



BEA WebLogic Server

Programming WebLogic Management Services

BEA WebLogic Server 6.0
Document Date: March 3, 2001

Copyright

Copyright © 2001 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, WebLogic, Tuxedo, and Jolt are registered trademarks of BEA Systems, Inc. How Business Becomes E-Business, BEA WebLogic E-Business Platform, BEA Builder, BEA Manager, BEA eLink, BEA WebLogic Commerce Server, BEA WebLogic Personalization Server, BEA WebLogic Process Integrator, BEA WebLogic Collaborate, BEA WebLogic Enterprise, and BEA WebLogic Server are trademarks of BEA Systems, Inc.

All other product names may be trademarks of the respective companies with which they are associated.

Programming WebLogic Management Services

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

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 | viii |

1. Overview of Management Services

| | |
|-----------------------------------------------------------------|-----|
| Advantages of the WebLogic Server Development Environment | 1-1 |
| WebLogic Server MBeans | 1-2 |
| Domains, the Administration Server, and Managed Servers | 1-4 |

2. Getting Started

| | |
|-------------------------------------------------|-----|
| Registering with BEA..... | 2-1 |
| Obtaining the WebLogic Server Software | 2-2 |
| Licensing a BEA WebLogic Product | 2-2 |
| Choosing a License Key Format | 2-3 |
| Choosing a License Type | 2-3 |
| Creating an Entry Point to WebLogic Server..... | 2-4 |
| Running WebLogic Server at Startup | 2-4 |
| Using WebLogic Server JDBC Drivers | 2-4 |
| Types of JDBC Drivers | 2-5 |

3. Programming WebLogic Server MBeans

| | |
|---------------------------------------------------|-----|
| Programming Client Access to WebLogic MBeans..... | 4-2 |
| Getting MBeanServer and MBeanHome | 4-3 |

| | |
|-----------------------------------------------------------------------|------|
| Naming MBeans | 4-4 |
| Naming MBean Packages..... | 4-5 |
| Setting Up Monitoring | 4-5 |
| Setting Up Notifications | 4-7 |
| Writing Custom Notifications for WebLogic Server Error Messages | 4-8 |
| Registering a Notification Listener for Log Notifications | 4-8 |
| Contents of a Log Notification | 4-9 |
| Using Public MBean JavaDocs | 4-10 |

4. Getting Technical Support

| | |
|--------------------------------------|-----|
| Getting License Support | 5-1 |
| Getting Non-License Support | 5-2 |
| Upgrading an Evaluation License..... | 5-2 |

Index

About This Document

This document describes how to use the BEA WebLogic Server™ management 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 3, “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. |
| <i>italics</i> | 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. <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> |

| Convention | Usage |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Separates mutually exclusive choices in a syntax line. <i>Example:</i> <pre>java weblogic.deploy [list deploy undeploy update] password {application} {source}</pre> |
| ... | 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. |



1 Overview of Management Services

Some BEA Systems, Inc. customers choose to use WebLogic Server by configuring and deploying the software without any additional enhancements. This approach works for the vast majority of WebLogic Server users. A specific type of customer, sometimes referred to as an independent software vendor (ISV) or third-party developer, purchases special licenses for BEA software that allows such customers to build their own products on top of BEA's core technologies.

The following sections provide overviews of WebLogic Server services and interfaces that you use to customize WebLogic Server:

- [Advantages of the WebLogic Server Development Environment](#)
- [WebLogic Server MBeans](#)
- [Domains, the Administration Server, and Managed Servers](#)

Advantages of the WebLogic Server Development Environment

As the leading e-commerce transaction platform and the first independently developed Java application server to receive J2EE certification, WebLogic Server provides a plug-and-play environment for creating multitier enterprise applications. J2EE APIs are implemented to work in an integrated manner with WebLogic Server services and facilities sharing common elements and objects.

