



BEA WebLogic Server™

BEA WebLogic Server Partners' Guide

Release 8.1
Revised: April 12 2004

Copyright

Copyright © 2003 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.

Contents

About This Document

Audience	v
e-docs Web Site	vi
How to Print the Document	vi
Related Information	vi
Contact Us!	vii
Documentation Conventions	vii

1. Distributing WebLogic Server

Upgrade an ISV License for Distributing a Newer Version of WebLogic Server	1-2
Enroll in the BEA Partner Program	1-2
Install the Partner Development Kit	1-3
Install the ISV License	1-3
Step 1: Preparing to Install an ISV License	1-4
Step 2: Extracting the License Data and Linking WebLogic Server Files	1-5
Step 3: Updating the WebLogic Server License	1-5
Next Steps: Configuring and Installing Your Application and WebLogic Server	1-6

2. Customizing WebLogic Server Configuration Files

Customizing the config.xml File	2-1
Pre-Configuring WebLogic Server Resources	2-3
Deployment of Application Components	2-4
Example Configuration	2-5

Domain Configuration	2-5
Database Connections	2-6
Messaging Resources	2-7
Application Components	2-9
Basic Server Setup	2-10
Security Realm	2-10
Customizing Files for Compatibility Security	2-12

3. Creating Startup Scripts

Choosing a Startup Technique	3-1
Using WebLogic Server Start Scripts	3-4
Creating Scripts that Invoke Node Managers	3-5
Creating Windows Services for Instances of WebLogic Server	3-6
Installing Your Startup Scripts	3-7

4. Using JDBC Profiling MBeans

Enabling JDBC Profiling	4-1
Accessing JDBC Profiles	4-3

About This Document

This document describes how to acquire and install an Independent Software Vendors (ISV) license, which enables you to bundle BEA's core technologies with your application and distribute both items as a single product. The document also suggests development techniques for bundling BEA WebLogic Server™ with your applications.

The document is organized as follows:

- [Chapter 1, “Distributing WebLogic Server,”](#) which describes how to acquire and install an ISV license and specifies which WebLogic Server files must be included in your distribution.
- [Chapter 2, “Customizing WebLogic Server Configuration Files,”](#) which highlights typical modifications that partners and ISVs make to the WebLogic Server configuration files that they distribute with their applications.
- [Chapter 3, “Creating Startup Scripts,”](#) which describes how to create and install startup scripts.
- [Chapter 4, “Using JDBC Profiling MBeans,”](#) which describes how to enable and use JDBC profiling.

Audience

This document is written for independent software vendors (ISVs) and other developers who are interested in creating custom applications that use BEA WebLogic Server core technologies. It is assumed that readers are already familiar with the BEA WebLogic Server platform, other guides in the WebLogic Server documentation set, and the Java programming language.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the WebLogic Server Product Documentation page at <http://e-docs.bea.com>.

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>.

Related Information

The BEA corporate Web site provides all documentation for WebLogic Server. The following BEA WebLogic Server documentation contains information that is relevant to understanding how to extend WebLogic Server.

- BEA WebLogic Server Documentation (available online):
 - *Administration Guide*
 - *Programming Guides*
 - WebLogic Server API
- The Sun Microsystems, Inc. Java site at <http://java.sun.com/>

For more information about BEA WebLogic Server and Java, refer to the Bibliography at <http://edocs.bea.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 your 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.

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
...	<p>Indicates one of the following in a command line:</p> <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

Distributing WebLogic Server

Instead of requiring your customers to purchase, install, and maintain both your application and a J2EE application server, you can bundle BEA's core technologies with your application and distribute both items as a single product. The plug-and-play environment of WebLogic Server makes it an ideal choice for integration with your product.

To distribute WebLogic Server, you must obtain and install a special license called an **ISV license**. (You do not need an ISV license to develop your application or to configure WebLogic Server.) Installing an ISV license for a WebLogic Server modifies the server and inextricably links files. Your distribution must include these modified WebLogic Server files.

To set up WebLogic Server files that you can distribute, complete the following tasks:

- [“Upgrade an ISV License for Distributing a Newer Version of WebLogic Server” on page 1-2](#)
- [“Enroll in the BEA Partner Program” on page 1-2](#)
- [“Install the Partner Development Kit” on page 1-3](#)
- [“Install the ISV License” on page 1-3](#)
- [“Next Steps: Configuring and Installing Your Application and WebLogic Server” on page 1-6](#)

Upgrade an ISV License for Distributing a Newer Version of WebLogic Server

If you distributed an older version of WebLogic Server, you must upgrade your ISV license before distributing a newer version of WebLogic Server.

You do not need to upgrade your ISV license when you upgrade to a newer WebLogic Server service pack. For example, if you distributed WebLogic Server 8.1 sp1 and you want your subsequent distributions to include WebLogic Server 8.1 sp2, you do not need to upgrade your ISV license.

In addition, if your customers upgrade their installation with a WebLogic Server service pack release that they purchase from BEA, the license file that you distributed to them will be updated to enable the new service pack release. You do not need to distribute a new license file to customers who install new service packs of WebLogic Server on their own.

To upgrade your existing ISV license:

1. Send an email to licensing@bea.com. In the email, request a new ISV license and attach the `isv.jar` file that contains your old license. If you are upgrading from WebLogic Server 5.1, attach your old license file.
2. Install the WebLogic Server 8.1 Partner Development Kit.
3. After you receive your new `isv.jar` file from BEA, install the new ISV license as described in [“Install the ISV License” on page 1-3](#).
4. Update your installer to include the new WebLogic Server files.

