BEA WebLogic Server™ and WebLogic Express™

Release Notes

Release 8.1 Beta
Document Revised: December 9, 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA WebLogic Server 8.1 Beta Release Notes

Part Number	Document Revised	Software Version
N/A	December 9, 2002	BEA WebLogic Server Version 8.1 Beta

Contents

About This Document

1. WebLogic Server 8.1 Beta Features and Changes

2. Known Issues

About This Document

This document introduces the current release of BEA WebLogic Server™. It contains important information on new features, supported specifications, and known problems.

The document is organized as follows:

- Chapter 1, “WebLogic Server 8.1 Beta Features and Changes,” is an overview of new features and changes in WebLogic Server 8.1 Beta.
- Chapter 2, “Known Issues,” is a list of known problems in WebLogic Server 8.1 Beta.

Audience

This document is written for all users of WebLogic Server 8.1 Beta.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation.

How to Print the Document

You can print a copy of this document from a Web browser, one main topic at a time, by using the File→Print option on your Web browser.

A PDF version of this document is available on the WebLogic Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

Adobe Acrobat Reader is available at no charge from the Adobe Web site at http://www.adobe.com.

Contact Us!

Your feedback on BEA documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the documentation.

In your e-mail message, please indicate the software name and version you are using, as well as the title and document date of your documentation. If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using

- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Usage
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis and book titles.

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

`monospace italic text`	Variables in code. *Example*: `String` *`CustomerName`*`;`
UPPERCASE TEXT	Device names, environment variables, and logical operators. *Example*s: LPT1 BEA_HOME OR
`{ }`	A set of choices in a syntax line.

<table>
<tr><th>Convention</th><th>Usage</th></tr>
<tr><td>[ ]</td><td>Optional items in a syntax line. *Example*:
java utils.MulticastTest -n *name* -a *address*
     [-p *portnumber*] [-t *timeout*] [-s *send*]</td></tr>
<tr><td>|</td><td>Separates mutually exclusive choices in a syntax line. *Example*:
java weblogic.deploy [list|deploy|undeploy|update]
     password {application} {source}</td></tr>
<tr><td>...</td><td>Indicates one of the following in a command line:
■ An argument can be repeated several times in the command line.
■ The statement omits additional optional arguments.
■ You can enter additional parameters, values, or other information</td></tr>
<tr><td>.
.
.</td><td>Indicates the omission of items from a code example or from a syntax line.</td></tr>
</table>

CHAPTER

1 WebLogic Server 8.1 Beta Features and Changes

Welcome to BEA WebLogic Server 8.1 Beta! As the leading Web application server, WebLogic Server™ implements J2EE 1.3 technologies, Web services, and other leading Internet standards to provide a reliable framework for highly available, scalable, and secure applications. WebLogic Server's seamless integration of disparate, heterogeneous platforms and applications enables your network to leverage existing software investments and share the enterprise-class services and data that are crucial to building mission-critical e-business applications.

WebLogic Server 8.1 Beta is the latest version of WebLogic Server. The following sections describe the new features and major improvements made in the WebLogic Server 8.1 Beta release:

- "Packaging and Deployment Features" on page 1-2
- "Administration Features" on page 1-5
- "Sample Applications" on page 1-9
- "Security Features" on page 1-10
- "EJB Features and Changes" on page 1-11
- "J2EE Connector Changes" on page 1-14
- "JDBC Features" on page 1-14
- "JTA Features" on page 1-17

Packaging and Deployment Features

WebLogic Server includes the following new and changed features related to J2EE modules and module deployment.

Custom Classloading for J2EE Modules

You can now create custom classloader hierarchies for an Enterprise Application, allowing for better control over class visibility and reloadability of modules within an `.EAR`. You achieve this by defining a `classloader-structure` element in the `weblogic-application.xml` deployment descriptor file. See Custom Module Classloader Hierarchies in *Developing WebLogic Server Applications*.

New Application Lifecycle Events

You can extend the abstract class `weblogic.j2ee.ApplicationLifeCycleListener` to perform application-specific actions when various application lifecycle events occur. WebLogic Server defines the following lifecycle events:

- Initialization—WebLogic Server parses the application's deployment descriptors before deploying its module(s).
- Preparation—WebLogic Server has identified (and in some cases, started) EJBs, Web Applications, and application-scoped DataSources that are defined in the Enterprise Application.
- Activation—The application is available for processing client requests.
- Update—WebLogic Server detects that one of the application's deployment descriptors has changed.

You can perform actions for each of the above lifecycle events by supplying the associated method. See Application Lifecycle Events in *Developing WebLogic Server Applications*.

Application-Level Class Library

The `APP-INF/lib` directory helps you organize shared class files in an application. WebLogic Server automatically appends classes included in `APP-INF/lib` to the end of the Application's `CLASSPATH`; this ensures that all application modules can access the shared classes. See Developing WebLogic Server Applications.

Dynamic Descriptor Changes

WebLogic Server supports updating deployment descriptor attributes in deployed module containers.

Dynamic Changes to Application-Scoped Pools

WebLogic Server supports updating application-scoped JDBC connection pool properties in the `weblogic-application.xml` deployment descriptor for deployed applications.

Alternate Deployment Descriptors

You can specify an alternate deployment descriptor file to use when deploying an archive file or exploded archive directory. This ability enables you to change the run-time deployment configuration of an application without having to modify and repackage the contents of the archive itself. To use an alternate deployment descriptor, you use one or both of the following options with the `weblogic.Deployer` utility:

- `-altappdd`—specifies the name of an alternate J2EE deployment descriptor, such as `application.xml`.
- `-altwlsappdd`—specifies the name of an alternate WebLogic Server deployment descriptor, such as `weblogic-application.xml`.