WebLogic Server offers the independent developer:

- A robust and reliable server
- Faster development of e-commerce applications that use the J2EE standards for application services, for example, Java Management Extensions (JMX) and Enterprise Java Beans (EJB)
- Facilities for development and deployment, such as integrated security (WebLogic SSL and WebLogic ACLs), logging, and a graphical management console (the WebLogic Server Administration Console).
- Scalability that supports aggressive growth rates in a customer base

The WebLogic Server management architecture is based on the Sun Microsystems Java Management Extensions (JMX) specification. Configuration is done using *management beans*, or *MBeans*, which retrieve their values from the *domain configuration* and *state*. MBeans provide developers with a means to access all configuration and monitoring information about WebLogic Server programmatically via the JMX standard API. An *administrative domain* represents a group of servers, clusters, or both servers and clusters, which must be administered together. The configuration information is stored in a configuration repository. The only repository supported for BEA WebLogic 6.0 is an XML file-based repository.

For more detailed information, see the following WebLogic Server documentation:

- [WebLogic Server Administration Guide](#)
- [Programming WebLogic Enterprise Java Beans](#)
- [Programming WebLogic XML](#)
- [Programming WebLogic Security](#)

WebLogic Server MBeans

WebLogic Server Management Beans (MBeans) are based on Java Management Extensions (JMX) for server configuration and server run time data. For the MBean interfaces exposed via JMX, see the WebLogic Server [API Reference](#). The MBean interfaces are documented as follows:

- Administration and configuration — `weblogic.management.configuration`
- Run-time — `weblogic.management.runtime`

An MBean is an object that implements an interface using a “getter,” a “setter,” or an operation.

There are three types of MBeans:

- Administration MBean (`ServerMBean`)—Represents the configured properties of various subcomponents of the server. They are created as an image of the configuration repository when the Administration server starts up.
- Configuration MBean (`ServerMBean`)—Each administration MBean has an "active" instance, which is cloned from it and used by the relevant subsystems after applying any overrides (for example, at the command line). These are called configuration MBeans.
- Run-time MBean (`ServerRuntimeMBean`)—Represents the run-time transient state of the underlying resource or subsystem which it represents. For instance, `ServerRuntimeMBean` has metrics like `'OpenSocketsCurrentCount'` and `'SocketsOpenedTotalCount'`.

An administration MBean and its corresponding configuration MBean implement the same interface. At any point in time the configuration MBeans contain the property values actually being used by the server. Any configuration changes are applied to administration MBeans, and these changes are propagated to the relevant configuration MBeans, when appropriate.

The following example of an MBean interface declares one read-only attribute (`Foo`), one read/write attribute (`Bar`), one collection attribute (`Baz`), and one operation (`deploy`).

```
public interface FooMBean {
    int getFoo();
    BarMBean getBar();
    void setBar(BarMBean bar);
    BazBean[] getBaz();
    void addBaz();
    void removeBaz();

    void deploy()
}
```

See also [Chapter 3, “Programming WebLogic Server MBeans.”](#)

Domains, the Administration Server, and Managed Servers

A WebLogic Server domain is an inter-related set of WebLogic Server resources managed as one unit. A domain includes one Administration Server and zero or more managed servers. It can include WebLogic Server clusters.

An Administration Server is a WebLogic Server that runs the Administration Service. The Administration Service is the central point of control for configuring and monitoring the domain. The Administration Server must be running in order to perform any kind of management operation on that domain.

Typically, the applications and components with your business logic are deployed across Managed Servers, and the Administration Server is used to configure and monitor the Managed Servers.

The Administration Server must be running in order to perform any kind of management operation. For detailed information about the WebLogic Server management architecture, see the WebLogic Server [Administration Guide](#).

Detailed information about the WebLogic Server domain configuration is available in the *Administration Guide* section “[Domain Configuration](#).”

Each WebLogic Server domain configuration is stored in an XML file-based repository named `config.xml`. The `config.xml` file resides in a subdirectory with the same name as the domain. An example of the default `config.xml` file is provided in the WebLogic Server [Configuration Reference](#).

Warning: Manual editing of the `config.xml` file is *not* recommended. Customers who want to change their default WebLogic configurations are strongly urged to use the Administration Console to make configuration changes.

When manual editing of the `config.xml` file is required, make a backup copy of your original `config.xml` file prior to incorporating any changes.

See also the WebLogic Server Programming Guide [Programming WebLogic XML](#).

