BEA WebLogic Server™

BETA

Introduction to WebLogic Server®

Version 9.0 BETA
Revised: JDecember 15, 2004
Part Number:

Copyright

Copyright © 2005 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.

BETA

Contents

1. Introduction to WebLogic Server

BETA

2. WebLogic Server Services

3. Overview of WebLogic Server System Administration

BETA

BETA

4. Overview of the Administration Console

BETA

CHAPTER 1

Introduction to WebLogic Server

Note: For a complete discussion of new features in WebLogic Server 9.0, see What's New in WebLogic Server 9.0 Beta.

The following sections provide an overview of WebLogic Server:

BETA

The WebLogic Server Solution

Today's business environment demands Web and e-commerce applications that accelerate your entry into new markets, help you find new ways to reach and retain customers, and allow you to introduce new products and services quickly. To build and deploy these new solutions, you need a proven, reliable e-commerce platform that can connect and empower all types of users while integrating your corporate data, mainframe applications, and other enterprise applications in a powerful, flexible, end-to-end e-commerce solution. Your solution must provide the performance, scalability, and high availability needed to handle your most critical enterprise-scale computing.

As the industry-leading e-commerce transaction platform, WebLogic Server allows you to quickly develop and deploy reliable, secure, scalable and manageable applications. It manages system-level details so you can concentrate on business logic and presentation.

J2EE Platform

WebLogic Server implements Java 2 Platform, Enterprise Edition (J2EE) version 1.4 technologies (`http://java.sun.com/j2ee/1.4/index.jsp`). J2EE is the standard platform for developing multitier enterprise applications based on the Java programming language. The technologies that make up J2EE were developed collaboratively by Sun Microsystems and other software vendors, including BEA Systems.

J2EE applications are based on standardized, modular components. WebLogic Server provides a complete set of services for those components and handles many details of application behavior automatically, without requiring programming.

Application Deployment Across Distributed, Heterogeneous Environments

WebLogic Server provides essential features for developing and deploying mission-critical e-commerce applications across distributed, heterogeneous computing environments. These features include the following:

- Standards leadership—Comprehensive enterprise Java support to ease the implementation and deployment of application components. WebLogic Server is the first independently developed Java application server to achieve J2EE certification. In addition, BEA actively participates in the development of J2EE and Web Services standards that drive innovation and advancement in Java and XML technology.
- Rich client options—WebLogic Server supports Web browsers and other clients that use HTTP; Java clients that use RMI (Remote Method Invocation) or IIOP (Internet Inter-ORB Protocol);

BETA

SOAP clients on any SOAP-enabled plaftorm; and mobile devices that use (WAP) Wireless Access Protocol. Connectors from BEA and other companies enable virtually any client or legacy application to work with a WebLogic Server application.

Note: Flexible Web services—WebLogic Server provides a solid platform for deploying Web services as components of a heterogeneous distributed application. Web services use a cross-platform, cross-language data model (XML) to provide interoperability among application components on diverse hardware and software platforms.

- Enterprise e-business scalability—Efficient use and high availability of critical resources are achieved through Enterprise JavaBean business components and mechanisms such as WebLogic Server clustering for dynamic Web pages, backend resource pooling, and connection sharing.
- Robust administration—WebLogic Server offers a Web-based Administration Console for configuring and monitoring WebLogic Server services. A scripting based on Jython makes it convenient to configure and monitor WebLogic Server instances with scripts.
- E-commerce-ready security—WebLogic Server provides Secure Sockets Layer (SSL) support for encrypting data transmitted across WebLogic Server, clients, and other servers. User authentication and authorization for all WebLogic Server services are provided through roles and security providers. External security stores, such as Lightweight Directory Access Protocol (LDAP) servers, can still be adapted to WebLogic realms, enabling single sign-on for the enterprise. The Security Service Provider Interface makes it possible to extend WebLogic Security services and to implement WebLogic Security features in applications.
- Maximum development and deployment flexibility—WebLogic Server provides tight integration with and support for leading databases, development tools, and other environments.
- Java Message Service (JMS)—An enterprise messaging system, also referred to as message-oriented middleware (MOM), enables applications to communicate with one another through the exchange of messages. A message is a request, report, and/or event that contains information needed to coordinate communication between different applications. A message provides a level of abstraction, allowing you to separate the details about the destination system from the application code.

  The Java Message Service (JMS) is a standard API for accessing enterprise messaging systems. Specifically, JMS enables Java applications sharing a messaging system to exchange messages, and it simplifies application development by providing a standard interface for creating, sending, and receiving messages.

BETA

WebLogic Server Application Architecture

WebLogic Server is an application server: a platform for developing and deploying multitier distributed enterprise applications. WebLogic Server centralizes application services such as Web server functionality, business components, and access to backend enterprise systems. It uses technologies such as caching, connection pooling and work managers to improve resource use and application performance. WebLogic Server also provides enterprise-level security and powerful administration facilities.

WebLogic Server operates in the middle tier of a multitier (or *n*-tier) architecture. A multitier architecture determines where the software components that make up a computing system are executed in relation to each other and to the hardware, network, and users. Choosing the best location for each software component lets you develop applications faster; eases deployment and administration; and provides greater control over performance, utilization, security, scalability, and reliability.

WebLogic Server implements J2EE, the Java Enterprise standard. Java is a network-savvy, object-oriented programming language, and J2EE includes component technologies for developing distributed objects. This functionality adds a second dimension to the WebLogic Server application architecture—a layering of application logic, with each layer selectively deployed among WebLogic Server J2EE technologies.

The next two sections describe these two views of WebLogic Server architecture: software tiers and application logic layers.

BETA

Software Component Tiers

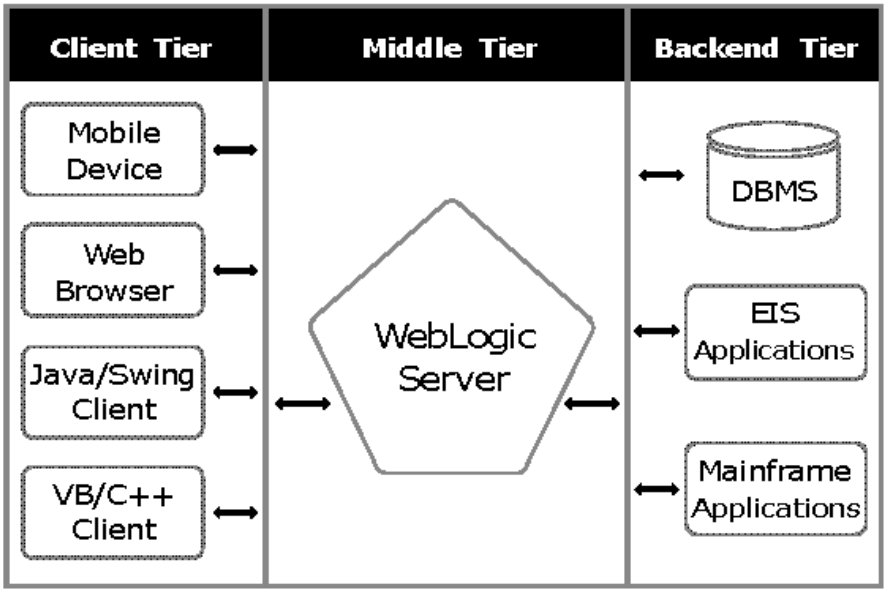
The software components of a multitier architecture consist of three tiers:

- The *client tier* contains programs executed by users, including Web browsers and network-capable application programs. These programs can be written in virtually any programming language.
- The *middle tier* contains WebLogic Server and other servers that are addressed directly by clients, such as existing Web servers or proxy servers.
- The *backend tier* contains enterprise resources, such as database systems, mainframe and legacy applications, and packaged enterprise resource planning (ERP) applications.

Client applications access WebLogic Server directly, or through another Web server or proxy server. WebLogic Server generally connects with backend services on behalf of clients. However, clients may directly access backend services using a multitier JDBC driver.

Figure 1-1 illustrates the three tiers of the WebLogic Server architecture.

Figure 1-1 Three-Tier Architecture



Client Tier Components

WebLogic Server clients use standard interfaces to access WebLogic Server services. WebLogic Server has complete Web server functionality, so a Web browser can request pages from WebLogic Server using the Web's standard HTTP protocol. WebLogic Server servlets and JavaServer Pages (JSPs) produce the dynamic, personalized Web pages required for advanced e-commerce Web applications.

