BEA WebLogic Server® and WebLogic Express™

Release Notes

Version 9.0 BETA
Revised: December 15, 2004
Part Number: N/A for BETA

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

1. What's New in WebLogic Server 9.0

2. WebLogic Server 9.0 Known Issues

CHAPTER 1

What's New in WebLogic Server 9.0

Welcome to BEA WebLogic Server 9.0 Beta! WebLogic Server 9.0 is the most significant BEA application server release to date. Fully compliant with J2EE 1.4, this release tackles head-on the biggest challenge facing enterprise networks today: reducing overall cost of management and operations while delivering high reliability, continuous uptime, scalability, and mission-critical integration solutions.

Caution: The Beta release of WebLogic Server 9.0 should not be used in production environments. WebLogic Server 9.0 Beta is not guaranteed to be compatible with the General Availability release of WebLogic Server 9.0 or with any other WebLogic Server® release. Configuration file formats, configuration data, and persisted data may change between this Beta release and the General Availability release of WebLogic Server 9.0.

The following sections describe new and changed functionality in WebLogic Server 9.0.

What's New: The Big Picture

WebLogic Server 9.0 is fully compliant with the J2EE 1.4 Specification. This release implements and extends the latest J2EE standards -- Enterprise Web Services 1.1, JMS 1.1, JMX 1.2, JDBC 3.0, Connector Architecture 1.5, EJB 2.1, and more -- to deliver unparalleled quality-of-service across the enterprise. The following sections summarize innovations in key areas of functionality.

Systems Management

WebLogic Server 9.0 focuses on simplifying and streamlining the day-to-day management of production systems with minimal disruption to live production environments. A comprehensive, centralized diagnostics service lets administrators identify and resolve issues in real time, and accommodates the integration of third-party analytic tools. This release of WebLogic Server introduces a more extensible, "portalized" Administration Console as well as a tightly controlled, predictable process for managing changes to a domain's configuration regardless of the configuration tool you use. A new scripting tool, a simplified domain directory structure, and modular deployment capabilities automate and facilitate application configuration and deployment.

See "System Administration and Management" on page 1-4.

Performance and Availability

Out-of-the-box support for WAN and MAN failover address catastrophic data center failures. Overall server performance improves greatly in WebLogic Server 9.0, with automated server migration and provisioning of services, policy-driven processing, self-tuning capabilities, increased overload protection, cross-cluster failover, and more.

See "Server Reliability, Availability, Scalability, and Performance" on page 1-14.

Enterprise Web Services

Now a J2EE standard, Web Services increase developer flexibility and choice by providing a common run-time environment and industry-standard support for Java annotations and Web Services extensions. The WebLogic Server implementation of Web Services 1.1 lets developers respond more effectively to changing business requirements, with true asynchronous messaging support for conversational applications; interoperability features designed for enterprise service-oriented architecture (SOA); and improved XML processing. Developers can leverage WebLogic Web Services to rapidly create, deploy, and adapt secure and fully interoperable Enterprise Web Services.

See "J2EE Standard Web Services" on page 1-17.

Enterprise-Ready Messaging Infrastructure

WebLogic Server 9.0 implements cross-domain communication; bi-directional transactions from Enterprise Information Systems; automatic destination migration for high availability; message store-and-forward for improved reliability; and tightly controlled order of message delivery. In addition, this release improves management and performance of enterprise and messaging-intensive applications.

See "Java Message Service (JMS)" on page 1-22 and "Resource Adapters" on page 1-26.

System Administration and Management

The following sections describe new, enhanced, and deprecated features in WebLogic Server 9.0 that affect overall server administration and management.

Server Operations

The following sections describe key changes and improvements to WebLogic Server core operations.

Server Life Cycle State Isolates Administration Operations

A new life cycle state, `ADMIN`, facilitates application redeployment, maintenance, and troubleshooting. In the `ADMIN` state, WebLogic Server is running, but available only for administration operations, allowing you to perform server and application-level administration tasks without risk to running applications.

See "Understanding Server Lifecycle" in *Managing Server Startup and Shutdown*.

Work Contexts Ease Implementation and Maintenance

Developers can define properties as work contexts and can pass properties without explicitly including them in a remote call. A work context is propagated with each remote call—allowing the called component to add or modify properties defined in the work context. Similarly, the calling component can access the work context to obtain new or updated properties.

Work contexts facilitate diagnostics monitoring, application transactions, and application load-balancing, which require communication with remote components. They also provide information to third-party components.

Work contexts can be used by both clients and servers, and are enabled separately for each.

See *Developing WebLogic Server Applications*.

Network Channels Can Manage Traffic Between Server Instances

In addition to managing external network traffic, network channels can now manage network traffic between server instances. Other new and improved configuration and control options for network channels include:

- SSL behaviors configurable on a per-channel basis
- Dynamic configuration and starting of channels without re-booting the server instance
- New capabilities for closing and restarting a channel
- Replication channel for replication traffic among server instances in a WebLogic Server cluster

See *Designing and Configuring WebLogic Server Environments*.

Configuration Persistence Changes That Affect Core Functionality

The following changes to configuration data storage affect core functionality:

- New domain directory structure—Domain configuration data now resides in multiple files, including `config.xml`, in the new domain `config` directory. Configuration files are archived in JAR files in the `configArchive` directory under the domain directory. These changes affect recommended backup procedures. See "Avoiding and Recovering From Server Failure" in *Managing Server Startup and Shutdown.*

- Managed Servers cache configuration—A Managed Server always maintains a local copy of the domain configuration. See "Managed Server Independence Mode" in *Managing Server Startup and Shutdown*.