2 Getting Started

The following sections describe what you need to do before starting to develop applications that extend WebLogic Server:

- [Registering with BEA](#)
- [Obtaining the WebLogic Server Software](#)
- [Licensing a BEA WebLogic Product](#)
- [Creating an Entry Point to WebLogic Server](#)

Registering with BEA

To register with BEA Systems, Inc. and participate in the BEA WebLogic Server Developer Community you register and set your membership login account only once. The [BEA Systems Download Center](#) provides access to the registration and sign-in sections, as well as a set of [Frequently Asked Questions \(FAQ\)](#) to help guide you through the process.

If you are an existing BEA customer, your profile information is filled in automatically during registration, and you only have to make updates as necessary. You can also edit your profile at any time after registration.

Note: Customers who have an active WebSUPPORT account can use the same login password for software downloads.

Obtaining the WebLogic Server Software

Once you are a registered BEA developer, you can download the BEA WebLogic Server software from the [WebLogic Server web site](#). The software will have an evaluation license. For information about obtaining licenses, see [Licensing a BEA WebLogic Product](#).

Licensing a BEA WebLogic Product

Customers choose WebLogic Server product licenses based on their development needs.

The following products are licensed separately:

- **WebLogic Core Server**

A Core Server license provides developers and their customers full access to WebLogic Server's implementation of J2EE standards, including EJBs, Servlets, Java Server Pages (JSP), Java Messaging Service (JMS), Java Database Connectivity Service (JDBC), connection pools, workspaces, and events.

- **Clustering**

A special set of license keys is required to enable clustering, and must be specifically requested. Clustered WebLogic servers can provide transparent failover services for greater scalability and fault tolerance.

- **Secure Sockets Layer (SSL)**

A special license provides encrypted, authenticated connections through Secure Sockets Layer (SSL). ISVs who have major security issues with an application that is built and deployed on WebLogic Server, such as applications developed for distribution outside the United States, should request an SSL license key. See also http://www.weblogic.com/docs/classdocs/API_secure.html.

- **Type 4 JDBC Drivers**

As an optional feature, WebLogic Server provides Type 4 JDBC drivers for Microsoft SQL Server and Informix. These are pure Java drivers that do not require any native client libraries, and must be obtained separately. The licenses are in XML format, but special code must be written to enable the Type 4 driver license.

Choosing a License Key Format

You can choose from two license key formats, Java or XML, based on the following criteria:

- **Java license key format**—For customers who use WebLogic Server 5.1 or older, prefer to use a Java class file license, and do not need to support Type 4 JDBC drivers.
- **XML license key format** —For customers who use WebLogic Server 6.0 or newer, or who need to support Type 4 JDBC drivers.

Choosing a License Type

BEA provides three types of licenses.

- **Evaluation License**—An evaluation license is a limited license that a customer uses to evaluate WebLogic Server. It has a time limit, and is generally obtained when the customer downloads WebLogic Server for evaluation purposes only.
- **Developer License**—A developer license gives a customer access to all WebLogic Server functionality, but restricts the number of server connections. This type of license is typically used in a development environment where a developer is preparing an application to run on top of WebLogic Server; a production license is required for product deployment.
- **Production License**—A production license allows the development of unlimited connections for an unrestricted period of time for a specific IP address. ISVs receive an extended version of a production license that allows them to distribute the server to many customers, while keeping control of the license through a special encryption key that is specified during server startup.

Creating an Entry Point to WebLogic Server

Developers who want to extend WebLogic Server to support their own products use the `ISVServer` API to create a programmatic entry point into WebLogic Server, or start WebLogic Server using command-line arguments.

Creating an entry point into WebLogic Server is explored in the following sections:

- [Running WebLogic Server at Startup](#)
- [Using WebLogic Server JDBC Drivers](#)

Running WebLogic Server at Startup

The WebLogic Server [Administration Guide](#) provides extensive information about creating and using simple command-line startup scripts in the following sections:

- [“Starting the WebLogic Administration Server”](#)
- [“Starting the WebLogic Managed Servers Using Scripts”](#)
- [“Starting the WebLogic Administration Server from the Command Line”](#)

Using WebLogic Server JDBC Drivers

Java Database Connectivity (JDBC) is a Java API for executing SQL statements. The API consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool and database developers, and makes it possible to write database applications using a pure Java API. JDBC is a low-level interface, which means that it is used to directly call or invoke SQL commands. In addition, JDBC is a base upon which to build higher-level interfaces and tools, such as Java Messaging Service (JMS) and Enterprise Java Beans (EJB).