Client programs written in Java may include highly interactive graphical user interfaces built with Java Swing classes. They can also access WebLogic Server services using standard J2EE APIs.

All these services are also available to Web browser clients by deploying servlets and JSP pages in WebLogic Server.

This version of WebLogic Server supports a true J2EE application client. In previous versions, a WebLogic client that could fully utilize WLS features such as clustering, security, transactions and JMS, required locating the complete WebLogic JAR on the client machine.

A J2EE application client runs on a client machine and can provide a richer user interface than can be provided by a markup language. Application clients directly access EJBs running in the business tier, and can, as appropriate, communicate through HTTP with servlets running in the Web tier. An application client is typically downloaded from the server, but can be installed on a client machine.

Although a J2EE application client is a Java application, it differs from a stand-alone Java application client because it is a J2EE component, hence it offers the advantages of portability to other J2EE-compliant servers, and can access J2EE services.

Middle Tier Components

The middle tier includes WebLogic Server and other Web servers, firewalls, and proxy servers that mediate traffic between clients and WebLogic Server. The Nokia WAP server, part of the BEA mobile commerce solution, is an example of another middle tier server that provides connectivity between wireless devices and WebLogic Server.

Applications based on a multitier architecture require reliability, scalability, and high performance in the middle tier. The application server you select for the middle tier is, therefore, critical to the success of your system.

The WebLogic Server cluster option allows you to distribute client requests and back-end services among multiple cooperating WebLogic Servers. Programs in the client tier access the cluster as if it were a single WebLogic Server. As the workload increases, you can add WebLogic Servers to the cluster to share the work. The cluster uses a selectable load-balancing algorithm to choose a WebLogic Server in the cluster that is capable of handling the request.

When a request fails, another WebLogic Server that provides the requested service can take over. Failover is transparent whenever possible, which minimizes the amount of code that must be written to recover from failures. For example, servlet session state can be replicated on a secondary WebLogic Server so that if the WebLogic Server that is handling a request fails, the client's session can resume uninterrupted on the secondary server. WebLogic EJB, JMS, JDBC, and RMI services are all implemented with clustering capabilities.

Backend Tier Components

The backend tier contains services that are accessible to clients only through WebLogic Server. Applications in the backend tier tend to be the most valuable and mission-critical enterprise resources. WebLogic Server protects them by restricting direct access by end users. With technologies such as connection pools and caching, WebLogic Server uses back-end resources efficiently and improves application response.

BETA

Backend services include databases, enterprise resource planning (ERP) systems, mainframe applications, legacy enterprise applications, and transaction monitors. Existing enterprise applications can be integrated into the backend tier using the Java Connector Architecture specification from Sun Microsystems. WebLogic Server makes it easy to add a Web interface to an integrated backend application.

A database management system is the most common backend service, required by nearly all WebLogic Server applications. WebLogic EJB and WebLogic JMS typically store persistent data in a database in the backend tier.

A JDBC connection pool, defined in WebLogic Server, opens a predefined number of database connections. Once opened, database connections are shared by all WebLogic Server applications that need database access. The expensive overhead associated with establishing a connection is incurred only once for each connection in the pool, instead of once per client request. WebLogic Server monitors database connections, refreshing them as needed and ensuring reliable database services for applications.

Application Logic Layers

WebLogic Server implements J2EE component technologies and services. J2EE component technologies include servlets, JSP Pages, and Enterprise JavaBeans. J2EE services include access to standard network protocols, database systems, and messaging systems. To build a WebLogic Server application, you must create and assemble components, using the service APIs when necessary.

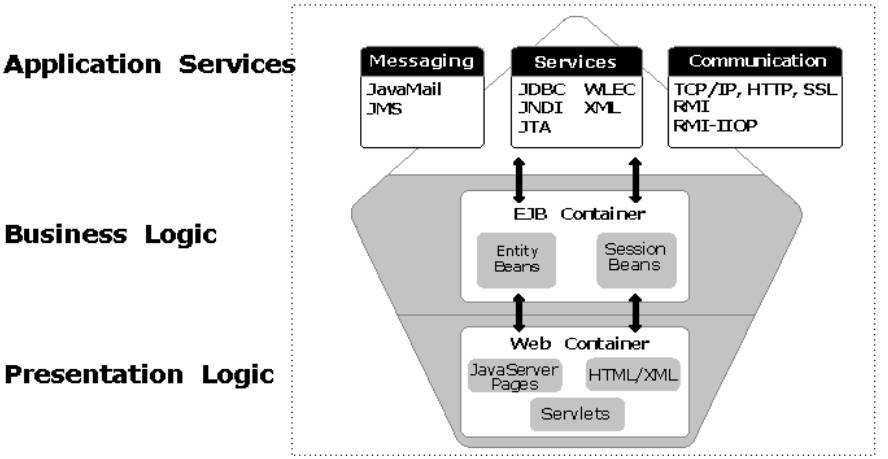
Components are executed in the WebLogic Server Web container or EJB container. Containers provide the life cycle support and services defined by the J2EE specifications so that the components you build do not have to handle underlying details.

Web components provide the presentation logic for browser-based J2EE applications. EJB components encapsulate business objects and processes. Web applications and EJBs are built on J2EE application services, such as JDBC, JMS (Java Messaging Service), and JTA (Java Transaction API).

Figure 1-2 illustrates WebLogic Server component containers and application services.

BETA

Figure 1-2 Application Logic Layers



The following sections discuss the presentation logic, business logic, and application services layers.

Presentation Logic Layer

The presentation layer includes an application's user interface and display logic. Most J2EE applications use a Web browser on the client machine because it is much easier than deploying client programs to every user's computer. In this case, the presentation logic is the WebLogic Server Web container. Client programs written in any programming language, however, must contain either logic to render HTML or their own presentation logic.

Web Browser Clients

Web-based applications built with standard Web technologies are easy to access, maintain, and port. Web browser clients are standard for e-commerce applications.

In Web-based applications, the user interface is represented by HTML documents, JavaServer Pages (JSP), and servlets. The Web browser contains the logic to render the Web page on the user's computer from the HTML description.

JavaServer Pages (JSP) and servlets are closely related. Both produce dynamic Web content by executing Java code on WebLogic Server each time they are invoked. The difference between them is that JSP is written with an extended version of HTML, and servlets are written with the Java programming language.

BETA

JSP is convenient for Web designers who know HTML and are accustomed to working with an HTML editor or designer. Servlets, written entirely in Java, are more suited to Java programmers than to Web designers. Writing a servlet requires some knowledge of the HTTP protocol and Java programming. A servlet receives the HTTP request in a request object and typically writes HTML or XML in its response object.

JSP pages are converted to servlets before they are executed on WebLogic Server, so ultimately JSP pages and servlets are different representations of the same thing. JSP pages are deployed on WebLogic Server the same way an HTML page is deployed. The `.jsp` file is copied into a directory served by WebLogic Server. When a client requests a `.jsp` file, WebLogic Server checks whether the page has been compiled or has changed since it was last compiled. If needed, it calls the WebLogic JSP compiler, which generates Java servlet code from the `.jsp` file, and then it compiles the Java code to a Java class file.

Non-Browser Clients

A client program that is not a Web browser must supply its own code for rendering the user interface. Non-browser clients usually contain their own presentation and rendering logic, depending on WebLogic Server only for business logic and access to back-end services. This makes them more difficult to develop and deploy and less suited for Internet-based e-commerce applications than browser-based clients.

Java programs can use the Java Swing classes to create powerful and portable user interfaces. Client programs written in Java can use any WebLogic Server service over Java RMI (Remote Method Invocation)., allowing the client to operate on a WebLogic Server object the same way it would operate on a local object in the client. Because RMI hides the details of making calls over a network, J2EE client code and server-side code are very similar.

To leverage WebLogic Server services over RMI, WebLogic Server classes must be available on the client. WebLogic Server 8.1 supports a true J2EE application client, referred to herein as the *thin client*. Small footprint standard and JMS jars—each about 400 KB—are provided. The WebLogic Server client can leverage standared J2EE artifacts such as InitialContext, UserTransaction, and EJBs. It supports iiop, iiops, http, https, t3, and t3s—each of which can be selected by using a different URL in InitialContext.

