.

In cases where you invoke an application by using RMI/T3 or RMI/IIOP, WebLogic Server 7.0 and 8.1 are interoperable. Within a domain, however, all servers must be of the same version.

Prepared Statement Cache Algorithm

A new prepared statement cache algorithm has been introduced. It removes the least recently used statements from the cache. The old algorithm kept a fixed number of statements in the cache (the first n , where n is the configured size of the cache). If you want to get the old cache algorithm behavior, the connection pool can be configured to use the "FIXED" algorithm for the pool.

Default Names for Execute Queues Have Changed

Default names for execute queues have changed in WebLogic Server 8.1. If you upgrade a configuration that specifies execute queues, the default queue names will automatically alias the new queue names.

Table 2-1 Default Queue Names

Pre-8.1 Default Queue Names	WebLogic Server 8.1 Default Queue Names
default	weblogic.kernel.Default
__weblogic_admin_html_queue	weblogic.admin.HTTP
__weblogic_admin_rmi_queue	weblogic.admin.RMI

Security

The following sections discuss general changes to security in WebLogic Server 8.1.

Troubleshooting Problems with Certificates

In versions preceding 7.0 Service Pack 2, WebLogic Server did not ensure each certificate in a certificate chain was issued by a certificate authority. This problem was resolved in Service Pack 2.

If SSL communications worked properly before upgrading to 8.1 but fail unexpectedly after the upgrade, the problem is most likely because the certificate chain is failing validation.

To troubleshoot problems with certificates, try the following methods:

- If you know where the certificate chains for the processes using SSL communication are located, use the `ValidateCertChain` command-line utility to check whether the certificate chains will be accepted. For information about `ValidateCertChain`, see [ValidateCertChain](#) in *Using the WebLogic Server Java Utilities*.
- Turn on SSL debug tracing for the processes using SSL communication. The syntax for SSL debug tracing is:

```
-Dssl.debug=true -Dweblogic.StdoutDebugEnabled=true
```

The following message indicates the SSL failure is due to problems in the certificate chain:

CA certificate rejected. The basic constraints for a CA certificate were not marked for being a CA, or were not marked as critical.

When using one-way SSL, look for this error in the client log. When using two-way SSL, look for this error in the client and server logs.

- Determine where the certificate chain is being rejected, and decide whether to update the certificate chain with one that will be accepted, or make the enforcement of constraints less severe using the `-Dweblogic.security.SSL.enforceConstraints` command-line argument.

Security Data Is Now Written to the `config.xml` File

The persistence model for the COMMO MBeans changed in this release of WebLogic Server. All security configuration data is now stored in the `config.xml` file. Existing security configuration data is written to the `config.xml` file when the WebLogic Server 8.1 server is initially booted. The `config.xml` file must be writable for this upgrade to occur.

Web Resource Is Replaced by URL Resource

The Web resource available in previous releases of WebLogic Server has been replaced by the URL resource. If you wrote a custom Authorization provider that uses the Web resource (instead of the URL resource), enable the Use Deprecated Web Resource attribute. This attribute changes the runtime behavior of the Servlet container to use a Web resource rather than a URL resource when performing authorization.

To use an existing Web resource:

1. Expand the Security Realms node.

All the security realms available for the WebLogic domain are listed in the Realms table.

2. In the Realms table, click the name of the desired security realm.
3. Select the General tab.
4. Check the Use Deprecated Web Resource attribute.
5. Reboot WebLogic Server.

Keystores Are Supported

A new keystore implementation is available in WebLogic Server 8.1. The keystore retrieves trusted CAs, private keys, and server certificates from JDK keystores. The keystore implementation in WebLogic Server 8.1 offers several improvements:

- All JDK keystore types are supported. In WebLogic Server 7.x, only JKS keystores were supported.
- The keystore retrieves the server's certificate from a JDK keystore. In WebLogic Server 7.0, a server's certificate could only be retrieved from a file.
- Each server has its own keystore configuration.
- The keystore can be protected with a password to add another a layer of security to the WebLogic Server deployment.

By default, WebLogic Server has SSL enabled and the server's identity and trust is established with a demonstration certificate and demonstration and standard (JDK) certificate authorities.

Custom Keystore Providers Are No Longer Supported

Custom Keystore providers cannot be used with this release of WebLogic Server. Private keys and trusted CAs should be stored in keystores associated with a particular instance of WebLogic Server. For more information about configuring a keystore for a server and loading private keys and trusted CAs into the keystore, see “Configuring Keystores and SSL” in the Security section of the Administration Console online help.

The WebLogic Keystore Provider Is Deprecated

The WebLogic Keystore provider is deprecated in this release of WebLogic Server. If you are using the WebLogic Keystore provider to store private keys and trusted CAs, the server, when first booted with this release of WebLogic Server, will update the SSL MBean in the `config.xml` file to include the following attribute:

```
IdentityAndTrustLocations=FilesorKeystoreProviders
```

This attribute tells the server to use the 7.x SSL configuration rules instead of the new SSL rules. BEA recommends using this configuration only until you can upgrade to the keystores available in this release of WebLogic Server.

Upgrading to WebLogic Server 8.1 Keystores

Storing identity (private keys and certificates) and trust (certificate authorities) in files is no longer supported. To upgrade to the keystores available in WebLogic Server 8.1:

1. Create a trust keystore. See [Configuring Keystores](#) in *Managing WebLogic Security*.
2. Load the trusted CAs into the keystore. Use the JDK `keytool` utility to perform this step.
3. Create an identity keystore. See [Configuring Keystores](#) in *Managing WebLogic Security*.

4. Load the private key and certificate into the keystore. Use the WebLogic Server `ImportPrivateKey` utility to perform this step. See [ImportPrivateKey](#) in the *WebLogic Server Command Reference*.
5. In the Administration Console, reconfigure SSL to use these trust and identity keystores. See [Configuring Keystores and SSL](#) in the Administration Console help.

Upgrading to the WebLogic Server 8.1 Trust Mechanism for Java Clients

In WebLogic Server 7.0, Java clients (fat clients) retrieved trusted CAs from a jks keystore. The following command-line argument specified the pathname to the keystore:

```
-Dweblogic.security.SSL.trustCAkeystore
```

If the command-line argument was not specified, WebLogic Server defaulted to the trusted CAs in the JDK `jre/lib/security/cacerts` keystore.

The way Java clients retrieve trusted CAs has been improved in the following ways:

- All JDK keystore types are now supported. In WebLogic Server 7.x, only JKS keystores were supported.
- Keystore passwords can be specified.
- Clients can trust more types of certificate authorities. Command-line arguments specify whether to use standard trust (that is, `cacerts` in the JDK), demo trust (that is, `cacerts` in the JDK and the demo trusted CA provided by WebLogic Server), or custom trust.

This release of WebLogic Server still supports the trust command-line argument in WebLogic Server 7.0. As in WebLogic Server 7.0, WebLogic Server 8.1 defaults to the trusted CAs in the JDK `jre/lib/security/cacerts` keystore.

To upgrade to the new trust mechanism in this release of WebLogic Server, specify one of the following command-line arguments.

To trust the JDK standard trusted CAs, use:

```
-Dweblogic.security.TrustKeystore=JavaStandardTrust
```

To trust the JDK standard trusted CAs and the demo trusted CAs provided by WebLogic Server, use:

```
-Dweblogic.security.TrustKeystore=DemoTrust
```

To use a custom keystore, use:

```
-Dweblogic.security.TrustKeystore=CustomTrust
```

```
-Dweblogic.security.CustomTrustKeystorePathname=keystorepathname
```

URIs

Previous versions of WebLogic Server resolved URIs that contained extra spaces. WebLogic Server 8.1 no longer resolves extra spaces, and a URI request that contains extra spaces will result in a 404.

For example, `http://server:port/mywebapp/foo%20%20` formerly resolved to the resource `foo` in the Web application "mywebapp," but beginning with 8.1 it no longer does.

Web Applications

- 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.
- It is no longer possible to use the Administration Console to define the default Web application. Define the default Web application by setting the `context-root` element in the application's `application.xml` file, if the application is part of an enterprise application, or the `weblogic.xml` file if it is a standalone Web application, to `"/`.
- Servlet Request and Response objects have a new API. Some serializable, lightweight implementations of these objects 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.
- HTML Kona, marked deprecated in previous releases, has been removed.
- Five WebLogic-specific context parameters supported by the Web application container have been deprecated:
 - `weblogic.httpd.servlet.reloadCheckSecs` (has been replaced with the `weblogic.xml` `servlet-reload-check-secs`).
 - `weblogic.httpd.servlet.classpath` (instead use the manifest classpath, or `WEB-INF/lib` or `WEB-INF/classes`, or virtual directories).
 - `weblogic.httpd.clientCertProxy` (not yet replaced in `weblogic.xml`).
 - `weblogic.httpd.defaultServlet` (instead define a `servlet-mapping` with `pattern=` in `weblogic.xml`).
 - `weblogic.httpd.inputCharset` (instead use `weblogic.xml` `charset-params`).

For more information about `weblogic.xml`, see [weblogic.xml Deployment Descriptor Elements](#) in *Developing Web Applications for WebLogic Server*.

Web Services

When upgrading Web Services from WebLogic Server 7.0 to WebLogic Server 8.1, you will need to re-run `servicegen` to regenerate the Web Service deployment units.

For detailed information on upgrading a 7.0 WebLogic Web Service to 8.1, see [Upgrading 7.0 WebLogic Web Services to 8.1](http://e-docs.bea.com/wls/docs81/webServices/migrate.html) at <http://e-docs.bea.com/wls/docs81/webServices/migrate.html> in *Programming WebLogic Web Services*.

For examples of using JAX-RPC to invoke WebLogic Web services, see [Invoking Web Services](http://e-docs.bea.com/wls/docs81/webServices/client.html) at <http://e-docs.bea.com/wls/docs81/webServices/client.html> in *Programming WebLogic Web Services*.

For general information on the differences between 7.0 and 8.1 Web services, see [Overview of WebLogic Web Services](http://e-docs.bea.com/wls/docs81/webServices/overview.html) at <http://e-docs.bea.com/wls/docs81/webServices/overview.html> in *Programming WebLogic Web Services*.

EJB 2.0

To see a complete listing of the specification changes, you can view and download the [EJB 2.0 final specification](http://java.sun.com/products/ejb/2.0.html) at <http://java.sun.com/products/ejb/2.0.html>.

Run DDConverter Before Using New Features

Before using any new EJB features in WebLogic Server 8.1, be sure to run the `DDConverter` tool to convert existing EJB deployment descriptors to 8.1 descriptors. For more information, see the discussion of the [DDConverter tool](http://e-docs.bea.com/wls/docs81/ejb/EJB_tools.html#ddconverter) at http://e-docs.bea.com/wls/docs81/ejb/EJB_tools.html#ddconverter.

New Default Values for EJB Deployment Descriptor Elements

In WebLogic Server 8.1, there are new default values for the following EJB deployment descriptor elements:

The new default values are only in effect for deployment descriptors newly created or generated in WebLogic Server 8.1. Existing beans that use pre-8.1 versions of the deployment descriptors can be deployed on the 8.1 release with no change in behavior.

Table 2-2 Element Default Values

Element	Deployment Descriptor	New Default Value
enable-call-by-reference	weblogic-ejb-jar.xml	False
include-updates	weblogic-cmp-rdbms-jar.xml	The default value is <code>False</code> for beans that use optimistic concurrency. The default value is <code>True</code> for beans that use other concurrency types, such as database.
check-exists-on-method	weblogic-cmp-rdbms-jar.xml	True

For more information on EJB deployment descriptors, see [The weblogic-ejb-jar.xml Deployment Descriptor at http://e-docs.bea.com/wls/docs81/ejb/reference.html](http://e-docs.bea.com/wls/docs81/ejb/reference.html) and [The weblogic-cmp-rdbms-jar.xml Deployment Descriptor at http://e-docs.bea.com/wls/docs81/ejb/EJB_reference.html](http://e-docs.bea.com/wls/docs81/ejb/EJB_reference.html) in *Programming WebLogic Enterprise Java Beans*.

Inserting a New CMP Bean at Commit Time

In WebLogic Server 7.0, to have the EJB container perform bulk inserts, you set the `weblogic-cmp-rdbms-jar.xml` element `delay-database-insert-until` to `commit`.

As of the current release, the `commit` value is no longer supported for `delay-database-until-insert`. To permit bulk inserts, set the new `weblogic-cmp-rdbms-jar.xml` element `enable-batch-operations` to `true`.

`enable-batch-operations` takes effect on the jar level, so you need only set this tag once per jar. By contrast, `delay-database-insert-until` had to be set for every bean.

For more information on entity batch operations see [Entity Batch Operations at `http://e-docs.bea.com/wls/docs81/ejb/EJB_environment.html#entity_batch_operations`](http://e-docs.bea.com/wls/docs81/ejb/EJB_environment.html#entity_batch_operations) in *Programming WebLogic Enterprise Java Beans*.

Writable config.xml File

WebLogic Server 8.1 automatically updates configuration information read from the 7.0 `config.xml` file to include WebLogic Server 8.1 information. In order for these changes to be retained between invocations of the 8.1 server, the `config.xml` file must be writable. If you have made your `config.xml` read-only, access its file properties and change the attribute so that it is writable. For example, in Windows, right-click the file in Windows Explorer, select Properties, and make sure that the Read-Only attribute is unchecked.

Upgrading WebLogic Server 7.0 to Version 8.1

Upgrading WebLogic Server 6.x to Version 8.1

Upgrading WebLogic Server 6.x to version 8.1, under the simplest circumstances, involves changing your WebLogic Server start command scripts and environment settings.

BEA Systems recommends copying your WebLogic Server 6.x domain directory to a new directory. If you do this, ensure that relative paths to external files remain accurate.

Upgrading may also require changes specific to the subsystem being upgraded.

The following sections contain information necessary to upgrade your system from WebLogic Server 6.x to WebLogic Server 8.1:

- [“Contents of a Domain Directory” on page 3-2](#)
- [“Understanding the WebLogic Server 8.1 Directory Structure” on page 2-2](#)
- [“Upgrading Your WebLogic Server 6.x Domain to Version 8.1” on page 3-3](#)
- [“Modifying Start Scripts” on page 3-6](#)
- [“Upgrading the Pet Store Application from WebLogic Server 6.1 Service Pack 4 to WebLogic Server 8.1” on page 3-8](#)
- [“Upgrading WebLogic Server 6.x Security to Version 8.1” on page 3-17](#)
- [“Additional Upgrade Procedures and Information” on page 3-53](#)

For instructions on how to upgrade the Pet Store application from WebLogic Server 6.1 to WebLogic Server 8.1, see [“Upgrading the Pet Store Application from WebLogic Server 6.1 Service Pack 4 to WebLogic Server 8.1” on page 3-8](#).

For information on upgrading to WebLogic Platform 8.1, see the [Upgrading](#) section of the WebLogic Server FAQs.

Understanding the WebLogic Server 8.1 Directory Structure

BEA WebLogic recommends that you locate domain directories outside the WebLogic Server installation directory.

. For complete information on the updated directory structure see [Understanding the WebLogic Server Directory Structure](#) in *Performing Post-Installation Tasks* in the *Installation Guide*, and [Domain Restrictions](#) in Overview of WebLogic Server Domains.

In WebLogic Server 6.x, domain directories were created within the directory structure of the WebLogic Server installation. In subsequent versions, domain directories can be in any location that can access the WebLogic Server installation and the JDK.

If you change the location of your domain directory, remember to update any custom tools or scripts relative to the new directory structure. Similarly, if you use a scripted tool for creating domains, change its scripts. The [Configuration Wizard](#) is the recommended tool for creating domains, and it can be scripted.

Contents of a Domain Directory

When you upgrade from WebLogic Server 6.x to 8.1, you are upgrading a WebLogic Server domain. For a full description of WebLogic Server domains see [Overview of WebLogic Server Domains](#) in *Configuring and Managing WebLogic Server*.

The configuration of a domain is stored in the `config.xml` file of the domain directory on the Administration Server. `config.xml` stores the name of the domain and the configuration parameter settings for each server instance, cluster, resource, and service in the domain.

The domain directory also contains server start scripts that start the Administration Server and the Managed Servers in the domain.

The domain directory structure should have a root directory with the same name as the domain, such as `mydomain` or `petstore`. This directory should contain the following:

- The configuration file (`config.xml`) for the domain.
- Any scripts you use to start server instances and establish your environment.
- Optionally, a subdirectory for storing the domain's application files.

For more information about WebLogic Server domains, see [WebLogic Server Domains](#) in *Configuring and Managing WebLogic Server*.

Upgrading Your WebLogic Server 6.x Domain to Version 8.1

To upgrade a domain with multiple server instances, upgrade as you would a single-server domain. In a nutshell, you upgrade a domain by installing WebLogic Server 8.1 alongside your 6.x installation, copying the contents of your 6.x domain into a new 8.1 domain directory, and changing the domain scripts and environment settings to point to the new 8.1 domain and server instances.

If the domain includes a cluster and you want to take advantage of new clustering features in WebLogic Server 8.1, refer to [Setting Up WebLogic Clusters](#) in *Using WebLogic Server Clusters* for WebLogic Server 8.1 cluster configuration guidelines.

As a prerequisite to performing the upgrade procedures in this section, install WebLogic Server 8.1 on all the machines that contain 6.x server instances whose domain you will upgrade. See [Installing WebLogic Platform](#) for installation instructions for WebLogic Server 8.1.

- [“Shut Down WebLogic 6.x Server Instances”](#) on page 3-3
- [“Create a WebLogic Server 8.1 Domain Directory”](#) on page 3-4
- [“Copy Contents of Your 6.x Domain into the 8.1 Directory”](#) on page 3-4
- [“Delete WebLogic Server 6.x Settings from the 8.1 Domain”](#) on page 3-4
- [“Modify Server Start Scripts”](#) on page 3-4
- [“Upgrade Applications”](#) on page 3-5
- [“Start 8.1 Administration and Managed Servers”](#) on page 3-6
- [“Configure and Deploy Your Applications”](#) on page 3-6

Shut Down WebLogic 6.x Server Instances

Shut down all server instances in the WebLogic Server 6.x domain you are upgrading.

Shutting down the servers ensures that any changes you have made to the domain or its applications have been persisted. See [Starting and Stopping Servers: Quick Reference](#).

Create a WebLogic Server 8.1 Domain Directory

Create a new directory where your 8.1 domain will reside.

Consider that after you move the contents of your 7.0 domain directory to this new 8.1 domain directory, any references from within the 7.0 domain to resources outside the domain will need to change unless the new domain directory is in the same relative location to the external resources.

Copy Contents of Your 6.x Domain into the 8.1 Directory

Copy the contents of your WebLogic Server 6.x domain directory to the new 8.1 domain directory, including the server start scripts and configuration settings (see [“Contents of a Domain Directory” on page 3-2](#)).

Note that once you have upgraded the domain to WebLogic Server 8.1, you may not be able to convert it back to WebLogic Server 6.x.

Note: At this point, application paths in this new domain which will become your 8.1 domain still point to the applications that were deployed in the 6.x domain. Do not try to start server instances in either domain until you have completed the domain conversion—if you simultaneously run server instances that deploy the same applications under different versions of WebLogic Server, problems are likely to occur.

Delete WebLogic Server 6.x Settings from the 8.1 Domain

In the new 8.1 domain directory, delete files with the `.log` suffix, and the `\.wlnotdelete` folder. These artifacts may contain 6.x-specific settings that can cause problems in the 8.1 domain.

Modify Server Start Scripts

Modify the server start scripts in the new 8.1 domain directory to point to WebLogic Server 8.1 server instances instead of 6.x server instances. Do this for all Administration Servers and Managed Servers in the upgraded domain. The names of default start scripts created with new WebLogic Server domains are `startWebLogic.cmd` (or `.sh`) (for Administration Servers) and `startManagedWebLogic.cmd` (or `.sh`) (for Managed Servers).

The server start scripts in both the 6.x and the 8.1 domains reference a server start script in the `WL_HOME\server\bin` directory, where `WL_HOME` is the WebLogic Server installation.

If your start script calls the `startWLS.cmd` script in your WebLogic Server 6.x `WL_HOME\server\bin` directory, change it to call instead the `startWLS.cmd` in your WebLogic Server 8.1 `WL_HOME\server\bin` directory.

Depending on your server start script, it is likely that you need to change the settings of several of its properties. See [“Modifying Start Scripts” on page 3-6](#).

Upgrade Applications

Upgrade your applications to WebLogic Server 8.1 by checking references to external resources, updating utilities and optionally your JVM, and upgrading EJBs and security and other factors.

1. Make the relative paths in the application’s deployment descriptor files point to the actual locations of any external files they reference, in case the location of the external files relative to the application has changed.

In this step, you are making sure that when your application references an external file, it references the correct address of the external file. This may involve editing your application’s deployment descriptor files and the domain configuration file, `config.xml`.

