BEA WebLogic Server

Programming WebLogic Management Services

BEA WebLogic Server 6.0
Document Date: March 3, 2001

Programming WebLogic Management Services

Part Number	Document Date	Software Version
N/A	March 3, 2001	BEA WebLogic Server Version 6.0

Contents

Index

About This Document

This document describes how to use the BEA WebLogic Server™ managment APIs to enhance WebLogic Server to support your applications.

The document is organized as follows:

- Chapter 1, "Overview of Management Services," describes the WebLogic Server management interface, and provides overviews of MBeans, the administrative domain, and server configurations.
- Chapter 2, "Getting Started," describes what you must do before you begin to extend WebLogic Server.
- Chapter 3, "Creating an Entry Point to WebLogic Server," describes how to create a programmatic entry point from which you can extend WebLogic Server.
- Chapter 2, "Programming WebLogic Server MBeans," describes how to create and deploy WebLogic Server MBeans.
- Chapter 4, "Getting Technical Support," tells you how to get technical support for both licensed and non-licensed development, and how to upgrade an evaluation license.

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 have a familiarity with the BEA WebLogic Server platform 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/wls/docs60.

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.
italics	Emphasis and book titles.

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

`monospace italic text`	Variables in code. *Example*: `String CustomerName;`
UPPERCASE TEXT	Device names, environment variables, and logical operators. *Examples*: LPT1 BEA_HOME OR
`{ }`	A set of choices in a syntax line.
`[ ]`	Optional items in a syntax line. *Example*: `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`

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

1 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 a WebLogic Server that you can distribute, complete the following tasks:

- Upgrade an ISV License for Distributing a Newer Release of WebLogic Server
- Enroll in the BEA Star Partner Program
- Install the Partner Development Kit
- Install the ISV License
- Distribute Files

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

If you distributed an older version of WebLogic Server, complete the following steps to upgrade your existing ISV license:

1. Send an email to `licensing@bea.com`. In the email, request a new ISV license and attach your old ISV license file.
2. Install the WebLogic Server 6.0 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.

Update your installer to include the new WebLogic Server files.

Enroll in the BEA Star Partner Program

BEA Systems, Inc. manages its relationships with Independent Software Vendors (ISVs) and Application Software Providers (ASPs) through the Star Partner Program. If you have not already enrolled in the BEA Star 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 Star 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 Star 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 6.0 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 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 Server* 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. Add `isv.jar` to the computer's `CLASSPATH` by entering the following command:

   `set CLASSPATH=.\isv.jar;%CLASSPATH%` (Windows systems)
   `export CLASSPATH=./isv.jar:$CLASSPATH` (UNIX systems)

5. Add the WebLogic Server JDK to the computer's `PATH` by entering one of the following commands:

   - `set PATH=.\jdk130\bin;%PATH%` (Windows systems)
   - `export PATH=./jdk130/bin:$PATH` (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 the following command from `BEA_HOME`:

```
java -Xmx128m -Dbea.home=BEA_HOME
-Dbea.jar=WL_HOME\lib\weblogic.jar install
```

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, enter 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 Your Application and WebLogic Server

After you install your ISV license, start the ISV-licensed WebLogic Server, deploy your application, and configure the server components. For more information, refer to the following topics (available from http://edocs.bea.com):

- "Starting the WebLogic Administration Server" in the *Administration Guide*
- "Starting the WebLogic Managed Servers Using Scripts" in the *Administration Guide*
- "Starting the WebLogic Administration Server from the Command Line" in the *Administration Guide*
- "Overview of JDBC Drivers" in *Programming WebLogic JDBC*
- "Overview of JDBC Drivers" in *Programming WebLogic JDBC*
- The remaining sections of this document, which provide development tips that are specific to ISVs.

Distribute Files

When you are ready to distribute WebLogic Server with your application, you must make sure that your installer includes 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.For information on using the WebLogic Server silent install process, see Installing WebLogic Server Using Silent Installation.

CHAPTER

2 Programming WebLogic Server MBeans

Using the J2EE Java Management Extensions (JMX) specification with the WebLogic Server Management API, developers can create and deploy Management Beans (or MBeans) to extend and customize WebLogic Server.