WebLogic RMI-IIOP allows CORBA-enabled programs to execute WebLogic Server enterprise beans and RMI classes as CORBA objects. The WebLogic Server RMI and EJB compilers can generate IDL (Interface Definition Language) for RMI classes and enterprise beans. IDL generated this way is compiled to create skeletons for an ORB (Object Request Broker) and stubs for the client program. WebLogic Server parses incoming IIOP requests and dispatches them to the RMI run-time system.

BETA

Business Logic Layer

Enterprise JavaBeans are the business logic components for J2EE applications. The WebLogic Server EJB container hosts enterprise beans, providing life cycle management and services such as caching, persistence, and transaction management.

There are three types of enterprise beans: entity beans, session beans, and message-driven beans. The following sections describe each type in detail.

Entity Beans

An entity bean represents an object that contains data, such as a customer, an account, or an inventory item. Entity beans contain data values and methods that can be invoked on those values. The values are saved in a database (using JDBC) or some other data store. Entity beans can participate in transactions involving other enterprise beans and transactional services.

Entity beans are often mapped to objects in databases. An entity bean can represent a row in a table, a single column in a row, or an entire table or query result. Associated with each entity bean is a unique primary key used to find, retrieve, and save the bean.

An entity bean can employ one of the following:

- Bean-managed persistence—the bean contains code to retrieve and save persistent values.
- Container-managed persistence—the EJB container loads and saves values on behalf of the bean.

When container-managed persistence is used, the WebLogic EJB compiler can generate JDBC support classes to map an entity bean to a row in a database.

Entity beans can be shared by many clients and applications. An instance of an entity bean can be created at the request of any client, but it does not disappear when that client disconnects. It continues to live as long as any client is actively using it. When the bean is no longer in use, the EJB container may passivate it: that is, it may remove the live instance from the server.

Session Beans

A session bean is a transient EJB instance that serves a single client. Session beans tend to implement procedural logic; they embody actions more than data.

The EJB container creates a session bean at a client's request. It then maintains the bean as long as the client maintains its connection to the bean. Sessions beans are not persistent, although they can save data to a persistent store if needed.

BETA

A session bean can be stateless or stateful. Stateless session beans maintain no client-specific state between calls and can be used by any client. They can be used to provide access to services that do not depend on the context of a session, such as sending a document to a printer or retrieving read-only data into an application.

A stateful session bean maintains state on behalf of a specific client. Stateful session beans can be used to manage a process, such as assembling an order or routing a document through a workflow process. Because they can accumulate and maintain state through multiple interactions with a client, session beans are often the controlling objects in an application. Because they are not persistent, session beans must complete their work in a single session and use JDBC, JMS, or entity beans to record the work permanently.

Message-Driven Beans

Message-driven beans, introduced in the EJB 2.0 specification, are enterprise beans that handle asynchronous messages received from JMS Message Queues. JMS routes messages to a message-driven bean, which selects an instance from a pool to process the message.

Message-driven beans are managed in the WebLogic Server EJB container. Because they are not called directly by user-driven applications, they cannot be accessed from an application using an EJB home. A user-driven application can, however, instantiate a message-driven bean indirectly by sending a message to the bean's JMS Queue.

Application Services Layer

WebLogic Server supplies the fundamental services that allow components to concentrate on business logic without concern for low-level implementation details. It handles networking, authentication, authorization, persistence, and remote object access for EJBs and servlets. Standard Java APIs provide portable access to other services that an application can use, such as database and messaging services.

XML Implementation

WebLogic Server consolidates Extensible Markup Language (XML) technologies applicable to WebLogic Server and XML applications based on WebLogic Server. A simplified version of the Standard Generalized Markup Language (SGML) markup language, XML describes the content and structure of data in a document and is an industry standard for delivering content on the Internet. Typically, XML is used as the data exchange format between J2EE applications and client applications, or between components of a J2EE application. The WebLogic Server XML subsystem

BETA

supports the use of standard parsers, the WebLogic FastParser, the WebLogic PullParser, XSLT transformers, and DTDs and XML schemas to process and convert XML files.

Data and Access Services

WebLogic Server implements standard J2EE technologies to provide data and access services to applications and components. These services include the following APIs:

- Java Naming and Directory Interface (JNDI)
- Java Database Connectivity (JDBC)
- Java Transaction API (JTA)
- J2EE Connector Architecture
- eXtensible Markup Language (XML)

The following sections discuss these services in detail.

JNDI

The Java Naming and Directory Interface (JNDI) is a standard Java API that enables applications to look up an object by name. WebLogic Server or a user application binds the Java objects it serves to a name in a naming tree. An application can look up objects, such as RMI objects, Enterprise JavaBeans, JMS Queues and Topics, and JDBC DataSources, by getting a JNDI context from WebLogic Server and then calling the JNDI lookup method with the name of the object. The lookup returns a reference to the WebLogic Server object.

WebLogic JNDI supports WebLogic Server cluster load balancing and failover. Each WebLogic Server in a cluster publishes the objects it serves in a replicated cluster-wide naming tree. An application can get an initial JNDI context from any WebLogic Server in the cluster, perform a lookup, and receive an object reference from any WebLogic Server in the cluster that serves the object. A configurable load-balancing algorithm is used to spread the workload among the servers in the cluster.

JDBC

Java Database Connectivity (JDBC) provides access to backend database resources. Java applications access JDBC using a JDBC driver, which is a database vendor-specific interface for a database server. Although any Java application can load a vendor's JDBC driver, connect to the database, and perform database operations, WebLogic Server provides a significant performance advantage by offering JDBC connection pools.

BETA

A JDBC connection pool is a named group of JDBC connections managed through WebLogic Server. At startup time WebLogic Server opens JDBC connections and adds them to the pool. When an application requires a JDBC connection, it gets a connection from the pool, uses it, and then returns it to the pool for use by for other applications. Establishing a database connection is often a time-consuming, resource-intensive operation, so a connection pool, which limits the number of connection operations, improves performance.

WebLogic Server also provides JDBC multipools for achieving load balancing or high availability capabilities with database connections in single-server configurations. Multipools are a "pool of pools" that provide a configurable algorithm for choosing which pool to provide a connection for a given request. Currently, WebLogic Server provides algorithms to support either high availability or load balancing behavior for database connections.

To register a connection pool in the JNDI naming tree, define a DataSource object for it. Java client applications can then get a connection from the pool by performing a JNDI lookup on the DataSource name.

Server-side Java classes use the WebLogic JDBC pool driver, which is a generic JDBC driver that calls through to the vendor-specific JDBC driver. This mechanism makes application code more portable, even if you change the brand of database used in the backend tier.

The client-side JDBC driver is the WebLogic JDBC/RMI driver, which is an RMI interface to the pool driver. Use this driver the same way you use any standard JDBC driver. When the JDBC/RMI driver is used, Java programs can access JDBC in a manner consistent with other WebLogic Server distributed objects, and they can keep database data structures in the middle tier.

WebLogic EJB and WebLogic JMS rely on connections from a JDBC connection pool to load and save persistent objects. By using EJB and JMS, you can often get a more useful abstraction than you can get by using JDBC directly in an application. For example, using an enterprise bean to represent a dataful object allows you to change the underlying store later without modifying JDBC code. If you use persistent JMS messages instead of coding database operations with JDBC, it will be easier to adapt your application to a third-party messaging system later.

JTA

The Java Transaction API (JTA) is the standard interface for managing transactions in Java applications. By using transactions, you can protect the integrity of the data in your databases and manage access to that data by concurrent applications or application instances. Once a transaction begins, all transactional operations must commit successfully or all of them must be rolled back.

BETA

WebLogic Server supports transactions that include EJB, JMS, JCA, and JDBC operations. Distributed transactions, coordinated with two-phase commit, can span multiple databases that are accessed with XA-compliant JDBC drivers, such as BEA WebLogic jDriver for Oracle/XA.

The EJB specification defines bean-managed and container-managed transactions.When an enterprise bean is deployed with container-managed transactions, WebLogic Server coordinates the transaction automatically. If an enterprise bean is deployed with bean-managed transactions, the EJB programmer must provide transaction code.