WebLogic Server configurations rely on a number of files that can be stored anywhere on the file system (for example, log files, file-based repositories, the Java compiler). Unless all such external files are referenced using relative paths and are located in or below the domain directory, do one of the following:

- Re-create the directory locations of the external files relative to the new locations of the files that reference them
 - Edit the references to the actual locations of the files.
2. To update utilities such as the Java compiler, use the WebLogic Server 8.1 Administration Console to configure the appropriate attribute to use the new file or utility.
 3. Consider upgrading your JVM to JRockit. See [“Upgrading Your JVM to JRockit” on page 3-60](#).
 4. If you are upgrading EJBs from WebLogic Server 6.0, you may have to make manual changes as described in [“Manual Changes to Deployment Descriptors for EJB 2.0” on page 3-55](#).
 5. Your application may require additional upgrade steps to account for other factors, for example new parsers and altered APIs. See [“Additional Upgrade Procedures and Information” on page 3-53](#) for information about these factors.

6. For instructions on configuring security in WebLogic Server 8.1, see [Upgrading WebLogic Server 6.x Security to Version 8.1 on page -17](#).

Start 8.1 Administration and Managed Servers

Start WebLogic Server 8.1 Administration and Managed servers, and configure and deploy your applications.

Note: WebLogic Server 8.1 automatically updates configuration information in the 6.x `config.xml` file to include WebLogic Server 8.1 information. In order for these changes to be retained between invocations of the server, the `config.xml` file must be writable. If you have made your `config.xml` read-only, access its file properties and change the attribute so that it is writable. For example, in Windows, right-click the file in Windows Explorer, select Properties, and make sure that the Read-Only attribute is unchecked.

For information about starting WebLogic Server 8.1, see [Starting and Stopping Servers: Quick Reference](#).

Configure and Deploy Your Applications

For information about configuring and deploying your applications, see [Deployment](#).

Modifying Start Scripts

If you used WebLogic Server start scripts with a previous version of the product, modify them to work with WebLogic Server 8.1.

For a concrete example of a start script being modified, see “[Upgrading the Pet Store Application from WebLogic Server 6.1 Service Pack 4 to WebLogic Server 8.1](#)” on page 3-8.

In general, modify the start scripts as described here. Your domain’s start scripts may differ significantly from the `startWebLogic.cmd` (or `.sh`) script on which the following instructions are based. These instructions are valid for both Administration Server start scripts and Managed Server start scripts—all server start scripts that reference the 7.0 servers must be made to reference the 8.1 servers.

These instructions assume that you have performed the first two steps in the previous section, “[Upgrading Your WebLogic Server 6.x Domain to Version 8.1](#)” on page 3. That is, before performing the following steps you should have done the following:

- Shut down your server instance
- Copied your domain to a new location

To modify a start script:

1. Remove or comment out the `JAVA_HOME` setting.

For example, remove:

```
set JAVA_HOME=C:\bea61\jdk131
```

The `JAVA_HOME` variable is defined elsewhere (in the `commEnv.cmd` script).

2. Remove the lines that specify the JDK and the `startWebLogic.cmd` (or `.sh`) if they are present.

Remove:

```
@rem Check that script is being run from the appropriate directory
if not exist lib\weblogic.jar goto wrongplace
goto checkJDK
:wrongplace
echo startWebLogic.cmd must be run from the config\mydomain 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 startWebLogic.cmd script so that the JAVA_HOME
echo variable points to the root directory of your JDK installation.
goto finish
```

3. Change the `PATH` setting.

Remove the reference to `.\bin`.

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

4. Remove the `CLASSPATH` setting.

```
set CLASSPATH=.;.\lib\weblogic_sp.jar;.\lib\weblogic.jar
```

5. Add the following lines:

```
set WL_HOME=(the WebLogic Server 8.1 installation directory)
call %WL_HOME%\common\bin\commEnv.cmd
```

```
set CLASSPATH=%WEBLOGIC_CLASSPATH%;%CLASSPATH%
```

6. In the start command line (the line that ends with “weblogic.Server”), remove the reference to `bea.home` (for instance, “-Dbea.home=C:\bea61”)

WebLogic Server 8.1 installs the JVM, JDK 1.4.1, with the server installation. The `setenv.cmd` and `.sh` scripts provided with the server all point to the JVM. The latest information regarding certified JVMs is available at the [Certifications Page](#).

Upgrading the Pet Store Application from WebLogic Server 6.1 Service Pack 4 to WebLogic Server 8.1

This section walks through an actual upgrade of Sun’s Pet Store application from WebLogic Server 6.1 to 8.1. This walkthrough uses the version of Pet Store that is included with WebLogic Server 6.x Service Pack 4.

In the following procedures it is assumed that WebLogic Server 6.1 and WebLogic Server 8.1 are both installed.

Before the actual upgrade steps, some problems in the Pet Store application need to be fixed. Stricter JSP and XML parsing in WebLogic Server 8.1 requires that minor XML and JSP errors that were acceptable in WebLogic Server 6.x be fixed.

Repair Pet Store

This section describes corrections that need to be made to Pet Store in order to deploy it on WebLogic Server 8.1.

Versions of Pet Store from WebLogic Server 6.1SP6 and later do not require these repairs.

The procedures are as follows:

- “Fix Erroneous <ejb-ref-name>” on page 3-9
- “Add Missing <ejb-ref-name>” on page 3-10
- “Fix Encoding Error” on page 3-11
- “Clean fileRealm.properties” on page 3-11
- “Fix JSP Parsing Errors” on page 3-11
- “Rebuild Petstore” on page 3-14

Following this preliminary section on repairing Pet Store in preparation for upgrade, [“Upgrade the Pet Store Domain” on page 14](#) provides instructions for upgrading the Pet Store domain.

Fix Erroneous <ejb-ref-name>

If you are upgrading from a WebLogic Server Service Pack later than 6.1 Service Pack 4, you do not need to complete this step.

Post-6.0 versions of WebLogic Server require that resource references defined in WebLogic-specific deployment descriptors match resource references in Sun deployment descriptors.

The following steps fix resource references that are mismatched in the Pet Store deployment descriptor files `customer_weblogic_ejb.xml` and `customer_ejb.xml`.

1. In a command console or file browser, navigate to `WL_HOME\samples\petStore\src\components\customer\src`. For example:

```
C:\> cd WL_HOME\samples\petStore\src\components\customer\src
```

2. Open `customer_weblogic_ejb.xml` in a text editor. For example:

```
WL_HOME\samples\petStore\src\components\customer\src> notepad  
customer_weblogic_ejb.xml
```

3. In `customer_weblogic_ejb.xml`, add the text bolded below.

```
<ejb-name>TheCustomer</ejb-name>  
<reference-descriptor>  
<ejb-reference-description>  
<ejb-ref-name>ejb/account/Account</ejb-ref-name>  
<jndi-name>estore/account</jndi-name>  
</ejb-reference-description>  
<ejb-reference-description>  
<ejb-ref-name>ejb/order/Order</ejb-ref-name>  
<jndi-name>estore/order</jndi-name>  
</ejb-reference-description>  
</reference-descriptor>  
<jndi-name>estore/customer</jndi-name>
```

4. Add the bolded text in the following:

```
<ejb-name>TheOrder</ejb-name>
<reference-descriptor>
<resource-description>
<res-ref-name>jdbc/EstoreDataSource</res-ref-name>
<jndi-name>jdbc.EstoreDB</jndi-name>
</resource-description>
<ejb-reference-description>
<ejb-ref-name>ejb/account/Account</ejb-ref-name>
<jndi-name>estore/account</jndi-name>
</ejb-reference-description>
</reference-descriptor>
<jndi-name>estore/order</jndi-name>
```

5. Save and close `customer_weblogic_ejb.xml`.

Add Missing `<ejb-ref-name>`

1. In a command console, navigate to `WL_HOME\samples\petStore\src\petstore\src\docroot\WEB-INF`, where `WL_HOME` is the WebLogic Server 6.1 installation directory. For example:

```
C:\> cd WL_HOME\samples\petStore\src\petstore\src\docroot\WEB-INF
```

2. Open `weblogic.xml` in a text editor. For example:

```
WL_HOME\samples\petStore\src\petstore\src\docroot\WEB-INF>notepad
weblogic.xml
```

3. `weblogic.xml` defines a number of `resource-description` and `ejb-reference-description` elements for the Web application. Add the following `ejb-reference-description` to the file:

```
<ejb-reference-description>
  <ejb-ref-name>ejb/profilemgr/ProfileMgr</ejb-ref-name>
  <jndi-name>estore/profilemgr</jndi-name>
</ejb-reference-description>
```

4. Save and close `weblogic.xml`.

Fix Encoding Error

Because of stricter XML parsing, WebLogic Server 8.1 will not allow this encoding error to pass, even though it was acceptable to earlier versions of WebLogic Server.

1. In a command console, navigate to

`WL_HOME\samples\petStore\src\petstore\src\docroot\WEB-INF\xml\ja`. For example:

```
C:\>cd WL_HOME\samples\petStore\src\petstore\src\docroot\WEB-INF\xml\ja
```

2. Open `screendefinitions.xml` in a text editor. For example:

```
WL_HOME\samples\petStore\src\petstore\src\docroot\WEB-INF\xml\ja>  
notepad screendefinitions.xml
```

3. Change the encoding in `screendefinitions.xml` from Unicode to utf-16.

```
<?xml version="1.0" encoding="Unicode"?>
```

becomes:

```
<?xml version="1.0" encoding="utf-16"?>
```

4. Save and close `screendefinitions.xml`.

Clean fileRealm.properties

Delete a wildcard setting from a Pet Store property file because it may cause a security error.

1. In a command console, navigate to `WL_HOME\config\petstore`.

2. Open `fileRealm.properties` in a text editor. For example:

```
WL_HOME\config\petstore>notepad fileRealm.properties
```

3. Locate and delete this line:

```
acl.reset.weblogic.jdbc.connectionPool=system
```

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`

Upgrading WebLogic Server 6.x to Version 8.1

```
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\blueprints\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\blueprints\petstore>taglib\list
```

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

`WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\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 `CartListTag.java`.

6. Repeat steps 1 through 5 for the remaining files:

```
MyListTag.java
ProductItemListTag.java
ProductListTag.java
SearchListTag.java
```

Rebuild Petstore

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

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

```
WL_HOME\config\examples> setexampleseenv.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 application builds `petstore.ear` to

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

3. Copy the new `petstore.ear` from `WL_HOME\samples\petStore\src\petstore\build` to `WL_HOME\config\petstore\applications`.

Upgrade the Pet Store Domain

In this upgrade procedure your WebLogic Server 8.1 configuration continues to point to the Cloudscape database used in WebLogic Server 6.1.

1. Create a new directory, `C:\petstorefrom61to81`, in which to upgrade the WebLogic Server 6.1 Pet Store domain to a WebLogic Server 8.1 Pet Store domain.

In WebLogic Server 8.1, it is advisable to locate domains outside the WebLogic Server installation directory.

2. Copy these two directories and their contents from the WebLogic Server 6.1 `WL_HOME` directory to `C:\petstorefrom61to81`:

```
WL_HOME\config
WL_HOME\samples
```

3. Open the `config.xml` file in `C:\petstorefrom61to81\config\petstore` in a text editor. For example:

```
C:\petstorefrom61to81\config\petstore>notepad config.xml
```

4. Make the following edits to `config.xml`.

- a. Edit the application paths for `petstore.ear` and `petstoreadmin.ear` to point to `C:\petstorefrom61to81`. Replace:

```
Path="WL_HOME\config\petstore\applications">
```

with:

```
Path="c:\petstorefrom61to81\config\petstore\applications">
```

- b. For the server's `RootDirectory` setting, replace the WebLogic Server 6.x `WL_HOME` path with the `c:\petstorefrom61to81` path. That is, replace:

```
Name="petstoreServer" RootDirectory="WL_HOME"
```

with

```
Name="petstoreServer" RootDirectory="C:\petstore61to81"
```

- c. Change the path from the WebLogic Server 6.x Java compiler to the WebLogic Server 8.1 Java compiler. For example, change the following setting (in which `WL_HOME` is the WebLogic Server 6.1 install directory)

```
JavaCompiler="WL_HOME\jdk131\bin\javac"
```

to the following (in which `WL_HOME` is the WebLogic Server 8.1 install directory):

```
JavaCompiler="WL_HOME\jdk141_03\bin\javac"
```

5. Open the `startpetstore.cmd` (or `.sh`) script in `C:\petstorefrom61to81\config\petstore`. For example:

```
C:\petstorefrom61to81\config\petstore>notepad startpetstore.cmd
```

6. Make the following edits to `startpetstore.cmd` or `.sh`:

- a. Change `%JAVA_HOME%` from the JDK your WebLogic Server 6.x installation uses to the JDK your WebLogic Server 8.1 installation uses. For example:

```
set JAVA_HOME=WL_HOME\jdk131
```

where `WL_HOME` is the WebLogic Server 6.1 installation directory, becomes:

```
set JAVA_HOME=WL_HOME\jdk141
```

where `WL_HOME` is the WebLogic Server 8.1 installation directory.

- b. Edit the line that checks which directory contains the script from:

```
if not exist lib\weblogic.jar goto wrongplace
```

to read:

```
if not exist WL_HOME\server\lib\weblogic.jar goto wrongplace
```

where `WL_HOME` is the WebLogic Server 8.1 installation directory.

- c. Change the classpath from

```
set
CLASSPATH=.;C:\bea\81feb26\weblogic81\server\lib\weblogic.jar;.WL_HOME61\samples\eval\cloudscape\lib\cloudscape.jar;.\config\petStore\serverclasses
```

to

```
set
CLASSPATH=.;WL_HOME81\server\lib\weblogic.jar;.C:\petstorefrom61to81\samples\eval\cloudscape\lib\cloudscape.jar;.C:\petstorefrom61to81\config\petstore\serverclasses
```

- d. Change the `-Dbea.home` setting from the 6.x `BEA_HOME` to the 8.1 `BEA_HOME`, where `BEA_HOME` is the directory that contains the WebLogic Server installation directory. For example, change

```
-Dbea.home=C:\bea\bea61
```

to

```
-Dbea.home=C:\bea\bea81
```

- e. Change the security policy setting, `-Djava.security.policy`, from the 6.x security policy file to the 8.1 security policy file. In the setting below, `WL_HOME` is the 6.x installation directory, and needs to be changed to point to the 8.1 installation directory.

```
-Djava.security.policy=="WL_HOME/server/lib/weblogic.policy"
```

7. Test the upgrade by starting Pet Store using the

`C:\petstorefrom61to81\config\petstore\startpetstore.cmd` or `.sh` command, and then browsing to `http://localhost:7001/petstore` to run the application.

Upgrading WebLogic Server 6.x Security to Version 8.1

WebLogic Server 8.1 has a new security architecture. For details on the changes, see [“What Changed in WebLogic Security”](#) in *Introduction to WebLogic Security*.

WebLogic Server 8.1 detects whether you are upgrading from an earlier WebLogic Server version such as 6.x or whether you are a new customer starting with version 8.1. If you are upgrading from WebLogic Server 6.x, you can run WebLogic Server 8.1 in Compatibility security, which 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 8.1 offers improved and expanded security features—customers are encouraged to upgrade their security configuration. This section contains the following upgrade issues and procedures:

- [“Security Realms in WebLogic Server 8.1 Versus 6.x” on page 3-17](#)
- [“Upgrading Your WebLogic Server 6.x Security Realms to 8.1” on page 3-18](#)
- [“Access Control Lists \(ACLs\)” on page 3-43](#)
- [“Upgrading from Compatibility Security to WebLogic Server 8.1 Security” on page 3-47](#)
- [“Guest and <Anonymous> Users” on page 3-48](#)
- [“password.ini File” on page 3-49](#)
- [“Upgrading SSL” on page 3-49](#)

Security Realms in WebLogic Server 8.1 Versus 6.x

The scope of security realms changed in WebLogic Server 8.1. 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. For more information about WebLogic Server 6.x security realms, see [“Security Realms”](#) in *Programming WebLogic Security*.

In WebLogic Server 8.1, each security realm consists of a set of configured security providers, users, groups, security roles, and security policies. Authentication and Authorization providers within a security realm offer authentication and authorization services. For more information about WebLogic Server 8.1 security realms, see [“Security Realms”](#) in *Introduction to WebLogic Security*.

Upgrading Your WebLogic Server 6.x Security Realms to 8.1

You have the following choices when upgrading a WebLogic Server 6.x security realm to WebLogic Server 8.1:

- Use Compatibility security—Booting WebLogic Server 8.1 in Compatibility security provides easy access to your WebLogic Server 6.x security configuration, including your users, groups, and ACLs. When using Compatibility security, however, the only security realm available is the `CompatibilityRealm`. For detailed information about booting WebLogic Server 8.1 in Compatibility security and using Compatibility security, see [“Using Compatibility Security”](#) in *Managing WebLogic Security*.

Compatibility security also allows you to start exploring WebLogic Server 8.1 security features so you can later convert your 6.x security configuration to the new security mechanisms (such as security roles and security policies) that are available in WebLogic Server 8.1. For information about moving off Compatibility Security entirely, see [“Upgrading from Compatibility Security to WebLogic Server 8.1 Security”](#) on page 3-47.

- Use the Realm Adapter security providers—Configuring the Realm Adapter Authentication provider in a pure WebLogic Server 8.1 security realm (that is, a security realm other than the `CompatibilityRealm`) allows you to use the new security mechanisms available in WebLogic Server 8.1 while accessing the users and groups stored in a WebLogic Server 6.x LDAP, Windows NT, UNIX or RDBMS security realm.

Note: You will not be able to access ACLs stored in a WebLogic Server 6.x security realm if you configure the Realm Adapter Authentication provider as an Authentication provider in a WebLogic Server 8.1 security realm. You will have to protect your application resources with security roles and security policies.

The following sections provide step-by-step instructions for upgrading your WebLogic Server 6.x security realms to version 8.1.

Note: Before following the instructions in this section, be sure to upgrade your WebLogic Server 6.x domain to version 8.1, using the instructions in [“Upgrading Your WebLogic Server 6.x Domain to Version 8.1”](#) on page 3-3.

- [“Upgrading from an LDAP V2 Security Realm”](#) on page 3-19
- [“Upgrading from a Windows NT Security Realm”](#) on page 3-24
- [“Upgrading from a UNIX Security Realm”](#) on page 3-29
- [“Upgrading from an RDBMS Security Realm”](#) on page 3-35
- [“Upgrading from a Custom Security Realm”](#) on page 3-42

Upgrading from an LDAP V2 Security Realm

This section provides step-by-step instructions for upgrading your WebLogic Server 6.x LDAP V2 security realm to a WebLogic Server 8.1 security realm. This upgrade causes the users and groups defined in your LDAP server to be referenced from the `myrealm` security realm, which is the default (active) security realm in WebLogic Server 8.1.

Note: Security policies replace the access control lists (ACLs) and permissions that were used to protect WebLogic resources in WebLogic Server 6.x. Therefore, no ACLs will be referenced from the 8.1 security realm as a result of this upgrade. To learn about re-securing resources in WebLogic Server 8.1, see [“Verify the upgrade of ACLs by:” on page 3-44](#) and [Securing WebLogic Resources](#).

To upgrade from a WebLogic Server 6.x LDAP V2 security realm to a WebLogic Server 8.1 security realm:

- [“Step 1: Verify and Capture Your WebLogic Server 6.x LDAP V2 Security Realm Configuration” on page 3-19](#)
- [“Step 2: Configure a WebLogic LDAP Authentication Provider” on page 3-21](#)
- [“Step 3: Enable Communication Between Your LDAP Server and WebLogic Server 8.1” on page 3-22](#)
- [“Step 4: Specify How Users and Groups are Stored in the LDAP Directory” on page 3-22](#)
- [“Step 5: Restart the WebLogic Server 8.1 Instance and Verify the Referenced Users and Groups” on page 3-23](#)

Step 1: Verify and Capture Your WebLogic Server 6.x LDAP V2 Security Realm Configuration

1. In the left pane of the WebLogic Server 6.x Administration Console, expand Security.
2. Click Users.

A list of currently defined users for the WebLogic Server 6.x LDAP V2 security realm appears at the bottom of the right pane. You will be referencing these users from the WebLogic Server 8.1 security realm.

3. In the left pane of the WebLogic Server 6.x Administration Console, click Groups.

A table of currently defined groups for the WebLogic Server 6.x LDAP V2 security realm appears in the right pane. You will be referencing these groups from the WebLogic Server 8.1 security realm.

4. In the left pane of the WebLogic Server 6.x Administration Console, expand Realms.

The Realms node expands to show the security realms for the WebLogic Server 6.x domain, including the LDAP V2 security realm.

5. Click the name of the LDAP V2 security realm.

For example, click `defaultLDAPRealmForNetscapeDirectoryServer`.

6. On the Configuration tab:

- a. Select the contents of the Configuration Data field.

For example, select:

```
user.filter=( &(uid=%u) (objectclass=person) )
user.dn=ou=people, ou=Test,dc=companysys,dc=com
server.port=1155
membership.filter=( &(uniquemember=%M) (objectclass=groupofuniquenames) )
server.principal=uid=tadmin, ou=People,dc=companysys,dc=com
group.filter=( &(cn=%g) (objectclass=groupofuniquenames) )
group.dn=ou=Groups,ou=Test,dc=companysys,dc=com
server.host=testmachine
```

- b. Copy the selected text to a temporary location, such as a Notepad file.
7. Be sure you know the password for the user that can log into your LDAP server.
8. Check whether you have SSL enabled by navigating to the Servers → *myserver* → Configuration → SSL tab in the WebLogic Server 6.x Administration Console, where *myserver* is the name of your WebLogic Server 6.x instance.
9. Record the values of any Cache Size and Cache TTL fields in the LDAP V2 security realm's associated Caching realm by navigating to the subtabs of Security → Caching Realms → *myCachingRealm* → Configuration, where *myCachingRealm* is the name of your Caching realm.

Step 2: Create a WebLogic Server 8.1 Domain and Security Realm

Note: If you are running WebLogic Server 6.x and 8.1 on the same machine, be sure to stop your WebLogic Server 6.x instance before following these instructions.

1. In WebLogic Server 8.1, run the Configuration Wizard to create a WebLogic Server 8.1 domain called *mydomain*, by accepting most of the defaults.

Note: Accept all the defaults in the Configuration Wizard to create the WebLogic Server 8.1 domain, with the exception of the username and password combination that is

used to boot the server. This username and password combination must be in your LDAP server, and the user must be a member of the Administrators group.

Step-by-step instructions for using the Configuration Wizard are in “Creating and Configuring Domains Using the Configuration Wizard” in *Configuring and Managing WebLogic Server*.

2. Start the WebLogic Server 8.1 instance called `myserver`.
3. Log into the WebLogic Server 8.1 Administration Console.
4. In the left pane of the WebLogic Server 8.1 Administration Console, expand Security → Realms.

The Realms node expands to show the `myrealm` security realm, which has been configured for you as part of creating the WebLogic Server 8.1 domain.

Step 2: Configure a WebLogic LDAP Authentication Provider

1. In the left pane of the WebLogic Server 8.1 Administration Console, expand `myrealm` → Providers → Authentication.

The Authentication node expands to show the Authentication providers that have already been configured in the `myrealm` security realm. By default, this includes the WebLogic security providers called `DefaultAuthenticator` and `DefaultIdentityAsserter`.

2. Click on the Authentication node.
3. In the right pane of the WebLogic Server 8.1 Administration Console, click the Configure a New... link that corresponds to the type of LDAP server you have.

For example, if you have an iPlanet (Netscape) LDAP Server, click the Configure a New iPlantAuthenticator... link.

Note: If none of the Configure a New... links correspond to the type of LDAP server you have, see “[Accessing Other LDAP Servers](#)” in *Managing WebLogic Security*.

4. On the General tab, provide a name for the LDAP Authentication provider.
5. Select a value for the Control Flag field.

The Control Flag determines how this LDAP Authentication provider will be used in conjunction with other Authentication providers. Initially, the Control Flag is best left as `OPTIONAL`. For more information, see “[Setting the JAAS Control Flag Attribute](#)” in *Managing WebLogic Security*.

6. Click Create.

The LDAP Authentication provider you configured appears in the list under the Authentication node in the left pane of the WebLogic Server 8.1 Administration Console.

Step 3: Enable Communication Between Your LDAP Server and WebLogic Server 8.1

1. On the `<LDAPServerName>` LDAP tab of the WebLogic Server 8.1 Administration Console, use the information you saved in step 6 of “[Step 1: Verify and Capture Your WebLogic Server 6.x LDAP V2 Security Realm Configuration](#)” on page 3-19 to fill in the following fields:
 - a. Host—the value of `server.host`
 - b. Port—the value of `server.port`
 - c. Principal—the value of `server.principal` (after the first = sign)
 - d. In the Credential/Confirm Credential fields, enter the password for the user that can log into your LDAP server.
2. If your WebLogic Server 6.x instance had SSL enabled, check the SSL Enabled field so there *is* a checkmark in the box.
3. Accept the default values for the Cache Enabled, Cache Size, and Cache TTL fields.

Note: If you prefer, you can enter the values from the 6.x Caching realm you used with your WebLogic Server 6.x LDAP V2 security realm, instead of these defaults.
4. Click Apply.

Warning: Do not yet restart the WebLogic Server 8.1 instance called `myserver`, even if the restart icon is flashing.

Step 4: Specify How Users and Groups are Stored in the LDAP Directory

1. On the Users tab of the WebLogic Server 8.1 Administration Console, use the information you saved in step 6 of “[Step 1: Verify and Capture Your WebLogic Server 6.x LDAP V2 Security Realm Configuration](#)” on page 3-19 to fill in the following fields:

User Base DN—the value of `user.dn`

User From Name Filter—the value of `user.filter` (the default in the WebLogic Server 8.1 Administration Console may already be correct)

Note: Accept the defaults for other fields on the Users tab or fill in appropriate information.
2. Click Apply.

3. On the Groups tab, use the information you saved in step 6 of “[Step 1: Verify and Capture Your WebLogic Server 6.x LDAP V2 Security Realm Configuration](#)” on page 3-19 to fill in the following fields:

Group Base DN—the value of `group.dn`

Group From Name Filter—the value of `group.filter` (the default in the WebLogic Server 8.1 Administration Console may already be correct)

Note: Accept the defaults for other fields on the Groups tab or fill in appropriate information.

4. Click Apply.

Note: You are not required to set values for any fields on the Membership tab nor the Details tab. However, the WebLogic LDAP Authentication providers in WebLogic Server 8.1 have additional features available through these tabs, that you may want to take advantage of, such as dynamic groups. For more information about these features, see “[Configuring an LDAP Authentication Provider](#)” in *Managing WebLogic Security*.

Step 5: Restart the WebLogic Server 8.1 Instance and Verify the Referenced Users and Groups

1. Save a copy of the WebLogic Server 8.1 `config.xml.booted` file.

Note: Because the `config.xml.booted` file is a copy of the `config.xml` that existed *before* you made any changes, saving it allows you to restore the old configuration in case you run into any problems.
2. Stop the WebLogic Server 8.1 instance called `myserver` by following these steps:
 - a. In the left pane of the WebLogic Server 8.1 Administration Console, expand Servers.

The Servers node expands to show the WebLogic Server 8.1 instances that have already been configured in the `mydomain` WebLogic Server 8.1 domain.
 - b. Right-click `myserver`, and select the Start/Stop This Server... option from the menu.
 - c. On the Start/Stop tab, click the Graceful Shutdown of This Server... link.
 - d. Click Yes.
3. Restart the WebLogic Server 8.1 instance called `myserver` by selecting Programs →BEA WebLogic Platform 8.1 →User Projects →domains →mydomain →Start Server.
4. Log back into the WebLogic Server 8.1 Administration Console.

5. In the left pane of the WebLogic Server 8.1 Administration Console, expand Security → Realms → `myrealm`.
6. Click Users.
A table that lists all the users for the `myrealm` security realm appears in the right pane. This table includes the users from your WebLogic Server 6.x LDAP V2 security realm.
7. In the left pane of the WebLogic Server 8.1 Administration Console, click Groups.
A table that lists all the groups for the `myrealm` security realm appears in the right pane. This table includes the groups from your WebLogic Server 6.x LDAP V2 security realm.

Upgrading from a Windows NT Security Realm

This section provides step-by-step instructions for upgrading your WebLogic Server 6.x Windows NT security realm to a WebLogic Server 8.1 security realm. This upgrade causes the users and groups defined in your Windows NT server to be referenced from the `myrealm` security realm, which is the default (active) security realm in WebLogic Server 8.1.

Note: Security policies replace the access control lists (ACLs) and permissions that were used to protect WebLogic resources in WebLogic Server 6.x. Therefore, no ACLs will be referenced from the 8.1 security realm as a result of this upgrade. To learn about re-securing resources in WebLogic Server 8.1, see “[Verify the upgrade of ACLs by:](#)” on [page 3-44](#) and [Securing WebLogic Resources](#).

To upgrade from a WebLogic Server 6.x Windows NT security realm to a WebLogic Server 8.1 security realm:

- “[Step 1: Verify and Capture Your WebLogic Server 6.x Windows NT Security Realm Configuration](#)” on [page 3-24](#)
- “[Step 2: Configure a Realm Adapter Authentication Provider](#)” on [page 3-26](#)
- “[Step 3: Reconfigure Your WebLogic Server 6.x Windows NT Realm in WebLogic Server 8.1 Compatibility Security](#)” on [page 3-27](#)
- “[Step 4: Restart the WebLogic Server 8.1 Instance and Verify the Referenced Users and Groups](#)” on [page 3-29](#)

Step 1: Verify and Capture Your WebLogic Server 6.x Windows NT Security Realm Configuration

1. In the left pane of the WebLogic Server 6.x Administration Console, expand Security.

2. Click Users.

A list of currently defined users for the WebLogic Server 6.x Windows NT security realm appears at the bottom of the right pane. You will be referencing these users from the WebLogic Server 8.1 security realm.

3. In the left pane of the WebLogic Server 6.x Administration Console, click Groups.

A table of currently defined groups for the WebLogic Server 6.x Windows NT security realm appears in the right pane. You will be referencing these groups from the WebLogic Server 8.1 security realm.

4. In the left pane of the WebLogic Server 6.x Administration Console, expand Realms.

The Realms node expands to show the security realms for the WebLogic Server 6.x domain, including the Windows NT security realm.

5. Click the name of the Windows NT security realm.

For example, click `MyNTRealm`.

6. On the Configuration tab, copy the contents of the Primary Domain field to a temporary location, such as a Notepad file.

7. Be sure you know the password for the user that can log into your Windows NT server.

8. Check whether you have SSL enabled by navigating to the Servers → *myserver* → Configuration → SSL tab in the WebLogic Server 6.x Administration Console, where *myserver* is the name of your WebLogic Server 6.x instance.

9. Record the values of any Cache Size and Cache TTL fields in the Windows NT security realm's associated Caching realm by navigating to the subtabs of Security → Caching Realms → *myCachingRealm* → Configuration, where *myCachingRealm* is the name of your Caching realm.

Step 2: Create a WebLogic Server 8.1 Domain and Security Realm

Note: If you are running WebLogic Server 6.x and 8.1 on the same machine, be sure to stop your WebLogic Server 6.x instance before following these instructions.

1. In WebLogic Server 8.1, run the Configuration Wizard to create a WebLogic Server 8.1 domain called `mydomain`, by accepting most of the defaults.

Note: You can accept all the defaults in the Configuration Wizard to create the WebLogic Server 8.1 domain, with the exception of the username and password combination that is used to boot the server. This username and password combination must have

the correct privileges to be able to restart the WebLogic Server 8.1 instance. (It need not be the Administrator account.)

Step-by-step instructions for using the Configuration Wizard are in “Creating and Configuring Domains Using the Configuration Wizard” in *Configuring and Managing WebLogic Server*.

Warning: If you do not use the Configuration Wizard to create your WebLogic Server 8.1 domain, you will not be able to create the Realm Adapter Authentication provider (in “Step 2: Configure a Realm Adapter Authentication Provider” on page 3-26) without first copying over your 6.x `filerealm.properties` file.

2. Start the WebLogic Server 8.1 instance called `myserver`.
3. Log into the WebLogic Server 8.1 Administration Console.
4. In the left pane of the WebLogic Server 8.1 Administration Console, expand Security → Realms.

The Realms node expands to show the `myrealm` security realm, which has been configured for you as part of creating the WebLogic Server 8.1 domain.

Step 2: Configure a Realm Adapter Authentication Provider

Note: For a detailed explanation of the Realm Adapter Authentication provider, see “Configuring a Realm Adapter Authentication Provider” in *Managing WebLogic Security*.

- In the left pane of the WebLogic Server 8.1 Administration Console, expand `myrealm` → Providers → Authentication.

The Authentication node expands to show the Authentication providers that have already been configured in the `myrealm` security realm. By default, this includes the WebLogic security providers called `DefaultAuthenticator` and `DefaultIdentityAsserter`.
- Click on the Authentication node.
- In the right pane of the WebLogic Server 8.1 Administration Console, click the Configure a New Realm Adapter Authenticator... link.
- On the General tab, provide a name for the Realm Adapter Authentication provider.
- If it is not already selected, set the Control Flag field to `OPTIONAL`.

The Control Flag determines how this Realm Adapter Authentication provider will be used in conjunction with other Authentication providers. Initially, the Control Flag is best left as

OPTIONAL. For more information, see [“Setting the JAAS Control Flag Attribute”](#) in *Managing WebLogic Security*.

6. Click Create.

The General tab updates to show a Types chooser, which allows you to select token types for an Authentication provider that includes an Identity Assertion provider.

Warning: Do not select any token types from the Types chooser. The WebLogic Authentication provider (called DefaultAuthenticator in the WebLogic Server 8.1 Administration Console) is already configured to handle a specified token type. The Realm Adapter Authenticator includes an Identity Assertion provider, which if configured to handle the same token type as the WebLogic Authentication provider, will render the server unbootable.

The Realm Adapter Authentication provider you configured appears in the list under the Authentication node in the left pane of the WebLogic Server 8.1 Administration Console.

Step 3: Reconfigure Your WebLogic Server 6.x Windows NT Realm in WebLogic Server 8.1 Compatibility Security

1. Save a copy of the WebLogic Server 8.1 `config.xml.booted` file.

Note: Because the `config.xml.booted` file is a copy of the `config.xml` that existed *before* you made any changes, saving it allows you to restore the old configuration in case you run into any problems.

2. Stop the WebLogic Server 8.1 instance called `myserver` by following these steps:

- a. In the left pane of the WebLogic Server 8.1 Administration Console, expand Servers.
The Servers node expands to show the WebLogic Server 8.1 instances that have already been configured in the `mydomain` WebLogic Server 8.1 domain.
- b. Right-click `myserver`, and select the Start/Stop This Server... option from the menu.
- c. On the Start/Stop tab, click the Graceful Shutdown of This Server... link.
- d. Click Yes.

3. Restart the WebLogic Server 8.1 instance called `myserver` by selecting Programs →BEA WebLogic Platform 8.1 →User Projects →domains →mydomain →Start Server.

4. Log back into the WebLogic Server 8.1 Administration Console.

The Compatibility Security node appears in the left pane of the WebLogic Server 8.1 Administration Console.

5. In the left pane of the WebLogic Server 8.1 Administration Console, expand Compatibility Security.
6. Click Realms.
7. In the right pane of the WebLogic Server 8.1 Administration Console, click the Configure a New NT Realm (Deprecated)... link.

Re-configuring your WebLogic Server 6.x Windows NT realm in Compatibility security is what provides the connection to the Realm Adapter Authentication provider and allows you to view your 6.x users and groups in the WebLogic Server 8.1 Administration Console.

8. On the General tab, use the information you saved in step 6 of [“Step 1: Verify and Capture Your WebLogic Server 6.x Windows NT Security Realm Configuration”](#) on page 3-24 to:
 - Provide a name for the Windows NT realm.
 - Provide the primary domain of your original, 6.x Windows NT realm.

Note: If entering multiple sets of host numbers, delimit them with commas. If the local computer (where WebLogic Server executes) is the Primary Controller, you can just enter a period (“.”).

9. Click Create.
10. In the left pane of the WebLogic Server 8.1 Administration Console, click Caching Realms.
11. In the right pane of the WebLogic Server 8.1 Administration Console, select the Configure a New Caching Realm... link.
12. On the General tab:
 - Provide a name for the Caching realm.
 - Select the name of the Windows NT realm as the Basic Realm.
 - Click Create.

13. In the left pane of the WebLogic Server 8.1 Administration Console, click Compatibility Security.

14. Select the Compatibility tab.

15. On the File Realm tab:
 - Select the name of the Caching realm you created in step 12 from the Caching Realm drop-down menu.
 - Click Apply.

Step 4: Restart the WebLogic Server 8.1 Instance and Verify the Referenced Users and Groups

1. Save a copy of the WebLogic Server 8.1 `config.xml.booted` file.

Note: Because the `config.xml.booted` file is a copy of the `config.xml` that existed *before* you made any changes, saving it allows you to restore the old configuration in case you run into any problems.
2. Stop the WebLogic Server 8.1 instance called `myserver` by following these steps:
 - a. In the left pane of the WebLogic Server 8.1 Administration Console, expand Servers.

The Servers node expands to show the WebLogic Server 8.1 instances that have already been configured in the `mydomain` WebLogic Server 8.1 domain.
 - b. Right-click `myserver`, and select the Start/Stop This Server... option from the menu.
 - c. On the Start/Stop tab, click the Graceful Shutdown of This Server... link.
 - d. Click Yes.
3. Restart the WebLogic Server 8.1 instance called `myserver` by selecting Programs → BEA WebLogic Platform 8.1 → User Projects → `domains` → `mydomain` → Start Server.
4. Log back into the WebLogic Server 8.1 Administration Console.
5. In the left pane of the WebLogic Server 8.1 Administration Console, expand Compatibility Security.
6. Click Users.

The users from your WebLogic Server 6.x Windows NT security realm appear at the bottom of the right pane.
7. In the left pane of the WebLogic Server 8.1 Administration Console, click Groups.

A table that lists all the groups for the `CompatibilityRealm` security realm appears in the right pane. This table includes the groups from your WebLogic Server 6.x Windows NT security realm.

Upgrading from a UNIX Security Realm

This section provides step-by-step instructions for upgrading your WebLogic Server 6.x Unix security realm to a WebLogic Server 8.1 security realm. This upgrade causes the users and groups

defined in your Unix server to be referenced from the `myrealm` security realm, which is the default (active) security realm in WebLogic Server 8.1.

Note: Security policies replace the access control lists (ACLs) and permissions that were used to protect WebLogic resources in WebLogic Server 6.x. Therefore, no ACLs will be referenced from the WebLogic Server 8.1 security realm as a result of this upgrade. To learn about re-securing resources in WebLogic Server 8.1, see [“Verify the upgrade of ACLs by:” on page 3-44](#) and [Securing WebLogic Resources](#).

To upgrade from a WebLogic Server 6.x Unix security realm to a WebLogic Server 8.1 security realm:

- [“Step 1: Verify and Capture Your WebLogic Server 6.x Unix Security Realm Configuration” on page 3-30](#)
- [“Step 2: Configure a Realm Adapter Authentication Provider” on page 3-26](#)
- [“Step 3: Reconfigure Your WebLogic Server 6.x Unix Realm in WebLogic Server 8.1 Compatibility Security” on page 3-32](#)
- [“Step 4: Restart the WebLogic Server 8.1 Instance and Verify the Referenced Users and Groups” on page 3-29](#)

Step 1: Verify and Capture Your WebLogic Server 6.x Unix Security Realm Configuration

1. In the left pane of the WebLogic Server 6.x Administration Console, expand Security.
2. Click Users.

A list of currently defined users for the WebLogic Server 6.x Unix security realm appears at the bottom of the right pane. You will be referencing these users from the WebLogic Server 8.1 security realm.

3. In the left pane of the WebLogic Server 6.x Administration Console, click Groups.

A table of currently defined groups for the WebLogic Server 6.x Unix security realm appears in the right pane. You will be referencing these groups from the WebLogic Server 8.1 security realm.

4. In the left pane of the WebLogic Server 6.x Administration Console, expand Realms.

The Realms node expands to show the security realms for the WebLogic Server 6.x domain, including the Unix security realm.

5. Click the name of the Unix security realm.

For example, click `MyUnixRealm`.

6. On the Configuration tab, copy the contents of the Auth Program field to a temporary location, such as a Notepad file.
7. Be sure you know the password for the user that can log into your Unix server.
8. Check whether you have SSL enabled by navigating to the Servers → `myserver` → Configuration → SSL tab in the WebLogic Server 6.x Administration Console, where `myserver` is the name of your WebLogic Server 6.x instance.
9. Record the values of any Cache Size and Cache TTL fields in the Unix security realm's associated Caching realm by navigating to the subtabs of Security → Caching Realms → `myCachingRealm` → Configuration, where `myCachingRealm` is the name of your Caching realm.

Step 2: Create a WebLogic Server 8.1 Domain and Security Realm

Note: If you are running WebLogic Server 6.x and 8.1 on the same machine, be sure to stop your WebLogic Server 6.x instance before following these instructions.

1. In WebLogic Server 8.1, run the Configuration Wizard to create a WebLogic Server 8.1 domain called `mydomain`, by accepting most of the defaults.

Note: You can accept all the defaults in the Configuration Wizard to create the WebLogic Server 8.1 domain, but you must be logged into a Unix machine.

Step-by-step instructions for using the Configuration Wizard are in “Creating and Configuring Domains Using the Configuration Wizard” in *Configuring and Managing WebLogic Server*.

Warning: If you do not use the Configuration Wizard to create your WebLogic Server 8.1 domain, you will not be able to create the Realm Adapter Authentication provider (in “Step 2: Configure a Realm Adapter Authentication Provider” on page 3-26) without first copying over your `6.x filerealm.properties` file.

2. Start the WebLogic Server 8.1 instance called `myserver`.
3. Log in to the WebLogic Server 8.1 Administration Console.
4. In the left pane of the WebLogic Server 8.1 Administration Console, expand Security → Realms.

The Realms node expands to show the `myrealm` security realm, which has been configured for you as part of creating the WebLogic Server 8.1 domain.

Step 2: Configure a Realm Adapter Authentication Provider

Note: For a detailed explanation of the Realm Adapter Authentication provider, see [“Configuring a Realm Adapter Authentication Provider”](#) in *Managing WebLogic Security*.

1. In the left pane of the WebLogic Server 8.1 Administration Console, expand `myrealm` → Providers → Authentication.

The Authentication node expands to show the Authentication providers that have already been configured in the `myrealm` security realm. By default, this includes the WebLogic security providers called `DefaultAuthenticator` and `DefaultIdentityAsserter`.

2. Click on the Authentication node.
3. In the right pane of the WebLogic Server 8.1 Administration Console, click the [Configure a New Realm Adapter Authenticator...](#) link.
4. On the General tab, provide a name for the Realm Adapter Authentication provider.
5. If it is not already selected, set the Control Flag field to `OPTIONAL`.

The Control Flag determines how this Realm Adapter Authentication provider will be used in conjunction with other Authentication providers. Initially, the Control Flag is best left as `OPTIONAL`. For more information, see [“Setting the JAAS Control Flag Attribute”](#) in *Managing WebLogic Security*.

6. Click Create.

The General tab updates to show a Types chooser, which allows you to select token types for an Authentication provider that includes an Identity Assertion provider.

Warning: Do not select any token types from the Types chooser. The WebLogic Authentication provider (called `DefaultAuthenticator` in the WebLogic Server 8.1 Administration Console) is already configured to handle a specified token type. The Realm Adapter Authenticator includes an Identity Assertion provider, which if configured to handle the same token type, will render the server unbootable.

The Realm Adapter Authentication provider you configured appears in the list under the Authentication node in the left pane of the WebLogic Server 8.1 Administration Console.

Step 3: Reconfigure Your WebLogic Server 6.x Unix Realm in WebLogic Server 8.1 Compatibility Security

1. Save a copy of the WebLogic Server 8.1 `config.xml.booted` file.

Note: Because the `config.xml.booted` file is a copy of the `config.xml` that existed *before* you made any changes, saving it allows you to restore the old configuration in case you run into any problems.

2. Stop the WebLogic Server 8.1 instance called `myserver` by following these steps:
 - a. In the left pane of the WebLogic Server 8.1 Administration Console, expand Servers.
The Servers node expands to show the WebLogic Server 8.1 instances that have already been configured in the `mydomain` WebLogic Server 8.1 domain.
 - b. Right-click `myserver`, and select the Start/Stop This Server... option from the menu.
 - c. On the Start/Stop tab, click the Graceful Shutdown of This Server... link.
 - d. Click Yes.
3. Restart the WebLogic Server 8.1 instance called `myserver` by selecting Programs →BEA WebLogic Platform 8.1 →User Projects →domains →mydomain →Start Server.
4. Log back into the WebLogic Server 8.1 Administration Console.
The Compatibility Security node appears in the left pane of the WebLogic Server 8.1 Administration Console.
5. In the left pane of the WebLogic Server 8.1 Administration Console, expand Compatibility Security.
6. Click Realms.
7. In the right pane of the WebLogic Server 8.1 Administration Console, click the Configure a New Unix Realm (Deprecated)... link.
Re-configuring your Unix realm in Compatibility security is what provides the connection to the Realm Adapter Authentication provider and allows you to view your 6.x users and groups in the WebLogic Server 8.1 Administration Console.
8. On the Configuration tab, use the information you saved in step 6 of “[Step 1: Verify and Capture Your WebLogic Server 6.x Windows NT Security Realm Configuration](#)” on [page 3-24](#) to:
 - Provide a name for the Unix realm.
 - Provide the name of the authentication program used in your original, 6.x Unix realm.

Note: `wlauth` is the name of a small native program provided by default as part of WebLogic Server. `wlauth` is executed to look up users and groups and to authenticate users on the basis of their Unix login names and passwords.

9. Click Create.
10. In the left pane of the WebLogic Server 8.1 Administration Console, click Caching Realms.
11. In the right pane of the WebLogic Server 8.1 Administration Console, select the Configure a New Caching Realm... link.
12. On the General tab:
 - Provide a name for the Caching realm.
 - Select the name of the Unix realm as the Basic Realm.
 - Click Create.
13. In the left pane of the WebLogic Server 8.1 Administration Console, click Compatibility Security.
14. Select the Compatibility tab.
15. On the File Realm tab:
 - Select the name of the Caching realm you created in step 12 from the Caching Realm drop-down menu.
 - Click Apply.

Step 4: Restart the WebLogic 8.1 Server Instance and Verify the Referenced Users and Groups

1. Save a copy of the WebLogic Server 8.1 `config.xml.booted` file.

Note: Because the `config.xml.booted` file is a copy of the `config.xml` that existed *before* you made any changes, saving it allows you to restore the old configuration in case you run into any problems.
2. Stop the WebLogic Server 8.1 instance called `myserver` by following these steps:
 - a. In the left pane of the WebLogic Server 8.1 Administration Console, expand Servers.

The Servers node expands to show the WebLogic Server 8.1 instances that have already been configured in the `mydomain` WebLogic Server 8.1 domain.
 - b. Right-click `myserver`, and select the Start/Stop This Server... option from the menu.

- c. On the Start/Stop tab, click the Graceful Shutdown of This Server... link.
 - d. Click Yes.
3. Restart the WebLogic Server 8.1 instance called `myserver` by selecting Programs →BEA WebLogic Platform 8.1 →User Projects →domains →mydomain →Start Server.
 4. Log back into the WebLogic Server 8.1 Administration Console.
 5. In the left pane of the WebLogic Server 8.1 Administration Console, expand Compatibility Security.
 6. Click Users.

The users from your WebLogic Server 6.x Unix security realm appear at the bottom of the right pane.
 7. In the left pane of the WebLogic Server 8.1 Administration Console, click Groups.

A table that lists all the groups for the `CompatibilityRealm` security realm appears in the right pane. This table includes the groups from your WebLogic Server 6.x Unix security realm.

Upgrading from an RDBMS Security Realm

Notes: The instructions in this section illustrate how to upgrade the RDBMS example that was provided with WebLogic Server 6.x. The RDBMS example utilized a Cloudscape database. Customers who have modified this example or use other databases may need to make some modifications to these instructions for their environment.

The instructions in this section provide step-by-step instructions for upgrading your WebLogic Server 6.x RDBMS security realm to a WebLogic Server 8.1 security realm. This upgrade causes the users and groups defined in your Cloudscape database to be referenced from the `myrealm` security realm, which is the default (active) security realm in WebLogic Server 8.1.

Note: Security policies replace the access control lists (ACLs) and permissions that were used to protect WebLogic resources in WebLogic Server 6.x. Therefore, no ACLs will be referenced from the 8.1 security realm as a result of this upgrade. To learn about re-securing resources in WebLogic Server 8.1, see [Securing WebLogic Resources](#).

To upgrade from a WebLogic Server 6.x RDBMS security realm to a WebLogic Server 8.1 security realm:

- [“Step 1: Verify and Capture Your WebLogic Server 6.x RDBMS Security Realm Configuration” on page 3-36](#)

- “Step 2: Recompile the WebLogic Server 6.x RDBMS Realm Code for Use in WebLogic Server 8.1” on page 3-37
- “Step 3: Configure a Realm Adapter Authentication Provider” on page 3-38
- “Step 4: Modify the WebLogic Server 8.1 Start Script” on page 3-39
- “Step 5: Reconfigure Your WebLogic Server 6.x RDBMS Realm in WebLogic Server 8.1 Compatibility Security” on page 3-40
- “Step 6: Restart the WebLogic Server 8.1 Instance and Verify the Referenced Users and Groups” on page 3-41

Step 1: Verify and Capture Your WebLogic Server 6.x RDBMS Security Realm Configuration

1. In the left pane of the WebLogic Server 6.x Administration Console, expand Security.
2. Click Users.

A list of currently defined users for the WebLogic Server 6.x RDBMS security realm appears at the bottom of the right pane. These are the users that will be referenced from the WebLogic Server 8.1 security realm.
3. In the left pane of the WebLogic Server 6.x Administration Console, click Groups.

A table of currently defined groups for the WebLogic Server 6.x RDBMS security realm appears in the right pane. These are the groups that will be referenced from the WebLogic Server 8.1 security realm.
4. In the left pane of the WebLogic Server 6.x Administration Console, expand Realms.

The Realms node expands to show the security realms for the WebLogic Server 6.x domain, including the RDBMS security realms.
5. Click the name of an RDBMS security realm.

For example, click `DefaultRDBMSRealmForCloudscape`.
6. On the Configuration tab, copy the following information to a temporary location, such as a Notepad file:
 - a. On the General tab, the contents of the Name and Realm Class fields.

Note: The out-of-the-box RDBMS example used `examples.security.rdbmsrealm.RDBMSRealm` as the value for the Realm Class.

- b. On the Database tab, the contents of the Driver, URL, and User Name and Password fields, if applicable.

Note: The out-of-the-box RDBMS example used
`COM.cloudscape.core.JDBCdriver` as the value for the Driver, and
`jdbc:cloudscape:demo;create=true;autocommit=false` as the value for
the URL.

- c. On the Schema tab, the contents of the Schema Properties field.

For example:

```
getGroupMembers=SELECT GM_GROUP, GM_MEMBER from groupmembers WHERE
GM_GROUP = ? deleteGroup2=DELETE FROM aclentries WHERE A_PRINCIPAL
= ? deleteGroup1=DELETE FROM groupmembers WHERE GM_GROUP = ?
addGroupMember=INSERT INTO groupmembers VALUES ( ? , ? )
getUser=SELECT U_NAME, U_PASSWORD FROM users WHERE U_NAME = ?
getPermission=SELECT DISTINCT A_PERMISSION FROM aclentries WHERE
A_PERMISSION = ? deleteUser3=DELETE FROM aclentries WHERE
A_PRINCIPAL = ? getGroupNewStatement=true deleteUser2=DELETE FROM
groupmembers WHERE GM_MEMBER = ? deleteUser1=DELETE FROM users
WHERE U_NAME = ? getAcls=SELECT A_NAME, A_PRINCIPAL, A_PERMISSION
FROM aclentries ORDER BY A_NAME, A_PRINCIPAL getUsers=SELECT
U_NAME, U_PASSWORD FROM users getGroups=SELECT GM_GROUP, GM_MEMBER
FROM groupmembers getPermissions=SELECT DISTINCT A_PERMISSION FROM
aclentries getAclEntries=SELECT A_NAME, A_PRINCIPAL, A_PERMISSION
FROM aclentries WHERE A_NAME = ? ORDER BY A_PRINCIPAL
newUser=INSERT INTO users VALUES ( ? , ? ) removeGroupMember=DELETE
FROM groupmembers WHERE GM_GROUP = ? AND GM_MEMBER = ?
```

Step 2: Recompile the WebLogic Server 6.x RDBMS Realm Code for Use in WebLogic Server 8.1

The RDBMS realm sample included in WebLogic Server 6.x at

`WL_HOME\samples\examples\security\rdbmsrealm` used a private Admin API
(`weblogic.management.Admin.getActiveDomain()`) in the `RDBMSDelegate(RDBMSRealm
realm)` method of the `RDBMSDelegate.java` file. This private Admin API has been replaced
with a public Admin API in WebLogic Server 8.1. Therefore, customers must:

1. Modify the `RDBMSDelegate.java` file so that the RDBMS realm sample code uses the public Admin API rather than the private one. For more information, see [MBean API Change](#).
2. Save the `RDBMSDelegate.java` file.
3. Run the `setenv` script provided with WebLogic Server 8.1.

4. Recompile all the RDBMS realm sample code directly from the 6.x `WL_HOME\samples\examples\ directory`, using `javac`.

Step 3: Create a WebLogic Server 8.1 Domain and Security Realm

Note: If you are running WebLogic Server 6.x and 8.1 on the same machine, be sure to stop your WebLogic Server 6.x instance before following these instructions.

1. In WebLogic Server 8.1, run the Configuration Wizard to create a WebLogic Server 8.1 domain called `mydomain`, by accepting all of the defaults.

For step-by-step instructions for the Configuration Wizard, see “Creating and Configuring Domains Using the Configuration Wizard” in *Configuring and Managing WebLogic Server*.

2. Start the WebLogic Server 8.1 instance called `myserver`.
3. Log into the WebLogic Server 8.1 Administration Console.
4. In the left pane of the WebLogic Server 8.1 Administration Console, expand Security → Realms.

The Realms node expands to show the `myrealm` security realm, which as been configured for you as part of creating the WebLogic Server 8.1 domain.

Step 3: Configure a Realm Adapter Authentication Provider

Note: For a detailed explanation of the Realm Adapter Authentication provider, see “Configuring a Realm Adapter Authentication Provider” in *Managing WebLogic Security*.

1. In the left pane of the WebLogic Server 8.1 Administration Console, expand `myrealm` → Providers → Authentication.

The Authentication node expands to show the Authentication providers that have already been configured in the `myrealm` security realm. By default, this includes the WebLogic security providers called `DefaultAuthenticator` and `DefaultIdentityAsserter`.

2. Click on the Authentication node.
3. In the right pane of the WebLogic Server 8.1 Administration Console, select the Configure a New Realm Adapter Authenticator... link.
4. On the General tab, provide a name for the Realm Adapter Authentication provider.
5. If it is not already selected, set the Control Flag field to `OPTIONAL`.

The Control Flag determines how this Realm Adapter Authentication provider will be used in conjunction with other Authentication providers. Initially, the Control Flag is best left as `OPTIONAL`. For more information, see [“Setting the JAAS Control Flag Attribute”](#) in *Managing WebLogic Security*.

6. Click Create.

The General tab updates to show a Types chooser, which allows you to select token types for an Authentication provider that includes an Identity Assertion provider.

Warning: Do not select any token types from the Types chooser. The WebLogic Identity Assertion provider (called `DefaultAuthenticator` in the WebLogic Server 8.1 Administration Console) is already configured to handle a specified token type. The Realm Adapter Authenticator includes an Identity Assertion provider, which if configured to handle the same token type, will render the server unbootable.

The Realm Adapter Authentication provider you configured appears in the list under the Authentication node in the left pane of the WebLogic Server 8.1 Administration Console.

Step 4: Modify the WebLogic Server 8.1 Start Script

1. Stop the WebLogic Server 8.1 instance called `myserver` by following these steps:

a. In the left pane of the WebLogic Server 8.1 Administration Console, expand Servers.

The Servers node expands to show the WebLogic Server 8.1 instances that have already been configured in the `mydomain` WebLogic Server 8.1 domain.

b. Right-click `myserver`, and select the Start/Stop This Server... option from the menu.

c. On the Start/Stop tab, click the Graceful Shutdown of This Server... link.

d. Click Yes.

2. Open the `StartWebLogic.cmd/sh` file in a text editor and update the `CLASSPATH` to include paths to:

- The 6.x RDBMS realm code you re-compiled in [“Step 2: Recompile the WebLogic Server 6.x RDBMS Realm Code for Use in WebLogic Server 8.1”](#) on page 3-37.
- The WebLogic Server 6.x `cloudscape.jar` file, which is installed under `WL_HOME\samples\eval\cloudscape\lib`.

3. Save the modified `StartWebLogic.cmd/sh` file.

Step 5: Reconfigure Your WebLogic Server 6.x RDBMS Realm in WebLogic Server 8.1 Compatibility Security

1. Save a copy of the WebLogic Server 8.1 `config.xml.booted` file.
Note: Because the `config.xml.booted` file is a copy of the `config.xml` that existed *before* you made any changes, saving it allows you to restore the old configuration in case you run into any problems.
2. Stop the WebLogic Server 8.1 instance called `myserver` by following these steps:
 - a. In the left pane of the WebLogic Server 8.1 Administration Console, expand Servers.
The Servers node expands to show the WebLogic Server 8.1 instances that have already been configured in the `mydomain` WebLogic Server 8.1 domain.
 - b. Right-click `myserver`, and select the Start/Stop This Server... option from the menu.
 - c. On the Start/Stop tab, click the Graceful Shutdown of This Server... link.
 - d. Click Yes.
3. Restart the WebLogic Server 8.1 instance called `myserver` by selecting Programs → BEA WebLogic Platform 8.1 → User Projects → ~~domains~~ → ~~mydomain~~ → Start Server.
4. Log back into the WebLogic Server 8.1 Administration Console.
The Compatibility Security node appears in the left pane of the WebLogic Server 8.1 Administration Console.
5. In the left pane of the WebLogic Server 8.1 Administration Console, expand Compatibility Security.
6. Click Realms.
7. In the right pane of the WebLogic Server 8.1 Administration Console, select the Configure a New RDBMS Realm (Deprecated)... link.
Re-configuring your RDBMS realm in Compatibility security is what provides the connection to the Realm Adapter Authentication provider and allows you to view your 6.x users and groups in the WebLogic Server 8.1 Administration Console.
8. On the Configuration tab:
 - Provide a name for the RDBMS realm.
 - Click Create.

9. In the left pane of the WebLogic Server 8.1 Administration Console, click Caching Realms.
10. In the right pane of the WebLogic Server 8.1 Administration Console, select the Configure a New Caching Realm... link.
11. On the General tab:
 - Provide a name for the Caching realm.
 - Select the name of the RDBMS realm as the Basic Realm.
 - Click Create.
12. In the left pane of the WebLogic Server 8.1 Administration Console, click Compatibility Security.
13. Select the Compatibility tab.
14. On the File Realm tab:
 - Select the name of the Caching realm you created in step 11 from the Caching Realm drop-down menu.
 - Click Apply.

Step 6: Restart the WebLogic Server 8.1 Instance and Verify the Referenced Users and Groups

1. Restart the WebLogic Server 8.1 instance called `myserver` by selecting Programs →BEA WebLogic Platform 8.1 →User Projects →domains →mydomain →Start Server.
2. Log into the WebLogic Server 8.1 Administration Console.
3. In the left pane of the WebLogic Server 8.1 Administration Console, expand Compatibility Security.
4. Click Users.

The users from your WebLogic Server 6.x RDBMS security realm appear at the bottom of the right pane.
5. In the left pane of the WebLogic Server 8.1 Administration Console, click Groups.

A table that lists all the groups for the `CompatibilityRealm` security realm appears in the right pane. This table includes the groups from your WebLogic Server 6.x RDBMS security realm.

Upgrading from a Custom Security Realm

WebLogic Server 6.x Custom security realms can only be used in Compatibility security. The following list provides advice for common Custom security realm upgrade situations:

- [“You Have a 5.x weblogic.properties File Instead of a 6.x config.xml File”](#) on page 3-42
- [“You Upgraded a WebLogic Server 6.x Domain “In Place””](#) on page 3-42

You Have a 5.x weblogic.properties File Instead of a 6.x config.xml File

If your Custom security realm has a `weblogic.properties` file rather than a `config.xml` file, you will not be able to boot a WebLogic Server 8.1 instance in Compatibility security until you convert this file. For instructions on performing this conversion and more information about the conversion, see [“Converting the weblogic.properties File to .xml files”](#) and [“weblogic.properties Mapping Table”](#) in *Migrating WebLogic Server 4.5 and 5.1 Applications to 6.x*, respectively.

WebLogic Server 8.1 requires changes to WebLogic Server 5.x Custom realm code to make it compatible with WebLogic Server 6.x:

- Implement the `BasicRealm.getUser(String)` or `BasicRealm.getUser(UserInfo)` method in your `CustomRealm` class. This method should return `null` for `system`, `everyone` and `Administrators`.
- Implement the `BasicRealm.getAcl()` method to ignore requests (that is, return `null`) for ACL names starting with `weblogic.server`. (For example, for an ACL name starting with `weblogic.server.myserver`.) The return value of `null` indicates to a WebLogic Server 6.x instance that it should boot by ignoring the ACLs that start with this prefix.

Warning: Failure to meet the above requirements means that your server may not boot.

You Upgraded a WebLogic Server 6.x Domain “In Place”

If you upgrade a WebLogic Server 6.x domain “in place” (that is, you use the same domain directory from 6.x and upgrade it to run under 8.1 by changing start scripts, classpaths, and so on as described in [“Upgrading Your WebLogic Server 6.x Domain to Version 8.1”](#) on page 3-3, WebLogic Server 8.1 will automatically be configured for Compatibility security. If this describes your upgrade situation, the only steps necessary to complete your security upgrade are:

- The recompilation of the 6.x realm code against WebLogic Server 8.1.
- Use of the new MBean API described in [MBean API Change](#).

An example of instructions for these steps (using the RDBMS realm) are available in [“Step 2: Recompile the WebLogic Server 6.x RDBMS Realm Code for Use in WebLogic Server 8.1”](#) on page 3-37.

Access Control Lists (ACLs)

In WebLogic Server 6.x, an ACL defined the permissions by which a user could interact with a WebLogic resource. ACLs were used to protect both MBeans and other types of WebLogic resources. This section contains the following information about the state of ACLs:

- [“ACLs on MBeans”](#) on page 3-43
- [“ACLs That Protected WebLogic Resources”](#) on page 3-43

ACLs on MBeans

ACLs on MBeans are not supported in WebLogic Server 8.1 and therefore cannot be upgraded to this version of WebLogic Server. ACLs on MBeans are replaced by security role protections. For information about MBean security role protections, see [“Protected MBean Attributes and Operations”](#) in *Securing WebLogic Resources*.

ACLs That Protected WebLogic Resources

If you are running WebLogic Server 8.x in Compatibility security, any out-of-the-box ACLs that protected resources in WebLogic Server 6.x are represented by default security policies. If you have created additional ACLs (except those on MBeans), these will remain ACLs and continue to protect the WebLogic resource. In other words, when you run WebLogic Server 8.1 in Compatibility security, you may have a mixture of 6.x ACLs and 8.1 security roles/security policies. See [“Default Security Policies”](#) in *Securing WebLogic Resources* for more information.

Note: Be sure you have upgraded your WebLogic Server 6.x domain to a WebLogic Server 8.1 domain and have upgraded your WebLogic Server 6.x security realm to a WebLogic Server 8.1 security realm before following these instructions. For instructions, see [“Upgrading Your WebLogic Server 6.x Domain to Version 8.1”](#) on page 3-3 and [“Upgrading Your WebLogic Server 6.x Security Realms to 8.1”](#) on page 3-18.

1. If it is not already running, start the WebLogic Server 8.1 instance called `myserver` by selecting Programs →BEA WebLogic Platform 8.1 →User Projects →domains →mydomain →Start Server.
2. Log into the WebLogic Server 8.1 Administration Console.

3. In the left pane of the WebLogic Server 8.1 Administration Console, expand Compatibility Security →ACLs.

The Realm Adapter Authentication provider configured in Compatibility Security allows you to see the ACLs defined in WebLogic Server 6.x security configurations. However, it is important to understand that because out-of-the-box ACLs protecting WebLogic resources are no longer used in WebLogic Server 8.1, these ACLs are *not* being enforced.

4. Verify the upgrade of ACLs by:
 - a. Right-clicking a WebLogic resource in the left pane of the WebLogic Server 8.1 Administration Console. For example, right-click Servers.

For more information about WebLogic resources, see [“Types of WebLogic Resources”](#) in *Securing WebLogic Resources*.
 - b. Select the Define Security Policy... option.
 - c. In the Policy Statement list box of the displayed Policy Editor page, you see the upgraded security policy for the WebLogic resource.

If the ACL stated, for example, that the group `Administrators` had permission to unlock servers, the Policy Statement on the Policy Editor page for ALL the Server resource’s methods would show: `Caller is Granted the Role Admin`. This is because in WebLogic Server 8.1, the `Administrators` group is granted the `Admin` security role.

Note: All out-of-the-box ACLs protecting WebLogic resources are upgraded to security policies that use global roles. However, in WebLogic Server 8.1, there are also scoped roles, which you may find useful for securing your WebLogic resources. For more information, see [“Types of Security Roles: Global Roles and Scoped Roles”](#) in *Securing WebLogic Resources*.

- d. Repeat steps a - c for each ACL you want to verify.

Note: The ACLs and permissions that have been upgraded to Weblogic Server 8.x from WebLogic Server 6.x are those listed in the `filerealm.properties` files from the `examples` and `petstore` domains. These are also listed in [Listing 3-1](#) and [Listing 3-2](#).

Listing 3-1 `fileRealm.properties` from 6.x examples Domain