System-Wide Default Persistent Store

The WebLogic Persistent Store is a built-in, high-performance storage solution for WebLogic Server subsystems and services that require persistence, especially subsytems that require the creation and deletion of short-lived data objects, such as transactional messages for JMS Servers. Each server instance in a domain has a default persistent store that requires no configuration and that can be used simultaneously by subsystems that do not require explicit selection of a particular store, but can use the system's default storage. These subsystems include JMS Servers, Web Services, EJB Timer services, Store-and-Forward services, and the JTA Transaction Log (TLOG). Optionally, administrators can configure dedicated file-based stores or JDBC-accessible stores to suit their environment.

See Using The WebLogic Persistent Store in *Designing and Configuring WebLogic Server Environments*.

Store-and-Forward Service for Highly Available Messaging

The WebLogic Store-and-Forward (SAF) service enables WebLogic Server to deliver messages reliably between applications that are distributed across WebLogic Server instances. For example, with the SAF service, an application that runs on or connects to a local WebLogic Server instance can reliably send messages to a destination that resides on a remote server. If the destination is not available at the moment the messages are sent, then the messages are durably saved on a local server instance, and are forwarded to the remote destination once it becomes available.

WebLogic JMS utilizes the SAF service to enable local JMS message producers to reliably send messages to remote JMS queues or topics. WebLogic Web Services build the implementation of the Web Services Reliable Messaging (WSRM) standard on top of the SAF service.

See "Configuring and Managing WebLogic Store-and-Forward Service" for information about the benefits and usage of the WebLogic Store-and-Forward service.

New Look, Feel, and Functionality in the Console

The WebLogic Server Administration Console has been completely re-designed in the current release. The Administration Console is now built on the WebLogic Portal Framework, which makes it more open and more readily extensible. Other new features include:

- Improved navigation and user interface design.
- New application deployment and configuration tools, including assistants for installing applications, more configuration screens, and new deployment and redeployment controls for production applications. Additional Console updates enable you to assign values more easily to deployment plan variables when deploying an exported application.
- The WebLogic Diagnostic Service, with many new features for configuring, collecting, and viewing diagnostic information in a run-time environment. You access the service through the Console. See "Centralized Diagnostic Service with More Visibility and Run-time Control" on page 1-9.
- Domain change management features, including a domain lock and reliable batch change capabilities.

The next two sections describe Console enhancements in more detail.

Portal Application Support in the Administration Console

The Administration Console has a new architecture based on the WebLogic Portal Framework and a model-view-controller approach built on Struts, which makes the Console more open and more readily extensible. The Administration Console can now be extended in the same ways in which portal applications generally can be extended. A console extension could include simple overrides of existing pages, new pages and sections, and JSR 168 or WSRP portlets.

As in previous versions, a Console extension can add or remove Console functionality. A Console extension can also change aspects of the Console's appearance, such as colors or branding images. However, the new architecture necessitates new procedures for extending the Administration Console. WebLogic Administration Console extensions built against prior releases of WebLogic Server will not function with the new Console infrastructure. The extent of architectural and structural changes in this release make compatibility and migration impractical.

Predictable Distribution of Domain Configuration Changes

Improvements in change management enable you to distribute configuration changes throughout a domain securely, consistently, and predictably, regardless of whether you are using the Administration Console, the new WebLogic Scripting Tool, JMX, or other APIs.

To protect your changes and to prevent others from making changes, a new region in the Administration Console called the Change Center requires you to lock the Administration Console before you begin to modify a domain configuration.

When you finish making changes, you save and distribute them to all server instances in the domain (or you can roll back changes and release the lock). Each server determines whether it can accept the change. If all servers can accept the change, they update their working configuration tree and the change is completed. If one or more servers do not accept the change, the changes are not made in any of the servers, thus avoiding an incomplete intermediate state. This feature helps ensure that WebLogic Server configuration information is correct and consistent at all times.

WebLogic Server controls configuration changes in generally the same manner, whether the changes are implemented using the Administration Console, the WebLogic Scripting Tool, or the Configuration Manager service and JMX APIs:

- See Editing Configuration MBeans in *WebLogic Scripting Tool* for information about change management and the WebLogic Scripting Tool.
- See Managing Configuration Changes in *Understanding Domain Configuration* for information about change management, the Configuration Manager service, and JMX.

WebLogic Scripting Tool Automates Domain Configuration Tasks

The WebLogic Scripting Tool (WLST) is a new command-line interface that you use to configure WebLogic Server instances and domains, and manage and persist WebLogic Server configuration changes.

The WLST enables you to:

- Retrieve domain configuration and runtime information
- Edit the domain configuration and persist the changes in the `config.xml` file
- Navigate and edit custom, user-created MBeans and non-WebLogic Server MBeans, such as WebLogic Integration Server and WebLogic Portal Server MBeans
- Automate configuration tasks and application deployment
- Clone WebLogic Server domains
- Access Node Manager and start, stop, and suspend server instances remotely or locally, without requiring the presence of a running Administration Server

Based on the Java scripting interpreter, Jython, WLST interprets commands either interactively, supplied one-at-a-time from a command prompt, or in batches, supplied in a file (script), or embedded

in your Java code. You can use the scripting tool *online* (connected to a running server) and *offline* (not connected to a running server).