See Deploying WebLogic Server Applications.

Improved Deployment Performance and Feedback

The speed of J2EE module deployment has improved over previous server versions. You also get additional feedback on the deployment process for both `weblogic.Deployer` and the Administration Console. Feedback is provided by a new JMX notification and filter (`weblogic.management.DeploymentNotification` and `weblogic.management.DeploymentNotificationFilter`), which you can use in your own applications. See the Javadocs for WebLogic Classes.

Administration Console Deployment Changes

The WebLogic Server Administration Console interface has been streamlined to make production-level deployment descriptors available for editing by the administrator. Although full deployment descriptor editing is no longer available in the Administration Console, many descriptor elements of interest to Administrators are directly editable via Administration Console fields. You can edit these descriptors without repackaging and redeploying the associated module.

Full J2EE module deployment descriptor editing remains available in the WebLogic Builder application for development use.

The Administration Console also provides new Deployment Assistants to help you deploy different types of J2EE modules. The assistants guide you through the process of selecting deployment files and target servers, and automates the selection of deployment staging modes.

See the Administration Console Online Help.

Weblogic.Deployer Changes

The `weblogic.Deployer` utility now includes the `-distribute`, `-start`, and `-stop` commands identified in JSR88. Command help has also been reorganized for easy access to basic and advanced commands. See Deploying WebLogic Server Applications.

Administration Features

The following new and improved server administration features are included in WebLogic Server 8.1 Beta.

JRockit JVM

This version of WebLogic Server includes BEA WebLogic JRockit 8.1 Beta, the first commercial server-side Java Virtual Machine. See the JRockit for Windows and Linux User Guide for information about the benefits and usage of JRockit.

Message Catalogs

WebLogic Server now provides message catalogs in HTML format on e-docs as part of the documentation deliverable. You can search for messages by error number using the search engine. See the Message Catalog Index.

Automatic config.xml Archiving

The Administration Server automatically archives old copies of the domain's `config.xml` file when you make changes to the configuration. By default, the Administration Server saves the five most recent versions of `config.xml` in the `/configArchive` subdirectory of the domain. You use the Administration Console to configure the maximum number of archived files to be stored for the domain. See WebLogic Server Archives Previous Versions of config.xml in *Configuring and Managing WebLogic Server*.

JRockit Monitoring

The Administration Console provides additional runtime data for servers running with the JRockit Virtual Machine (VM). See Monitoring the JRockit Virtual Machine in the *Administration Console Help*.

New Affinity Policies for Java Client Load Balancing

WebLogic Server provides three new load balancing algorithms that minimize the number of IP sockets opened between external Java clients and server instances in a cluster. The new algorithms preserve server affinity by considering a client's existing server connections when accessing objects in a cluster.

The new policies can be applied to EJBs and other RMI objects, as well as JMS client applications. See Load Balancing in a Cluster in *Using WebLogic Server Clusters*.

Improved Node Manager

You have the option of configuring the Node Manager process automatically during server installation (using demonstration SSL certificates). The new Node Manager also includes improved logging facilities and monitoring capabilities that help you better manage multiple servers in a domain. See Configuring, Starting, and Stopping Node Manager in *Configuring and Managing WebLogic Server*.

Improved Network Channel Configuration

The functionality of Network Channels has been enhanced to simplify the configuration process. Network Channels now encompass the features that, in WebLogic Server 7.x, required both Network Channels and Network Access Points. In this version of WebLogic Server, Network Access Points are deprecated.

Network Channels allow you to manage quality of service, meet varying connection requirements, and improve utilization of your systems and network resources. For example, you can use Network Channels to:

- Segregate different types of network traffic
- Support varied application or user requirements on the same Managed Server
- Prioritize network connections that servers use to connect to other servers in a domain

WebLogic Server 8.1 also has new guidelines that apply to configuring Network Channels. See Configuring Network Resources in *Configuring and Managing WebLogic Server*.

Administration Console Usability Improvements

The WebLogic Server Administration Console has been reorganized to provide better usability for both novice and advanced users. The following are some of the many Administration Console changes:

- **Simplified navigation tree** in the left pane of the Administration Console makes it easier to access configuration tabs for server resources and deployed modules.
- **Reorganized server configuration tabs** provide easier access to frequently-used controls.
- **Reorganized SSL and Key Store configuration tabs** facilitate enabling SSL security in a domain.
- **Advanced options toggle** enables you to access less-common, advanced fields by clicking the Advanced button on the relevant page. By default, the Administration Console now displays only the most commonly-used controls on each configuration tab.

- **Improved feedback and exception handling** in the Administration Console provides better feedback on deployment and other administration tasks. It also provides better access to server log files. Within each server log file, you can click on message ID numbers to view more information about an error or exception in the message catalog.

- **New inline help** for Administration Console controls provide help text directly on the configuration tab. You can also access more detailed help for a particular page by using the context-sensitive help icon.

- **Deployment Assistants** in the Administration Console help you deploy different J2EE modules.

- **JDBC Assistants** in the Administration Console help you configure JDBC Connection Pools and DataSources.

For more information about the Administration Console changes, see the Administration Console Online Help.

weblogic.Admin Features

The `weblogic.Admin` utility provides the following new commands:

- `BATCHUPDATE` runs multiple `weblogic.Admin` commands in an uninterrupted sequence. (You no longer have to invoke a separate JVM for each `weblogic.Admin` command.)

- `CLUSTERSTATE` returns the number and state of servers in a cluster.

- `DISCOVERMANAGEDSERVER` causes the Administration Server to re-establish administrative control over Managed Servers.