```
acl.modify.weblogic.jndi.weblogic.fileSystem=everyone
user.j2ee=0xe22513c99d9279bdf9318894033ef310b9aa830f
```

```
acl.lookup.weblogic.jndi.weblogic.fileSystem=everyone
acl.lookup.weblogic.jndi.weblogic.ejb=system,everyone
acl.lookup.weblogic.jndi=system,everyone
acl.list.weblogic.jndi.weblogic.rmi=system,everyone
acl.lockServer.weblogic.admin=system,guest
acl.shutdown.weblogic.admin=system,guest
group.gold=
acl.boot.weblogic.server=system,everyone
user.jps_admin=0xcc0b7594b451623dd59a3db8148de6984c60ac74
acl.list.weblogic.jndi.weblogic.fileSystem=everyone
acl.lookup.weblogic.jndi.weblogic=system,everyone
acl.modify.weblogic.jndi.weblogic.rmi=system,everyone
acl.modify.weblogic.jndi.weblogic=system,everyone
acl.list.weblogic.jndi.weblogic=system,everyone
acl.lookup.weblogic.management=system,everyone
group.cust=j2ee
acl.modify.weblogic.management=system,everyone
acl.findMBeans.weblogic.admin=Administrators
group.Administrators=system
group.admin=jps_admin
acl.lookup.weblogic.jndi.weblogic.rmi=system,everyone
acl.list.weblogic.jndi=system,everyone
acl.modify.weblogic.admin.acl=system,guest
acl.list.weblogic.jndi.weblogic.ejb=system,everyone
acl.modify.weblogic.jndi=system,everyone
acl.write.managedObject=system,guest
acl.reserve.weblogic.jdbc.connectionPool.demopool=system,everyone
acl.read.managedObject=system,guest
acl.admin.weblogic.jdbc.connectionPool.create=system,guest
user.system=0xc4d441a2bdd9c4bba6029d63066781512326d355
acl.unlockServer.weblogic.admin=system,guest
acl.reset.weblogic.jdbc.connectionPool=system
acl.execute.weblogic.servlet=everyone
acl.modify.weblogic.jndi.weblogic.ejb=system,everyone
```

Listing 3-2 fileRealm.properties from 6.x petstore Domain

```
acl.modify.weblogic.jndi.weblogic.fileSystem=everyone
acl.lookup.weblogic.jndi.weblogic.fileSystem=everyone
user.support=0xa227185fac962e564de4796154a80757860e32d4
acl.lookup.weblogic.jndi.weblogic.ejb=system,newdbuSer,NEWDBUSER,everyone
acl.lookup.weblogic.jndi=system,newdbuSer,NEWDBUSER,everyone
acl.list.weblogic.jndi.weblogic.rmi=system,newdbuSer,NEWDBUSER,everyone
acl.create.weblogic.jms.ConnectionConsumer=guest
acl.lockServer.weblogic.admin=system,newdbuSer,NEWDBUSER,guest
acl.reserve.weblogic.jdbc.connectionPool.oraclepool=system,newdbuSer,NEWDB
USER,everyone
acl.shutdown.weblogic.admin=system,newdbuSer,NEWDBUSER,guest
acl.boot.weblogic.server=vlatas,system,newdbuSer,NEWDBUSER,everyone
acl.list.weblogic.jndi.weblogic.fileSystem=everyone
acl.lookup.weblogic.jndi.weblogic=system,newdbuSer,NEWDBUSER,everyone
acl.modify.weblogic.jndi.weblogic.rmi=system,newdbuSer,NEWDBUSER,everyone
acl.modify.weblogic.jndi.weblogic=system,newdbuSer,NEWDBUSER,everyone
acl.list.weblogic.jndi.weblogic=system,newdbuSer,NEWDBUSER,everyone
acl.lookup.weblogic.management=system,newdbuSer,NEWDBUSER,everyone
acl.modify.weblogic.management=system,newdbuSer,NEWDBUSER,everyone
acl.findMBeans.weblogic.admin=Administrators
group.Administrators=system
acl.lookup.weblogic.jndi.weblogic.rmi=system,newdbuSer,NEWDBUSER,everyone
acl.list.weblogic.jndi=system,newdbuSer,NEWDBUSER,everyone
acl.modify.weblogic.admin.acl=vlatas,system,newdbuSer,NEWDBUSER,guest
acl.list.weblogic.jndi.weblogic.ejb=system,newdbuSer,NEWDBUSER,everyone
acl.modify.weblogic.jndi=system,newdbuSer,NEWDBUSER,everyone
acl.frob.aclexample=joeuser
acl.write.managedObject=system,newdbuSer,NEWDBUSER,guest
acl.reserve.weblogic.jdbc.connectionPool.demopool=system,newdbuSer,NEWDBUS
ER,everyone
user.joeuser=0x5923f72dc88665f59c119a2a965897fc28a2feaa
acl.read.managedObject=system,newdbuSer,NEWDBUSER,guest
acl.admin.weblogic.jdbc.connectionPoolcreate=system,newdbuSer,NEWDBUSER,gu
est
user.system=0xfafe4ee933bf59e193de3a8a506a57ddd2364943
acl.unlockServer.weblogic.admin=system,newdbuSer,NEWDBUSER,guest
```

```
acl.reset.weblogic.jdbc.connectionPool=system
acl.execute.weblogic.servlet=everyone
acl.modify.weblogic.jndi.weblogic.ejb=system,newdbuSer,NEWDBUSER,everyone
```

Upgrading from Compatibility Security to WebLogic Server 8.1 Security

If you want to leverage the new security features in WebLogic Server 8.1, you need to upgrade your existing security configuration to a WebLogic Server 8.1 security configuration.

When you boot your WebLogic Server 6.x configuration in WebLogic Server 8.1, a security realm called `CompatibilityRealm` is created and set as the default (active) security realm. This `Compatibility` realm contains all your 6.x security data. In addition, a security realm called `myrealm` is created when you boot WebLogic Server 8.1.

To upgrade from Compatibility security, you need to make the `myrealm` security realm the default security realm:

1. In the WebLogic Server 8.1 Administration Console, click 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 the `myrealm` node.
3. Select 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 the WebLogic Server 8.1 instance to the `Administrators` group. This user replaces the `system` user. To add a user to the `Administrators` group:
 - a. In the left pane of the WebLogic Server 8.1 Administration Console, expand Security → Realms.
 - b. Expand the name of the realm you are configuring (for example, `myrealm`).
 - c. Click Groups.
 - d. The Groups tab appears. This tab displays the names of all groups defined in the WebLogic Authentication provider.

- e. Click the `Administrators` link on the `Groups` tab.
 - f. Select the `Membership` tab and add the user who can boot WebLogic Server to the `Administrators` group.
 - g. Click `Apply`.
5. Make sure the users and groups you had configured in the 6.x security realm have been added to an Authentication provider. For more information, see [“Upgrading Your WebLogic Server 6.x Security Realms to 8.1”](#) on page 3-18.
 6. Optionally, define security roles for your WebLogic Server 6.x users and groups. For more information, see [“Security Roles”](#) in *Securing WebLogic Resources*.
 7. Express any ACLs you added in WebLogic Server 6.x as security policies. For more information, see [“Working with Security Policies”](#) in *Securing WebLogic Resources*.
 8. Set `myrealm` as the default (active) security realm. For more information, see [“Setting a New Security Realm as the Default \(Active\) Security Realm”](#) in *Managing WebLogic Security*.
 9. Reboot the WebLogic Server 8.1 instance.

Each time the WebLogic Server 8.1 instance is booted, the security roles and security policies are applied. Subsequent access to the server and its methods are constrained by these security roles and security policies until they are changed.

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. Because of a potential security risk, 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 the anonymous 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 use great caution when using the `guest` user in a production environment.

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 8.1. 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 [“Boot Identity Files”](#) in the *Administration Console Online Help*. There is no direct upgrade of the old `password.ini` file.

Upgrading SSL

This section contains the following information about upgrading SSL:

- [“Upgrading SSL Identity and Trust”](#) on page 3-49
- [“New Options for Storing SSL Client Trust”](#) on page 3-50
- [“Using Host Name Verification”](#) on page 3-51
- [“Using a Certificate Authenticator”](#) on page 3-52
- [“Upgrading Private Keys and Digital Certificates”](#) on page 3-52

Upgrading SSL Identity and Trust

Using files or the WebLogic Keystore provider as a way to store identity and trust is deprecated in WebLogic Server 8.1. Also, private keys stored in files may or may not be password protected. Private keys that are not password protected can be vulnerable to security attacks. BEA recommends upgrading to keystores as a way to store identity and trust. If you choose not to use a keystore, a password is required in order to use the private key and associated digital certificate with this release of WebLogic Server. For more information, see [“Configuring SSL”](#) in *Managing WebLogic Security*.

For the purpose of backward compatibility, the WebLogic Server 8.1 Administration Console provides mechanisms for configuring SSL using file-based identity and trust. However, you should upgrade to keystores as soon as possible.

New Options for Storing SSL Client Trust

By default in WebLogic Server 8.1, SSL clients check the trusted certificate authority of the server. This check is done whenever a client and server connect using SSL, including when WebLogic Server is acting as a client. The client rejects the server's trusted certificate authority if the certificate authority is not trusted by the client. Previous releases of WebLogic Server allowed you to store SSL client trust in a file. In WebLogic Server 8.1, a client's trusted CA certificates must be stored in a keystore. WebLogic Server 8.1 offers different keystore options for storing client trust.

By default, all the trusted certificate authorities available from the JDK (`...\jre\lib\security\cacerts`) are trusted by SSL clients. Optionally, use the following command-line argument to specify a password for the JDK cacerts trust keystore:

```
-Dweblogic.security.JavaStandardTrustKeystorePassPhrase=password
```

where *password* is the password for the Java Standard Trust keystore. This password is defined when the keystore is created.

You also have the option of specifying one of the following types of trust:

- **Demo Trust**—The trusted CA certificates in the demonstration Trust keystore (`DemoTrust.jks`) located in the `WL_HOME\server\lib` directory. In addition, the CAs in the JDK `cacerts` keystore are trusted. To use the Demo Trust, specify the following command-line argument:

```
-Dweblogic.security.TrustKeyStore=DemoTrust
```

Optionally, use the following command-line argument to specify a password for the JDK cacerts trust keystore:

```
-Dweblogic.security.JavaStandardTrustKeystorePassPhrase=password
```

where *password* is the password for the Java Standard Trust keystore. This password is defined when the keystore is created.

- **Custom Trust**—A trust keystore you create. To use Custom Trust, specify the following command-line arguments:

```
weblogic.security.CustomTrustKeystoreFileName=filename
```

This required command-line argument specifies the fully qualified path to the trust keystore

```
weblogic.security.CustomTrustKeystoreType=type
```

This optional command-line argument specifies the type of the keystore. Generally, this value for type is jks.

```
weblogic.security.CustomTrustKeystorePassPhrase=password
```

This optional command-line argument specifies the password defined when creating the keystore.

For more information about using keystores, see [“Configuring SSL”](#) in *Managing WebLogic Security*.

Using Host Name Verification

In this release of WebLogic Server, SSL clients perform a host name verification check by default. The SSL client compares the Common Name (CN) field in the digital certificate received from the server with the server name in the URL the client used to connect to the server. The CN field and the server name must match to pass the host name verification check. In previous releases of WebLogic Server, this check had to be enabled.

You may need to disable host name verification to ensure your SSL client continues to work properly. Note that disabling host name verification will make your environment vulnerable to man-in-the-middle attacks. To disable the check, specify the following command-line argument:

```
-Dweblogic.security.SSL.ignoreHostnameVerification=true
```

Note: If the SSL server specified an IP address in its URL, disable the host name verification check.

The SSL client can also use a custom host name verifier instead of the host name verifier supplied by WebLogic Server. A custom host name verifier is an implementation of the `weblogic.security.SSL.HostnameVerifier` interface. Use the following command-line argument to specify a custom host name verifier:

```
-Dweblogic.security.SSL.hostnameVerifier=classname
```

where `classname` specifies the implementation of the `weblogic.security.SSL.HostnameVerifier` interface.

Using a Certificate Authenticator

If your WebLogic Server 6.x security configuration used an implementation of the `weblogic.security.acl.CertAuthenticator` class, you need to configure the Realm Adapter Authentication provider's included Identity Assertion provider.

To enable identity assertion in the Realm Adapter Authentication provider:

1. Expand the Security → Realms nodes.
2. Select `CompatibilityRealm`.
3. Expand the Providers node.
4. Select Authentication Providers.
5. Click the Realm Adapter Authenticator link in the Realms table.
6. On the General tab, enter X.509 in the Active Types list box.

Note: This step enables the use of 6.x Cert Authenticators.

7. Click Apply.
8. Reboot the WebLogic Server 8.1 instance.

When the Identity Assertion provider is configured, the Cert Authenticator is called before any username/password authentication. This change in behavior may cause a Cert Authenticator written in WebLogic Server 6.x to fail if clients use both two-way SSL and username and password for authentication. To fix this problem, configure the Cert Authenticator in the Realm Adapter Authentication provider to return `NULL` for unrecognized certificates or for all certificates if certificates are being used to establish the SSL connection.

Upgrading Private Keys and Digital Certificates

You can still use private keys and digital certificates used with WebLogic Server 6.1 with WebLogic Server 8.1. Convert the private key and digital certificate from privacy-enhanced mail (PEM) format to distinguished encoding rules (DER) format. For more information, see the description of the [der2pem](#) utility in *Using WebLogic Server Java Utilities*.

After converting the files, ensure the digital certificate file has the -----BEGIN CERTIFICATE----- header and the -----END CERTIFICATE----- footer. Otherwise, the digital certificate will not work with WebLogic Server 8.1.

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 8.1.

Note: WebLogic Server 8.1 uses PointBase 4.2 as a sample database and does not bundle the Cloudscape database.

- [“Apache Xalan XML Transformer” on page 3-54](#)
- [“Apache Xerces XML Parser” on page 3-54](#)
- [“Deployment” on page 3-55](#)
- [“Manual Changes to Deployment Descriptors for EJB 2.0” on page 3-55](#)
- [“jCOM” on page 3-57](#)
- [“Upgrading Your JDK” on page 3-58](#)
- [“JMS” on page 3-59](#)
- [“JMX” on page 3-60](#)
- [“Jolt Java Client” on page 3-60](#)
- [“Upgrading Your JVM to JRockit” on page 60](#)
- [“JSP” on page 3-61](#)
- [“6.1 Managed Servers Cannot Boot from an 8.1 Administrative Server” on page 3-62](#)
- [“MBean API Change” on page 3-63](#)
- [“Default Queue Names Have Changed” on page 3-63](#)
- [“ThreadPoolSize Is Now ThreadCount” on page 3-64](#)
- [“Web Application API Changes from 6.0” on page 3-65](#)
- [“Clusters on Solaris Perform Better with Client JVM” on page 3-65](#)
- [“Change WebLogic Tuxedo Connector Configuration” on page 3-66](#)
- [“Upgrade Web Services” on page 3-71](#)
- [“Deprecated APIs and Features” on page 3-72](#)

- “Removed APIs and Features” on page 3-73

Apache Xalan XML Transformer

The built-in transformer in WebLogic Server 8.1 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 8.1 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 8.1 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. For more information, see [Administering WebLogic Server XML](#) in *Programming WebLogic Server XML*.

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.

Runtime Modes

In WebLogic Server 6.1, 7.0, and 8.1 there is a division between runtime modes. The two modes are “development” and “production.” The runtime mode with a command- line parameter when starting the Weblogic Server (`-Dweblogic.ProductionModeEnabled=true | false`). If this

parameter is set to true, the server runs in development mode. In development mode the server behavior is consistent with WebLogic Server 6.0. In production mode in versions 6.1 and later, however, the auto-deployment feature is disabled, with the result that deployment units in the applications directory that are not explicitly deployed in the configuration repository (`config.xml`) will not be automatically deployed.

Deployment

WebLogic Server 8.1 uses a two-phase deployment model that fails a clustered deployment on all servers if it fails on some servers. For more information on this deployment model and other 8.1 deployment features, see [Deploying WebLogic Server Applications](#). 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.

Unlike previous versions, WebLogic Server 8.1 will not deploy an application that has any errors in its deployment descriptor. For example, if your WebLogic Server 6.x application was missing a reference description stanza in the deployment descriptor, the application will not deploy in WebLogic Server 8.1 until you add that stanza. A typical stanza looks like this:

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

Using WebLogic Server 8.1, 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 are starting your server, the applications are deployed with one-phase deployment. The `weblogic.deploy` and `weblogic.refresh` command line utilities and the `weblogic.management.tools.WebAppComponentRefreshTool` and are deprecated in 8.1.

See “[Deprecated APIs and Features](#)” on page 3-72 for information on deprecated MBean attributes and operations.

Manual Changes to Deployment Descriptors for EJB 2.0

The EJB 2.0 specification has changed substantially between WebLogic Server 6.0 and WebLogic Server 8.1, and somewhat between WebLogic Server 6.1 and WebLogic Server 8.1.

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 8.1, 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 8.1 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 8.1, 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 8.1:
 - In 8.1, 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 8.1, 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 8.1, 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 8.1, 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 8.1.

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 and 8.1.
- Remote relationships do not exist in 6.0, but do exist in 6.1 and 7.0 and 8.1.
- The new implementation of read-only beans in WebLogic Server 7.0 and after 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 the WebLogic Server 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 implementation of read-only beans in version 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 3-55](#).

weblogic.management.configuration.EJBComponentMBean Changes

Beginning in WebLogic Server 6.1 and continuing in WebLogic Server 8.1, 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 upgrade from WebLogic Server 6.0 to 8.1.

max-beans-in-cache Parameter

In WebLogic Server 8.1 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.

Note: Evaluate the amount of memory available with the amount you require as, assigning a very high value to the `max-beans-in-cache` parameter may result in an `OutOfMemoryError`. The amount of memory cache required can be calculated as the number of beans in the cache times the size of the EJB.

Fully Qualified Path Expressions

In an EJB QL Query on WebLogic Server 8.1, 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 8.1 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 8.1 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 8.1 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 8.1. See [JDBCConnectionPool](#) in *WebLogic Server Configuration Reference*.

Upgrading Your JDK

If you compile your application in WebLogic Server 8.1, it is advisable to do subsequent builds referencing JDK 1.4 rather than earlier JDKs.

JDK 1.4 requires stricter adherence to specifications than did earlier JDKs. For example, the Blueprint example application Pet Store contains methods that declare `int` while their setters specify `String`. These methods were acceptable in earlier versions of WebLogic Server, but they need to be repaired before they can run in WebLogic Server 8.1, because JDK 1.4 disallows them. See [“Fix JSP Parsing Errors” on page 3-11](#).

Stricter JSP Parsing in JDK 1.3.1_09

Improper code that was acceptable under JDKs before 1.3.1_09 may cause errors under JDK 1.3.1_09 and later. This failure occurs for JSPs and all beans that are subject to the Introspector API. Specifically, disagreement between a property’s setter and getter types may cause startup errors like the following after you migrate to JDK 1.3.1_09 or later:

```
Error in using tag library uri='/WEB-INF/tlds/taglib.tld' prefix='j2ee':
There is no setter method for property 'numItems', for Tag class
'com.sun.j2ee.blueprints.petstore.taglib.list.SearchListTag' probably
occurred due to an error in /template.jsp line 8: <%@ taglib
uri="/WEB-INF/tlds/taglib.tld" prefix="j2ee" %>
```

When a class fails conform to Java bean specifications about type agreement, the `java.beans.Introspector` API no longer returns read or write methods for the offending property. Correct such errors by ensuring that the setters and getters in your classes do not disagree in type. If the setter is an integer, the getter must be an integer also, and must not be a string.

For example, the following snippet shows a valid setter and getter from which we can expect the Introspection API to return valid read and write method(s):

```
private String foo;

public void setFoo(String f) {
```

```

        foo = f;
    }
    public String getFoo() {
        return foo;
    }
}

```

The following snippet demonstrates bad code that does not conform to Java bean specifications:

```

private int foo; // note that foo is an int
public void setFoo(String f) {
    foo = Integer.parseInt(f);
}
public int getFoo() {
    return foo;
}

```

In the second case, the type disagreement between the setter and getter will cause an error under JDK 1.3.1_09 and later, because the newer JDKs adhere strictly to the Java bean specifications. */

WebLogic Server 8.1 Dependencies on JDK 1.4

Some APIs and other items added in JDK 1.4 have been removed in the WebLogic Server 8.1 version of `weblogic.jar`. Such APIs and other items cease to be available to your application in WebLogic Server 8.1 after you compile your application with the 8.1 `weblogic.jar` in place of the earlier version of `weblogic.jar` in your classpath.

JAXP, for example, was included in previous versions of `weblogic.jar`, but is absent from the WebLogic Server 8.1 `weblogic.jar`. JAXP is not in the JDK 1.3, so if your WebLogic Server 6.x application uses JAXP and you compile it after replacing the 6.x `weblogic.jar` with the 8.1 `weblogic.jar`, build your classes using the JDK 1.4.

Other WebLogic Server 8.1 dependencies on JDK 1.4 include the JAAS package and the Principal Authenticator object.

JMS

WebLogic Server 8.1 supports the *JavaSoft JMS specification version 1.0.2*.

All WebLogic JMS 6.x applications are supported in WebLogic JMS 8.1. However, in order for your applications to use 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 8.1. However, if you are employing internal MBeans or attributes, you may encounter porting issues.

See “[Deprecated APIs and Features](#)” on page 3-72 for information on deprecated MBean attributes and operations.

Jolt Java Client

Jolt users may need to upgrade to Jolt Java Client 8.1.0 to enable BEA Tuxedo services for the Web using BEA WebLogic Server 8.1. For more information, see [Platform Support for Jolt](#). The Jolt Java Client is available from the [BEA Product Download Center at `http://commerce.bea.com/downloads/products.jsp`](#).

Upgrading Your JVM to JRockit

When you upgrade a domain to WebLogic Server 8.1, consider upgrading your JVM to JRockit. WebLogic JRockit is a JVM designed for running server-side applications in Windows and Linux running on Intel architectures. For server-side applications, JRockit has these advantages over other virtual machines:

- It employs adaptive optimization, which detects and removes bottlenecks in the deployed application.
- It is designed specifically for the special requirements of server-side applications, which tend to be parallel and thread-intensive, to run for longer periods of time, and not to use graphical interfaces.
- You can monitor JRockit using the WebLogic Server Administration Console.

To switch a WebLogic Server domain to the JRockit JVM:

1. In the server start scripts, set `JAVA_HOME` (or equivalent) shell variables to point to the JRockit root directory. For example, change:

```
@rem Set user-defined variables.
```

```
set JAVA_HOME=WL_HOME\jdk131
```

where `WL_HOME` is the WebLogic Server 6.x installation directory, to

```
@rem Set user-defined variables.
```

```
set JAVA_HOME=WL_HOME\jrockit81_141_02
```

where `WL_HOME` is the WebLogic Server 8.1 installation directory.

2. Change the domain's `config.xml` to use the JRockit `javac.exe`. For example, change

```
JavaCompiler="WL_HOME\jdk131\bin\javac"
```

where `WL_HOME` is the WebLogic Server 6.x installation directory, to

```
JavaCompiler=WL_HOME\jrockit81_141_02\bin\javac"
```

where `WL_HOME` is the WebLogic Server 8.1 installation directory.

3. Remove from server start scripts any switches specific to the Sun JVM. For example, from the start command:

```
echo on "%JAVA_HOME%\bin\java" -hotspot .... weblogic.Server
```

```
delete "-hotspot".
```

4. Start and configure JRockit. See the [Starting and Configuring the WebLogic JRockit JVM](#) section for the appropriate version of the JRockit documentation.

For JRockit platform and user information, see the appropriate version of the JRockit [User Guide](#).

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 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>
```

New Behavior of Load Order Methods for Startup Classes

The behavior of the load order methods in `StartupClassMBean` has changed between versions 6.x and 8.1.

If you set load order in version 6.x by setting `LoadBeforeAppDeployment` to true or false, you should now use `LoadBeforeAppActivation`.

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 version 8.1, achieve the same load order by setting `LoadBeforeAppActivation` to true.

`LoadBeforeAppDeployments` still exists in version 8.1 but its behavior has changed. It now determines whether a startup class is loaded and run before the server activates JMS and JDBC services or deploys applications and EJBs.

See details about these methods in version 8.1 at

<http://e-docs.bea.com/wls/docs81/javadocs/weblogic/management/configuration/StartupClassMBean.html>.

6.1 Managed Servers Cannot Boot from an 8.1 Administrative Server

A Managed Server running WebLogic Server 6.x cannot obtain its configuration and boot using an Administration Server running WebLogic Server 8.1. 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 8.1 installation directory.

Default Queue Names Have Changed

Default names for execute queues have changed in WebLogic Server 8.1. If you upgrade a configuration that specifies execute queues, the default queue names will automatically alias the new queue names.

Table 3-1 Default Queue Names

Pre-8.1 Default Queue Names	WebLogic Server 8.1 Default Queue Names
default	weblogic.kernel.Default
__weblogic_admin_html_queue	weblogic.admin.RMI
__weblogic_admin_rmi_queue	weblogic.admin.HTTP

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*.

Update Your web.xml file with New Classes

For 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 3-55](#).

ThreadPoolSize Is Now ThreadCount

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 8.1 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.

In WebLogic Server 6.x you set the thread count on the command line as follows:

```
<Server
Name="myserver"
ThreadPoolSize="23"
...
/Server>
```

Starting in WebLogic Server 7.0 you set the thread count on the command line as follows:

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

To change the thread count value via the Administration Console in WebLogic Server 8.1:

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.

404 Message for URIs with Extra Spaces

Previous versions of WebLogic Server resolved URIs that contained extra spaces. WebLogic Server 8.1 no longer resolves extra spaces, and a URI request that contains extra spaces will result in a 404.

For example, `http://server:port/mywebapp/foo%20%20` used to resolve to the resource `foo` in the Web application "mywebapp," but beginning with 8.1 it no longer does.

Web Application API Changes from 6.0

- 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 and 8.1. You will have to update your source code and recompile if you are using `weblogic.management.runtime.ServletRuntimeMBean.getName()`.
- It is no longer possible to use the Administration Console to define the default Web application. Define the default Web application by setting the `context-root` element in the application's `application.xml` file, if the application is part of an enterprise application, or the `weblogic.xml` file if it is a standalone Web application, to `"/`.
- 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 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 8.1 supports Servlet 2.3, so you should be able to take advantage of the new `ServletRequest/ResponseWrapper` objects.

Clusters on Solaris Perform Better with Client JVM

Certain applications (large EJB applications) 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.

Change WebLogic Tuxedo Connector Configuration

You must make the following application and configuration changes to use WebLogic Tuxedo Connector in WebLogic Server 8.1:

- “Start the WebLogic Tuxedo Connector” on page 3-66
- “Convert WebLogic Tuxedo Connector XML Configuration Files” on page 3-66
- “Update Inbound RMI-IIOP Applications” on page 3-68
- “Authenticate Remote Users” on page 3-70
- “Set WebLogic Tuxedo Connector Properties” on page 3-71
- “Using the Runtime WTC ORB” on page 3-71

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/docs81/wtc_admin/Install.html#WTCuseCLI.

Releases prior to WebLogic Server 7.0 use 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 8.1, WebLogic Tuxedo Connector is managed as a WebLogic Server Service.

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

For more information on starting and ending a WebLogic Tuxedo Connector session, see *Assign a WTC Service to a Server* at <http://e-docs.bea.com/wls/docs81/ConsoleHelp/wtc.html#AssignWTCServer>.

Convert WebLogic Tuxedo Connector XML Configuration Files

In 8.1, 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 pre-8.1 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:

Table 3-2 Conversion Tool Arguments

Argument	Description
<code>-Dweblogic.wtc.migrateDebug</code>	WebLogic property used to turn on <code>wtc.migrateDebug</code> mode.
URL	URL passed to your server. Example: <code>\myServer:7001</code>
USERNAME	User Name passed to your server. Example: <code>system</code>
PASSWORD	Password passed to your server. Example: <code>mypasswd</code>
CONFIGWTC	Fully qualified path and name of the WebLogic Tuxedo Connector XML configuration file to migrate to the <code>config.xml</code> file. Example: <code>d:\bea\weblogic81\server\samples\examples\wtc\atmi\simpapp\bdmconfig.xml</code>
SERVERNAME	Optional. The name of the administration or managed server that you want the new <code>WTCServer MBean</code> assigned to. Default: the current active administration server.
DOMAIN	Optional. The name of the WebLogic Server 8.1 domain that you want the new <code>WTCServer MBean</code> assigned to. Default: the current active domain.

Table 3-2 Conversion Tool Arguments

Argument	Description
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 and placed in the `config.xml` file of the server specified in step 2. The migration utility sets the AppKey Generator attribute to `TpUsrFile` and the Default AppKey attribute to `-1`.

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-IIORP at http://e-docs.bea.com/wls/docs81/wtc_atmi/CORBA.html](http://e-docs.bea.com/wls/docs81/wtc_atmi/CORBA.html).

To use inbound RMI-IIOP in 8.1, 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 3-3 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 where the WTC Service is running.

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/docs81/wtc_admin/BDCONFIG.html#WTCuserAuth.

WebLogic Tuxedo Connector uses Access Control Lists (ACLs) to 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 8.1 Administration 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. 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/docs81/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/docs81/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/docs81/wtc_admin/BDCONFIG.html#1111404.

- `-Dweblogic.wtc.PasswordKey=mykey`

where *mykey* is the key value.

Using the Runtime WTC ORB

This release of WebLogic Tuxedo Connector implements a new WTC ORB which uses WebLogic Server RMI-IIOP runtime and CORBA support. Previous releases used a JDK based WTC ORB. A wrapper is provided to allow users with legacy applications to use the new WTC ORB without modifying their existing applications. BEA recommends that users migrate to the new WTC ORB. For more information, see [How to Develop WebLogic Tuxedo Connector Client Beans using the CORBA Java API](#) at http://e-docs.bea.com/wls/docs81/wtc_atmi/CORBA.html.

Upgrade Web Services

Due to changes in the Web service runtime system and architecture between versions 6.1 and 8.1 of WebLogic Server, you must upgrade Web services created in version 6.1 to run on version 8.1.

The WebLogic Web services client API included in WebLogic Server 6.1 of has been deprecated and you cannot use it to invoke 8.1 Web services. WebLogic Server 8.1 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 8.1, see *Upgrading 6.1 WebLogic Web Services to 8.1* at [http://e-docs.bea.com/wls/docs81/web Services/migrate.html](http://e-docs.bea.com/wls/docs81/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/docs81/webServices/client.html>.

For general information on the differences between 6.1 and 8.1 Web services, see *Overview of WebLogic Web Services* at <http://e-docs.bea.com/wls/docs81/webServices/overview.html>.

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 8.1. It is replaced by JavaWebStart.
- WebLogic Enterprise Connectivity (WLEC) is deprecated in this release of WebLogic Server 8.1. 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 8.1 and has been replaced by `weblogic.Deployer`. For more information, see *Deploying WebLogic Server Applications*.
- `weblogic.management.tools.WebAppComponentRefreshTool` and `weblogic.refresh` are both deprecated in this release of WebLogic Server 8.1. They have been replaced by `weblogic.Deployer`.
- 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 8.1 [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.

Upgrading WebLogic Server 6.x to Version 8.1

Upgrading WebLogic Server 5.1 to Version 8.1

Upgrading WebLogic Server 5.1 to version 8.1 is a multi-step process that involves, among other tasks, using a utility to convert your `weblogic.properties` file(s) into a new XML file format. There are also several specification changes that affect the upgrade process. The following sections address most known 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 5.1 to WebLogic Server 8.1; to upgrade your applications from WebLogic Server 5.1 to WebLogic Server 8.1; and deploy these applications. Instructions apply to upgrades from both WebLogic Server 5.1 to WebLogic Server 8.1.

To see an example of an application being upgraded from WebLogic Server 5.1 to WebLogic Server 8.1, see [“Upgrading the Banking Application from WebLogic Server 5.1 to WebLogic Server 8.1”](#) on page 4-17.

- [“Understanding the WebLogic Server 8.1 Directory Structure”](#) on page 4-2
- [“Contents of a Domain Directory”](#) on page 4-2
- [“Upgrading WebLogic Server 5.1 to Version 8.1: Main Steps”](#) on page 4-3
- [“Converting the weblogic.properties File to config.xml”](#) on page 4-5
- [“Upgrading the Banking Application from WebLogic Server 5.1 to WebLogic Server 8.1”](#) on page 4-17
- [“Classloading in WebLogic Server 8.1”](#) on page 4-7

- “Modifying Startup Scripts” on page 4-8
- “Converting Applications to Web Applications” on page 4-8
- “Upgrading Enterprise Java Beans Applications” on page 4-12
- “Upgrading JMS” on page 4-16
- “Upgrading Oracle” on page 4-17
- “Additional Upgrade and Deployment Considerations” on page 4-21

Understanding the WebLogic Server 8.1 Directory Structure

When you upgrade to WebLogic Server 8.1, your servers and applications will be managed in WebLogic Server domains. For a full description of WebLogic Server domains see [Overview of WebLogic Server Domains](#) in *Configuring and Managing WebLogic Server*.

BEA WebLogic recommends that you locate domain directories outside the WebLogic Server installation directory. Domain directories can be in any location that can access the WebLogic Server installation and the JDK.

If you change the location of your domain directory, remember to update any custom tools or scripts relative to the new directory structure. Similarly, if you use a scripted tool for creating domains, change its scripts. The *Configuration Wizard* is the recommended tool for creating domains, and it can be scripted.

Contents of a Domain Directory

The configuration of a domain is stored in the `config.xml` file of the domain directory on the Administration Server (for information about Administration Servers and Managed Servers, see [Overview of WebLogic Server Domains](#) in *Configuring and Managing WebLogic Server*). `config.xml` stores the name of the domain and the configuration parameter settings for each server instance, cluster, resource, and service in the domain.

The domain directory also contains default script files that you can use to start the Administration Server and Managed Servers in the domain.

The domain directory should have:

- A root directory with the same name as the domain, such as `mydomain` or `petstore`. This directory should contain the following:
 - The configuration file (usually `config.xml`) for the domain.

- Any scripts you use to start server instances and establish your environment.
- Optionally, a subdirectory for storing the domain’s application files.

Upgrading WebLogic Server 5.1 to Version 8.1: Main Steps

These general steps are a checklist of issues to consider when upgrading from version 5.1 to version 8.1. For a detailed example of such an upgrade, see [“Upgrading the Banking Application from WebLogic Server 5.1 to WebLogic Server 8.1”](#) on page 4-17.

To upgrade a cluster of server instances, you must install WebLogic Server 8.1 on each computer that hosts server instances. If you are upgrading a server cluster, refer to [Setting Up WebLogic Clusters](#) in *Using WebLogic Server Clusters* for WebLogic Server cluster configuration guidelines.

1. Convert your `weblogicLicense.class` or a `WebLogicLicense.XML` license file before installing WebLogic Server 8.1. See [“Upgrading WebLogic Server License Files”](#) on page 4-4.

2. Install WebLogic Server 8.1 in a location accessible from the WebLogic Server 5.1 installation directory. See [Installing WebLogic Platform](#).

Prior to WebLogic Server 6.0, a separate download was required if you wanted to use 128-bit encryption instead of 56-bit encryption. WebLogic Server 8.1 has a single download for both 56-bit encryption and 128-bit encryption. For details about how to enable 128-bit encryption see [Enabling 128-Bit Encryption](#) in the *Installation Guide*.

3. Convert your `weblogic.properties` file to an 8.1 domain configuration using the Convert `weblogic.properties` utility in the Administration Console, located under Information Resources. For instructions, see [“Converting the weblogic.properties File to config.xml”](#) on page 4-5 and the [Console Help](#) documentation.
4. Add classes to your Java system `CLASSPATH`. For more information see [“Classloading in WebLogic Server 8.1”](#) on page 4-7.
5. If you are using custom startup scripts from WebLogic Server 5.1, modify them to point to WebLogic 8.1 Server. See [“Modifying Startup Scripts”](#) on page 4-8.
6. Package your WebLogic Server server-side business object implementations (referred to as Web applications in WebLogic Server 6.0 and later) to run on WebLogic 8.1. See [“Converting Applications to Web Applications”](#) on page 4-8.
7. If you need to upgrade EJBs, see [“Upgrading Enterprise Java Beans Applications”](#) on page 4-12.

8. Upgrade JMS. Many new configuration attributes have been added to JMS since WebLogic Server 5.1. For more information, see [“Upgrading JMS” on page 4-16](#).
9. Upgrade Security. Many new security features are available in WebLogic Server 8.1. For more information, see [“Upgrading WebLogic Server License Files” on page 4-4](#).
10. Your application may require additional upgrade steps to account for other factors, for example new parsers and altered APIs. See [“Additional Upgrade and Deployment Considerations” on page 4-21](#) for information about these factors.

For example, if you compile an upgraded application with WebLogic Server 8.1, 8.1 dependencies on JDK 1.4 require that your 8.1 domain reference JDK 1.4 or an equivalent such as JRockit. For more information on upgrading to JRockit, see [“Upgrading Your JVM to JRockit” on page 4-25](#).

11. Start WebLogic Server 8.1 Administration and Managed Servers, and configure and deploy your application.

For information about starting WebLogic Server 8.1, see [Starting and Stopping Servers: Quick Reference](#). For information about configuring and deploying your application, see [Deployment](#).

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 8.1:

1. Convert the `WebLogicLicense.class` license file to a `WebLogicLicense.XML` file using the [`licenseConverter` utility](#) at <http://www.weblogic.com/docs51/techstart/utills.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 8.1), 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_wlsxx.bea` file through e-mail. To update the `license.bea` file on your system, see “[Installing and Updating WebLogic Platform License Files](#)” in *Installing WebLogic Platform*.

Converting the weblogic.properties File to config.xml

Prior to WebLogic Server 6.0, WebLogic Server releases used a `weblogic.properties` file to configure applications. In WebLogic Server 8.1, configuration is handled by a domain configuration file, `config.xml`, and by deployment descriptor files. Converting a `weblogic.properties` file to the `config.xml` file creates a WebLogic Server 8.1 domain for your applications and generates the XML files that define how your applications are set up.

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. The `domain` element includes child elements, such as the `server`, `cluster`, and `application` elements. These child elements often 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.