Configuration is done using MBeans, which retrieve their values from the domain configuration and state. MBeans provide developers with a way to programmatically access all configuration and monitoring information about WebLogic Server via the JMX standard API. Using this JMX specification with the WebLogic Server Management API, developers can create and deploy MBeans to extend and customize WebLogic Server. WebLogic Server MBeans also provide management access to domain resources.

WebLogic Server MBeans are based on JMX extensions for server configuration and server runtime data. For the MBean interfaces that are exposed by JMX, see the WebLogic Server *Management API*.

The following sections describe how to program WebLogic Server MBeans:

- Programming Client Access to WebLogic MBeans
- Setting Up Monitoring
- Setting Up Notifications
- Using Public MBean JavaDocs

For documentation describing the WebLogic Server architecture and management services, see the WebLogic Server *Administration Guide*. The Administration Server must be running in order to perform any kind of management operation. For details on how to start and stop the server, see the WebLogic Server *Administration Guide* section, "Starting and Stopping WebLogic Servers."

For additional information, see the WebLogic Server *Programming Guides*.

Programming Client Access to WebLogic MBeans

Two interfaces are used to provide client access to MBeans:

- `MBeanServer`—Each WebLogic Server contains an `MBeanServer` interface. The `MBeanServer` interface hosts all of the MBeans that are contained within its host WebLogic Server. Using the JMX API, JMX clients can get MBean attribute values and perform other JMX operations on the `MBeanServer` interface. For more information about the `MBeanServer` interface, see the `javax.managment.MBeanServer API`.
- `MBeanHome`—Each WebLogic Server publishes an `MBeanHome` interface, which is a wrapper on `MBeanServer`, exposing a stongly-typed MBean interface. For more information about the `MBeanHome` interface, see the `weblogic.management.MBeanHome API`.

`MBeanServer` and `MBeanHome` provide access to the same set of MBeans in a given server. Each server in a domain contains an `MBeanHome` (and a corresponding MBeanServer), which hosts configuration and run-time MBeans on that server. In addition, the Administration server has an administration `MBeanHome` that provides access to all MBeans in the entire domain The Administration MBeans reside only in the Administration Server, and are only available through the Administration `MBeanHome`. For example, a query for server run-time MBeans on the Admiinistration `MBeanHome` returns one server MBean for each running server in the domain. The same query on the `MBeanHome` of a managed server (or the regular `MBeanHome` of the Administration Server) returns only the server run-time MBean for that managed server.

Getting MBeanServer and MBeanHome

The `MBeanHome` of any server is available from the relevant server's JNDI tree at:

`weblogic.management.MBeanHome.JNDI_NAME.`*`relevantserverName`*

An Administration server publishes an `MBeanHome` for each server in the domain on its JNDI tree. The administration `MBeanHome` is available only from the JNDI tree of the Administration server at:

`weblogic.management.MBeanHome.ADMIN_JNDI_NAME`

The underlying `MBeanServer` for any `MBeanHome` can be obtained by invoking the `getMBeanServer()` method on that `MBeanHome`.

The following is an example of a JNDI lookup for an Administration server `MBeanHome`:

```
import javax.naming.Context;
import javax.naming.NamingException;
import javax.naming.AuthenticationException;
import javax.naming.CommunicationException;
import weblogic.jndi.Environment;          ...
import weblogic.management.MBeanHome;
        ...
    String url = "t3://localhost:7001"; //URL of the Administration server
    String username = "guest"; //Only works if guest logins are enabled
    String password = "guest";
    MbeanHome home = null;
    try {
      Environment env = new Environment();
      env.setProviderUrl(URL);
      env.setSecurityPrincipal(username);
      env.setSecurityCredentials(password);
      ctx = env.getInitialContext();
      home = (MBeanHome) ctx.lookup(MBeanHome.ADMIN_JNDI_NAME);
    }
    catch (AuthenticationException e) {
        ... //Error handling
    } catch (CommunicationException e) {
        ... //Error handling
    } catch (NamingException e) {
        ... //Error handling
    }
```

Naming MBeans

The WebLogic Server Management API distinguishes between three types of MBeans:

- Administration
- Configuration
- Run-time

All MBeans have a name, a type and a domain. These attributes are reflected in the MBean's JMX Object Name. The Object Name is the unique identifier for a given MBean across all domains, and has the following structure:

domain name:Name=*name*,Type=*type*[,*attr*=*value*]...