- `QUERY` searches for WebLogic Server MBeans whose `WebLogicObjectName` matches a pattern that you specify.

- `STARTCLUSTER` and `STOPCLUSTER` start and stop all server instances in a cluster.

- `TEST_POOL` tests a connection pool by reserving and releasing a connection from it.

- `VALIDATECLUSTERCONFIG` verifies the formatting of cluster-related attributes in the domain's `config.xml` file.

In addition, you can use the new `-adminurl` argument to access runtime MBeans for all server instances through the Administration Server.

All `weblogic.Admin` commands now return an exit code of 0 if the command succeeds and an exit code of 1 if the command fails.

For more information, refer to weblogic.Admin Command-Line Reference in the *WebLogic Server Command Reference*.

New Default OOTB Performance Parameter Settings

The following performance-related attributes have been adjusted for improving the out-of-the-box performance of WebLogic Server. Optimal WebLogic Server production tuning values vary according to your environment and applications.

- The default number of execute queues has been increased to 25.
- The default JDBC connection pool Maximum Capacity database connection parameter now equals the number of default execute queue Thread Count parameter (15).
- The default JDBC connection pool Prepared Statement Cache Size parameter has been reduced to 5.
- To further improve performance on Solaris platforms, specify the following command-line parameter when starting WebLogic Server instances: `-Dweblogic.PosixSocketReaders=1`

Sample Applications

A new J2EE sample application, Avitek Medical Records (or MedRec), concisely demonstrates all aspects of the J2EE platform, and is designed as an educational tool for all levels of J2EE developers. MedRec showcases the use of each J2EE component, and illustrates best practice design patterns for component interaction and client development. Medical Records is available from the Start menu on Windows machines. On Linux and other platforms it can be started from the `WL_HOME\samples\server\config\medrec` directory.

WebLogic Server sample applications have also been updated to use the PointBase® version 4.3 server as the sample datastore.

Security Features

The following new and improved security features are included in WebLogic Server 8.1 Beta.

Improved Functionality for Creating Roles and Policies

New windows and improved options facilitate managing access to WebLogic resources such as the Administration Console, the `weblogic.Admin` tool, MBeans, applications, COM, EIS, EJB, JDBC, JNDI, JMS, servers, and Web applications. See Security in the *Administration Console Online Help*.

Improved Support for Keystores and SSL Configuration

The SSL implementation of WebLogic Server supports the use of keystores for storing private keys and trusted CAs. Keystores add a level of protection to the flat files used in past release of WebLogic Server.

The default configuration of SSL and demonstration keystores provide users with secure communication out of the box. The configuration of keystores and SSL for a production environment has been simplified by the implementation of a wizard. See Configuring Keystores and SSL in the *Administration Console Online Help*.

Support for the Sun Java Cryptography Extension (JCE) Package

The Java Cryptography Extension (JCE) is a set of packages that provide a framework for encryption using strong ciphers, key generation and agreement, and Message Authentication Code algorithms. See WebLogic Server Security Service in *Introduction to WebLogic Server and WebLogic Express*.

EJB Features and Changes

This release introduces the following EJB features and changes.

Performance Monitoring Improvements

Performance monitoring is improved through the implementation of new tab pages in the WebLogic Server Administration Console.

appc

`appc` is a single tool for compiling and validating a J2EE `.EAR` file, an EJB `.JAR` file, or `.WAR` file for deployment. `appc` replaces and improves upon `ejbc`, which is deprecated as of this release.

Previously, a user wanting to compile all modules within an `.EAR` file had to extract the individual components of an ear and manually execute the appropriate compiler (`jspc` or `ejbc`) to prepare the module for deployment. `appc` automates this process and performs additional pre-deployment validation checks not previously performed.

See appc in *Programming WebLogic Enterprise JavaBeans*.

Batch Operations

WebLogic Server now supports batch updates and deletes, in addition to the existing batch insert (previously known as “bulk insert”) support. In addition, the EJB container now prevents exceptions by performing dependency checks between batch operations. See Batch Operations in *Programming WebLogic Enterprise JavaBeans*.

Automatic Database Detection

As application developers develop their entity beans, the underlying table schema must change. With the automatic database detection feature enabled, WebLogic Server automatically changes the underlying table schema as entity beans change. See Automatic Database Detection in *Programming WebLogic Enterprise JavaBeans*.

EJB QL Compiler Enhancements

Compiler error messages in EJB QL provide a visual aid to identify which part of the query is in error and allow the reporting of more than one error per compilation. See EJB QL Error-Reporting Enhancements in *Programming WebLogic Enterprise JavaBeans*.

ejbc Deprecated

The `ejbc` compiler has been deprecated. Use `appc` in its place. See “appc” on page 1-11.

sql-select-distinct Deprecated

This version of WebLogic Server deprecates the `sql-select-distinct` element in `weblogic-cmp-rdbms-jar.xml`. Use the `DISTINCT` clause directly in finder queries instead of this XML element. For finder queries that have a `DISTINCT` clause, the container defers duplicate elimination to the database if `FOR UPDATE` is not used and filter duplicates is used.

If `sql-select-distinct` is set to true, but the finder query does not have a `DISTINCT` clause, the value of `sql-select-distinct` is ignored and it is equivalent to specifying a `DISTINCT` clause in the finder query.

If `sql-select-distinct` is set to false, but the finder query has a `DISTINCT` clause, the value of `sql-select-distinct` is ignored.

For more information on `sql-select-distinct`, see sql-select-distinct in *Programming WebLogic Enterprise JavaBeans*.

For more information on the `SELECT DISTINCT` clause in EJB QL, see Using SELECT DISTINCT.