Enroll in the BEA Partner Program

BEA Systems, Inc. manages its relationships with Independent Software Vendors (ISVs) and Application Software Providers (ASPs) through the Partner Program. If you have not already enrolled in the BEA Partner Program:

1. Verify that your target platforms are certified for use with WebLogic Server by referring to the BEA WebLogic Server Platform Support page, <http://e-docs.bea.com/wls/certifications/certifications/index.html>.
2. Visit the BEA Partner Program Web site to learn about the program and to enroll. You can use the following URL to access the site: <http://www.bea.com/partners>.

Install the Partner Development Kit

After you enroll in the program, BEA ships a CD collection of all major BEA products. When the Partner Development Kit arrives, install the software from the CDs. For information on installing WebLogic Server, refer to the [Installation Guide](#) on the BEA documentation Web site, <http://edocs.bea.com>.

Caution: If you already have BEA products installed on the computer that you want to host your distributable WebLogic Server, **back up your current `BEA_HOME\license.bea` file** before installing the Partner Development Kit. For more information about the BEA home directory and the `license.bea` file, refer to [BEA Home Directory](#) in the *Installing BEA WebLogic Server* guide.

Instead of waiting for the CDs, you can download BEA software from the BEA Systems Download Center, <http://commerce.beasys.com>. If you have an active WebSUPPORT account, you can use your WebSUPPORT login password for software downloads.

Install the ISV License

After verifying your eligibility for the Partner Program, BEA sends an email that includes your customized ISV license in an attached file named `isv.jar`. This section describes how to install the ISV license file **for WebLogic Server version 8.1 only**. If you are installing an ISV license for other versions of WebLogic Server, please consult the relevant installation instructions for your software version.

There are three main steps to installing an ISV license:

- [Step 1: Preparing to Install an ISV License](#)
- [Step 2: Extracting the License Data and Linking WebLogic Server Files](#)
- [Step 3: Updating the WebLogic Server License](#)

Step 1: Preparing to Install an ISV License

Before you install an ISV license file, do the following:

1. If you have not already done so, install WebLogic Server version 8.1 as described in the previous section, [“Install the Partner Development Kit” on page 1-3](#).

Note the location of the BEA home directory that the BEA installer uses. It contains a `license.bea` file, which will be updated in subsequent steps of this process. For more information about the BEA home directory and the `license.bea` file, refer to [BEA Home Directory](#) in the *Installing BEA WebLogic Platform* guide.

2. Copy the `isv.jar` file from your email to the BEA home directory that the installer used.
3. Open a command shell and change directories to `BEA_HOME`, where `BEA_HOME` is the name of your BEA home directory.
4. Run one of the following scripts to set environment variables:

- `WL_HOME\server\bin\setWLSEnv.cmd` (Windows systems)
- `WL_HOME/server/bin/setWLSEnv.sh` (UNIX systems)

where `WL_HOME` is the directory in which you installed WebLogic Server. By default, this directory is directly below the BEA home directory.

For example, in a `bash` shell on UNIX, enter the following command:

```
[me@workstation]$ . WL_HOME/server/bin/setWLSEnv.sh
```

5. Add `isv.jar` to the computer's `CLASSPATH` by entering one of the following commands:
 - `set CLASSPATH=./isv.jar;%CLASSPATH%` (Windows systems)
 - `export CLASSPATH=./isv.jar:$CLASSPATH` (UNIX systems)

You are now ready to extract the ISV license data and link it to WebLogic Server files.

Step 2: Extracting the License Data and Linking WebLogic Server Files

To extract the ISV license data and link it to WebLogic Server files, enter one of the following commands from `BEA_HOME`:

- `java -Xmx128m -Dbea.home=BEA_HOME
-Dbea.jar=WL_HOME\server\lib\weblogic.jar install` (Windows systems)
- `java -Xmx128m -Dbea.home=BEA_HOME
-Dbea.jar=WL_HOME/server/lib/weblogic.jar install` (UNIX systems)

where `BEA_HOME` is an absolute pathname for your BEA home directory, and `WL_HOME` is an absolute pathname for the directory in which you installed WebLogic Server.

Caution: Do not interrupt this process once it has started.

The command generates a file named `BEA_HOME\license_isv.bea`, which contains the ISV license data. It also links files within the `WL_HOME` directory to the specific ISV license. Only the files in the `WL_HOME` directory that you specified will be able to use the ISV license data that you extracted to `license_isv.bea`.

Note: With some platforms and JDKs, you might encounter an "Out of Memory Error." To address this error, increase the value for the `-Xmx` argument (which sets the maximum heap size in megabytes) and run the command again. For example, `-Xmx150m` increases the default heap size to 150 megabytes.

To complete the process for installing an ISV license, you must update the Weblogic Server license with the data in `license_isv.bea`.

Step 3: Updating the WebLogic Server License

To update the `license.bea` file with the newly generated `license_isv.bea` file, run one of the following commands from `BEA_HOME`:

- `UpdateLicense license_isv.bea` (Windows systems)
- `sh UpdateLicense.sh license_isv.bea` (UNIX systems)

The `UpdateLicense` command merges the `license_isv.bea` file with the `license.bea` file. After you run `UpdateLicense`, you do not need to keep the `license_isv.bea` file.