Application code based on the JMS or JDBC API can initiate a transaction, or participate in a transaction started earlier. A single transaction context is associated with the WebLogic Server thread executing an application; all transactional operations performed on the thread participate in the current transaction.

J2EE Connector Architecture

The J2EE Connector Architecture adds simplified Enterprise Information System (EIS) integration to the J2EE platform. It provides a Java solution to the problem of connectivity between the multitude of application servers and EISes. By using the Connector Architecture, it is no longer necessary for EIS vendors to customize their product for each application server. By conforming to the J2EE Connector Architecture, BEA WebLogic Server does not require added custom code in order to extend its support connectivity to a new EIS.

The J2EE Connector Architecture is implemented both in WebLogic Server and in an EIS-specific resource adapter. A resource adapter is a system library specific to an EIS and provides connectivity to the EIS. A resource adapter is analogous to a JDBC driver. The interface between a resource adapter and the EIS is specific to the underlying EIS, and can be a native interface.

The J2EE Connector Architecture comprises the system-level contracts between WebLogic Server and a given resource adaptor, a common interface for clients to access the adaptor, and interfaces for packaging and deploying resource adaptors to J2EE applications. See Programming WebLogic Server J2EE Connectors for more information.

XML

WebLogic Server consolidates Extensible Markup Language (XML) technologies applicable to WebLogic Server and XML applications based on WebLogic Server. For more information, refer to "XML Implementation" on page 1-11.

BETA

Messaging Technologies

The J2EE messaging technologies provide standard APIs that WebLogic Server applications can use to communicate with one another as well as with non-WebLogic Server applications. The messaging services include the following APIs:

- Java Message Service (JMS)
- JavaMail

The following sections describe these APIs in detail.

JMS

Java Messaging Service (JMS) enables applications to communicate with one another by exchanging messages. A message is a request, report, and/or event that contains the information needed to coordinate communication between different applications. A message provides a level of abstraction, allowing you to separate details about the destination system from the application code.

WebLogic JMS implements two messaging models: point-to-point (PTP) and publish/subscribe (pub/sub). The PTP model allows any number of senders to send messages to a Queue. Each message in the Queue is delivered to a single reader. The pub/sub model allows any number of senders to send messages to a Topic. Each message on the Topic is sent to every reader with a subscription to the Topic. Messages can be delivered to readers synchronously or asynchronously; the particular messaging mode can be controlled either using the Administration Console or via the method used to send messages in the JMS application.

JMS messages can be persistent or non-persistent. Persistent messages are stored in a database and are not lost if WebLogic Server restarts. Non-persistent messages are lost if WebLogic Server is restarted. Persistent messages sent to a Topic can be retained until all interested subscribers have received them.

JMS supports several message types that are useful for different types of applications. The message body can contain arbitrary text, byte streams, Java primitive data types, name/value pairs, serializable Java objects, or XML content.

JavaMail

WebLogic Server includes the Sun JavaMail reference implementation. JavaMail allows an application to create e-mail messages and send them through an SMTP server on the network.

Table 1-1

BETA

BETA

CHAPTER 2

WebLogic Server Services

Note: For a complete discussion of new features in WebLogic Server 9.0, see What's New in WebLogic Server 9.0 Beta.

The following sections describe WebLogic Server services:

BETA

WebLogic Server as a Web Server

WebLogic Server can be used as the primary Web server for advanced Web-enabled applications. A J2EE Web application is a collection of HTML or XML pages, JavaServer Pages, servlets, Java classes, applets, images, multimedia files, and other types of files.

How WebLogic Server Functions as a Web Server

A Web application runs in the Web container of a Web server. In a WebLogic Server environment, a Web server is a logical entity, deployed on one or more WebLogic Servers in a cluster.

The files in a Web application are stored in a directory structure that, optionally, can be packed into a single `.war` (Web ARchive) file using the Java `jar` utility. A set of XML deployment descriptors define the components and run-time parameters of an application, such as security settings. Deployment descriptors make it possible to change run-time behaviors without changing the contents

of Web application components, and they make it easy to deploy the same application on multiple Web servers.

Web Server Features

When used as a Web server, WebLogic Server supports the following functionality:

- Virtual hosting
- Support for proxy server configurations
- Load balancing
- Failover

This section describes how each function is supported by WebLogic Server.

Virtual Hosting

WebLogic Server supports virtual hosting, an arrangement that allows a single WebLogic Server instance or WebLogic Server cluster to host multiple Web sites. Each logical Web server has its own host name, but all Web servers are mapped in DNS to the same cluster IP address. When a client sends an HTTP request to the cluster address, a WebLogic Server is selected to serve the request. The Web server name is extracted from the HTTP request headers and is maintained on subsequent exchanges with the client so that the virtual host name remains constant from the client's perspective.

Multiple Web applications can be deployed on a WebLogic Server, and each Web application can be mapped to a virtual host.

Using Proxy Server Configurations

WebLogic Server can be integrated with existing Web servers. Requests can be proxied from a WebLogic Server to another Web server or, using a native plug-in supplied with WebLogic Server, from another Web server to WebLogic Server. BEA supplies plug-ins for Apache Web Server, Netscape Enterprise Server, and Microsoft Internet Information Server.

The use of proxy Web servers between clients and a set of independent WebLogic Servers or a WebLogic Server cluster makes it possible to perform load balancing and failover for Web requests. To the client, there appears to be only one Web server.

BETA

Load Balancing

You can set up multiple WebLogic Servers behind a proxy server to accommodate large volumes of requests. The proxy server performs load-balancing by distributing requests across the multiple servers in the tier behind it.

The proxy server can be a WebLogic Server Web server, or it can be an Apache, Netscape, or Microsoft Web server. WebLogic Server includes native code plug-ins for some platforms that allow these third-party Web servers to proxy requests to WebLogic Server.

The proxy server is set up to redirect certain types of requests to the servers behind it. For example, a common arrangement is to configure the proxy server to handle requests for static HTML pages and redirect requests for servlets and JavaServer Pages to a WebLogic Server cluster behind the proxy.

Failover and Replication

When a Web client starts a servlet session, the proxy server may send subsequent requests that are part of the same session to a different WebLogic Server. WebLogic Server provides session replication to ensure that a client's session state remains available.

There are two types of session replication:

- *JDBC session replication* is used with a WebLogic Server cluster or with a set of independent WebLogic Servers. It does not require the WebLogic Server clustering option.
- *In-memory session replication* requires the WebLogic Server clustering option.

JDBC session replication writes session data to a database. Once a session has been started, any WebLogic Server the proxy server selects can continue the session by retrieving the session data from the database.

When a WebLogic Server cluster is deployed behind a proxy server, servlet sessions can be replicated over the network to a secondary WebLogic Server selected by the cluster, thus avoiding the need to access a database. In-memory replication uses fewer resources and is much faster than JDBC session replication, so it is the best way to provide failover for servlets in a WebLogic Server cluster.

WebLogic Server Security Service

The open, flexible security architecture of WebLogic Server delivers advantages to all levels of users and introduces an advanced security design for application servers. Companies now have a unique application server security solution that, together with clear and well-documented security policies and procedures, can assure the confidentiality, integrity and availability of the server and its data.

The key features of the new WebLogic Security Service include:

BETA

- A comprehensive and standards-based design.
- End-to-end security for WebLogic Server-hosted applications, from the mainframe to the Web browser.
- Legacy security schemes that integrate with WebLogic Server security, allowing companies to leverage existing investments.
- Security tools that are integrated into a flexible, unified system to ease security management across the enterprise.
- Easy customization of application security to business requirements through mapping of company business rules to security policies.
- Easy updates to security policies.
- Easy adaptability for customized security solutions.
- A modularized architecture, so that security infrastructures can change over time to meet the requirements of a particular company.
- Support for configuring multiple security providers, as part of a transition scheme or upgrade path.
- A separation between security details and application infrastructure, making security easier to deploy, manage, maintain, and modify as requirements change.
- Default, WebLogic security providers that provide you with a working security scheme out of the box.
- Customization using WebLogic custom security providers
- Unified management of security rules, security policies, and security providers through the WebLogic Server Administration Console.
- Support for standard J2EE security technologies such as the Java Authentication and Authorization Service (JAAS), Java Secure
- Sockets Extensions (JSSE), and Java Cryptography Extensions (JCE).