New dbms-column-type Values

WebLogic Server supports two additional values for the `dbms-column-type` element in `weblogic-cmp-rdbms.xml`: `LongString` and `SybaseBinary`. See dbms-column-type in *Programming WebLogic Enterprise JavaBeans*.

Performance Improvements

WebLogic Server provides improved performance for EJB bulk updates, optimistic concurrency, field groups, relationship caching, and EJB redeployment.

Reloadable EJB Modules

With the Custom Classloading for J2EE Modules feature, you can redeploy EJBs independently of other components in an Enterprise Application.

EJB Deployment Assistants

The Administration Console provides an EJB Module Deployment Assistant to help you deploy EJBs. See the Administration Console Online Help.

J2EE Connector Changes

The DTD for the WebLogic Server Connector deployment descriptor, `weblogic-ra.xml`, has changed in this version. See weblogic-ra.xml Deployment Descriptor Elements in *Programming WebLogic Server J2EE Connectors*.

The Connector implementation now makes connections shareable unless a suitable “hint” is specified at deployment time. EJB 2.0 uses the `res-sharing-scope` deployment descriptor (with values `Shareable` or `Unshareable`) to specify this hint.

JDBC Features

WebLogic Server provides the following new JDBC features, along with internal performance enhancements in the JDBC subsystem.

JDBC Assistants

The Administration Console includes the JDBC Connection Pool Assistant and the JDBC Data Source Assistant. These assistants help ease database connectivity configuration by prompting you for database, JDBC driver, and connection pool information, and then constructing the connection attributes required by your JDBC driver. See the Administration Console Online Help.

JDBC Connection Pool Attributes

JDBC connection pools include several new attributes and features that you can configure from the Administration Console or MBean attributes using the JMX API, including among others:

- Connection Reserve Timeout—Enables connection requests to wait for a connection from a connection pool when all connections are currently in use.
- Connection Creation Retry Frequency—Enables WebLogic Server to retry to create a database connection after the original attempt to create the connection failed.
- Test Created Connections—Enables testing and initialization of physical database connections when each connection is created.
- Test Pool—Tests a JDBC connection pool by reserving and releasing a connection from it.

See the Administration Console Online Help.

Enhanced Support for JDBC Extensions

Some database vendors provide additional proprietary methods for working with data from their DBMS. These methods extend the standard JDBC interfaces.WebLogic Server provides enhanced support for vendor extensions to JDBC by supporting most extension methods exposed in a public interface in the vendor's JDBC driver. See Using Vendor Extensions to JDBC Interfaces in *Programming WebLogic JDBC*.

Physical Connection from a Connection Pool

When you get a connection from a connection pool, WebLogic Server provides a logical connection rather than a physical connection so that WebLogic Server can manage and maintain the connection. In some cases, you may want to use a physical connection, such as if you need to pass the connection to a method that checks the class name of the object for a particular class. WebLogic Server includes the `getVendorConnection()` method in the `weblogic.jdbc.extensions.WLConnection` interface that you can use to get the underlying physical connection from a logical connection. See Getting a Physical Connection from a Connection Pool in *Programming WebLogic JDBC*.

Support for RowSets

WebLogic Server includes support for RowSets, which are a JDBC 2.0 extension to ResultSets. RowSets allow a user to read and modify a cached query result and then commit the resulting changes back to a database. RowSets use a disconnected model which uses optimistic concurrency control to ensure database consistency. This allows work to be performed without holding open long transactions or database and application server resources.

Statement Caching

The statement cache for JDBC connection pools was enhanced to include a Least Recently Used caching algorithm and controls for clearing the statement cache.

When you use a prepared statement or callable statement in an application or EJB, there is considerable processing overhead. To minimize the processing costs, WebLogic Server can cache statements used in your applications in the statement cache. When an application or EJB calls any of the statements stored in the cache, WebLogic Server reuses the statement stored in the cache, which reduces CPU usage on the database server, and thus improves performance for the current statement and leaves CPU cycles on the database server for other tasks. See the Administration Console Online Help.

JTA Features

WebLogic Server 8.1 Beta provides the following new JTA features.

Manual Completion of Current Transactions

In some cases, a transaction may not complete normally due to system or network failures. In such situations there may be locks held on behalf of the pending transaction that are inhibiting the progress of other transactions. You can use the Administration Console or methods on the JTA runtime MBean to manually complete transactions that did not complete normally. See the Administration Console Online Help.

Non–XA-Compliant Resource Participation in a Global Transaction

A single, non–XA-compliant resource adapter can participate in a global transaction with other XA-compliant resources. WebLogic Server uses a last agent commit optimization so that after all participating XA-compliant resources are prepared, the result of the local transaction for the non-XA resource is used to determine the outcome of the global transaction. The resource adapter must provide local transaction semantics. You can use this functionality with the WebLogic Server J2EE Connector architecture to enable non-XA legacy systems to participate in a global transaction.

JMS Features

WebLogic Server 8.1 Beta provides the following new JMS features.

JMS Thin Client

At approximately 800k, the `wljmsclient.jar` file provides full WebLogic JMS functionality, yet greatly reduces the client-side WebLogic footprint by using a smaller library that contains only the set of supporting files required by client-side programs. The new client `.jar` file is available in the `/server/lib` subdirectory of the WebLogic Server installation directory (for example, `c:\bea\weblogic81b\server\lib`). This `.jar` provides for full-featured WebLogic Server clients that can support clustering, load balancing, transactions, security, and failover. See WebLogic JMS Thin Client in *Programming WebLogic JMS*.