Next Steps: Configuring and Installing Your Application and WebLogic Server

After you install your ISV license:

1. Start the ISV-licensed WebLogic Server.

See [“Starting and Stopping Servers”](#) in the *Administration Console Online Help*.

2. Configure the server components and develop and deploy your application.

See:

- [“Developing WebLogic Server Applications”](#)
- [“Creating, Configuring, and Monitoring Servers”](#) in the Administration Console Online Help (adding resources to a domain that is currently active)
- [“Extending Domains”](#) in the *Creating WebLogic Configurations Using the Configuration Wizard* guide (adding resources to a domain that is inactive)

3. (Optional) Create customized management components.

See the [“Extending the Administration Console”](#) guide and the [“Programming WebLogic Management Services with JMX”](#) guide.

4. Create scripts for starting WebLogic Server and your application.

See [“Creating Startup Scripts”](#) on page 3-1.

5. Use the Template Builder to save your application, server configuration, and startup scripts as a configuration template.

See [“Creating Configuration Templates Using the Template Builder”](#) in the *Creating WebLogic Configurations Using the Configuration Wizard* guide.

6. Configure the WebLogic Server installation program and the Configuration Wizard to run in silent mode.

Silent-mode is a way of setting installation and domain configurations only once and then using those configurations to duplicate the installation on many machines. The installation program installs the BEA license file, the WebLogic Server software (including the Configuration Wizard), and your configuration template. The Configuration Wizard creates domains based on your configuration templates.

While running in silent mode, the installer and the Configuration Wizard read installation options from configuration files that you create. The programs do not display configuration options during the installation process. Silent-mode installation works on both Windows and UNIX systems.

In previous releases of WebLogic Server, you could optionally run the Configuration Wizard as part of the silent installation process. In WebLogic Server 8.1, the Configuration Wizard cannot automatically be run with the WebLogic Server installation program. It must be run separately. However, you can run both the installation program and the Configuration Wizard in silent mode from the same script.

For more information, see [“Installing WebLogic Platform Using Silent-Mode Installation”](#) in the *Installation Guide* and [“Starting in Silent Mode”](#) in *Creating WebLogic Configurations Using the Configuration Wizard*.

Make sure that you configure the installer to include the BEA license file (BEA_HOME\license.bea) and the WL_HOME\lib\weblogic.jar file that you specified in [“Step 3: Updating the WebLogic Server License”](#) on page 1-5. If you do not install both of the files that you specified, your embedded WebLogic Server will not start.

You can use this same license.bea-weblogic.jar pair for all of your licensed installations.

Distributing WebLogic Server

Customizing WebLogic Server Configuration Files

WebLogic Server stores configuration information, such as security credentials and the list of deployable resources and applications, in a set of configuration files.

The following sections highlight typical modifications that partners and ISVs make to the WebLogic Server configuration files that they distribute with their applications:

- [Customizing the config.xml File](#)
- [Example Configuration](#)
- [Customizing Files for Compatibility Security](#)

Customizing the config.xml File

The `config.xml` file defines the majority of configuration settings for all WebLogic Servers in a management domain. For example, `config.xml` controls all details of a given domain, including the name, number and configuration of servers and cluster; the list of deployable resources and applications; and the mapping of deployable resources and applications to servers and clusters.

Usually, BEA recommends that you use such WebLogic Server tools as the Administration Console, the `weblogic.Admin` utility, or the Configuration Wizard to modify the `config.xml` file. Partners, however, may need to edit this file directly in order to customize an installation.

The following sections highlight elements of the `config.xml` file that partners might modify for their installations:

- [“Pre-Configuring WebLogic Server Resources”](#) on page 2-3
- [“Deployment of Application Components”](#) on page 2-4

If you are unfamiliar with the role of the `config.xml` file or management domains, refer to the following:

- [“Directory Structure”](#) in *Configuring and Managing WebLogic Server*
- [“Understanding Cluster Configuration and Application Deployment”](#) in *Using WebLogic Server Clusters*

If you are unfamiliar with editing `config.xml` directly, see the [WebLogic Server Configuration Reference](#), which provides conventions for editing `config.xml`.

Pre-Configuring WebLogic Server Resources

Partner applications typically rely on several WebLogic Server resources, each of which is defined in the `config.xml` file. [Table 2-1](#) provides an overview of the `config.xml` elements that partners typically use to pre-configure WebLogic Server resources.

Table 2-1 WebLogic Server Resource Definitions in the config.xml File

Resources	config.xml Elements	Notes
Domain	Domain	To act as a cohesive unit, all WebLogic Servers that host a component of your application must reside within a single WebLogic Server administrative domain.
Server Names and Connection Information	Server	<p>Partner applications can be configured to access one or more WebLogic Server names, IP addresses, and/or port numbers, or, if necessary for your application, you can hard-code a WebLogic Server domain to use specific server names and connection ports.</p> <p>IP Addresses can be configured dynamically by the application installer and embedded into a <code>config.xml</code> before installing the configuration.</p>
JDBC Datasources	JDBCConnectionPool JDBCDataSource JDBCMultiPool JDBCTxDataSource	Partner applications that install WebLogic Server also frequently install an RDBMS or other datastore for maintaining the application data. If your product installer installs a datastore along with the application, you may want to pre-configure the installed WebLogic Server to set up a default datasource and connection pool for the datastore.