For more information about the WebLogic Security Service, see *Introduction to WebLogic Security*.

WebLogic Server Clusters

A WebLogic Server cluster is a group of WebLogic Server instances that work together to provide a powerful and reliable Web application platform. A cluster appears to its clients as a single server but

BETA

it is, in fact, a group of servers acting as one. It provides two key benefits that are not provided by a single server: scalability and availability.

Using WebLogic Server Clusters provides complete information about planning and configuring WebLogic Server clusters.

Benefits of Using Clusters

WebLogic Server clusters bring scalability and high-availability to J2EE applications in a way that is transparent to application developers. The benefit of scalability is that it expands the capacity of the middle tier beyond that of a single instance of WebLogic Server or a single computer. The only limitation on cluster membership is that all WebLogic Server instances must be able to communicate by IP multicast. New WebLogic Servers can be added to a cluster dynamically to increase capacity.

A WebLogic Server cluster also guarantees high availability by using the redundancy of multiple servers to insulate clients from failures. The same service can be provided on multiple servers in a cluster. If one server fails, another can take over. The ability to have a functioning server take over from a failed server increases the availability of the application to clients.

Cluster Architecture

A WebLogic Server cluster consists of a number of WebLogic Server instances deployed on a network, coordinated with a combination of Domain Name Service (DNS), JNDI naming tree replication, session data replication, and WebLogic RMI.

Web proxy servers between Web clients and the WebLogic Server cluster coordinate clustering services for servlets and JavaServer Pages. Web proxy servers can be other WebLogic Servers, or third-party Web servers from Netscape, Microsoft, or Apache, used with a plug-in supplied with WebLogic Server.

Web clients connect with a WebLogic Server cluster by directing requests to a proxy server. Java RMI-based clients connect with a WebLogic Server cluster using a cluster address defined on the network.

Server-side code also benefits from the load-balancing and failover services provided by a WebLogic Cluster. In J2EE applications, most application code runs in the middle tier and can use services distributed among several WebLogic Servers. For example, a servlet running on WebLogic Server *A* could use an enterprise bean on WebLogic Server *B* and read messages from a JMS Queue on WebLogic Server *C*.

BETA

Server Management and Monitoring

WebLogic Server administration is accomplished by setting attributes for the servers in a domain, using either the Administration Console or the command-line interface. The Administration Console is a Web browser application that allows you to configure WebLogic Server services, manage security, deploy applications, and monitor services dynamically.

Both the Administration Console and the command-line interface connect to the Administration Server.

Administration Server

The Administration Server is the WebLogic Server used to configure and manage all the WebLogic Servers in its domain. A domain may include multiple WebLogic Server clusters and independent WebLogic Server instances. If a domain contains only one WebLogic Server, then that server is the Administration Server. In a domain with multiple instances of WebLogic Server, the first instance to start must be the Administration Server.

Administration Console

The WebLogic Server Administration Console runs in a Web browser. It displays the components of the domain it administers, including clusters and independent WebLogic Servers, in a graphical tree in the left pane. The right pane displays details about the object selected in the left pane. See "The following sections describe the WebLogic Administration Console:" on page 4-1.

BETA

CHAPTER 3

Overview of WebLogic Server System Administration

BETA

The following sections provide an overview of system administration for WebLogic Server:

Introduction to System Administration

You manage a WebLogic Server installation by using any of several system administration tools provided with WebLogic Server. A WebLogic Server installation can consist of a single WebLogic Server instance or multiple instances, each hosted on one or more physical machines. The system administration tools include the Administration Console, command line utilities, the WebLogic Scripting Tool, (WLST) and the JMX API, with which you manage security, database connections, messaging, transaction processing, and the runtime configuration of your applications. The tools also allow you to monitor the health of the WebLogic Server environment to ensure maximum availability and performance for your applications.

WebLogic Server Domains

The basic administrative unit for a WebLogic Server installation is called a *domain*. A domain is a logically related group of WebLogic Server resources that you manage as a unit. A domain always includes at least one WebLogic Server instance called the *Administration Server*. The Administration Server serves as a central point of contact for server instances and system administration tools. A domain may also include additional WebLogic Server instances called *Managed Servers*.

You can configure some or all of these Managed Servers to be part of a WebLogic Server *cluster*. A cluster is a group of WebLogic Server instances that work together to provide scalability and high-availability for applications. A Managed Server in a cluster can act as a backup for services such as JMS and JTA that are hosted on another server instance in the cluster. A Managed Server can also function as a virtual host. Your applications are also deployed and managed as part of a domain.

For more information on configuring domains, see Understanding Domain Configuration.

You can organize your domains based on criteria such as:

- *Logical divisions of applications*. For example, a domain devoted to end-user functions such as shopping carts and another domain devoted to back-end accounting applications.
- *Physical location*. Domains for different locations or branches of your business.
- *Size*. Domains organized in small units that can be managed more efficiently, perhaps by different personnel.

For more information about WebLogic Server domains, see:

- Understanding WebLogic Server Domains
- "Creating WebLogicConfigurations Using the Configuration Wizard
- Configuring Domains in the *Administration Console Online Help*

BETA

Figure 3-1 WebLogic Server Domain

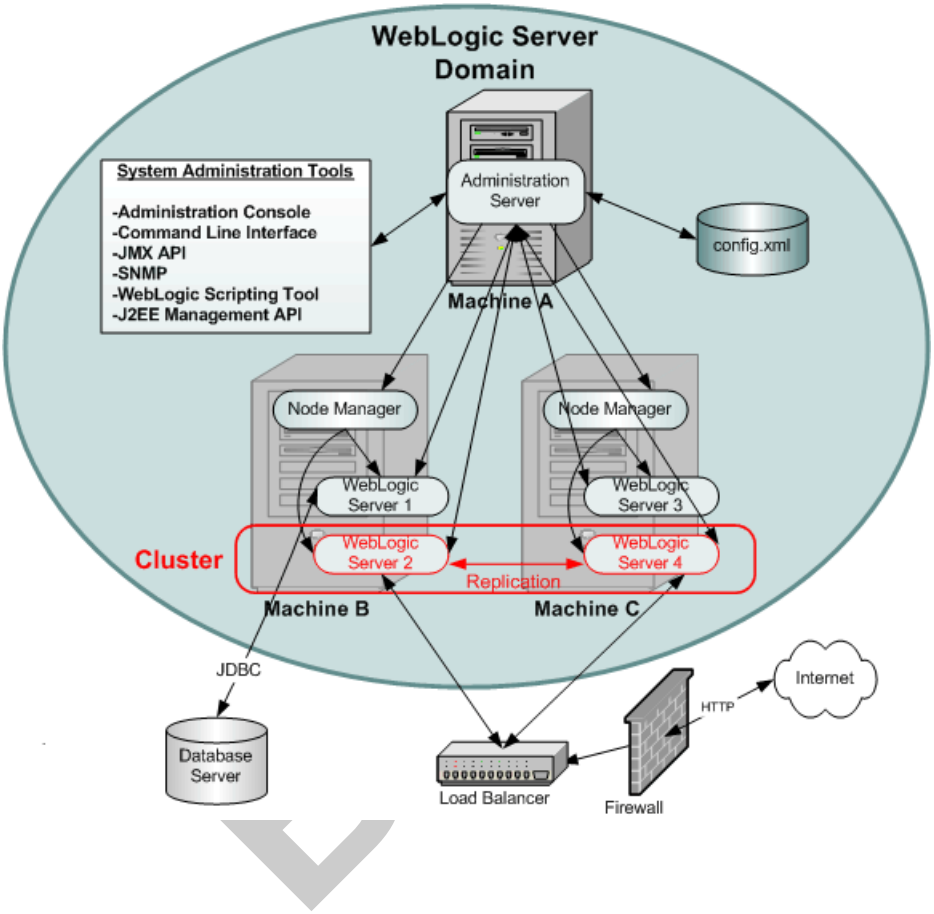


Figure 3-1 depicts a possible configuration of a WebLogic Server Domain—one of many possible configurations.

In the depicted domain, there are three physical machines:

Machine A hosts one instance of WebLogic Server, the Administration Server. The System Administration Tools communicate with the Administration Server to perform configuration and monitoring of the servers and applications in the domain. The Administration Server

communicates with each Managed Server on behalf of the System Administration Tools. The configuration for all the servers in the domain is stored in the configuration repository, the `config.xml` file, and other configuration files which reside on the machine hosting the Administration Server.