Accessing Foreign JMS Providers

Using the Foreign JMS Server node on the Administration Console, you can quickly map a foreign JMS provider so that its connection factories and destinations appear in the WebLogic JNDI tree as a local JMS objects. A Foreign JMS Server configuration can also be used to reference remote instances of WebLogic Server in another cluster or domain in the local WebLogic JNDI tree. See Accessing Foreign JMS Providers in *Programming WebLogic JMS*.

Accessing JMS via Servlets and EJBs

New “wrappers” make it easier to use JMS inside a J2EE component, such as an EJB or servlet. The wrappers provide features including automatic pooling of JMS Connection and Session objects (and some pooling of `MessageProducer` objects as well); automatic transaction enlistment for JMS providers that support XA; monitoring of the JMS connection and re-establishment after a failure; and security credentials that are managed by the container. See Using WebLogic JMS with EJBs and Servlets in *Programming WebLogic JMS*.

Better Expired Message Handling

Active message expiration ensures that expired messages are cleaned up immediately. Moreover, expired message auditing gives you the option of tracking expired messages, either by logging when a message expires or by redirecting expired messages to a special destination. See Handling Expired Messages in the *Administration Console Help*.

Improved Message Flow Control by Blocking Producers

The “Blocking Send” features help you avoid receiving message quota errors by temporarily blocking message producers from sending messages to a destination (queue or topic) when the destination has exceeded its specified maximum message quota. See Avoiding Quota Excptions by Blocking Message Producers in the *Administration Console Help*.

Web Application Features

WebLogic Server 8.1 Beta includes the following new Web Application features and changes.

Performance Improvements

Performance is improved for JSP string handling as well as JSP compilation time.

Security Principle for init Method

Use the new `init-as-principal-name` element in `weblogic.xml` to declare a principle name for running a servlet’s `init` method. See Developing Web Applications for WebLogic Server.

Allow Access to WEB-INF on forward/include

You can now call one servlet from inside another servlet. This is accomplished using either a forward or an include request from within the original servlet. Should you forward to a second servlet, all future action takes place according to the second servlet, as with any forward. Including a second servlet allows you to gather data from a source already accessed by another servlet without having to rewrite all the code. See Dispatching Requests to Another Resource.

Integrated jspc Functionality into appc

The `appc` compiler now incorporates the functionality of `jspc`. You can use `appc` to compile and generate both EJBs and JSPs for deployment. For more information about `appc` usage and syntax see appc and jspc Compilers in *Developing Web Applications for WebLogic Server*.

Class Reloading for Servlet Filters

When responding to a request for a servlet, WebLogic Server checks the time stamp of the servlet class file prior to applying any filters associated with the servlet, and compares it to the servlet instance in memory. If a newer version of the servlet class is found, WebLogic Server re-loads the servlet class before any filtering takes place. You can configure the interval at which WebLogic Server checks the timestamp using the `Server Reload` attribute. See Servlet Development Tips in *Programing WebLogic HTTP Servlets*.

FileServlet File Sorting Options

WebLogic Server introduces new `weblogic.xml` deployment descriptors to provide sorting options for directory listings. The new element, `index-directory-sort-by` has valid sorting styles of `NAME`, `LAST_MODIFIED`, and `SIZE`. For example, to enable directory listing sorted by file size, the XML would resemble:

```
<weblogic-web-app>
       <index-directory-enabled>true</index-directory-enabled>
       <index-directory-sort-by>SIZE</index-directory-sort-by>
</weblogic-web-app>
```

Web Services Features

The following new Web Services features are available with WebLogic Server 8.1 Beta.

Digital Signatures and Encryption

You can configure data security for Web Services and Web Service clients using new elements in the `web-services.xml` deployment descriptor. See Configuring Security in *Programming WebLogic Web Services*.

Reliable Messaging

Reliable messaging is a framework whereby an application running in one WebLogic Server instance can asynchronously and reliably invoke a Web service running on another WebLogic Server instance. See Using Reliable Messaging in *Programming WebLogic Web Services*.

SOAP 1.2

WebLogic Server supports SOAP 1.2 as the message transport when a client invokes a Web Service operation. See Using SOAP 1.2 in *Programming WebLogic Web Services*.

JMS Transport Protocol

You can optionally configure a Web Service to use JMS as the transport protocol (in addition to HTTP/S, the default protocol) when a client accesses the service. See Using JMS Transport to Invoke a WebLogic Web Service in *Programming WebLogic Web Services*.

Asynchronous Invocation of WebLogic Web Services

The `clientgen` Ant task can now generate stubs for invoking a Web service operation asynchronously. The stub contains two methods: the first invokes the operation with the required parameters but does not wait for the result; later, the second method returns the actual results. You use this asynchronous client when using reliable messaging. See Writing an Asynchronous Client in *Programming WebLogic Web Services*.

Portable Stubs

You can now use portable stubs (versioned client JAR files used to invoke WebLogic Web services) to avoid class clashes when invoking a Web service from within WebLogic Server. See Creating and Using Portable Stubs in *Programming WebLogic Web Services*.

WebLogic Tuxedo Connector Features

WebLogic Server 8.1 Beta provides the following new WebLogic Tuxedo Connector features.

Enhanced Security Administration

WebLogic Tuxedo Connector provides to users the ability to select one of the following APPKEY generators to access Tuxedo services:

- TPUSER plug-in—Enables users to use a `tpuser` file to provide user information to the Tuxedo authentication server.
- LDAP—Enables users create a single source of security administration by allowing WebLogic Server embedded LDAP information to be used by a Tuxedo 8.1 authentication server.
- Custom—Enables users to create a custom APPKEY to generate user information to access Tuxedo services.