Deployment of Application Components

Partner applications can also be installed by adding the necessary elements to `config.xml`. Installing an application into a pre-configured WebLogic Server, however, requires coordination between the `config.xml` settings and the installed location of application component files (`.war`, `.jar`, `.html` and so forth).

Table 2-2 provides an overview of elements used to pre-deploy application components within WebLogic Server. See “Example Configuration” on page 2-5 for an example of how these elements correspond to the installed location of actual application component files.

Table 2-2 Application Component Definitions in the config.xml File

Components	config.xml Elements	Notes
Startup Classes	StartupClass	WebLogic Server startup classes can be used to initialize resources required by other components of the partner application.
Webserver	WebServer	Web applications typically require standard web resources, such as static <code>.html</code> content, in addition to business logic. Use the <code>config.xml</code> file to configure the default location of these static files for the application.
Web Applications	Application	EAR and WAR files can be stored anywhere in your application directory or the WebLogic Server directory. Reference the final installed location from within <code>config.xml</code> to deploy the application on startup.

Example Configuration

When you install WebLogic Server, by default you also install the Avitek Medical Records sample domain. This sample domain includes a server configuration that defines resources for database connectivity and messaging. The domain also contains a enterprise applications that include EJBs and Web applications.

The following sections highlight key aspects of the `config.xml` file that configures the Avitek Medical Records domain. The file is located in the root directory of the domain's Administration Server: `WL_HOME\samples\domains\medrec\config.xml` where `WL_HOME` is the directory in which you installed WebLogic Server:

- [“Domain Configuration” on page 2-5](#)
- [“Database Connections” on page 2-6](#)
- [“Messaging Resources” on page 2-7](#)
- [“Application Components” on page 2-9](#)
- [“Basic Server Setup” on page 2-10](#)
- [“Security Realm” on page 2-10](#)

Domain Configuration

The parent element in the `config.xml` file, `<Domain>`, provides the configuration for the `medrec` domain. All of the application's servers, resources, and components are defined within this element.

Listing 2-1 Parent Element

```
<Domain
  Name="medrec "
  ConfigurationVersion="8.1.0.0"
>
```

Database Connections

The Avitek Medical Records domain defines two JDBC connection pools and one transactional data source. Each connection pool connects to a different type of database.

The elements in the `config.xml` file include information on how to connect to the database, definitions for the database driver, credentials for logging in to the database, and capacity properties of the connection pool.

Each `JDBCConnectionPool` element also lists the server instances to which it has been targeted. The connection pools are a domain-wide resource: they can be targeted to any server in the domain and used by any application that is deployed on one of those servers.

Listing 2-2 Elements that Configure Database Connections

```
<!-- PointBase -->
<JDBCConnectionPool
  CapacityIncrement="1"
  DriverName="com.pointbase.jdbc.jdbcUniversalDriver"
  InitialCapacity="1"
  MaxCapacity="10"
  Name="MedRecPool-PointBase"
  Password="MedRec"
  Properties="user=MedRec"
  RefreshMinutes="0"
  ShrinkPeriodMinutes="15"
  ShrinkingEnabled="true"
  Targets="MedRecServer"
  TestConnectionsOnRelease="false"
  TestConnectionsOnReserve="false"
  URL="jdbc:pointbase:server://localhost/demo"
/>

<!-- Oracle -->
<JDBCConnectionPool
  CapacityIncrement="2"
  DriverName="oracle.jdbc.driver.OracleDriver"
  InitialCapacity="4"
  LoginDelaySeconds="1"
  MaxCapacity="10"
  Name="MedRecPool-Oracle"
  Password="tiger"
  Properties="user=scott"
  RefreshMinutes="10"
  ShrinkPeriodMinutes="15"
```



```

    ShrinkingEnabled="true"
    Targets=" "
    TestConnectionsOnRelease="false"
    TestTableName="dual"

URL="jdbc:oracle:thin:@my-oracle-server:my-oracle-server-port:my-oracle-sid"
/>

<JDBCTxDataSource
    JNDIName="MedRecTxDataSource"
    Name="MedRecTxDataSource"
    PoolName="MedRecPool-PointBase"
    Targets="MedRecServer"
    EnableTwoPhaseCommit="true"
/>

```

Messaging Resources

The Avitek Medical Records domain contains JMS messaging resources for distributing messages between the applications in the domain.

Note that one of the `JMSJDBCStore` elements is surrounded by comment tags `<!-->`. Because it is surrounded by comment tags, the element is invisible to the Administration Console and other utilities that manage WebLogic Server. In addition, the Administration Server ignores the element and therefore the JDBC store that the element describes is unavailable to the domain.

The comment tags were added by editing the `config.xml` file in a text editor. BEA utilities such as the Administration Console do not use comment tags to hide or disable resources. To make the JDBC store available to the domain:

1. Stop the Administration Server.
2. Open the `config.xml` file in a text editor and remove the comment tags.
3. Restart the Administration Server.