`Name` is a name that is unique for a given domain and a given type. Examples of `type` include Server, WebComponent or JDBCConnectionPoolRuntime. `Type` is used to distinguish between the various types of MBeans; for example, the value of `Type` for a Server MBean is:

- `Server` for an Administration MBean
- `ServerConfig` for a Configuration MBean
- `ServerRuntime` for a Run-time MBean

Note that the "MBean" suffix is removed from the MBean interface name to get the base type of an MBean. Therefore, `Config` or `Runtime` suffices are added to the base name (`Server`) to distinguish configuration and run-time MBeans from administration MBeans.

Specific MBean types have additional components. All run-time and configuration MBeans have a `Location` component that uses the name of the server on which that MBean is located as its value. For example:

```
mydomain:Name=myServlet,Type=ServletRuntime,Location=myserver
```

Any MBean which has a child relationship with another parent MBean, has an extra attribute in its object name in the following format: *TypeOfParentMBean*=*NameOfParentMBean*. In the following example, `Server` is the type of Parent MBean, and `myserver` is the name of the Parent MBean:

```
mydomain:Name=mylog,Type=Log,Server=myserver
```

Naming MBean Packages

All interface types for administration or configuration MBeans are located in the `weblogic.management.configuration` API.

All interfaces types for run-time MBeans are located in the `weblogic.management.runtime` API.

Setting Up Monitoring

A WebLogic client can set up monitors to monitor MBean properties. The various monitors are defined in the JMX documentation for the package `javax.management.monitor`, and are as follows: `CounterMonitor`, `GaugeMonitor`, `StringMonitor`. Without repeating the details available in the JMX documentation, the following is an example of how to set up a counter monitor for receiving JMX Notifications.

1. Implement `weblogic.management.RemoteNotificationListener`.

   The following code example is a simple implementation of `CounterListener`:

```
public class CounterListener implements RemoteNotificationListener  {
  public void handleNotification(Notification p1,java.lang.Object p2){
    System.out.println(">>><<<<<<----->GotNotified!!" + p1.toString());
  }
}
```

2. Get `RemoteMBeanServer` from `MBeanHome`, and create a `Listener` object as follows:

```
(CounterListener), and create a Monitor object (CounterMonitor).
RemoteMBeanServer rmbs = mbHome.getMBeanServer();
CounterMonitor monitor = new CounterMonitor();
CounterListener listener = new CounterListener();
. . .
```

3. Register the `Monitor` to listen on the `ServerSecurityRuntime.InvalidLoginAttemptsTotalCount` attribute. This attribute indicates the number of failed logins to the server.

   Then set up the `Listener` on the `Monitor`. A simple implementation example follows:

```
ObjectName monitorObjectName = new
WebLogicObjectName("mydomain:Type=CounterMonitor,Name=MyCounter");
rmbs.registerMBean((Object)monitor, monitorObjectName);

ObjectName securityRtObjectName = new
WebLogicObjectName("mydomain:Name=myserver,Location=myserver,
   Type=ServerSecurityRuntime");

rmbs.setAttribute(monitorObjectName, new Attribute("Threshold",
   new Integer(5)));
rmbs.setAttribute(monitorObjectName, new Attribute("Offset", new Integer(0)));
rmbs.setAttribute(monitorObjectName, new Attribute("GranularityPeriod",
   new Long(5000)));
rmbs.setAttribute(monitorObjectName, new Attribute("ObservedObject",
   securityRtObjectName));
rmbs.setAttribute(monitorObjectName, new Attribute("ObservedAttribute",
   "InvalidLoginAttemptsTotalCount"));

rmbs.addNotificationListener(monitorObjectName,listener, null, null);
```

4. Run the WebLogic Remote Method Invocation (RMI) utility `weblogic.rmic` to compile the class that implements the `RemoteNotificationListener`, `CounterListener`, as shown in the following example:

```
java -classpath $CLASSPATH\;.  weblogic.rmic -keepgenerated
   -compiler sj -d $WL_HOME/classes -classpath $CLASSPATH\;.
   -d $CLASSOUTDIR -keepgenerated
   -nomanglednames CounterListener
```

   See also *Programming WebLogic RMI*.

5. When the invalid login attempts exceed the threshold value of 5, the `handleNotification` method is invoked by the notification listener, `CounterListener.handleNotification()`.

Setting Up Notifications

Note: For an overview of JMX notifications and how they work, see the Sun Microsystems J2EE JMX specification.