Asynchronous tpacall

The asynchronous `tpacall()` method allows you to send a request to a Tuxedo service and release the thread resource that performed the call to the thread pool. This allows a very large number of outstanding requests to be serviced with a much smaller number of threads. See Request/Response Communication in the *WebLogic Tuxedo Connector Programmer's Guide*.

XML Features

You can now use the WebLogic XPath API to perform XPath matching against an XML document represented as a DOM, XMLNode, or XMLInputStream. See Using the WebLogic XPath API in *Programming WebLogic XML*.

Developer Tools

The following new developer tool features are available in WebLogic Server 8.1 Beta.

New appc Compiler

The appc compiler compiles and generates EJBs and JSPs for deployment. It also validates the descriptors for compliance with the current specifications at both the individual module level and the application level. The application level checks include checks between the application-level deployment descriptors and the individual modules as well as validation checks across the modules.

New J2EE Client .JARs

Prior to version 8.1 Beta, client applications that incorporated WebLogic Server functionality required the entire WebLogic Server distribution (`weblogic.jar` and `weblogicaux.jar`) on the client machine. WebLogic Server now provides two new client `.jar` files that include only the functionality needed for small-footprint J2EE client functionality. The new files are:

- `wlclient.jar` for basic WebLogic functionality such as clustering, security, and transactions.
- `wljmsclient.jar` for basic WebLogic functionality plus JMS features.

The new client `.jar` files are located in the `/server/lib` subdirectory of the WebLogic Server installation directory (for example, `c:\bea\weblogic81b\server\lib`). The new client `.jar` files are not supported with JDK 1.3.x or earlier.

See Programming WebLogic Server Applications for more information.

Builder Changes

The following features and changes apply to the WebLogic Builder tool.

Deployment Descriptor Editing

Use the Builder tool to edit J2EE module deployment descriptors. Deployment Descriptor editing features are no longer available via the Administration Console. See the WebLogic Builder Online Help.

Optimistic Concurrency

It is now possible to configure your CMP entity beans to use optimistic concurrency for parallel transactions using Builder. See Working with EJBs in the *WebLogic Builder Online Help*.

Internationalization Features

WebLogic Server introduces the following changes to internationalization utilities:

- `weblogic.i18ngen` has updated command line options.
- `weblogic.i10ngen` has updated command line options.
- `weblogic.gettxt` is a new command line utility.
- `weblogic.i18ntools.GetText` is a new API.
- `weblogic.MsgEditor` has an updated GUI. The main Message Editor window also provides the ability to retire and unretire messages. Retiring a message does not mean that the message is deleted from the master catalog. It simply means it is hidden from user view. This feature is useful for removing obsolete messages. If you need to bring a retired message back into view, you can unretire it.

Other Available Resources

Here are some pointers to useful information related to this release. The hyperlinks require Internet access.

Fast Track Procedures

High-level procedures to help you quickly deploy an HTML file, JSPs, and servlets, are available at `http://e-docs.bea.com/wls/docs70/quickstart/quick_start.html`.

Examples

Code examples, if installed, are located in the *`SAMPLES_HOME`*`\server\src\examples` directory of your WebLogic Server installation, where *`SAMPLES_HOME`* is the location of all examples for the WebLogic Platform. By default, this location is `c:\bea\weblogic81b\samples`. Examples are also available from the Start menu for Windows users.

Introduction

For an overview of WebLogic Server features and the J2EE application architecture, see *Introduction to WebLogic Server*.

Additional Documentation

The full documentation set for BEA WebLogic Server 8.1 Beta, including administration, programming, and reference guides, is provided on the BEA Web site at `http://e-docs.bea.com/wls/docs81b/index`. This link is password protected and available only to Beta customers.

Newsgroups

BEA WebLogic Server newsgroups provide community support for BEA products. Information about BEA-related newsgroups can be found at `http://newsgroups.bea.com/` and `news://newsgroups.bea.com`.

Dev2Dev Online

The BEA site Dev2Dev Online provides resources to make your e-commerce development easier and faster. To reach Dev2Dev online, go to `http://developer.bea.com/`.

CHAPTER

2 Known Issues

The following sections describe known issues WebLogic Server 8.1 Beta:

- "WebLogic Server 8.1 Beta Interoperability" on page 2-1
- "General WebLogic Server Issues" on page 2-3
- "Web Services Issues" on page 2-8

WebLogic Server 8.1 Beta Interoperability

WebLogic Server 8.1 Beta does not support mixed-version domains. All Managed Servers in a domain must be at the same version (8.1 Beta).

Earlier WebLogic Server versions cannot act as clients to WebLogic Server 8.1 Beta over the IIOP protocol. For example, this version does not support a WebLogic Server 7.0 SP1 EJB using IIOP to call an EJB deployed on WebLogic Server 8.1 Beta.

In order to interoperate with WebLogic Server version 6.1, you must add the following elements to the `<Domain>` section of the WebLogic Server 8.1 Beta `confix.xml` file:

```
<!-- START OF 70 INTEROP SPECIFIC SETTING -->
<Security CompatibilityMode="true" GuestDisabled="false"
  InteropEnabled="true"
  InteropPassword="{3DES}OuQN48TIXRYFVVez4VBFTA=="
  InteropUsername="system" Name="@DOMAIN1"
  PasswordPolicy="wl_default_password_policy"
Realm="wl_default_realm"
  RealmSetup="true" SystemUser="system"/>
<!-- START PER CR071185, ADDING BELOW SETTING TO 70 ONLY -->
<SecurityConfiguration Credential="gumby1234" Name="@DOMAIN1" />
```

```
<!-- END PER CR071185, ADDING BELOW SETTING TO 70 ONLY -->
<!-- END OF 70 INTEROP SPECIFIC SETTING -->
```

(The above elements are also required for WebLogic Server version 7.0 to interoperate with WebLogic Server version 6.1.)