Listing 2-3 Elements for Configuring JMS Resources

```
<JMSJDBCStore
  ConnectionPool="MedRecPool-PointBase"
  Name="MedRecJMSJDBCStore"
  PrefixName="MedRec"
/>

<!-- For Oracle user Scott
<JMSJDBCStore
  ConnectionPool="MedRecPool-Oracle"
  Name="MedRecJMSJDBCStore"
  PrefixName="Scott"
/>
-->

<JMSServer
  Name="MedRecJMSServer"
  Store="MedRecJMSJDBCStore"
  Targets="MedRecServer"
>

  <JMSQueue
    JNDIName="jms/REGISTRATION_MDB_QUEUE"
    Name="jms/REGISTRATION_MDB_QUEUE" />

  <JMSQueue
    JNDIName="jms/MAIL_MDB_QUEUE"
    Name="jms/MAIL_MDB_QUEUE" />

  <JMSQueue
    JNDIName="jms/XML_UPLOAD_MDB_QUEUE"
    Name="jms/XML_UPLOAD_MDB_QUEUE" />

</JMSServer>
```

Application Components

The Avitek Medical Records domain includes three enterprise applications: `medrecEar`, `physicianEar`, `opc.ear`, and `startupEar`. The `physicianEar` enterprise application includes Web applications and EJBs. On Windows, the element in [Listing 2-4](#) configures the `physicianEar` application.

Note that the `c:/bea/wlserver810` portion of the application component path is determined during the WebLogic Server installation, while the remaining portion of the path is hard-coded. Your application installer can use a similar technique to install application components in a subdirectory unrelated to WebLogic Server, if necessary.

Listing 2-4 Elements for Configuring Applications

```
<!-- MedRec Enterprise Applications -->
<Application
  Name="MedRecEAR"
  Deployed="true"
  Path="c:/bea/wlserver810/samples/server/medrec/build/medrecEar"
  StagingMode="nostage"
  TwoPhase="true"
  LoadOrder="1">

  <WebAppComponent Name="AdminWAR" Targets="MedRecServer" URI="adminWebApp"/>
  <WebAppComponent Name="MainWAR" Targets="MedRecServer" URI="mainWebApp"/>
  <WebAppComponent Name="PatientWAR" Targets="MedRecServer"
    URI="patientWebApp"/>
  <EJBComponent Name="EntityEJB" Targets="MedRecServer" URI="entityEjbs"/>
  <EJBComponent Name="MdbEJB" Targets="MedRecServer" URI="mdbEjbs"/>
  <EJBComponent Name="SessionEJB" Targets="MedRecServer" URI="sessionEjbs"/>
  <EJBComponent Name="WebServicesEJB" Targets="MedRecServer"
    URI="webServicesEjb"/>
  <WebServiceComponent Name="WebServicesWAR" Targets="MedRecServer"
    URI="ws_medrec"/>
</Application>
```

Basic Server Setup

The Avitek Medical Records domain uses a single server named `MedRecServer`. The `Server` element configures the server's listen port, communication protocols, Java compiler, and other attributes.

Listing 2-5 Elements for Server Configuration

```
<!-- WebLogic Server Configuration -->

<Server
  JavaCompiler="javac"
  ListenPort="7001"
  Name="MedRecServer"
  IIOPEnabled="false"
  InstrumentStackTraceEnabled="false">

  <ExecuteQueue
    Name="default"
    ThreadCount="15"
  />

  <SSL
    Name="MedRecServer"
    Enabled="true"
    ListenPort="7002"
  />

</Server>
```

Security Realm

All WebLogic Server domains must configure a default security realm, which determines who can access resources within the domain. The elements in [Listing 2-6](#) configure the default security realm for the Avitek Medical Records domain.

Elements such as

`<weblogic.security.providers.authentication.DefaultAuthenticator>` specify an MBean that manages a Security Provider. For example, the aforementioned element configures the realm to use the Authenticator Provider that is managed by an MBean named `Security:Name=myrealmDefaultAuthenticator` Realm="Security:Name=myrealm.

This element also configures the realm to treat this Authenticator Provider as SUFFICIENT for authenticating users.

The last elements in [Listing 2-6](#) configure compatibility security, which enables the domain to use security configurations from WebLogic Server 6.x. For more information, refer to [“Customizing Files for Compatibility Security” on page 2-12](#).

Listing 2-6 Elements for Configuring the Security Realm

```
<!-- Security -->
<Security
  Name="medrec"
  PasswordPolicy="wl_default_password_policy"
  Realm="wl_default_realm"
  RealmSetup="true">

  <weblogic.security.providers.authentication.DefaultAuthenticator
    ControlFlag="SUFFICIENT"
    Name="Security:Name=myrealmDefaultAuthenticator"
    Realm="Security:Name=myrealm" />
  <weblogic.security.providers.authentication.DefaultIdentityAsserter
    ActiveTypes="AuthenticatedUser"
    Name="Security:Name=myrealmDefaultIdentityAsserter"
    Realm="Security:Name=myrealm" />
  <weblogic.security.providers.authorization.DefaultRoleMapper
    Name="Security:Name=myrealmDefaultRoleMapper"
    Realm="Security:Name=myrealm" />
  <weblogic.security.providers.authorization.DefaultAuthorizer
    Name="Security:Name=myrealmDefaultAuthorizer"
    Realm="Security:Name=myrealm" />
  <weblogic.security.providers.authorization.DefaultAdjudicator
    Name="Security:Name=myrealmDefaultAdjudicator"
    Realm="Security:Name=myrealm" />
  <weblogic.security.providers.credentials.DefaultCredentialMapper
    Name="Security:Name=myrealmDefaultCredentialMapper"
    Realm="Security:Name=myrealm" />
  <weblogic.management.security.authentication.UserLockoutManager
    Name="Security:Name=myrealmUserLockoutManager"
    Realm="Security:Name=myrealm" />

  <weblogic.management.security.Realm
    Adjudicator="Security:Name=myrealmDefaultAdjudicator"
    AuthenticationProviders="Security:Name=myrealmDefaultAuthenticator|
      Security:Name=myrealmMedRecSampleAuthenticator|
      Security:Name=myrealmDefaultIdentityAsserter"
    Authorizers="Security:Name=myrealmDefaultAuthorizer"
```

```
CredentialMappers="Security:Name=myrealmDefaultCredentialMapper"
DefaultRealm="true"
DeployPolicyIgnored="false"
DeployRoleIgnored="false"
DisplayName="myrealm"
FullyDelegateAuthorization="true"
Name="Security:Name=myrealm"
RoleMappers="Security:Name=myrealmDefaultRoleMapper"
UserLockoutManager="Security:Name=myrealmUserLockoutManager"/>

<com.bea.medrec.security.MedRecSampleAuthenticator
  ControlFlag="SUFFICIENT"
  Name="Security:Name=myrealmMedRecSampleAuthenticator"
    Realm="Security:Name=myrealm"/>

</Security>

<PasswordPolicy Name="wl_default_password_policy"/>
<Realm FileRealm="wl_default_file_realm" Name="wl_default_realm"/>
<FileRealm Name="wl_default_file_realm"/>
```