- Online, WLST provides simplified access to MBeans. You can perform administrative tasks and initiate WebLogic Server configuration changes while connected to a running server.
- Offline, WLST only provides access to persisted configuration information. You can create a new domain or update an existing domain without connecting to a running WebLogic Server— this functionality resembles that of the Configuration Wizard.

The weblogic.Admin utility is deprecated but still supported in the current release of WebLogic Server. Use the WebLogic Scripting Tool (WLST) for equivalent functionality.

See *WebLogic Scripting Tool*.

Centralized Diagnostic Service with More Visibility and Run-time Control

The new WebLogic Diagnostic Service integrates all diagnostics features and functionality into a centralized, unified framework that enables you to create, collect, analyze, and archive diagnostic data in a standard format. The Diagnostic Service provides more visibility into and control of the run-time performance of a server and its applications, allowing you to diagnose and isolate faults as they occur.

The following sections further describe WebLogic Diagnostic Service functionality. For more detailed information, see *Understanding the WebLogic Diagnostic Service*.

- "Increased Visibility" on page 1-10
- "Logging in a Standard Format" on page 1-10
- "Dynamic, Custom Instrumentation" on page 1-10
- "Diagnostic Context for Reconstruction and Correlation of Events" on page 1-10
- "Event Capture for a Single Request or a Single Client" on page 1-10
- "Data Harvesting from the Server and Applications" on page 1-11
- "Server Image Capture for Post-Failure Analysis" on page 1-11
- "Notifications Provide Tighter Integration with Legacy Monitoring Software" on page 1-11
- "Dynamic and Offline Access to Current and Historical Data" on page 1-11

- "Standard Integration of Third-Party Analytic Tools" on page 1-12

Increased Visibility

The WebLogic Diagnostic Service provides a dynamic view of metrics, data events, and logging and debug information. It also exposes new diagnostic data in areas of the server that were not previously exposed.

Logging in a Standard Format

Previous releases of WebLogic Server included a number of independent logging capabilities, such as the standard server and domain logs, JDBC log, and access log. The logs relied on different implementations and thus did not benefit from the same services and facilities (such as rotation) as the standard log. These logging solutions are now unified with a single implementation that provides the same qualities of service for all server logging. You can configure the WebLogic Diagnostic Service to collect, analyze, and archive all events generated by the WebLogic Logging Services.

Dynamic, Custom Instrumentation

You can add and remove diagnostic code selectively to Weblogic Server and to user-written applications in a running environment. You can collect and analyze particular types of run-time data events. Also, under certain circumstances, you can dynamically change the behavior of the diagnostic code executed at specific locations while the server is running by changing the diagnostic actions active at those locations.

Diagnostic Context for Reconstruction and Correlation of Events

The diagnostic context is a means of reconstructing transactional events and of correlating events based on the timing of the occurrence or on logical relationships. You can reconstruct or piece together a thread of execution from request to response, or generate diagnostic information only when contextual information in the diagnostic context satisfies certain criteria. This capability keeps the volume and overhead of generated information to manageable levels.

Event Capture for a Single Request or a Single Client

Most event collection is accomplished by designating key points in the application flow and then monitoring every request that passes through those points. However, to minimize collection volume and facilitate event isolation, it is sometimes preferable to capture only events for a single request or requests coming from a single client. The WebLogic Diagnostic Service allows you to mark, or dye, a particular request in such a way that the WebLogic Diagnostic Service can determine whether it

should be monitored. The WebLogic Diagnostic Service marks requests when they enter the system by setting flags in the diagnostic context, discussed earlier.

Data Harvesting from the Server and Applications

The Harvester component can be configured to collect metrics contained in server MBeans. Additionally, because the WebLogic Diagnostic Service enables you to harvest metrics from custom MBeans that you provide, you can also collect metrics from your own applications.

Server Image Capture for Post-Failure Analysis

The Server Image Capture component of the WebLogic Diagnostic Service creates a diagnostic snapshot from the server, either on-demand or automatically. The most common sources of server state— configuration, Log Cache, WorkManager, JNDI state, and harvestable data— are captured, as well as images from JMS, JDBC, EJB, JNDI, and other server subsystems.

When a server fails and recovers repeatedly over a short period—for example, in storm-related power failures— system administrators can also specify an image-lockout period, which prevents the server from repeatedly capturing and persisting similar diagnostic images that unnecessarily consume system resources.

Notifications Provide Tighter Integration with Legacy Monitoring Software

Customers who have developed software for trapping, routing, and handling of events from systems within their enterprise can configure the Diagnostic Service to analyze these custom application events and automatically generate notifications to alert system administrators. This capability enables a tighter integration with legacy monitoring frameworks in larger customer environments.

To monitor WebLogic Server, watches can be configured to detect specific conditions and to analyze logging and debugging log records, data events, and harvested metrics. Watches can also trigger different types of notification listeners, including Simple Mail Transfer Protocol (SMTP), Simple Network Management Protocol (SNMP), Java Management Extensions (JMX), and Java Message Service (JMS).

Dynamic and Offline Access to Current and Historical Data

All data collected from the server and applications running on it are persisted to permanent storage. The Data Accessor component provides access to all current and historical data collected by the WebLogic Diagnostic Service, including data events, log records, and harvested metrics. You can access archived data in on-line mode (on a running server) and off-line mode (after the server shuts down).