JVM Interoperability

Interoperability with servers using the BEA WebLogic JRockit 8.1 Beta JVM has not been tested and is not supported in WebLogic Server 8.1 Beta. This release supports only the JVM installed with Sun’s JDK version 1.4.1.

WebLogic Server versions 6.1 and 7.0 are certified to use Sun’s JDK version 1.3.1_x. However, version 6.1 and 7.0 servers acting as clients to WebLogic Server version 8.1 Beta must use Sun’s JDK version 1.4.1 as the client-side JVM.

General WebLogic Server Issues

Change Request Number	Description
058211	WebLogic Server version 7.0 introduced connection proxies to provide advanced features such as late transaction enlistment and idle connection detection for resource adapters. However, some resource adaptors such as IBM's IMS Connector for Java 1.2.5 could not be used because the server could not create a suitable connection proxy. This is caused by the resource adapter casting the returned proxy as a connection implementation class instead of one if its implemented interfaces. WebLogic Server 8.1 Beta, enhances the implementation by testing whether connection proxies can be used. The test is done when a connection is first allocated to the pool. If the resource adaptor throws a ClassCastException, proxy generation is turned off and the connection is returned, unchanged, to the resource adaptor. In this case the resource adaptor is available for use, but late transaction enlistment and idle connection detection features are not available. Note that even when the connection is returned unchanged to the resource adaptor, it is possible for client applications to receive a ClassCastException if they attempt to cast a proxy. Note that this testing only is possible for resource adapters which return a connection to a client via the `getConnection()` method taking no security credentials.

Change Request Number	Description

079630	Most WebLogic Tuxedo Connector tBridge users will expect to set the Correlation ID using the `setJMSCorrelationID(String)` method. This will take a 32 character string and turn it into a 64 byte array. JMS stores the string as UTF-16BE. When the tBridge receives the Correlation Id from Tuxedo it is 32 characters as 32 bytes. The tBridge then uses the `setJMSCorrelationIDAsBytes(byte[])` method to set the message for JMS receiving queue. The characters may be the same in ASCII but the two will not compare due the length difference.
	When it is necessary to compare the Correlation ID with the ID returned from Tuxedo in the receiving JMS queue, create a byte array containing the hexadecimal values of the Correlation ID. Then use the `setJMSCorrelationIDAsBytes()` and `getJMSCorrelationIDAsBytes()methods` to create an ID to compare with the ID returned from Tuxedo in the receiving JMS queue.
	For example, if the string Correlation ID is `“1234567890ABCDEFGHIJKLMNOPQRSTUV”`:
	private byte[] coridbyte={0x31,0x32,0x33,0x34,0x35,0x36,0x37,
	0x38,0x39,0x30,0x41,0x42,0x43,0x44,
	0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,
	0x4c,0x4d,0x4e,0x4f,0x50,0x51,0x52,
	0x53,0x54,0x55,0x56};
	`msg.setJMSCorrelationIDAsBytes(coridbyte);`
	`corIDasBytes = msg.getJMSCorrelationIDAsBytes();`
	`corIDasBytes` contains the correct value to compare with the Correlation ID returned from Tuxedo.

084387	If a member of the cluster that plays host to a member of a distributed destination crashes, the behavior of the distributed destination upon reboot is incorrect. Failed reconnection attempts cause all ports on the machine to be tied up. The workaround is to restart all servers that host a member of the distributed destination.

Change Request Number	Description

085718	The following exception may occur when starting the server:
	`java.lang.ArrayIndexOutOfBoundsException`
	`at com.octetstring.vde.Entry.bytesToInt(Entry.java:227)`
	`at com.octetstring.vde.Entry.readBytes(Entry.java:440)`
	`at com.octetstring.vde.Entry.<init>(Entry.java:94)`
	`at com.octetstring.vde.backend.standard.`
	`TransactionProcessor.run(TransactionProcessor.java:22)`
	This exception may occur the first or second time a 7.x WebLogic Server domain is used with this release of WebLogic Server. To resolve the problem, delete the `EmbeddedLDAP.tran` file in the *`domain_name/servername/`*`ldap/ldapfiles` directory.

089243	The performance of the Beta thin client `.jar` files (`wlclient.jar` and `wljmsclient.jar`) degrades excessively over time. This problem will be addressed with a future version of Sun's JVM.
090774	If a JDBC connection pool is in use and its associated database server goes down at runtime, connections in the pool are not restored, even if `TestOnReserve` or `TestOnRelease` are enabled. The workaround is to set the `ConnectionCreationRetryFrequencySeconds` attribute of the pool to a positive value using the Administration Console; the recommended value is 30 seconds. Note that setting this attribute is critical if you are using a JDBC connection pool for a JMS store.
090888	WebLogic Server console does not target Jolt, WLEC or WTC service objects. To target an object, users will need to assign the object to the server by editing the config.xml file.
091181	Builder does not support setting values for the `create-default-dbms-tables` element. Valid values for manually editing this element are `CreateOnly`, `Disabled`, `DropAndCreate`, and `AlterOrCreate`.
091266	The `weblogic.jspc` compiler generates extra (unused) local variables in the generated code for jsps with tag libraries. This can lead to ClassCastExceptions when accessing JSPs compiled with the `jspc` and running on the JRockit JVM.