Upgrading WebLogic Server 5.1 to Version 8.1

Convert your WebLogic Server 5.1 `weblogic.properties` file to a WebLogic Server 8.1 `config.xml` file following these steps:

1. Shut down the WebLogic Server 5.1 server instance or instances.
2. Start the WebLogic Server 8.1 Examples server.

For information on starting the WebLogic Server 8.1 examples server, see [Starting and Stopping Servers: Quick Reference](#).

You will be prompted for a user name and password.

3. 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.
4. In the first page of the “Convert `weblogic.properties`” path (“Step 1 - Locate `weblogic` root”), browse to select the directory that contains the `weblogic.properties` file.
The second page of the “Convert `weblogic.properties`” path appears.
5. From a list of available application directories, select the root directory that contains your application directories. The `weblogic.properties` converter will convert the application directories into a WebLogic Server 8.1 domain.
6. Fill in the remaining text fields. Do not target a directory in the currently running instance of Weblogic Server as Output Directory.
 - a. Admin Server Name
 - b. Output Directory
 - c. WebLogic Home
 - d. Name for New Domain
7. Click Convert.

When you convert your `weblogic.properties` file, the `web.xml` and `weblogic.xml` files for the default Web application are created for you and placed inside the `domain\applications\DefaultWebApp_myserver\WEB-INF` directory. 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 editing `config.xml`, see [WebLogic Server Configuration Reference](#).
- Security properties are stored in the `fileRealm.properties` file located in `domain`.
- 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 “[Mapping weblogic.properties to 6.x config.xml Configuration Attributes](#)” on page A-1.

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 8.1 distribution and start the Administration Server in the new domain. You may need to edit this startup script to further specify your startup preferences for the domain. See “[Modifying Startup Scripts](#)” on page 4-8.

Classloading in WebLogic Server 8.1

Before WebLogic Server 6.0, 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 `%MyOldClassspath%` contains only the directories that point to your old applications.

Modifying Startup Scripts

The `weblogic.properties` converter created a new startup script (called `startdomainName.cmd` or `.sh`) for your new WebLogic Server 8.1 domain. If you need to edit this script to specify your domain startup preferences, keep the following in mind.

- The WebLogic classpath is no longer used; use the Java system classpath as described in the preceding section, [“Classloading in WebLogic Server 8.1” on page 4-7](#).
- WebLogic Server 8.1 is started from the domain directory. Paths in the startup script assume that the script is located in the domain directory.
- It is no longer necessary to include the license file in the classpath.
- There is now a distinction between an Administration Server and Managed Servers (see [The Administration Server and Managed Servers](#) in *Configuring and Managing WebLogic Server*). Scripts that start servers may need to 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 *Configuring and Managing WebLogic Server*.

Converting Applications to Web Applications

In order to convert an application to a Web application and upgrade it to WebLogic Server 8.1, the application’s files must be placed within a directory structure that follows a specific pattern. For more information on Web applications see [Developing Web Applications for WebLogic Server](#).

The following sections discuss upgrading and deploying Web applications. They include a procedure for upgrading a simple servlet from WebLogic Server 5.1 to WebLogic Server 8.1.

- [“XML Deployment Descriptors” on page 4-9](#)
- [“WAR Files” on page 4-9](#)
- [“Session Porting” on page 4-9](#)
- [“JavaServer Pages \(JSPs\) and Servlets” on page 4-10](#)
- [“Upgrading a Simple Servlet from WebLogic Server 5.1 to WebLogic Server 8.1” on page 4-11](#)

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.

There is also a WebLogic-specific Web application 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 [weblogic.xml Deployment Descriptor Elements](#) in *Developing Web Applications for WebLogic Server*. 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`. See [Developing Web Applications for WebLogic Server](#) 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. 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.

Session Porting

WebLogic Server 8.1 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 8.1, cookies are named `JSESSIONID` by default.

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 have converted the properties in your `weblogic.properties` file to XML elements and attributes in `config.xml`, `web.xml` and `weblogic.xml`. For information on this process, see [“Converting the weblogic.properties File to config.xml”](#) on page 4-5.
- 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 8.1 is fully compliant with the Servlet 2.3 specification.
- Your `web.xml` file should use:

```
weblogic.servlet.proxy.HttpClusterServlet
```

instead of

```
weblogic.servlet.internal.HttpClusterServlet
```

and

```
weblogic.servlet.proxy.HttpProxyServlet
```

instead of

```
weblogic.t3.srvr.HttpProxyServlet
```

Upgrading a Simple Servlet from WebLogic Server 5.1 to WebLogic Server 8.1

The following procedure upgrades the Hello World Servlet provided with WebLogic 5.1 Server to WebLogic Server 8.1. The procedures assume that you have both 5.1 and 8.1 installed on a single server, and only 8.1 is running.

1. In WebLogic Server 8.1, create a directory structure for a Web application 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.
2. Place the `HelloWorldServlet.java` file (located in the `WL_HOME\examples\servlets` directory of your 5.1 installation) inside the `C:\hello\WEB-INF\classes` directory.
3. Create a `web.xml` file for this servlet. If you 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 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-class>
</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 *web.xml Deployment Descriptor Elements* in *Developing Web Applications for WebLogic Server*. A `weblogic.xml` file is not necessary with such a simple, stand-alone servlet as `HelloWorld`.

For more information on `weblogic.xml` files, see *weblogic.xml Deployment Descriptor Elements* in *Developing Web Applications for WebLogic Server*.

4. Move the `web.xml` file from `domain\applications\DefaultWebApp_myserver\WEB-INF` to `C:\hello\WEB-INF\`.
5. 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.

6. 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.

7. 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.

8. 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.

Upgrading Enterprise Java Beans Applications

The following sections describe Enterprise Java Beans upgrade procedures and related information.

EJB Upgrade Considerations

Consider the following when upgrading Enterprise Java Beans to WebLogic Server 8.1.

- WebLogic Server Version 8.1 supports the Enterprise Java Beans 1.1 and 2.0 specifications. However, the finder expressions feature, which was a WebLogic Server extension to EJB 1.1, is not supported for both the EJB 1.1 and EJB 2.0 specifications.
- EJB 1.1 beans are deployable in WebLogic Server 8.1. 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 [DDConverter](#) in *Programming WebLogic Enterprise Java Beans*.
- Upgrade EJB 1.0 deployment descriptors to EJB 2.0 by first upgrading them to EJB 1.1 and then using the [DDConverter](#) utility to upgrade them to EJB 2.0. Details on the [DDConverter](#) utility are provided in the section [DDConverter](#) in *Programming WebLogic Enterprise Java Beans*.

- If `ejbc` has not been run on an EJB, WebLogic Server 8.1 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 “[EJB Development Task Guide](#)” in *Programming WebLogic Enterprise Java Beans*.
- 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 8.1 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 8.1 EJB 1.1 DTD, or the WebLogic Server 8.1 EJB 2.0 DTD.
- In WebLogic Server 8.1 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.
- Use a fast compiler with `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 8.1.

The WebLogic Server 8.1 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 8.1 finds and complains about the error. These errors must be corrected before the EJB is deployed in WebLogic Server 8.1.

For instance, WebLogic Server 8.1 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.

- 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 8.1 CMP deployment descriptor (`weblogic-cmp-rdbms.xml`) supports `TxDataSources` and should be used instead of the WebLogic Server 5.1 CMP deployment descriptor which only specifies a connection pool.

- The following table shows which EJB versions are supported by which versions of WebLogic Server and the CMP deployment descriptor.

Table 4-1 Supported Descriptor Combinations

EJB Version	WebLogic Server Version	The CMP Version
Existing WebLogic Server 5.1 deployments use the following combination. EJBs can be deployed without changing descriptors or code in WebLogic Server 8.1.		
1.1	5.1	5.1
The following combinations include a WebLogic Server 8.1 CMP deployment descriptor. The WebLogic Server 8.1 EJB 1.1 CMP deployment descriptor allows multiple EJBs to be specified within a single EJB JAR file, and it supports using a <code>TxDataSource</code> , which is required when an EJB is enlisted in a two-phase /XA transaction.		
1.1	5.1	8.1
1.1	8.1	8.1
EJB 2.0 beans always use the WebLogic Server 6.x or 8.1 deployment descriptors.		
2.0	6.x	8.1
2.0	8.1	8.1

For more information on Enterprise Java Beans, see [Enterprise Java Bean Components](#) and [Programming WebLogic Enterprise Java Beans](#).

Steps for Upgrading a 1.1 EJB from WebLogic Server 5.1 to WebLogic Server 8.1

The WebLogic Server 5.1 `weblogic.properties` file only allows the exclusive or read-only concurrency options. The database concurrency option is available when upgrading to the WebLogic Server 8.1 `weblogic-ejb-jar.xml` file. See “[Choosing a Concurrency Strategy](#)” in *Programming WebLogic Enterprise Java Beans*.

The WebLogic Server 8.1 CMP deployment descriptor, `weblogic-cmp-rdbms-jar.xml`, allows multiple EJBs to be specified and 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 upgrade a 1.1 EJB from WebLogic Server 5.1 to WebLogic Server 8.1:

1. Open the WebLogic Server 8.1 Administration Console. From the home page, click on Install Applications under the Getting Started heading.
2. Locate the JAR file you wish to upgrade using the Browse button, then click Open and Upload. Your bean should automatically deploy on WebLogic Server 8.1.
3. Compile all the needed client classes. For example, using the Stateless Session Bean sample that was provided with WebLogic Server 8.1, you would use the following command:

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

4. 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 8.1. 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 8.1.

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`. EJB 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;
```

3. Run `DDConverter` on the JAR and specify 2.0 output. See [DDConverter](#) in *Programming WebLogic Enterprise Java Beans*.

Porting EJBs from Other J2EE Application Servers

Any EJB that complies with the EJB 1.1 or EJB 2.0 specifications can be deployed in the WebLogic Server 8.1 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.

Upgrading JMS

WebLogic Server 8.1 supports the [JavaSoft JMS specification version 1.0.2](#).

- Customers running Weblogic Server 5.1 SP07 or SP08 should contact BEA Support before upgrading existing JDBC stores to version 8.1.
- In order to upgrade object messages, the object classes need to be in the Weblogic Server 8.1 server CLASSPATH.
- For destinations that are not configured in Weblogic Server 8.1, the upgraded messages will be dropped and the event will be logged.

To upgrade your WebLogic Server JMS applications, see the procedures in *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.

For 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.

Upgrading the Banking Application from WebLogic Server 5.1 to WebLogic Server 8.1

In the following procedures it is assumed that you have WebLogic Server 5.1 and 8.1 both installed on a single computer.

Use the following three main steps to upgrade the banking application from WebLogic Server 5.1 to WebLogic Server 8.1:

- [Set Up the Banking Application on WebLogic Server 5.1](#)
- [Convert the weblogic.properties File](#)
- [Configure the Banking Application for WebLogic Server 8.1](#)

Set Up the Banking Application on WebLogic Server 5.1

1. Download the banking application from the link on http://dev2dev.bea.com/products/wlserver61/tutorials/wls_migration.jsp.
2. Follow the steps in the tutorial in the ZIP file with the banking application to install the banking application on WebLogic Server 5.1.

3. Shut down the instance of WebLogic Server 5.1.

Convert the weblogic.properties File

The conversion utility writes the properties in your WebLogic Server 5.1 `weblogic.properties` file to a WebLogic Server 8.1 domain configuration file, `config.xml`. For more information about domains in WebLogic Server, see [WebLogic Server Configuration Reference at `http://e-docs.bea.com/wls/docs70/config_xml/overview.html`](http://e-docs.bea.com/wls/docs70/config_xml/overview.html).

Use the WebLogic 8.1 Administration Console to convert the WebLogic Server 5.1 application's `weblogic.properties` file.

1. Launch the WebLogic Server 8.1 Examples Server.
 - a. In a command console, navigate to `WL_HOME\samples\domains\examples`, where `WL_HOME` is your WebLogic Server installation directory.
 - b. Run the script that sets up the Examples environment. In Windows, enter `setExamplesEnv.cmd`. In Unix, use `setExamplesEnv.sh`.
 - c. Start the Examples Server. In Windows, enter `StartExamplesServer.cmd`. In Unix, use `StartExamplesServer.sh`.
2. In the WebLogic Server Examples page that the Examples Server launches, click the Administration Console link.
3. Log in to the Administration Console.
4. Access the conversion utility by clicking “Convert weblogic.properties” on the left-hand side of the Administration Console.

The conversion utility is a sequence of screens that begins with “Step 1 - Locate weblogic root.”
5. In the “Step 1 - Locate weblogic root” screen, select the directory that contains the `weblogic.properties` file.

The second page of the “Convert weblogic.properties” path appears.
6. Select the root directory of your 5.1 application (that is, the directory that contains all of the application files and subdirectories).
7. Fill in the remaining text fields.
 - a. Admin Server Name: “migrationserver”

- b. Output Directory “c:\banco”
 - c. WebLogic Home
 - d. Name for New Domain “migrationdomain”
8. Click Convert.

The `weblogic.properties` converter converts the application directories in your root directory into a WebLogic Server 8.1 domain. The Administration Console displays the results of the conversion:

```
New Domain name is migrationdomain
*****
Server Name is migrationserver
This server doesn't belong to any cluster
*****
Converting Server properties
Converting Server Debug Properties
Converting WebServer properties
Converting WebApp Component Properties
Converting JDBC Specific properties
Converting CORBA IIOP properties
Converting EJB Specific Properties
--- Warning Source File D:\510sp12\migrationserver\app_banking.jar does not
exist copy the correct file manually after conversion to
C:\banco\applications
Converting StartupClass properties
Converting Shutdown Class properties
Converting MailSession Properties
Converting FileT3 properties
Converting JMS properties
Converting Security Properties
```

Converting the PasswordPolicy properties

Converting User Group and ACL properties

Creating webApp for the servlets registered in the properties file

Startup Scripts for the Server are created in the ResultDir C:\banco

Conversion successful.

Configure the Banking Application for WebLogic Server 8.1

This section discusses the two main steps needed to deploy and run the banking application on WebLogic Server 8.1:

- [Edit the startmigration Script](#) in the banking application's directory.
- [Copy Banking Application Files to the Output Directory](#) specified in the `weblogic.properties` converter.

Edit the startmigration Script

The `weblogic.properties` conversion utility generated a script called `startmigrationdomain` for starting up the banking application's domain. Edit this script to specify additional variables needed to run the upgraded banking application in this new 8.1 domain.

1. Edit the `startmigrationdomain` script, adding the following variables:

```
set APPLICATIONS=%WL51_HOME%\config\migrationdomain\applications
set CLIENT_CLASSES=%WL51_HOME%\config\migrationdomain\clientclasses
set SERVER_CLASSES=%WL51_HOME%\config\migrationdomain\serverclasses
set
BANKING_WEBAPP_CLASSES=D:\banking\510sp12\migrationserver\serverclasses
\examples\tutorials\migration\banking
set
CLOUDSCAPE_CLASSES=%WL51_HOME%\samples\eval\cloudscape\lib\cloudscape.jar
```

2. Append these variables to the classpath in `startmigrationdomain`:

```
CLASSPATH=...%APPLICATIONS%;%CLIENT_CLASSES%;%SERVER_CLASSES%;%BANKING_
WEBAPP_CLASSES%;%CLOUDSCAPE_CLASSES%
```

3. Add the following setting to the `startmigrationdomain` script, making sure to add it before the final `weblogic.Server`:

```
-Dcloudscape.system.home=WL51_HOME\eval\cloudscape\data
```

Copy Banking Application Files to the Output Directory

Copy the application JAR file and the Web application classes and files to the banking application directory.

1. Copy `AccountDetail.jsp`, `error.jsp`, and `login.html` to `C:\banco\applications\DefaultWebApp_migrationserver`.
2. Copy `app_banking.jar` to `C:\banco\applications\`.
3. Copy `AccountDetail.jsp`, `error.jsp`, and `login.html` to `C:\banco\applications\DefaultWebApp_migrationserver`.
4. Copy `BankAppServlet.class` into `C:\banco\applications\DefaultWebApp_migrationserver\WEB-INF\classes`.

Deploy and Run the Banking Application

Start the server for the migration domain by navigating to `c:\banco\` in a command console and entering the command `startmigration`.

To use the application, browse to `http://localhost:7001/banking`.

Log in using the following:

```
username: system
```

```
password: password
```

```
account: 1000
```

Additional Upgrade and Deployment Considerations

The following sections provide additional information that may be useful when you deploy applications on WebLogic Server 8.1. Deprecated features, upgrades, and the important changes that have been made in WebLogic Server 8.1 are noted.

Note: WebLogic Server 8.1 uses PointBase 4.2 as a sample database and does not bundle the Cloudscape database.

- [“Applications and Managed Servers” on page 4-22](#)

- “Reset Default Mime Type in weblogic.xml” on page 4-23
- “Two-Phase Deployment Is the Default Deployment Model” on page 4-23
- “Changes to Internationalization (I18N) Log Files” on page 4-23
- “8.1 Classes Must Be Built Under JDK 1.4” on page 4-24
- “Support for Java Transaction API (JTA)” on page 4-24
- “Changes to Java Database Connectivity (JDBC)” on page 4-24
- “Upgrading Your JVM to JRockit” on page 4-25
- “Changes to JSPs” on page 4-26
- “JVM” on page 4-27
- “Plug-ins Support SSL Communication” on page 4-27
- “Default Queue Names Have Changed” on page 4-27
- “Tips for Using RMI” on page 4-27
- “Security” on page 4-28
- “Standalone HTML and JSPs” on page 4-30
- “Web Components” on page 4-30
- “Define MIME Types for Wireless Application Protocol Applications in web.xml” on page 4-31
- “XML 8.1 Parser and Transformer Are Updated” on page 4-31
- “Deprecated APIs and Features” on page 4-32
- “Removed APIs and Features” on page 4-33

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*.

Reset Default Mime Type in weblogic.xml

WebLogic Server 5.1 provided the `weblogic.httpd.defaultMimeType` parameter to set the default mime-type for a Web application. The default value in 5.1 was `text/plain`.

WebLogic Server 8.1 replaces the parameter with the `default-mime-type` element in `weblogic.xml`, with the default value `null`.

The `weblogic.properties` conversion tool does not migrate this parameter, so if you have it set, you have to reset it manually.

A sample configuration of the `default-mime-type` element in `weblogic.xml`:

```
<weblogic-web-app>
  <container-descriptor>
    <default-mime-type>text/plain</default-mime-type>
  </container-descriptor>
</weblogic-web-app>
```

Two-Phase Deployment Is the Default Deployment Model

By default, WebLogic Server version 8.1 uses a two-phase deployment model includes a prepare phase and an activate phase, which helps prevent inconsistent server states by allowing deployments to be validated before being committed to the server. For more information on this deployment model and other 8.1 deployment features, see [Deploying WebLogic Server Applications](#). If you deploy a 5.1 application in WebLogic Server 8.1 without specifying the deployment model, the server will use the two-phase deployment. For more information, see the WebLogic Server 8.1 [Release Notes](#).

FileServlet Behavior Has Changed

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` setting for determining the source filename. This setting makes it possible to explicitly only serve files from specific directories by mapping the FileServlet to `/dir/*` etc.

See [Setting Up a Default Servlet](#) in *Developing Web Applications for WebLogic Server*.

Changes to Internationalization (I18N) Log Files

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](#).

8.1 Classes Must Be Built Under JDK 1.4

WebLogic Server 8.1 has dependencies on JDK 1.4. If you compile an application after putting the 8.1 `weblogic.jar` in your classpath, the classes must be built under JDK 1.4. This means that your server start script (or `config.xml`, or environment setting) must reference JRockit or JDK 1.4.x.

Support for Java Transaction API (JTA)

JTA has changed as follows:

- WebLogic Server 8.1 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.

Changes to Java Database Connectivity (JDBC)

The initial capacities set for JDBC connection pools in WebLogic Server 5.1 are not persisted in the `config.xml` generated by the `weblogic.properties` converter utility.

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.

- 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.

Upgrading Your JVM to JRockit

When you upgrade a domain to WebLogic Server 8.1, consider upgrading your JVM to JRockit. WebLogic JRockit is a JVM designed for running server-side applications in Windows and Linux running on Intel architectures. For server-side applications, JRockit has these advantages over other virtual machines:

- It employs adaptive optimization, which detects and removes bottlenecks in the deployed application.
- It is designed specifically for the special requirements of server-side applications, which tend to be parallel and thread-intensive, to run for longer periods of time, and not to use graphical interfaces.
- You can monitor JRockit using the WebLogic Server Administration Console.

To switch a WebLogic Server domain to the JRockit JVM:

1. In the server start scripts, set `JAVA_HOME` (or equivalent) shell variables to point to the JRockit root directory. For example, change:

```
@rem Set user-defined variables.
```

```
set JAVA_HOME=WL_HOME\jdk131
```

where `WL_HOME` is the WebLogic Server 5.1 installation directory, to

```
@rem Set user-defined variables.
```

```
set JAVA_HOME=WL_HOME\jrockit81_141_02
```

where `WL_HOME` is the WebLogic Server 8.1 installation directory.

2. Change the domain's `config.xml` to use the JRockit `javac.exe`. For example, change

```
JavaCompiler="WL_HOME\jdk131\bin\javac"
```

where `WL_HOME` is the WebLogic Server 5.1 installation directory, to