Standard Integration of Third-Party Analytic Tools

The WebLogic Diagnostic Service provides standard integration capabilities for third-party monitoring and analytic tools:

- Standard interfaces and patterns of integration—Third-party tools and clients can collect data similar to the way in which the WebLogic Sever Administration Console collects data. Third-party monitoring tools with well-defined publishing interfaces can be integrated with the WebLogic Diagnostic Service; can collect data and consume events via JMX, JMS, SMTP, and SNMP; and can display data through a user-defined console.
- Historical archive of diagnostic data—Data collected by the WebLogic Diagnostic Service is archived so that 1) the Administration Console and third-party tools can access the current state of a running server and historical data and 2) file system mechanisms can have off-line access to persisted, historical data for a server that has shut down.

The WebLogic Diagnostic Service is compatible with existing diagnostic tools and capabilities, such as instrumentation, developed by BEA partners. The WebLogic Diagnostic Service makes it easier for partners to integrate their tools, while BEA Systems takes ownership of the integration interfaces and provides support, documentation, and an improved quality of service.

WebLogic Logging Services

WebLogic Logging Services provide the following new features and configuration options.

Logging Volume Control

Enhancements to LogMBean interfaces let you control logging output by setting the severity level and filters on both loggers and handlers.

In earlier versions of WebLogic Server, system administrators and developers had only programmatic access to loggers and handlers. In this release, you can configure handlers by using MBeans, eliminating the need to write code for most basic logging configurations. The Administration Console and WebLogic Scripting Tool (WLST) provide an interface for interacting with logging MBeans. Loggers are configured only through the API.

Support for Log4J

WebLogic Server supports Jakarta Project Log4j. In the Administration Console, you specify Log4j or the default Java Logging implementation. Alternatively, you can configure Log4j logging through the `LogMBean` interface using the `LogMBean.isLog4jLoggingEnabled` attribute.

Support for Commons API

The Jakarta Commons Logging APIs provide an abstraction layer that insulates users from the underlying logging implementation. WebLogic Server provides an implementation of the Commons `LogFactory` interface, letting you issue requests to the server `Logger` using this API.

New Log File Directories and Locations

- The server log file is located in the `logs` directory below the server instance root directory; for example, *`root-directory`*`\servers\`*`server-name`*`\logs\`*`server-name`*`.log`.
- The domain log resides in the Administration Server `logs` directory. The default name and location for the domain log file is *`root-directory`*`\servers\`*`AdminServer-name`*`\logs\`*`domain-name`*`.log`.
- By default, the rotated files are stored in the same directory where the log file is stored. You can specify a different directory location for the archived log files by using the Administration Console or setting the `LogFileRotationDir` property of the `LogFileMBean` from the command line.

Log Content Enhancements

All log messages include:

- Diagnostic context information and a request identifier to correlate messages coming from a specific request or application.
- Time stamp in milliseconds.

See *Configuring Log Files and Filtering Log Messages*.

Web Application and Resource Adapter Logging

In this release of WebLogic Server, you can configure Web application and resource adapter logging behavior using WebLogic-specific deployment descriptors. The logging configuration deployment descriptor elements define similar attributes used to configure server logging through the `LogFileMBean` interface, such as the log file name, location, and rotation policy.

Similarly, WebLogic logging services are provided to J2EE resource adapters for `ManagedConnectionFactory` scoped logging. You configure the log file name, location, and rotation policy for resource adapter logs through the `weblogic-ra.xml` deployment descriptor.

See *Using WebLogic Logging Services for Application Logging*.

Server Reliability, Availability, Scalability, and Performance

WebLogic Server 9.0 significantly improves server and cluster RASP:

Automatic and Manual Server Migration to Another Machine

WebLogic Server now supports automatic and manual migration of a clustered server instance and all the services it hosts from one machine to another. This feature is designed for environments with high availability requirements. Use the server migration capability to:

- Increase availability of services that must run on only a single server instance at any given time in a cluster, such as JMS, when the hosting server instance fails.
- Manually migrate a server instance at any time, not just when it fails, for administrative reasons.

See "Server Migration" in *Using WebLogic Server Clusters*.

Support for CommonJ Timer and Work Manager API Specification

WebLogic Server 9.0 supports part of the BEA and IBM Joint Specifications (CommonJ) described at http://dev2dev.bea.com/technologies/commonj/index.jsp. In particular, this release implements the Timer and Work Manager 1.1 Specification, available at http://dev2dev.bea.com/technologies/commonj/twm/index.jsp.

The Timer API provides a simple API that applications can use to create and use timers managed within the application server, and is a recommended alternative to the J2SE `java.util.Timer` class.

The Work Manager API enables an application to break a single request task into multiple work items, and assign those work items to execute concurrently using multiple Work Managers configured in WebLogic Server. (Applications that do not need to execute concurrent work items can also use configured Work Managers by referencing or creating Work Managers in their deployment descriptors or, for J2EE Connectors, using the JCA API.) See also "Server Self-Tuning for Production Environments" on page 1-16 for more information about the new Work Manager tuning strategies.

HTTP Session Replication and Failover Across Wide or Metropolitan Area Networks

The state of an HTTP session running on a server instance in one cluster can be replicated on a server instance in a different WebLogic Server cluster. The clusters can be located on different LANs within the same metropolitan area network (MAN), or be in geographically distant locations—in different cities or states—within a wide area network (WAN). Upon a failure of the primary server instance, another member of the same cluster can recover the session data from the remote instance (in another cluster) and make it available in the primary cluster. If no members of the cluster in which the primary failed are available, the request can fail over to the remote cluster hosting the session replica.