Machines B and C each host two instances of WebLogic Server, WebLogic Servers 1 through 4. These instances are called *Managed Servers*. The Administration Server communicates with an instance of Node Manager running on each machine to control startup and shutdown of the Managed Servers.

WebLogic Servers 2 and 4 are part of a *WebLogic Cluster* (outlined in red). This cluster is running an application that responds to HTTP requests routed to the cluster from a hardware load balancer. (An instance of WebLogic Server or a third-party Web server with one of the WebLogic Server plug-ins can also provide load balancing.) The load balancer processes HTTP requests from the Internet after they have passed through a firewall. The load balancer and firewall are not part of the domain. A replicated copy of objects such as HTTP sessions is passed between the two cluster members to provide failover capability.

WebLogic Server 1 runs an application that uses Java Database Connectivity (JDBC) to access a database server running on another physical machine that is not part of the WebLogic Domain.

Note: The pictured domain is only intended to illustrate the concepts of a WebLogic Server domain and how you manage the domain. Many possible configurations of servers, clusters, and applications are possible in a WebLogic Server domain.

The Administration Server and Managed Servers

One instance of WebLogic Server in each domain acts as an *Administration Server*. The Administration Server provides a central point for managing a WebLogic Server domain. All other WebLogic Server instances in a domain are called *Managed Servers*. In a domain with only a single WebLogic Server instance, that server functions both as Administration Server and Managed Server.

For a typical production system, BEA recommends that you deploy your applications only on Managed Servers. This practice allows you to dedicate the Administration Server to configuration and monitoring of the domain, while one or more Managed Servers service your applications.

For more information, see Overview of Starting and Stopping Servers in *Managing Server Startup and Shutdown*.

BETA

Recovery of a Failed Administration Server

A running Administration Server is always required to configure and monitor a domain. By maintaining backups of the `config.xml` file and certain other resources for a domain, you can replace a failed Administration Server with a backup WebLogic Server instance that can assume the role of Administration Server. For more information, see Overview of Starting and Stopping Servers in *Managing Server Startup and Shutdown*.

Managed Server Independence

When a Managed Server starts, it tries to contact the Administration Server to retrieve its configuration information. If a Managed Server cannot connect to the Administration Server during startup, it can retrieve its configuration by reading configuration and security files directly. A Managed Server that starts in this way is running in *Managed Server Independence (MSI)* mode. By default, MSI mode is enabled. Failover for Managed Servers is applicable to both clustered and non-clustered servers.

Domain-Wide Administration Port

You can enable an administration port for use with servers in a domain. The administration port is optional, but it provides two important capabilities:

- You can start a server in standby state. While in the standby state, the administration port remains available to activate or administer the server. However, the server's other network connections are unavailable to accept client connections. See Starting and Stopping WebLogic Server in *Managing Server Startup and Shutdown* for more information on the standby state.
- You can separate administration traffic from application traffic in your domain. In production environments, separating the two forms of traffic ensures that critical administration operations (starting and stopping servers, changing a server's configuration, and deploying applications) do not compete with high-volume application traffic on the same network connection.

  For more information, see "Enabling the Domain-Wide Administration Port" in *Administration Console Online Help*

BETA

System Administration Tools

Using JMS as the underlying architecture, System administration tools are provided for a variety of management functions. An Administration Server must be running when you use system administration tools to manage a domain. The following tools are discussed in the next sections:

BETA

Security Protections for System Administration Tools

All system administration operations are protected based on the user name employed to access a system administration tool. A user (or the group a user belongs to) must be a member of one of four security roles. These roles grant or deny a user access to various sets of system administration operations. The roles are Admin, Operator, Deployer, and Monitor. You can also set a security policy on WebLogic Server instances in a domain. For more information, see "Protecting User Accounts" in *Managing WebLogic Security*.

System Administration Console

The Administration Console is a Web Application hosted by the Administration Server. You access the Administration Console from any machine on the local network that can communicate with the Administration Server through a Web browser (including a browser running on the same

machine as the Administration Server). The Administration Console allows you to manage a WebLogic Server domain containing multiple WebLogic Server instances, clusters, and applications. The management capabilities include:

- Configuration
- Stopping and starting servers
- Monitoring server health and performance
- Monitoring application performance
- Viewing server logs
- Assistants, which help you create various configurations.

Through the Administration Console, System administrators can easily perform all WebLogic Server management tasks without having to learn the JMX API or the underlying management architecture. The Administration Server persists changes to attributes in the `config.xml` file for the domain you are managing.

See:

- "Starting the Administration Console" on page 4-2
- Administration Console Online Help. (The online help is also available from the Administration Console by clicking on the "?" icon located in the upper right portion of the console.)

WebLogic Scripting Tool

The WebLogic Scripting Tool (WLST) is a command-line scripting interface that system administrators and operators use to monitor and manage WebLogic Server instances and domains. The WLST scripting environment is based on the Java scripting interpreter, Jython. In addition to WebLogic scripting functions, you can use common features of interpreted languages, including local variables, conditional variables, and flow control statements. WebLogic Server developers and administrators can extend the WebLogic scripting language to suit their environmental needs by following the Jython language syntax. See http://www.jython.org.

WLST lets you perform the following tasks:

- Retrieve domain configuration and runtime information.
- Edit the domain configuration and persist the changes in the `config.xml` file.

BETA

- Edit custom, user-created MBeans and non-WebLogic Server MBeans, such as WebLogic Integration Server and WebLogic Portal Server MBeans.
- Automate domain configuration tasks and application deployment.
- Clone WebLogic Server domains..
- Control and manage the server life cycle.
- Access Node Manager and start, stop, and suspend server instances remotely or locally, without requiring the presence of a running Administration Server.

For more information on WLST, see WebLogic Scripting Tool.

JMX

Advanced Java programmers with knowledge of the JMX API from Sun Microsystems Inc. and WebLogic Server Mbeans can write their own management components as a Java class, or implement management components in applications deployed on WebLogic Server.

See:

- Developing Managable Applications with JMX
- Monitoring and Managing with the J2EE Management API
- Type-Safe Access for WebLogic Server MBeans (Deprecated)

Configuration Wizard

The Configuration Wizard is a tool for creating a new WebLogic Server domain or modifying an existing, inactive domain. The Configuration Wizard creates the appropriate directory structure for your domain, a basic `config.xml` file, and scripts you can use to start the servers in your domain. Depending on the options you select in the wizard, other files may also be created.

You can run the Configuration Wizard either using a graphical user interface (GUI) or in a text-based command line environment (this is called *console mode*—do not confuse this mode with the Administration *Console*). You can also create user-defined domain configuration templates for use by the Configuration Wizard.

See Creating WebLogic Domains Using the Configuration Wizard

BETA

Configuration Template Builder

A configuration template defines the full set of resources within a domain, including infrastructure components, applications, services, security options, and general environment and operating system options. BEA provides a number of templates and template extensions as part of the WebLogic Platform product.

The Configuration Template Builder provides the capability to easily create your own templates, to enable, for example, the definition and propagation of a standard domain across a development project, or to enable the distribution of a domain along with an application that has been developed to run on that domain. The templates you create with the Configuration Template Builder are used as input to the Configuration Wizard as the basis for creating a domain that is customized for your target environment. See Creating Templates Using the Domain Template Builder.

Java Utilities

Utility programs are provided for common tasks such as deploying applications and testing DBMS configurations. For more information, see Using the WebLogic Java Utilities in the *WebLogic Server Command Reference*.

Ant Tasks

You can use two Ant tasks provided with WebLogic Server to help you perform common configuration tasks in a development environment. Ant is a Java-based build tool similar to Make.The configuration tasks enable you to start and stop WebLogic Server instances as well as create and configure WebLogic Server domains. When combined with other WebLogic Ant tasks, you can create powerful build scripts for demonstrating or testing your application with custom domains.

See:

Using Ant Tasks to Configure a WebLogic Server Domain in *Developing Applications for WebLogic Server*.

Node Manager

Node Manager is a Java program provided with WebLogic Server that enables you to start, shut down, restart, and monitor remote WebLogic Server instances. To enable these capabilities, you run an instance of Node Manager on each physical machine in your domain.