Customizing Files for Compatibility Security

Compatibility security refers to the capability of running security configurations from WebLogic Server 6.x in WebLogic Server 8.1. If you run WebLogic Server with Compatibility security, your distribution must include the following:

- A `fileRealm.properties` file, which defines the ACLs, groups, and security principles for the default WebLogic Server security realm
- The following minimal set of elements in `config.xml`:

```
<Domain Name="mydomain">
  <Security Name="mydomain" Realm="mysecurity"/>
  <Realm Name="mysecurity" FileRealm="myrealm"/>
  <FileRealm Name="myrealm"/>
  <Server ListenPort="7001" Name="myserver">
    </Server>
</Domain>
```

If your application requires integration with a third-party security realm (for example, single sign-on using the Windows NT security realm), you must also configure a caching realm.

For more information on WebLogic Server security, refer to the following topics:

- [Using Compatibility Security](#) in the *Managing WebLogic Security* guide.
- The [Security](#) page on the WebLogic Server documentation Web site.
- The [BEA WebLogic Server Configuration Reference](#), which provides conventions for editing `config.xml`.

Creating Startup Scripts

WebLogic Server provides several techniques for starting server instances and clusters, all of which can be encapsulated in startup scripts that you install along with your configuration template. The following sections provide an overview of the techniques:

- [“Choosing a Startup Technique” on page 3-1](#)
- [“Using WebLogic Server Start Scripts” on page 3-4](#)
- [“Creating Scripts that Invoke Node Managers” on page 3-5](#)
- [“Creating Windows Services for Instances of WebLogic Server” on page 3-6](#)
- [“Installing Your Startup Scripts” on page 3-7](#)

Choosing a Startup Technique

The technique that you use to start an embedded server depends on the complexity of the WebLogic Server domain that you install, the number of computers that host server instances within the domain, and the underlying operating system. [Table 3-1](#) compares the various techniques.

Table 3-1 Startup Techniques

This Technique...	Is Recommended When...
Using WebLogic Server start scripts	<p>Your WebLogic Server domain runs on a single WebLogic Server host. A WebLogic Server host is a computer on which you have installed the WebLogic Server software.</p> <p>A WebLogic Server domain is a self-contained administrative unit that contains servers, applications, and other resources such as database connection pools.</p> <p>In the simplest configuration of a WebLogic Server domain, all applications run on a single server instance on a single WebLogic Server host. With this simple configuration, invoking the startup scripts that WebLogic Server provides is the easiest technique for starting a server. The scripts configure an environment and start a server instance on the current WebLogic Server host. You must log on to a WebLogic Server host (either directly or by opening a shell in a remote session) to invoke these scripts.</p> <p>For more information, see “Using WebLogic Server Start Scripts” on page 3-4.</p>

Table 3-1 Startup Techniques

This Technique...	Is Recommended When...
Creating scripts that invoke a Node Manager	<p>Your WebLogic Server domain includes multiple server instances and you want these server instances to run on multiple WebLogic Server hosts. Such a domain can include clusters, which provide scalability and failover capabilities.</p> <p>Each WebLogic Server host can run a Node Manager, which is a Java utility that can start and stop Managed Server instances on the current WebLogic Server host. Because a Node Manager can receive startup and shutdown requests from remote WebLogic Server hosts, you can create one script that invokes Node Managers running on multiple WebLogic Server hosts to start all server instances in a distributed cluster or a domain.</p> <p>For more information, see “Creating Scripts that Invoke Node Managers” on page 3-5.</p>
Installing a server as a Windows service	<p>Your domain runs on a platform that includes the Microsoft Windows operating system.</p> <p>The Windows operating system can automatically start Windows services as part of the computer’s boot up process. On a single WebLogic Server host, you can configure dependencies so that a Managed Server starts only after an Administration Server starts. However, you cannot configure a dependency between Windows services that run on different computers.</p> <p>For more information, see “Creating Windows Services for Instances of WebLogic Server” on page 3-6.</p>

Using WebLogic Server Start Scripts

WebLogic Server provides a layered set of scripts that set environment variables and configure a Java command. The Java command invokes the `weblogic.Server` class, which is the main class for a server instance.