Change Request Number	Description
091353	The following exception may occur when starting the server: `Exception:weblogic.server.ServiceFailureException:Could not get exclusive access to the embedded LDAP data files due to existing server.` This problem occurs occasionally when stopping and immediately restarting a server. Two WebLogic Server instances cannot run at the same time in the same directory. If this exception occurs, most likely the first server instance was not completely shutdown before the second server instance was started. Check to see that the first server instance is completely shutdown and then restart the second server instance.
092029	The Configuration Wizard installed with WebLogic Server 8.1 Beta includes a template for creating domains that include the Sun Pet Store application. However, the Pet Store application is no longer installed with the product. (It will instead be available via the dev2dev site). Use the WebLogic Server Medical Records application in place of Pet Store for sample J2EE application code.
092037	In the Solaris installation of WebLogic Server, the `license.bea` file contains extraneous `^M` characters. However, the file provides a valid license for WebLogic Server 8.1 Beta.
092041	Load Balancing and certain failover scenarios are not functional with the Beta thin-client `.jar` files (`wlclient.jar` and `wljmsclient.jar`).
092042	The Interactive Query example installed with WebLogic Server 8.1 Beta does not include a sample `build.xml` script. However, you can use the prebuilt sample that is installed with the Weblogic Server Examples domain.
092056	If an Administration Server fails or shuts down in a domain, you must restart the server using the same listen address and listen port numbers from the last startup. If you reboot the Administration Server with a different address or port number, Managed Servers in the domain will be unable to communicate with the new Administration Server instance. This problem limits certain failover scenarios for the Administration Server in this Beta release. For example, you cannot restart the Administration Server on a different machine after a hardware failure and resume administration of the domain.
092157	The Web Services runtime fails some TCK tests.

Change Request Number	Description
092225	You cannot use the Administration Console to target a JMS Server to a migratable target in this Beta release. If you attempt to target a JMS Server to a migratable target, the operation fails without displaying an error message. The only indication that the operation has failed is that the migratable target selection box reverts to its default value (none). If you try to target a JMS Server to both a server instance and a migratable target, the operation appears to succeed. However, when you navigate to another area of the console, the Administration Console prompts you with the question, "Save your changes before leaving this page?" If this occurs, click the Cancel button, return to the Targets tab and click the Apply button to apply your changes and target only to the server.
092251	If you start WebLogic Server in an empty directory, or if the ListenAddress element is left blank in `config.xml`, server startup messages indicate that the server is listening on IP address 0.0.0.0. However, the server actually listens on all IP addresses bound to the primary network interface. The 0.0.0.0 value simply indicates that the ListenAddress value is not set.
092452	In this Beta release, Builder does not persist values specified in the JDBC Data Source Setting panel.

Web Services Issues

Change Request Number	Description
063817	A complex type which contains an element of type xsd:schema causes an exception during autotyping.
074740	WebLogic Server does not convert Java data structures which contain Lists as items of Lists to an XML Schema.
078871	wsdl:fault element is not generated when an exception is thrown from a Java class.
080363	A service configured for HTTPS transport cannot be accessed from the web service test page in this Beta release.
083527	The typeMappingBase attribute of autotyper is ignored when generating holder classes.
083870	Holder classes for arrays of schema types that map to SOAPElement do not compile.
086326	The `-Dweblogic.webservice.verbose=true` does not work for SSL clients. The workaround to get verbose output is to use `-Dweblogic.webservice.binding.verbose=true`.
088053	Calendar to xsd:dateTime conversion is a lossy process, so trying to roundtrip Calendar to xsd:dateTime to Calendar does not work in all cases, especially with regard to daylight savings.
088519	WebLogic Server's Xpath implementation may fail to evaluate Xpaths containing more than one descendent axis when the input is an XML stream.
089627	On the client side, WebLogic Server encodes the nonce in base64, instead of using raw bytes, before hashing it. This is not compliant with the standard.
089799	Only void return services are supported over JMS transport. You cannot use JMS transport with two-way invokes.
090475	You cannot use remote JMS destinations as web service backend components in this release.
091077	You cannot pass headers implicitly using the webservice context. The values of headers do not show up in `WebServiceContext`.

Change Request Number	Description
091097	The wsse client jar file is not included in the beta distribution. This prevents access to the web service security extensions from a standalone client. The workaround is to run a client in a server environment or to contact the beta team for an updated jar.
091366	This issue applies to Web Services that use a pinned JMS destination in a WebLogic Server cluster. During startup, WebLogic Server may attempt to deploy a Web Service before the JNDI bindings for its pinned JMS destination are reflected in the cluster-wide JNDI tree. When this occurs, the deployment succeeds on the server that hosts the JMS destination, but fails on other servers that do not yet have the JNDI binding. If you experience this problem, try to redeploy the Web Service to the cluster. The deployment will succeed after the JMS destination's JNDI binding is reflected in all servers in the cluster.
091679	The reliable delivery feature does not work for a web service that is configured for both security extensions and reliability.
091681	JAX-RPC holder classes will not compile for types without targetNamespace.
092139	The passwords for retrieving keys from the server keystore are currently stored in plain text in the web serviced.
092157	The Web Services runtime fails some TCK tests.
092178	WebLogic Server does not correctly deserialize xsd:QName instances that have no prefix.
092184	A service configured to use the security extensions cannot be accessed from the web service test page.
091986	The `BeanCodecSimpleContentBase` class is missing from `wsclient70.jar`. To use portable 8.1 stubs running in a WLS 7.0 environment, first extract `weblogic/xml/schema/binding/ BeanCodecSimpleContentBase.class` from the WebLogic Server 8.1 Beta `webservices.jar` and add it to the `wsclient70.jar` file.
092065	If you are using a version of the JRE prior to 1.4.1, you must manually add the DOM classes to the classpath of a standalone web services client application. The classes are available in `weblogic.jar` in the `org/w3c/dom directory`.