BETA

See:

- Overview of Node Manager
- Configuring, Starting, and Stopping Node Manager
- Creating WebLogic Domains Using the Configuration Wizard

SNMP

WebLogic Server includes the ability to communicate with enterprise-wide management systems using Simple Network Management Protocol (SNMP). The WebLogic Server SNMP capability enables you to integrate management of WebLogic Servers into an SNMP-compliant management system that gives you a single view of the various software and hardware resources of a complex, distributed system.

See:

- WebLogic SNMP Management Guide
- WebLogic SNMP MIB Reference

Logs

Many WebLogic Server operations generate logs of their activity. Each server has its own log as well as a standard HTTP access log. These log files can be configured and used in a variety of ways to monitor the health and activity of your servers and applications.

See:

- Configuring Log Files and Filtering Log Messages
- Setting Up HTTP Access Logs

You can also configure a special *domain log* that contains a definable subset of log messages from all WebLogic Server instances in a domain. You can modify which logged messages from a local server appear in the domain log using the system administration tools. You can view this domain log using the Administration Console or a text editor/viewer.

WebLogic Diagnostic Service

The WebLogic Diagnostic Service is monitoring and diagnostic service that runs within WebLogic Server process and participates in the standard server life cycle. This service enables you to create, collect, analyze, archive, and access diagnostic data generated by a running server

BETA

and the applications deployed within its containers. This data provides insight into the runtime performance of servers and applications and enables you to isolate and diagnose faults when they occur.

The WebLogic Diagnostic Service provides a set of standardized application programming interfaces (APIs) that enable dynamic access and control of diagnostic data, as well as improved monitoring that provides visibility into the server. The interfaces are standardized to facilitate future enhancement and for easier integration of third-party tools, while maintaining the integrity of the server code base. Further, the service reduces customer total cost of ownership and support costs through improvements in WebLogic Server monitoring and diagnostic capabilities. The service is well suited to the server and the server's stack product components and targets operations and administrative staff as primary users.

See Understanding the WebLogic Diagnostic Service.

Managing Configuration Changes

To provide a secure, predictable means for distributing configuration changes in a domain, WebLogic Server imposes a change management process that loosely resembles a database transaction. The configuration of a domain is represented on the file system by a set of XML configuration files, centralized in the config.xml file, and at runtime by a hierarchy of Configuration MBeans. When you edit the domain configuration, you edit a separate hierarchy of Configuration MBeans that resides on the Administration Server. To start the edit process, you obtain a lock on the edit hierarchy to prevent other people from making changes. When you finish making changes, you save the changes. The changes do not take effect, however, until you activate them, distributing them to all server instances in the domain. When you activate changes, each server determines whether it can accept the change. If all servers are able to accept the change, they update their working configuration hierarchy and the change is completed.

See Understanding Domain Configuration.

Resources You Can Manage in a WebLogic Server Domain

This section discusses the domain resources you manage with the system administration tools.

Servers

See:

Creating, Configuring, and Monitoring Servers, in the *Administration Console Online Help*

Starting and Stopping Servers, in the *Administration Console Online Help*

BETA

weblogic.Server Command-Line Reference in the *WebLogic Server Command Reference*

Clusters

See:

- Using WebLogic Server Clusters
- Clusters, in the *Administration Console Online Help*

Machines

See:

- Using WebLogic Server Clusters
- Machines, in the *Administration Console Online Help*

Network Channels

See:

- Configuring Network Resources.

JDBC

See:

- JDBC, in the *Administration Console Online Help*
- Configuring and Managing JDBC

JMS

See:

- Configuring andManaging JMS

WebLogic Messaging Bridge

See:

- Configuring Messaging Bridge

BETA

Web Servers and Web Components

See::

- Configuring Web Server Functionality for WebLogic Server
- Virtual Hosts, in the *Administration Console Online Help*

You can also use Web Servers such as Apache, Microsoft IIS, and Netscape with WebLogic Server. For more information, see: Using WebLogic Server with Web Servers.

Applications

See:

- WebLogic Server Deployment
- Deploying Applications and Modules, in the *Administration Console Online Help*
- Securing WebLogic Resources.

Application Formats

See::

- Developing Applications
- Assembling and Configuring Web Applications
- Deploying Applications and Modules
- Developing WebLogic Server Applications
- Programming WebLogic Enterprise Java Beans
- Programming WebLogic J2EE Connectors
- Programming WebLogic Web Services
- Defining a Security Policy
- Setting Protections for WebLogic Resources

JNDI.

See:

BETA

- JNDI, in the *Administration Console Online Help*
- Programming WebLogic JNDI

Transactions

See:

- JTA, in the *Administration Console Online Help*
- Programming WebLogic JTA

XML

See:

- Administering WebLogic Server XML
- XML, in the *Administration Console Online Help*

Security

See:

- Managing WebLogic Security
- Using Compatibility Security in *Managing WebLogic Security*
- "Security" in the *Administration Console Online Help*
- Security Index Page

WebLogic Tuxedo Connector

See:

- WebLogic Tuxedo Connector
- WebLogic Tuxedo Connector (WTC), in the *Administration Console Online Help*

Jolt

See:

- BEA Jolt

BETA

- Jolt, in the *Administration Console Online Help*

Mail

See:

- "Using JavaMail with WebLogic Server Applications" under Programming Topics.
- Mail, in the *Administration Console Online Help*

Using WebLogic Server with Web Servers

You can proxy requests from popular Web servers to an instance of WebLogic Server or a cluster of WebLogic Servers by using one of the Web server plug-ins. Plug-ins are available for the following Web servers:

- Netscape Enterprise Server or IPlanet
- Microsoft Internet Information Server
- Apache

Because these plug-ins operate in the native environment of the Web server, you manage the plug-ins through the administration facilities of that Web server.

For more information, see Using WebLogic Server with Plug-ins.

Special servlets are also included with the WebLogic Server distribution to proxy requests from an instance of WebLogic Server to another instance of WebLogic Server or to a cluster of WebLogic Servers. For more information, see:

- Proxying Requests to a WebLogic Cluster in *Using WebLogic Server Clusters*

Monitoring

The system administration tools contain extensive capabilities for monitoring WebLogic Servers, domains, and resources. Using the tools you can monitor:

- Server health and performance:
  - Execute Queues
  - Connections
  - Sockets

BETA

  - Threads
  - Throughput
  - Memory Usage
- Security:
  - Locked-out users
  - Invalid Logins
  - Login attempts
- Transactions:
  - Committed transactions
  - Rolledback transactions
- JMS connections and servers
- WebLogic Messaging Bridge
- Applications:
  - Servlet sessions
  - Connector connection pools
  - EJB performance
- JDBC connections and connection pools.

BETA

Licenses

WebLogic Server requires a valid license to function. An evaluation copy of WebLogic Server is enabled for a limited time period so you can start using WebLogic Server immediately. To use WebLogic Server beyond the evaluation period, you will need to contact your salesperson about further evaluation or purchasing a license for each IP address on which you intend to use WebLogic Server. All WebLogic Server evaluation products are licensed for use on a single server with access allowed from up to 20 connections to the server.

If you downloaded WebLogic Server from the BEA Web site, your evaluation license is included with the distribution. The WebLogic Server installation program allows you to specify the location of the BEA home directory, and installs a BEA license file, `license.bea`, in that directory.

See “Licensing ” in *Installing WebLogic Platform.*

BETA

BETA

CHAPTER 4

Overview of the Administration Console

The following sections describe the WebLogic Administration Console:

BETA

About the Administration Console

The BEA WebLogic Server Administration Console is a Web browser-based, graphical user interface you use to manage a WebLogic Server domain. A WebLogic Server domain is a logically related group of WebLogic Server resources that you manage as a unit. A domain includes one or more WebLogic Servers and may also include WebLogic Server clusters. Clusters are groups of WebLogic Servers that work together to provide scalability and high-availability for applications. You deploy and manage your applications as part of a domain.

One instance of WebLogic Server in each domain is configured as an Administration Server. The Administration Server provides a central point for managing a WebLogic Server domain. All other WebLogic Server instances in a domain are called Managed Servers. In a domain with only a single WebLogic Server instance, that server functions both as Administration Server and Managed Server. The Administration Server hosts the Administration Console, which is a Web application accessible from any supported Web browser with network access to the Administration Server.