When you install the WebLogic Server software, the installation includes the `WL_HOME\common\bin\commEnv.sh` (and `commEnv.cmd` on Windows) script. This script sets environment variables that define defaults for all WebLogic software on the current host. It defines the following:

- The BEA home directory, which contains licensing information for BEA software.
- The home of the Java SDK that the Node Manager and WebLogic Server instances use.
- The Java classpath that the Node Manager and WebLogic Server software requires. **If you need to add classes to the WebLogic Server classpath**, add them to the `WEBLOGIC_CLASSPATH` variable in this script.
- Java options that configure basic attributes of the Java Virtual Machines (JVMs) that run instances of WebLogic Server, the Node Manager, and WebLogic Server utilities on this host. For example, for some JVMs, the options configure the heap size and whether the JVM uses Just In Time (JIT) compiling.

For each domain that you create based on a domain configuration template that BEA provides, the Configuration Wizard places startup scripts in the domain directory. These scripts do the following:

- Invoke the `commEnv.sh` (and `commEnv.cmd` on Windows) script.
The domain-specific startup scripts can override the values that the `commEnv` script sets for some of the variables.
- Set additional JVM parameters that are specific to running instances of WebLogic Server.
- Invoke the `weblogic.Server` class.

Your own templates can include the WebLogic Server scripts or similar startup scripts that you create.

The **Configuration Wizard** is a Java application that you use to create and configure domains. The Configuration Wizard prompts you to choose a **domain configuration template** to provide the basic structure for a domain.

For example, if you create a domain based on the Basic WebLogic Server Domain template, and if you accept all default values, the Configuration Wizard creates the following scripts in `BEA_HOME\user_projects\domains\mydomain`, where `BEA_HOME` is the BEA home directory:

- `startWebLogic.sh` (and `startWebLogic.cmd` on Windows), which starts Administration Server. The script calls the `commEnv` and `VMOptions` scripts to set environment variables. Then it sets values of domain-specific variables and invokes the `weblogic.Admin` class. You can configure this script to override some or all of the values that the `commEnv` and `VMOptions` scripts set.
- `startManagedWebLogic.sh` (and `startManagedWebLogic.cmd` on Windows), which starts Managed Servers. You can use this one script to start multiple Managed Servers by passing parameters when you invoke the script, or you can configure this script to start a single Managed Server. For more information, refer to [“Starting Managed Servers From a WebLogic Server Script”](#) in the Administration Console Help.

Creating Scripts that Invoke Node Managers

Each WebLogic Server host can run a Node Manager, which can start Managed Server instances on the current host. The Node Manager cannot start an Administration Server.

To create and use a script that uses Node Managers to start Managed Servers:

1. Make sure that the Node Manager is active on each WebLogic Server host.

On a Windows computer, you can configure the WebLogic Server silent installer to install the Node Manager as a Windows Service. On a UNIX computer, your customers can configure the Node Manager to run as a daemon.

For more information, see in [“Configuring, Starting, and Stopping Node Manager”](#) *Configuring and Managing WebLogic Server*.

2. Start the Administration Server by doing any of the following:
 - Calling the `startWebLogic.sh` script that the Configuration Wizard creates.
 - Calling a script that you create.
 - Installing the Administration Server as a Windows Service or a UNIX daemon. See [“Creating Windows Services for Instances of WebLogic Server”](#) on page 3-6.

You cannot use a Node Manager to start the Administration Server.

3. For each Managed Server in the domain, indicate which computer (machine) you want to host the Managed Server. See [“Configuring a Machine”](#) in the Administration Console Online Help.

4. Create a script that does the following:
 - a. Invokes `WL_HOME/common/bin/commEnv.sh` (or `commEnv.cmd` on Windows). This script sets environment variables that the `weblogic.Admin` utility requires.
 - b. Invokes the `java weblogic.Admin START` or `STARTINSTANDBY` command for each server instance in the domain. If your domain includes a cluster, your script can invoke the `java weblogic.Admin STARTCLUSTER` command.

For more information, refer to “[weblogic.Admin Command-Line Reference](#)” in the *WebLogic Server Command Line Reference*.

5. From a single WebLogic Server host, run the script that you created in the previous step.

The `weblogic.Admin` utility connects to the Administration Server, which causes the Node Manager to start the specified Managed Servers on the machines that you specified in step 3.

Creating Windows Services for Instances of WebLogic Server

If you want WebLogic Server instances within your domain to start automatically when you boot a Windows host computer, you can create a script that causes a WebLogic Server instance to run as a Windows service. The script must invoke the WebLogic Server `beasvc` utility, which creates a key in the Windows Registry under

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`. The registry entry contains such information as the name of the server and other startup arguments.

Neither the WebLogic Server installation program nor the Configuration Wizard can configure servers as Windows services when run in silent mode. Instead, you create a script that invokes the WebLogic Server `beasvc` utility.

If you want to install a Managed Server as a Windows service, your script should include a Java option that specifies how the Managed Server connects to the Administration Server.

If you want to install an Administration Server and one or more Managed Servers as Windows services on the same WebLogic Server host, your script should include a Java option that specifies how the Managed Server connects to the Administration Server **and** a `beasvc` option that configures dependencies between the Windows services. The Administration Server must complete its startup cycle before any Managed Servers start.

For more information, refer to “[Setting Up a WebLogic Server Instance as a Windows Service](#)” in *Configuring and Managing WebLogic Server*.