```
JavaCompiler=WL_HOME\jrockit81_141_02\bin\javac"
```

where WL_HOME is the WebLogic Server 8.1 installation directory.

3. Remove from server start scripts any switches specific to the Sun JVM. For example, from the start command:

```
echo on "%JAVA_HOME%\bin\java" -hotspot .... weblogic.Server  
delete "-hotspot".
```

4. Start and configure JRockit. See the [Starting and Configuring the WebLogic JRockit JVM](#) section for the appropriate version of the JRockit documentation.

For JRockit platform and user information, see the appropriate version of the JRockit [User Guide](#).

Changes to JSPs

The following sections detail changes to JSP behavior in WebLogic Server 8.1.

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 avoid the error by placing an empty file at the referenced location.

Null Requests

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 8.1 installs both the Java Virtual Machine (JVM), JDK 1.4.1, and the JRockit JVM with the server installation. The `setenv.sh` scripts provided with the server all point to the JDK 1.4.1 JVM. The latest information regarding certified JVMs is available at the [Certifications Page](#). For information about upgrading to JRockit, see “[Upgrading Your JVM to JRockit](#)” on [page 4-25](#).

Plug-ins Support SSL Communication

The communication between the proxy Plug-In and WebLogic Server 5.1 is clear text. WebLogic Server 8.1 supports SSL communication between the plug-ins (Apache, Microsoft IIS, and Netscape) and the back-end WebLogic Server.

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

For more information about 8.1 proxy Plug-Ins, see [Using Web Server Plug-Ins With WebLogic Server](#) at <http://e-docs.bea.com/wls/docs81/plugins/index.html>.

Default Queue Names Have Changed

Default names for execute queues have changed in WebLogic Server 8.1. If you upgrade a configuration that specifies execute queues, the default queue names will automatically alias the new queue names.

Table 4-2 Queue Names

Pre-8.1 Default Queue Names	WebLogic Server 8.1 Default Queue Names
default	weblogic.kernel.Default
__weblogic_admin_html_queue	weblogic.admin.RMI
__weblogic_admin_rmi_queue	weblogic.admin.HTTP

Tips for Using RMI

The following tips are for users upgrading to WebLogic Server 8.1 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 8.1.
- 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 [Using the WebLogic RMI Compiler](#) 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 WebLogic Server 5.1 to WebLogic Server 8.1 with the WebLogic Server 8.1 security functionality is a two-step process involving first upgrading to the WebLogic Server 6.x security functionality and then upgrading from WebLogic Server 6.x to WebLogic Server 8.1.

See “[Upgrading Security Realms from WebLogic Server 5.1 to 6.1](#)” on page 4-28 and “[Upgrading WebLogic Server 6.x Security to Version 8.1](#)” on page 3-17.

See also the security section of the [WebLogic Server Frequently Asked Questions](#).

Upgrading Security Realms from WebLogic Server 5.1 to 6.1

In WebLogic 6.x, WebLogic Server provides a new management architecture for security realms. The management architecture implemented through MBeans allows you to manage security realms through the Administration Console. If you have a security realm from a previous release of WebLogic Server, use the following information to upgrade to the new architecture:

- If you are using the Windows NT, UNIX, or LDAP security realms, use the `Convert weblogic.properties` option under Information and Resources in the Administration Console to convert the security realm to the 6.1 architecture. Note that you can view users, groups, and ACLs in a Windows NT, UNIX, or LDAP security realm in the Administration

Console. However, you still need to use the tools in the Windows NT, UNIX, or LDAP environments to manage users and groups.

- If you are using a custom security realm, follow the steps in *Installing a Custom Security Realm* in the section called *Specifying a Security Realm* in the *Administration Guide* to specify information about how the users, groups, and optional ACLs are stored in your custom security realm.
- The Delegating security realm is no longer supported. If you are using the Delegating security realm, you will have to use another type of security realm to store users, groups, and ACLs.
- If you are using the RDBMS security realm, use one of the following options to convert the security realm:
 - If you did not change the source for the RDBMS security realm, follow the steps in *Configuring the RDBMS Security Realm* in the section called *Specifying a Security Realm* in the *Administration Guide* to instantiate a new class for your existing RDBMS security realm and define information about the JDBC driver being used to connect to the database and the schema used by the security realm. In this case, you are creating a MBean in WebLogic Server for the RDBMS security realm.
 - If you customized the RDBMS security realm, convert your source to use the MBeans. Use the code example in the `\samples\server\src\examples\security\rdbmsrealm` directory as a guide to converting your RDBMS security realm. Once you have converted your RDBMS security realm to MBeans, follow the instructions in *Configuring the RDBMS Security Realm* in the section called *Specifying a Security Realm* in the *Administration Guide* to define information about the JDBC driver being used to connect to the database and the schema used by the security realm.
- The name of the default security realm changed from `WLPropertyRealm` to the File realm. Realm attributes are now stored in the `fileRealm.properties` file instead of the `weblogic.properties` file.
- Redefine your realm and authorization attributes through the Administration Console. The resulting information is stored in the `fileRealm.properties` file.
- It is highly recommended that at the end of installation, you check all security settings to make sure they are the appropriate ones for their environment.
- ACLs can no longer be used to specify security for stand-alone servlets because stand-alone servlets have been completely replaced by Web applications. Web applications

can only be secured using the Web application's deployment descriptors as defined in the Servlet 2.3 specification.

Upgrading WebLogic Server 6.1 Security to Version 8.1

Once you have configured WebLogic Server to use the 6.1 security functionality, see [Upgrading WebLogic Server 6.x to Version 8.1](#) to read about how to upgrade to the WebLogic Server 8.1 security functionality.

Standalone HTML and JSPs

In the original domain provided with WebLogic Server 8.1, 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.

URIs with Extra Spaces Result in a 404

Previous versions of WebLogic Server resolved URIs that contained extra spaces. WebLogic Server 8.1 no longer resolves extra spaces, and a URI request that contains extra spaces will result in a 404.

For example, `http://server:port/mywebapp/foo%20%20` used to resolve to the resource `foo` in the Web application "mywebapp," but beginning with 8.1 it no longer does.

Web Components

The following tips are for users upgrading to WebLogic Server 8.1 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 8.1 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.

- 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`, `JDBCConnectionTimeoutSecs`, `URLRewritingEnabled`)

For more information, see [Deploying Web Applications](#) in *Developing Web Applications for WebLogic Server*.

Define MIME Types for Wireless Application Protocol Applications in `web.xml`

To run a Wireless Application Protocol (WAP) application on WebLogic Server 8.1, you must now specify the MIME types associated with WAP in the `web.xml` file of the Web application. In WebLogic Server 5.1, MIME types were defined in the `weblogic.properties` 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 [web.xml Deployment Descriptor Elements](#) in *Developing Web Applications for WebLogic Server*.

XML 8.1 Parser and Transformer Are Updated

The built-in parser and transformer in WebLogic Server 8.1 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 8.1 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 8.1 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
- 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 8.1 and is replaced by `weblogic.Deployer`. For more information, see [Deploying WebLogic Server Applications](#).
- `weblogic.management.tools.WebAppComponentRefreshTool` and `weblogic.refresh` are both deprecated in this release of WebLogic Server 8.1. They have been replaced by `weblogic.Deployer`.
- Five WebLogic-specific context parameters supported by the Web application container have been deprecated:

- `weblogic.httpd.servlet.reloadCheckSecs` (has been replaced with the `weblogic.xml` `servlet-reload-check-secs`).
- `weblogic.httpd.servlet.classpath` (instead use the manifest classpath, or `WEB-INF/lib` or `WEB-INF/classes`, or virtual directories).
- `weblogic.httpd.clientCertProxy` (not yet replaced in `weblogic.xml`).
- `weblogic.httpd.defaultServlet` (instead define a `servlet-mapping` with `pattern=/*` in `weblogic.xml`).
- `weblogic.httpd.inputCharset` (instead use `weblogic.xml` `charset-params`).

For more information about `weblogic.xml`, see [weblogic.xml Deployment Descriptor Elements](#) in *Developing Web Applications for WebLogic Server*.

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

Upgrading WebLogic Server 5.1 to Version 8.1

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

Upgrading WebLogic Server 8.1 Service Pack 2 to Version 8.1 Service Pack 3

This section contains information about upgrading your domain from WebLogic Server 8.1 Service Pack 2 to WebLogic Server 8.1 Service Pack 3.

New SDKs Bundled with WebLogic Platform 8.1

The WebLogic Platform 8.1 SP3 installation includes new versions of the Java 2 SDKs, including:

- Sun Java 2 SDK 1.4.2_04
- BEA WebLogic JRockit SDK 1.4.2_04

If you are upgrading a domain from SP2 to SP3, the script in that domain that starts WebLogic Server needs to be modified to point to the location in which the new SDK has been installed. This script is located in the domain's root directory. Depending on the type of domain you are upgrading, this script by default is named either `setDomainEnv` or `startWebLogic`.

To modify this script, update the value of the `JAVA_HOME` variable. For example:

```
set JAVA_HOME=C:\bea\jrockit81sp3_142_04
```

It is also recommended that you update your Workshop applications, application startup scripts, and silent configuration scripts to reference the new Sun or JRockit SDK directory. For information about updating your Workshop applications to use the new SDK, see [WebLogic Workshop](#).

Note: WebLogic Platform 8.1 SP3 adds support for HP-UX SDK 1.4.2.03 on HP-UX PA-RISC 11.0 and 11i systems.

Upgrading WebLogic Server 8.1 Service Pack 2 to Version 8.1 Service Pack 3

Mapping `weblogic.properties` to 6.x `config.xml` Configuration Attributes

When upgrading from versions prior to WebLogic Server 6.0, use the `weblogic.properties` converter in the Administration Console to convert `weblogic.properties` parameters to `config.xml` attributes.

Some configuration attributes have changed in WebLogic Server versions subsequent to 6.x. The recommended upgrade path

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

<code>weblogic.properties</code> file Property	.xml Configuration Attribute
<code>weblogic.administrator.email</code>	<code>config.xml</code> : EmailAddress (Administrator element)
<code>weblogic.administrator.location</code>	<code>config.xml</code> : Notes (freeform, optional) (Administrator element)
<code>weblogic.administrator.name</code>	<code>config.xml</code> : Name (Administrator element)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
weblogic.administrator.phone	config.xml: PhoneNumber (Administrator element)
weblogic.cluster.defaultLoadAlgorithm	config.xml: DefaultLoadAlgorithm (Cluster element)
weblogic.cluster.multicastAddresses	config.xml: MulticastAddress (Cluster element)
weblogic.cluster.multicastTTL	config.xml: MulticastTTL (Cluster element)
weblogic.cluster.name	config.xml Cluster Address (Cluster element)
weblogic.httpd.authRealmName	config.xml: AuthRealmName (WebAppComponent element)
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)
weblogic.httpd.defaultServlet	web.xml: define a servlet-mapping with the URL pattern of / <servlet-mapping> element

weblogic.properties file Property	.xml Configuration Attribute
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)
weblogic.httpd.keepAlive.enable	config.xml : KeepAliveEnabled (WebServer element)
weblogic.httpd.logFileBufferKBytes	config.xml : LogFileBufferKBytes (WebServer element)
weblogic.httpd.logFileFlushSecs	config.xml : LogFileFlushSecs (WebServer element)
weblogic.httpd.logFileFormat	config.xml : LogFileFormat (WebServer element)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
weblogic.httpd.logFileName	config.xml: LogFileName (WebServer element)
weblogic.httpd.logRotationPeriod Mins	config.xml: LogRotationTimeBegin (WebServer element)
weblogic.httpd.logRotationPeriod Mins	config.xml: LogRotationPeriodMins (WebServer element)
weblogic.httpd.logRotationType	config.xml: LogRotationType (WebServer element)
weblogic.httpd.maxLogFileSizeKBy tes	config.xml: MaxLogFileSizeKBytes (WebServer element)
weblogic.httpd.mimeType	web.xml: mime-type (<mime-mapping> element)
weblogic.httpd.postTimeoutSecs	config.xml: PostTimeoutSecs (WebServer element)
weblogic.httpd.servlet.extension CaseSensitive	config.xml: ServletExtensionCaseSensitive (WebAppComponent element)
weblogic.httpd.servlet.reloadChe ckSecs	config.xml: ServletReloadCheckSecs (WebAppComponent element)
weblogic.httpd.servlet.SingleThr eadedModelPoolSize	config.xml: SingleThreadedServletPoolSize (WebAppComponent element)

weblogic.properties file Property	.xml Configuration Attribute
weblogic.httpd.session.cacheEntries	weblogic.xml: CacheSize <param-name>/<param-value> element pair
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.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.enabled	weblogic.xml: CookiesEnabled <param-name>/<param-value> element pair
weblogic.httpd.session.debug	weblogic.xml: SessionDebuggable <param-name>/<param-value> element pair

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
weblogic.httpd.session.enable	weblogic.xml: TrackingEnabled <param-name>/<param-value> element pair
weblogic.httpd.session.invalidat ionintervalSecs	weblogic.xml: InvalidationIntervalSecs <param-name>/<param-value> element pair
weblogic.httpd.session.jdbc.conn TimeoutSecs	weblogic.xml: JDBCConnectionTimeoutSecs <param-name>/<param-value> element pair
weblogic.httpd.session.persisten tStoreDir	weblogic.xml: PersistentStoreDir <param-name>/<param-value> element pair
weblogic.httpd.session.persisten tStorePool	weblogic.xml: PersistentStorePool <param-name>/<param-value> element pair
weblogic.httpd.session.persisten tStoreShared	weblogic.xml: SessionPersistentStoreShared <param-name>/<param-value> element pair
weblogic.httpd.session.persisten tStoreType	weblogic.xml: PersistentStoreType <param-name>/<param-value> element pair
weblogic.httpd.session.sessionID Length	weblogic.xml: IDLenght <param-name>/<param-value> element pair

weblogic.properties file Property	.xml Configuration Attribute
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
weblogic.httpd.session.URLRewriting.enable	weblogic.xml: URLRewritingEnabled <param-name>/<param-value> element pair
weblogic.httpd.tunneling.clientPingSecs	config.xml: TunnelingClientPingSecs (Server element)
weblogic.httpd.tunneling.clientTimeoutSecs	config.xml: TunnelingClientTimeoutSecs (Server element)
weblogic.httpd.tunnelingenabled	config.xml TunnelingEnabled (Server element)
weblogic.httpd.URLResource	config.xml: URLResource (WebServer element)
weblogic.iiop.password	config.xml: DefaultIIOPPassword (Server element)
weblogic.iiop.user	config.xml: DefaultIIOPUser (Server element)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
<p>weblogic.jdbc.connectionPool</p> <ul style="list-style-type: none"> • url=<i>URL for JDBC Driver</i> • driver=<i>full package name for JDBC driver</i> • loginDelaySecs=<i>seconds between connections</i> • initialCapacity=<i>initial number of JDBC connections</i> • maxCapacity=<i>maximum number of JDBC connections</i> • capacityIncrement=<i>increment interval</i> • allowShrinking=<i>true to allow shrinking</i> • shrinkPeriodMins=<i>interval before shrinking</i> • testTable=<i>name of table for autorefresh test</i> • refreshTestMinutes=<i>interval for autorefresh test</i> • testConnsOnReserve=<i>true to test connection at reserve</i> • testConnsOnRelease=<i>true to test connection at release</i> • props=<i>props for JDBC connection</i> 	<p>URL</p> <p>DriveName</p> <p>LoginDelaySeconds</p> <p>InitialCapacity</p> <p>MaxCapacity</p> <p>CapacityIncrement</p> <p>AllowShrinking</p> <p>ShrinkPeriodMinutes</p> <p>TestTableName</p> <p>RefreshMinutes</p> <p>TestConnectionsOnReserve</p> <p>TestConnectionsOnRelease</p> <p>Properties</p> <p>JDBCConnectionPool Element</p> <p>ConnLeakProfilingEnabled</p> <p>ACLName</p> <p>CapacityEnabled</p> <p>SupportsLocalTransaction</p> <p>KeepLogicalConnOpenOnRelease</p> <p>Password</p>

weblogic.properties file Property	.xml Configuration Attribute
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>> <ul style="list-style-type: none"> • 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)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
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 (JMSTopic element)
weblogic.jms.topicSessionPool	config.xml: ConnectionConsumer ConnectionFactory ListenerClass AcknowledgeMode SessionsMaximum Transacted (JMSSessionPool element)
weblogic.jndi.transportableObjectFactories	config.xml: JNDITransportableObjectFactoryList (Server element)
weblogic.login.readTimeoutMillisSSL	config.xml LoginTimeoutMillis (SSL element)
weblogic.security.audit.provider	config.xml AuditProviderClassName (Security element)

weblogic.properties file Property	.xml Configuration Attribute
weblogic.security.certificate.authority	config.xml ServerCertificateChainFileName (SSL element)
weblogic.security.certificate.server	config.xml: ServerCertificateFileName (SSL element)
weblogic.security.certificateCacheSize	config.xml: CertificateCacheSize (SSL element)
weblogic.security.clientRootCA	config.xml: TrustedCAFileName (SSL element)
weblogic.security.disableGuest	config.xml: GuestDisabled (Security element)
weblogic.security.enforceClientCertificate	config.xml: ClientCertificateEnforced (SSL element)
weblogic.security.key.export.lifespan	config.xml: ExportKeyLifespan (SSL element)
weblogic.security.key.server	config.xml: ServerKeyFileName (SSL element)
weblogic.security.ldaprealm.authentication	config.xml: AuthProtocol (LDAPRealm element)
weblogic.security.ldaprealm.credential	config.xml: Credential (LDAPRealm element)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
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.security.ldaprealm.userDN	config.xml : UserDN (LDAPRealm element)

weblogic.properties file Property	.xml Configuration Attribute
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.security.realm.cache.auth.size	config.xml: AuthenticationCacheSize (CachingRealm element)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
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.permission.enable	config.xml: PermissionCacheEnable (CachingRealm element)
weblogic.security.realm.cache.permission.size	config.xml: PermissionCacheSize (CachingRealm element)

weblogic.properties file Property	.xml Configuration Attribute
weblogic.security.realm.cache.permission.ttl.negative	config.xml: PermissionCacheTTLNegative (CachingRealm element)
weblogic.security.realm.cache.permission.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)
weblogic.security.SSL.ciphersuite	config.xml Ciphersuites (SSL element)
weblogic.security.ssl.enable	config.xml: Enabled (SSL element)
weblogic.security.SSL.hostnameVerifier	config.xml HostnameVerifier (SSL element)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
weblogic.security.SSL.ignoreHostNameVerification	config.xml HostNameVerificationIgnored (SSL element)
weblogic.security.SSLHandler.enable	config.xml: HandlerEnabled (SSL element)
weblogic.security.unixrealm.authProgram	config.xml: AuthProgram (UnixRealm element)
weblogic.system.AdministrationPort	config.xml AdministrationPort (Server element)
weblogic.system.bindAddr	config.xml: ListenAddress (Server element)
weblogic.system.defaultProtocol	config.xml: DefaultProtocol (Server element)
weblogic.system.defaultSecureProtocol	config.xml: DefaultSecureProtocol (Server element)
weblogic.system.enableConsole	config.xml: StdoutEnabled (Kernel element)
weblogic.system.enableIIOP	config.xml: IIOPEnabled (Server element)
weblogic.system.enableReverseDNSLookups	config.xml: ReverseDNSAllowed (Server element)

weblogic.properties file Property	.xml Configuration Attribute
weblogic.system.enableSetGID,	config.xml: PostBindGID
weblogic.system.enableSetUID,	config.xml: PostBindUIDEnabled
weblogic.system.enableTGIOP	config.xml TGIOPEnabled (Server element)
weblogic.system.helpPageURL	config.xml HelpPageURL (Server element)
weblogic.system.home	config.xml: RootDirectory (Server element)
weblogic.system.ListenPort	config.xml ListenPort (Server element)
weblogic.system.logFile	config.xml: FileName (Log element)
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)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes

weblogic.properties file Property	.xml Configuration Attribute
weblogic.system.maxLogFileSize	config.xml: FileMinxSize (Log element)
weblogic.system.nativeIO.enable	config.xml: NativeIOEnabled (Server element)
weblogic.system.nonPrivGroup	config.xml PostBindGID (UnixMachine element)
weblogic.system.nonPrivUser	config.xml PostBindUID (UnixMachine element)
weblogic.system.percentSocketReaders	config.xml: ThreadPoolPercentSocketReaders (Kernel element)
weblogic.system.readTimeoutMillis	config.xml: LoginTimeoutMillis (Server element)
weblogic.system.SSL.useJava	config.xml: UseJava (SSL element)
weblogic.system.SSLListenPort	config.xml: ListenPort (SSL element)
weblogic.system.startupFailureIsFatal	config.xml FailureIsFatal (StartupClass element)

weblogic.properties file Property	.xml Configuration Attribute
weblogic.system.user	config.xml: SystemUser (Security element)
weblogic.system.weight	config.xml ClusterWeight (Server element)

Mapping weblogic.properties to 6.x config.xml Configuration Attributes