You can use the Administration Console to:

- Configure, start, and stop WebLogic Server instances.
- Configure WebLogic Server clusters.
- Configure WebLogic Server services, such as database connectivity (JDBC) and messaging (JMS).
- Configure security parameters, including managing users, groups, and roles.
- Configure and deploy your applications.
- Monitor server and application performance.
- View server and domain log files.
- View application deployment descriptors.
- Edit selected runtime application deployment descriptor elements.

Administration Console Online Help

The Administration Console includes numerous links to help pages. These online help pages are not available in the Beta release.

Console Errors

Messages (including information, warning, and error messages) can be generated in the course of using the Administration Console. In the Beta release, these messages are incremented in the System Status panel of the Administration Console, but generally are not otherwise visible in the Administration Console. You can view WebLogic Server logs from the Diagnostics > Log Files page of the console.

BETA

Starting the Administration Console

This section contains instructions for starting the Administration Console.

To use the Administration Console, use one of the supported Web Browsers for your environment. See *Browser Support for the WebLogic Server Console*. If your Web browser is not on this list of supported browsers, you may experience functional or formatting problems when using the Administration Console.

To start the Administration Console:

1. Start an Administration Server.
2. Open one of the supported Web browsers and open the following URL:

   `http://`*`hostname`*`:`*`port`*`/console`

   where *`hostname`* is the DNS name or IP address of the Administration Server and *`port`* is the listen port on which the Administration Server is listening for requests (port 7001 by default). If you have configured a domain-wide Administration port, use that port number. If you configured the Administration Server to use Secure Socket Layer (SSL) you must add `s` after `http` as follows:

   `https://`*`hostname`*`:`*`port`*`/console`

   Note: A domain-wide administration port always uses SSL.
3. When the login page appears, enter the user name and the password you used to start the Administration Server (you may have specified this user name and password during the installation process) or enter a user name that belongs to one of the following security groups: Administrators, Operators, Deployers, or Monitors. These groups provide various levels of access to system administration functions in the Administration Console.

Using the security system, you can add or delete users to one of these groups to provide controlled access to the console.

Note: If you have your browser configured to send HTTP requests to a proxy server, then you may need to configure your browser to not send Administration Server HTTP requests to the proxy. If the Administration Server is on the same machine as the browser, then ensure that requests sent to localhost or 127.0.0.1 are not sent to the proxy.

Elements of the Administration Console

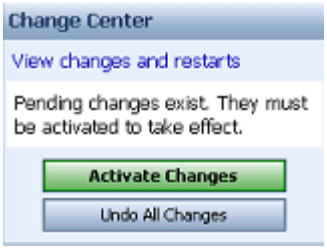
The Administration Console user interface includes the following panels:

Change Center

This is the starting point for using the Administration Console to make changes in WebLogic Server. See “Using the Change Center” on page 4-6.

BETA

Figure 4-1 Change Center



Domain Structure

This panel is a tree you can use to navigate to pages in the Administration Console. Click any of the nodes in the Domain Structure tree to go to that page. Click a + (plus) icon in the Domain Structure to expand a node and a - (minus) icon to collapse the node.

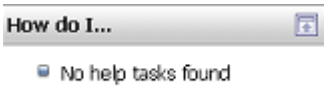
Figure 4-2 Domain Structure

BETA

How do I...

This panel includes links to online help tasks that are relevant to the current console page. No help tasks are included in the Beta release.

Figure 4-3 How do I...



Tool Bar

The tool bar at the top of the console includes the following:

Table 4-1

Tool Bar Element	Description
Welcome message	Indicates user name with which you have logged into the console.
Connected to:	The IP address and port you used to connect to the console.
Home	A link to the top page of the console.
Log Out	Click to log out of the console.
Preferences	A link to a page where you can change some console behavior.
Help	A link to the Administration Console Online Help. Note that the Online Help is not available in the Beta release.
Ask BEA	A link to BEA's eSupport Web site.

BETA

Figure 4-4 Tool Bar

Breadcrumb Navigation

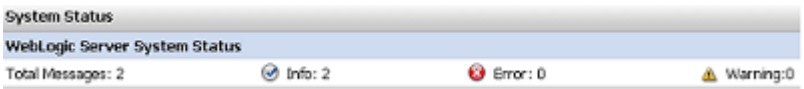
A series of links that show the path you have taken through the Administration Console's pages. You can click on any of the links to return to a previously-visited page.

Figure 4-5 Breadcrumb Navigation

System Status

The System Status panel reports on the number of information, error, and warning messages that have been logged. You can view these messages in the server log files, which you can access from the Administration Console at Diagnostics > Log Files.

Figure 4-6 System Status



Using the Change Center

The starting point for using the Administration Console to make changes in your WebLogic Server domain is the Change Center. To make any changes using the console, you must:

1. Locate the Change Center in the upper left of the Administration Console screen.
2. Click the Lock & Edit button to lock the configuration edit hierarchy for the domain. This enables you to make changes using the Administration Console.
3. Make the changes you desire on the relevant page of the console. Click Save on each page where you make a change.
4. When you have finished making all the desired changes, click Activate Changes in the Change Center.

As you make configuration changes using the Administration Console, you click Save (or in some cases Finish) on the appropriate pages. This does not cause the changes to take effect immediately. The changes take effect when you click Activate Changes in the Change Center. At that point, the configuration changes are distributed to each of the servers in the domain. If the changes are acceptable to each of the servers, then they take effect. If any server cannot accept a change, then all of the changes are rolled back from all of the servers in the domain. The changes are left in a pending state; you can then either edit the pending changes to resolve the problem or revert the pending changes.

Note that WebLogic Server's change management process applies to changes in domain and server configuration data, not to security or application data.

BETA

Undoing Changes

You can revert any pending (saved, but not yet activated) changes by clicking Undo All Changes in the Change Center. You can revert any individual change by going to the appropriate page in the Administration Console and restoring the attribute to its previous value.

Releasing the Configuration Lock

You can release the configuration lock in the following ways:

- Before you make any changes, you can click Release Configuration in the Change Center to release the lock explicitly.
- After you save any changes, you release the lock implicitly when you click Activate Changes or Undo All Changes in the Change Center.

Stopping the Administration Server does not release the configuration lock. When the Administration Server starts again, the configuration lock is in the same state it was in when the Administration Server was shut down, and any pending changes are preserved.

How Change Management Works

To provide a secure, predictable means for distributing configuration changes in a domain, WebLogic Server imposes a change management process that loosely resembles a database transaction. The configuration of a domain is represented on the file system by a set of XML configuration files, centralized in the `config.xml` file, and at runtime by a hierarchy of Configuration MBeans. When you edit the domain configuration, you edit a separate hierarchy of Configuration MBeans that resides on the Administration Server. To start the edit process, you obtain a lock on the edit hierarchy to prevent other people from making changes. When you finish making changes, you save the changes to the edit hierarchy. The changes do not take effect, however, until you activate them, distributing them to all server instances in the domain. When you activate changes, each server determines whether it can accept the change. If all servers are able to accept the change, they update their working configuration hierarchy and the change is completed.

For more information about change management, see Managing Configuration Changes in *Understanding Domain Configuration*.

Dynamic and Non-Dynamic Changes

Some changes you make in the Administration Console take place immediately when you activate them. Other changes require you to restart the server or module affected by the change.

BETA

These latter changes are called *non-dynamic changes*. Non-dynamic changes are indicated in the Administration Console with this warning icon:

Viewing Changes

You can view any changes that you have saved, but not yet activated, by clicking the View Changes and Restarts link in the Change Center. The View Changes and Restarts link presents two tabs, Change List and Restart Checklist:

- The Change List tab presents all changes that have been saved, but not yet activated.
- The Restart Checklist presents all non-dynamic changes, which require restarts before they become effective.

Deploying Multiple Applications

When you use the Administration Console to deploy multiple applications, clicking Activate Changes deploys only the first application installed. The remaining applications are not deployed, but are listed in the console's Deployments page as "distribute Initializing." To deploy the remaining applications, select the application names on the Deployments page and click Start.

BETA

Index

BETA

L

M

N

P

S

T

U

V

W

X

BETA