Here are two example environments in which WebLogic Server cross-cluster session replication features are useful:

- WAN replication—A company has one data center in San Francisco and another in Los Angeles, each with a WebLogic Server cluster. The company's service level agreement requires that, in the event of a complete data center failure, session requests can fail over to the other data center. WebLogic Server's *WAN session state replication* can be used. WAN session replication entails:
  - In-memory replication to a local server instance.
  - *Asynchronous* JDBC persistence to a remote cluster instance. Periodically, session states flushed to storage in a table on the server instance on the remote cluster. Because the JDBC persistence to the remote server instance is performed asynchronously, it provides improved performance over synchronous JDBC replication, in which the database write paces session completion.
- MAN replication—A company has two locations with a fast low-latency interconnect between the locations. In both sites, a WebLogic Server cluster hosts an application with high availability requirements. The session state is replicated between two clusters synchronously.

See *Using WebLogic Server Clusters*.

Server Self-Tuning for Production Environments

New self-tuning capabilities simplify the process of configuring WebLogic Server for production environments with service level requirements that vary over time or by application. Self-tuning helps prevent deadlocks during periods of peak demand. Self-tuning features are also useful if your WebLogic Server environment hosts multiple applications with different performance and availability requirements—for example, allowing you to allocate a greater percentage of resources to a user-facing order processing application than to a back-end inventory management application.

The new queue strategy enables administrators to allocate processing resources and manage performance more effectively, by avoiding the effort and complexity involved in configuring, monitoring, and tuning custom executes queues.

Key self-tuning features in WebLogic Server include:

- Workload management—Administrators can define scheduling policies and constraints at the domain level, application level, and module level.
- Automatic thread count tuning—A thread pool can maximize throughput by automatically changing its size, based on throughput history and queue size.
- Thread scheduling functionality—WebLogic Server 9.0 implements the commonj work manager API, exposing thread scheduling functionality to developers. Applications can also use the Work Manager API to execute work asynchronously and receive notifications on the execution status.

See *Designing and Configuring WebLogic Server Environments*.

Node Manager Enhancements

A number of enhancements make Node Manager more versatile and easier to use:

- Shell Script Node Manager—WebLogic Server 9.0 provides a version of Node Manager implemented as a shell script. The Node Manager shell script provides the same functionality as the Java Node Manager. In addition to Remote Shell (RSH) protocol, Node Manager can be configured with the Secure Shell (SSH) for secure remote control of server instances running on UNIX or Linux systems.
- WebLogic Scripting Tool (WLST) support— You can now use WLST commands to access Node Manager and to start, stop, and suspend server instances remotely or locally; obtain server status; and retrieve the contents of the server output log, without requiring the presence of a running Administration Server. In addition, you can configure the machine on which WLST is running to be monitored by Node Manager. See "WebLogic Scripting Tool Automates Domain Configuration Tasks" on page 1-8.

- Administration Server control—In previous versions, Node Manager required access to a running Administration Server, and could control and monitor only Managed Servers. In WebLogic Server 9.0, Node Manager can start, monitor, and restart Administration Servers.
- Node Manager runs as a Windows service—BEA Systems recommends running Node Manager as an operating system service so that it is automatically restarted in the event of system failure or reboot and using Node Manager to start or restart servers. See "Installing the Node Manager as a Windows Service" in the *Installation Guide*.
- Clustered server migration—In WebLogic Server 9.0, Node Manager is used to accomplish migration of servers in a WebLogic Server cluster. For more information, see "Server Migration" in *Using WebLogic Server Clusters*.
- Improved diagnostics and logging—Node Manager diagnostics are improved, and the logging strategy for Node Manager and the server instances it controls are simplified.
- Simplified setup—Among other improvements, Node Manager no longer requires two-way SSL. Only one-way SSL is required.
- Start Script Support—You can configure Node Manager to use a start script to start WebLogic Server instances.

See Using Node Manager to Control Servers in *Designing and Configuring WebLogic Server Environments*.

Dynamically Generated Cluster Address for Each Request

In this release, you can still explicitly define a cluster address, but if you do not, WebLogic Server dynamically generates the cluster address for each request. This capability eases the configuration effort, and ensures that the cluster address correctly reflects the cluster membership at startup. See "Cluster Address" in *Using WebLogic Server Clusters*.

New Overload Protection Increases Availability

New overload features protect a server instance from out-of-memory (OOM) exceptions, execute queue overloads, increasing the availability of a server or a cluster.

See *Designing and Configuring WebLogic Server Environments*.

J2EE Standard Web Services

Web Services are now a J2EE standard, which has resulted in many changes between 8.1 and 9.0 WebLogic Web Services.

WebLogic Server 9.0 includes the following new Web Services features:

For information about what has changed between 8.1 and 9.0 Web Services, and what has been deprecated, see "Differences Between WebLogic 8.1 and 9.0 Web Services" on page 1-18.

Differences Between WebLogic 8.1 and 9.0 Web Services

J2EE 1.4 introduces a standard Java component model for authoring Web Services with new specifications such as *Implementing Enterprise Web Services* (JSR-921, the 1.1 maintenance version of JSR-109) and *Java API for XML Registries* (JAX-R), as well as updated JAX-RPC and SAAJ specifications.