A WebLogic client can set up notification listeners to listen for WebLogic Server events. These notifications are generated when you attempt to add or remove a collection attribute. For example, you would get the appropriate notification if `add`*`SomeCollection`* or `remove`*`SomeCollection`* were called for an MBean.

The following notification methods are generated from the server:

- The JMX API defines two notification types:
  - `javax.management.AttributeChangeNotification`
  - `javax.management.MbeanServerNotification`
- The WebLogic Management API defines two additional notification types:
  - `weblogic.management.AttributeAddNotification`
  - `weblogic.management.AttributeRemoveNotification`

MBeans are notification broadcasters, which makes it possible for a client to set up a notification listener on any WebLogic Server MBean using the following API:

```
javax.management.NotificationBroadcaster.AddNotificationListener()
```

An example implementation of a notification listener follows:

```
import javax.management.NotificationLister;
import javax.management.Notification;
import javax.management.Notification.AttributeChangeNotification;
import weblogic.management.RemoteNotificationListener;

class FooListener implements RemoteNotificationListener {

  public void handleNotification(Notification n, Object hb) {
    if (notification instanceof AttributeChangeNotification) {
      AttributeChangeNotification acn = (AttributeChangeNotification)n;
      if ("Bar".equals(acn.getAttributeName())) {
        BarMBean oldValue = (BarMBean)acn.getOldValue();
        BarMBean newValue = (BarMBean)acn.getNewValue();
        // do my thing
      }
```

```
    }
  }
}
```

Writing Custom Notifications for WebLogic Server Error Messages

Note: For an overview of JMX notifications and how they work, see the Sun Microsystems J2EE JMX specification.

WebLogic Server can send JMX notifications for log messages. Users code can create `NotificationListeners` that can receive selective log messages as a notification based on a user-defined `NotificationFilter`; for example, certain log messages might need to be written or sent to an additional destination such as an RDBMS or an enterprise management console, or paged to an administrator.

Registering a Notification Listener for Log Notifications

You use published JMX interfaces to register a notification listener to the JMX notification broadcaster (`LogBroadcasterRuntimeMBean`) provided by the WebLogic Server logging system. `LogBroadcasterRuntimeMBean` is only responsible for generating notifications for log messages generated by the server. All notifications generated are of the type `WebLogicLogNotification`. There is only one `LogBroadcasterRuntimeMBean` per server, named `TheLogBroadcaster`, of the type `LogBroadcasterRuntime`. As defined in the JMX specification, the user can also provide a customized filter at the time of registration.

The `LogBroadcasterRuntimeMBean` can be fetched using the mechanisms described in "Writing Custom Notifications for WebLogic Server Error Messages." The following example shows a simple implementation of the registration of a notification listener.

```
...
WebLogicObjectName oname = new
WebLogicObjectName("mydomain:Name=TheLogBroadcaster,
  Type=LogBroadcasterRuntime,Location=myserver");
logBroadcaster = (LogBroadcasterRuntimeMBean) mbeanhome.getMBean(oname);
```

```
ANotificationFIlter filter = new ANotificationFIlter();
logBroadcaster.addNotificationListener(this, filter, null);
...
```

Contents of a Log Notification

A JMX notification contains the following fields:

- Type—The type field to which the log notification is mapped, for example:

  'weblogic.logMessage.*subSystem.messageID*'

- Time stamp—Contains the time at which the log message causing this notification was generated by the server
- Sequence number
- Message—Contains the actual message body of the log message.
- User data—The user data field is not currently used

All log notifications are of the type `WebLogicLogNotification`. The class interface provides getters for all individual fields of a message. This eases the filtering of log notifications when filtering messages based on their severity, user ID, subystem, and other fields. The following `NotificationFilter` example implementation only selects messages of a specific message ID (111000) to be sent as notifications.

```
import weblogic.management.logging.WebLogicLogNotification;
import javax.management.NotificationFilter;
....
class ALogNotificationFilter
   implements NotificationFilter, java.io.Serializable {
  public boolean isNotificationEnabled(Notification notif) {
    WebLogicLogNotification  wln = (WebLogicLogNotification) notif;
    return(wln.getMessageId() == 111000);
  }
}
```

Using Public MBean JavaDocs

The WebLogic Server *Management API* is fully documented online in JavaDocs. For additional information, see the WebLogic Server *Programming Guides*.

Index