Additional information is available in *Programming WebLogic JDBC*, [“Introduction to WebLogic JDBC.”](#)

Types of JDBC Drivers

WebLogic Server uses the following types of JDBC drivers that work in conjunction with each other to provide database access:

- Two-tier drivers provide database access directly between a Java application and the driver vendor's database. WebLogic Server uses a DBMS vendor-specific JDBC driver, such as the WebLogic jDrivers for Oracle, Informix and Microsoft SQL Server, to connect to a back-end database.
- Type 4 JDBC drivers contain no “native” code—they are pure Java, which means they can run on any standard Java platform without additional software from the DBMS vendor.
- Multitier drivers provide vendor-neutral database access. A Java client application can use a multitier driver to access any database configured in WebLogic Server. BEA offers three multitier drivers—Remote Method Invocation (RMI), Pool, and Java Transaction Service (JTS).

The middle tier architecture allows you to manage database resources centrally in WebLogic Server. The vendor-neutral multitier JDBC drivers makes it easier to adapt purchased components to your DBMS environment and to write more portable code.

For more details, see *Programming WebLogic JDBC*, [“Overview of JDBC Drivers.”](#)

3 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.management.MBeanServer API](#).
- **MBeanHome**—Each WebLogic Server publishes an `MBeanHome` interface, which is a wrapper on `MBeanServer`, exposing a strongly-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 Administration `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` suffixes 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 = mbeanHome.getMBeanServer();
CounterMonitor monitor = new CounterMonitor();
CounterListener listener = new CounterListener();
. . .
```

3 *Programming WebLogic Server MBeans*

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 *addSomeCollection* or *removeSomeCollection* 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.NotificationListener;
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, subsystem, 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*](#).

4 Getting Technical Support

The WebLogic Server installer automatically places an evaluation license (`license.bea`) in the BEA Home directory. An evaluation license does not entitle a user to get the level of technical support that is provided to license holders. To receive the extended technical support that software developers need, a customer must upgrade their evaluation license.

This section describes the two different support levels, and the information that a customer must submit to technical support with their request:

- [Getting License Support](#)
- [Getting Non-License Support](#)
- [Upgrading an Evaluation License](#)

Getting License Support

For license support, email `support@weblogic.com` and include the following information:

- Specify that the support request is for a developer license
- Give a full description of the problem
- Provide detailed contact information (for example, name, phone(s), email, availability)

- Output received after running the command `java.util.writeLicense`
- The procedure used to start WebLogic

Getting Non-License Support

For other non-license support, include the following information:

- Give a full description of the problem
- Describe the environment
 - Operating system and version
 - JDK vendor and version
 - WebLogic version
- Provide detailed contact information (for example, name, phone(s), email, availability)

Upgrading an Evaluation License

Customers with an existing WebLogic Server license key can upgrade to the most current version at the WebSUPPORT Customer Support Site

The following procedure describes the steps required to upgrade a WebLogic Server evaluation license, in this example an upgrade to WebLogic Server 6.0.

1. Acquire the upgraded license from BEA Customer Support.
2. Move the license upgrade file to your BEA Home directory.
3. Windows users need to open a command shell window.

4. Open the BEA Home directory that contains your existing `license.bea` file and the new license upgrade file.
5. Add the JDK to your `PATH` by entering one of the following commands:
 - a. Windows systems:
 - `set PATH=../jdk130/bin:%PATH%`
 - b. UNIX systems:
 - `set PATH=../jdk130/bin:`
6. Enter the following command:
 - a. Windows systems:
 - `UpdateLicense license_wls60.bea`
 - b. UNIX systems:
 - `sh UpdateLicense.sh license_wls60.bea`

Index

C

customer support contact information vii

D

documentation, where to find it vi

P

printing product documentation vi

S

support
 technical vii