As a result, the 9.0 programming model used to create WebLogic Web Services has also changed. The model now takes advantage of the powerful new metadata annotations feature introduced in Version 5.0 of the JDK (specified by JSR-175). Additionally, the runtime environment upon which WebLogic Web Services run now supports the *Implementing Enterprise Web Services* specification.

See Differences Between 8.1 and 9.0 WebLogic Web Services at http://e-docs.bea.com/wls/docs90/webserv/intro.html#8.1diff.

JWS Annotations-Based Programming Model

The programming model used to create 9.0 WebLogic Web Services is based on the new JDK 5.0 metadata annotations feature (specified by JSR-175). In this model, the implementation of your Web Service is a Java file that uses JWS annotations, defined by the *Web Services Metadata for the Java Platform* specification (JSR-181).

Note: Although using JWS annotations is the preferred programming model for creating WebLogic Web Services, you can also create one manually by programming the EJB or Java class that implements the Web Service and creating all the required components, such as the Web Service deployment descriptors and the WSDL file.

See Programming the JWS File at http://e-docs.bea.com/wls/docs90/webserv/jws.html.

Web Services for J2EE 1.1 Implementation

The *Web Services for J2EE*, Version 1.1 specification defines the standard J2EE runtime architecture for implementing Web Services in Java.

JSR-921 is the 1.1 maintenance release of JSR-109, which was the J2EE 1.3 specification for Web Services. JSR-921 is currently in final release of the JCP (Java Community Process).

See Anatomy of a WebLogic Web Service at http://e-docs.bea.com/wls/docs90/webserv/overview.html#Anatomy.

Asynchronous, Loosely Coupled Web Services

WebLogic Web Services support a variety of asynchronous features:

- *Reliable SOAP messaging*, in which two Web Services running on different WebLogic Server instances can communicate reliably in the presence of failures in software components, systems, or networks. This feature implements the WS-ReliableMessage specification.
- *Conversational Web Services*, in which two or more parties exchange related messages about a particular subject, typically a business transaction.
- *Callbacks*, in which two stateless Web Services interact and one calls back to the other.
- *Addressing,* which defines XML elements to identify Web Service endpoints and to secure end-to-end endpoint identification in messages. This feature implements the WS-Addressing specification.

See Using Reliable SOAP Messaging at http://e-docs.bea.com/wls/docs90/webserv/advanced.html#reliable_messaging.

Note: Documentation for conversations, callbacks, and addressing not available for Beta.

Digital Signatures and Encryption

WebLogic Web Services provide support for digitally signing and encrypting the request and response SOAP messages generated when invoking a Web Service. This capability derives from conformance with the following OASIS Standard 1.0 Web Services Security specifications:

- SOAP Message security
- Username Token Profile
- X.509 Token Profile

Message-level security is configured using security policy statements, as specified by the WS-Policy specification.

See Configuring Security at http://e-docs.bea.com/wls/docs90/webserv/security.html.

Data Binding Between XML and Java

As in previous releases, WebLogic Web Services support a full set of built-in XML Schema, Java, and SOAP types, as specified by the JAX-RPC 1.1 specification, that you can use in your Web Service operations without performing any additional programming steps.

Additionally, you can use a variety of user-defined XML and Java data types, including XMLBeans, as input parameters and return values of your Web Service. The WebLogic Web Services Ant tasks automatically generate the data binding components needed to convert the user-defined data types between their XML and Java representations.

See Data Types and Data Binding at http://e-docs.bea.com/wls/docs90/webserv/data_types.html.

Use of Policy Files

WebLogic Web Services 9.0 implement the WS-Policy specification, which provides a general purpose model and corresponding syntax to describe and communicate the policies of a Web Service. In this release, policy files are used only for configuring the reliable messaging and security features.

See Use of WS-Policy File for Message-Level Security Configuration at http://e-docs.bea.com/wls/docs90/webserv/security.html#ws_policy and Use of WS-Policy Files for Reliable SOAP Message Configuration at http://e-docs.bea.com/wls/docs90/webserv/advanced.html#ws_policy for information on how WS-Policy files are used to configure security and reliable messaging, respectively.

Ant Tasks That Handle JWS Files

A set of Ant tasks handle JWS annotated files and integrate them into the WebLogic split development directory environment. This development environment consists of a directory layout and associated Ant tasks that help you repeatedly build, change, and deploy J2EE applications, including Web Services.

See Web Services Ant Task Reference at http://e-docs.bea.com/wls/docs90/webserv/anttasks.html.

Implementations of Standard Web Services-Related Java Specifications

Web Services in WebLogic Server 9.0 implement the following standard Java specifications:

- Web Services Metadata for the Java Platform (JSR-181) at http://www.jcp.org/en/jsr/detail?id=181
- Enterprise Web Services 1.1 (JSR-921) at http://www.jcp.org/en/jsr/detail?id=921
- JAX-RPC 1.1 at http://java.sun.com/xml/jaxr/index.jsp
- SOAP Attachments for Java (SAAJ) 1.2 at http://java.sun.com/xml/saaj/index.jsp
- Java API for XML Registries 1.0 (JAX-R) at http://java.sun.com/xml/jaxr/index.jsp

Conformance with Standard Web Services Specifications

Web Services in WebLogic Server 9.0 conform to the following standard Web Services specifications:

- SOAP 1.1 at http://www.w3.org/TR/soap/
- WSDL 1.1 at http://www.w3.org/TR/wsdl
- UDDI 2.0 at http://www.uddi.org/
- WS-Security at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- WS-Addressing at http://www.w3.org/2002/ws/addr/
- WS-Policy at http://www-106.ibm.com/developerworks/library/specification/ws-polfram/
- WS-ReliableMessaging at http://www-106.ibm.com/developerworks/library/ws-rm/

Java Message Service (JMS)

WebLogic Server 9.0 introduces major changes in the configuration, deployment, and dynamic administration of WebLogic JMS.