Installing Your Startup Scripts

To install your startup scripts along with your application and WebLogic Server software:

1. Use the Template Builder to create a domain configuration template. See “[Creating Configuration Templates Using the Template Builder](#)” in the *Creating WebLogic Configurations Using the Configuration Wizard* guide.
2. Include your start scripts as part of the configuration template. On the Windows platform, you can also include your start script as an item on the Windows Start menu.
3. Configure the Configuration Wizard to run in silent mode and install your domain configuration template. See “[Starting in Silent Mode](#)” in *Creating WebLogic Configurations Using the Configuration Wizard*.

Creating Startup Scripts

Using JDBC Profiling MBeans

The WebLogic Server management system uses Java Management Extensions (JMX) and Managed Beans (MBeans) to configure servers. The [Programming WebLogic Management Services with JMX](#) guide provides detailed information and code samples for working with WebLogic Server MBeans.

BEA provides several JDBC MBeans that you can use to store and analyze metrics for SQL statements, prepared statements, and JDBC connection leaks. The following sections describe how to enable and use JDBC profiling. For additional information, refer to the Javadoc for the following WebLogic Server MBeans and related classes:

- [JDBCConnectionPoolMBean](#)
- [JDBCConnectionPoolRuntimeMBean](#)
- [JDBCStatementProfile](#)
- [JDBCConnectionLeakProfile](#)

Enabling JDBC Profiling

Before you can analyze SQL statements or connection leak profiles, you must enable profiling for the connection pool you want to observe. When profiling is enabled, the connection pool stores metrics in an external repository for later analysis.

Applications enable and disable JDBC profiling options using the `JDBCConnectionPoolMBean`. In addition to providing `get/set` methods for standard connection pool properties, `JDBCConnectionPoolMBean` provides the following methods for enabling and disabling profiling:

- `setConnLeakProfilingEnabled()` enables or disables profiling for JDBC connection leaks. Connection leaks represent connections that were checked out of the connection pool but never returned with a `close()` method. It is important to analyze the connection leak profiles, as leaked connections cannot be used to fulfill later connection requests.
- `setSqlStmtProfilingEnabled()` enables or disables profiling for SQL statements. When this type of profiling is enabled, the connection pool stores both SQL statement text as well as the statement execution time and other metrics. You can analyze the SQL statement profile to determine which queries consume the most time in your applications.
- `setSqlStmtParamLoggingEnabled()` enables or disables profiling for the bind parameters of prepared and callable statements. Because statement parameters can be very large, you can optionally use `setSqlStmtMaxParamLength()` to limit the size of parameters that are stored in the profile.

For information on obtaining MBeans in WebLogic Server, see “[Accessing WebLogic Server MBeans](#)” in *Programming WebLogic Management Services with JMX*. The following excerpt shows an application that obtains the `JDBCConnectionPoolMBean` and activates all profiling options. This example stores a maximum of 20 characters for each statement parameter:

```
// Obtain MBeanHome for the administration server.

...

JDBCConnectionPoolMBean mbean =
    (JDBCConnectionPoolMBean) home.getConfigurationMBean(poolName,
        "JDBCConnectionPoolConfig");

mbean.setConnLeakProfilingEnabled(true);

mbean.setSqlStmtParamLoggingEnabled(true);

mbean.setSqlStmtMaxParamLength(maxLen);

...
```

Accessing JDBC Profiles

Once you have enabled the desired profiling option(s), you can analyze the stored metrics using the `JDBCStatementProfile` and `JDBCConnectionLeakProfile` classes. Both of these profile classes can be easily obtained using the `JDBCConnectionPoolRuntimeMBean`.

`JDBCStatementProfile` stores the SQL statements and associated metrics (and optionally, bind parameters) for the connection pool. `JDBCConnectionLeakProfile` stores stack traces for leaked connections.

Obtaining all profiles at once may consume considerable resources. For this reason, applications should generally retrieve only a subset of profiles at a given time. You can accomplish this by first determining the total number of profiles in storage, then retrieving profiles in smaller subsets.

The following example shows a simple way to divide the number of profiles into smaller fractions.

```
// Obtain MBeanHome for the server that hosts the connection pool.
. . .

// Get the JDBCRuntimeMBean for the "testPool" connection pool.
String poolName = "testPool";

JDBCConnectionPoolRuntimeMBean mbean =
    (JDBCConnectionPoolRuntimeMBean) home.getRuntimeMBean
        (poolName, "JDBCConnectionPoolRuntime");

JDBCConnectionLeakProfile[] profiles = null;

// Get the total number of available prepared statement cache profiles
int profileCount = mbean.getConnectionLeakProfileCount();

// Request profilesPerStep number of profiles
int profilesPerStep = 10;

// Begin with profile number profileIndex
int profileIndex = 0;
boolean done = (profileCount > 0);
while (!done) {

    // Get profiles
    profiles = mbean.getConnectionLeakProfiles(profileIndex,
        profilesPerStep);
```

Using JDBC Profiling MBeans

```
// Go through retrieved profiles
    for (int index = 0; index < profiles.length; index++) {

        // Get pool name
        String poolName = profiles[index].getPoolName();

        // Get stack trace
        String stackTrace = profiles[index].getStackTrace();
    }

profileIndex = profileIndex + profilesPerStep - 1;

// Finish if number of retrieved profiles is
// less than requested
done = (profiles.length < profilesPerStep);
}
```