For more information about these and other new features and changes in WebLogic JMS, see "New and Changed JMS Features In This Release" in *Configuring and Managing WebLogic JMS*.

JMS 1.1 Specification Support for Production Environments

WebLogic Server 9.0 is compliant with the new JMS 1.1 Specification for use in production, and so supports unified APIs that work for both queues and topics. See the Java JMS technology page on the Sun Web site at `http://java.sun.com/products/jms/`.

Modular Configuration and Deployment of JMS Resources

JMS configurations in WebLogic Server 9.0 are stored as modules, defined by XML documents that conform to the new `weblogic-jmsmd.xsd` schema. You create and manage JMS resources either as *system modules*, similar to the way they were managed prior to version 9.0, or as *application modules*.

JMS application modules are a WebLogic-specific extension of J2EE modules and can be deployed either within a J2EE application (a *packaged module*) or as standalone modules. For more details, see "JMS and JDBC Deployable Resource Configuration" on page 1-46.

With modular deployment of JMS resources, you can migrate your application and the required JMS configuration from environment to environment, such as from a testing environment to a production environment, without opening an EAR file and without extensive manual JMS reconfiguration.

See Understanding JMS Resource Configuration in *Configuring and Managing WebLogic JMS*.

Store-and-Forward for Highly Available Message Production

The JMS Store-and-Forward feature builds on the WebLogic Store-and-Forward (SAF) service to provide highly-available JMS message production. For example, a JMS message producer connected to a local server instance can reliably forward messages to a remote JMS destination, even though that remote destination may be temporarily unavailable when the message was sent. JMS Store-and-forward is transparent to JMS applications; therefore, JMS client code still uses the existing JMS APIs to access remote destinations.

See "Configuring and Managing WebLogic Store-and-Forward Service".

Enhanced Run-time Message Management

Extensive message administration improvements greatly enhance a JMS administrator's ability to view and manipulate messages in a running JMS Server, using either the Administration Console or new public run-time APIs. These message management enhancements include message browsing (for sorting), message manipulation (such as move and delete), message import and export, as well as transaction management, durable subscriber management and JMS client connection management.

Pause and Resume Message Operations on Destinations

New WebLogic JMS configuration and run-time APIs enable an administrator to pause and resume message production, message insertion (in-flight messages), and message consumption operations on a given JMS destination, or on all the destinations hosted by a single JMS Server, either programmatically or administratively. This capability enables administrative control of the JMS subsystem behavior in the event of an external resource failure.

More Transparency with Message Life Cycle Logging

The message life cycle is an external view of the basic events that a JMS message traverses once it is accepted by a JMS server. The Message Life Cycle Logging feature provides more transparency of JMS

messages from a JMS server viewpoint, in particular basic life cycle events, such as message production, consumption, and removal. Logging can occur on a continuous basis and over a long period. It can be used in real-time mode while the JMS server is running, or in an off-line fashion when a JMS server is down.

See New and Changed JMS Features In This Release in *Configuring and Managing WebLogic JMS*.

Debug and Diagnostic Information More Readily Available

The JMS subsystem uses the new system-wide WebLogic Diagnostic service for centralized debug access and logging. Debug and diagnostic information is more readily available so you can easily diagnose and fix problems.

See *Understanding the WebLogic Diagnostic Service*.

Strict Message Ordering with Unit-of-Order

The Unit-of-Order feature goes beyond the message delivery ordering requirements in the JMS 1.1 Specification by enabling JMS message producers to group ordered messages into a single unit. This single unit is called a *Unit-of-Order* and it *guarantees* that all messages created from that unit are processed serially, by the same consumer, in the order in which they were created, until that consumer acknowledges them or is closed. For example, if a queue has messages with many consumers, and each message has an account number as a Unit-of-Order, then two consumers will not process messages with the same account number at the same time.

See Using Message Unit-of-Order in *Programming WebLogic JMS*.

Uniform Distributed Destinations

A new type of distributed destination, the *uniform distributed destination*, greatly simplifies the management and development of distributed destination applications. Using this feature, the administrator no longer needs to create or designate destination members, but relies on the system to uniformly create the necessary members on the JMS servers to which a JMS module is targeted. This feature ensures the consistent configuration of all distributed destination parameters, particularly in regard to weighting, security, persistence, paging, and quotas.

Access to JMS Applications from Programs Written in C

The Weblogic JMS C API enables programs written in "C" to participate in JMS applications. This implementation of the JMS C API uses JNI in order to access a Java Virtual Machine (JVM).

See WebLogic C API in *Programming WebLogic JMS*.

Message-Driven Bean (MDB) Enhancements for JMS

MDB enhancements enable support for transaction batching (via processing multiple messages in a single transaction), and for load balancing distributed destinations across member destinations in different clusters or domains, regardless of whether the MDB and destination reside in the same cluster or in different clusters or domains.

See "Message-Driven Bean Enhancements" on page 1-35.

Document Object Model (DOM) Support for XML Messages

The WebLogic JMS API is enhanced to provide native support for the Document Object Model (DOM) in the transmission of XML messages. This capability improves performance for implementations that already use a DOM, because those applications will not have to flatten the DOM before sending XML messages.

See Sending XML Messages in *Programming WebLogic JMS*.

Deprecated JMS Features, Methods, and Interfaces

In WebLogic Server 9.0, many changes were made to the JMS subsystem, including the removal of some classes and the deprecation of many MBeans. For a complete listing of deprecated JMS APIs, see New and Changed JMS Features In This Release in *Configuring and Managing WebLogic JMS*.

Legacy JMS Configuration Interfaces

The new descriptor-based method of configuring JMS resources uses Java Descriptor Bean interfaces to create deployable JMS resource modules. This fundamental change necessitated the deprecation of most JMS configuration MBean interfaces, with the exception of the JMSServerMBean interface.

JMS Helper APIs

The descriptor-based method of configuring JMS module resources necessitated the deprecation of the JMSHelper class for locating JMS run-time and configuration JMX MBeans. This class is replaced by a new JMSModuleHelper class, with methods for locating JMS runtime MBeans, and managing JMS Module configuration entities in a given module, including the JMS Interop Module.

See the JMSModuleHelper Javadoc.

JMS Session Pool and JMS Connection Consumer MBean Interfaces

The JMSSessionPoolMBean (and its associated methods on the JMSServerMBean) and the JMSConnectionConsumerMBean interfaces have been deprecated. These interfaces were used to automatically create a JMS session pool and start the JMS consumers on the server side. The ConnectionConsumer and ServerSessionPool APIs are still supported, but BEA strongly recommends using message-driven beans (MDBs), which are simpler, easier to manage, and more capable.

JMS File Store and JMS JDBC Store MBean Interfaces

The new WebLogic Persistent Store necessitated the deprecation of the JMSStoreMBean, JMSFileStoreMBean, and JMSJDBCStoreMBean interfaces. This deprecation also includes any associated JMS Store methods on the JMSServerMBean interface.

See Using The WebLogic Persistent Store in *Designing and Configuring WebLogic Server Environments*.

Resource Adapters

WebLogic Server now fully supports the J2EE 1.5 Connector Architecture as well as resource adapters based on the J2EE 1.0 Connector Architecture. Deployment descriptors for version 1.5 are schema-based, while deployment descriptors for version 1.0 are DTD-based. Except where noted, the following sections describe new functionality for version 1.5 adapters:

Multiple Outbound Connections

The J2EE 1.5 Connector Architecture now supports resource adapters with multiple outbound connections. The outbound resource adapter allows an application to connect to an EIS system and perform work. All communication is initiated by the application. The resource adapter serves as a passive library for connecting to an EIS and executes in the context of the application threads. See "Understanding Resource Adapters" in *Programming WebLogic Resource Adapters*.

Inbound-Outbound Transactions

Resource adapters in previous versions supported outbound messaging. Now, Version 1.5 resource adapters can also receive transactions, including messages, from an EIS. An EIS can send a transactional context under which messages are delivered or work is performed in the form of Work Requests. A message endpoint application (a message-driven bean and possibly other J2EE components) receives inbound messages from the EIS through the resource adapter.This functionality makes the WebLogic Server proprietary InterposedTransactionManager available through a J2EE standard interface, allowing J2EE applications to fully participate in an enterprise environment controlled by an external Transaction Manager. Prior to EJB 2.1, a message-driven bean (MDB) only supported Java Message Service (JMS) messaging. See "Messaging and Transaction Inflow" in *Programming WebLogic Resource Adapters*.

Connection Proxies No Longer Necessary

Late transaction enlistment and idle connection detection are now available to 1.5 resource adapters without requiring connection proxies. See "Connection Management" in *Programming WebLogic Resource Adapters*.

RAR Visibility to External Application Components

The Connector specification prohibits the application server from providing access to resource adapters defined in an EAR by application components outside the EAR. However, WebLogic Integration has a continuing requirement to provide such access. The `enable-access-outside-app` element of the `weblogic-ra.xml` deployment descriptor provides a configuration parameter for explicitly enabling such access. See "weblogic-ra.xml Schema" in *Programming WebLogic Resource Adapters*.

Resource Adapter Life Cycle Management

The application server calls new start() and stop() methods as part of its deployment and undeployment actions on 1.5 resource adapters. See "Packaging and Deploying Connectors" in *Programming WebLogic Resource Adapters*.

Suspend( ) and Resume( ) Speed In-Flight Transactions

New suspend() and resume() methods enable the 1.5 resource adapter to shut down all incoming messages while allowing in-flight transactions to complete and then to resume normal operations. See "Configuration" in *Programming WebLogic Resource Adapters*.

Self-Tuning Work Manager Avoids Direct Creation of Threads

The J2EE 1.5 Connector Architecture specification advises against a resource adapter creating threads. To enable resource adapters to perform work without direct creation of threads, a self-tuning, configurable Work Manager has been added to create WorkRequests under the control of the application server. The Work Manager is configurable and allows you to manage resources. See "Messaging and Transaction Inflow" in *Programming WebLogic Resource Adapters*.

Resource Adapter Security Identities

A number of new security identities can now be configured through elements of the weblogic-ra.xml deployment descriptor. See "Security" in *Programming WebLogic Resource Adapters*.

Adapters Bound in JNDI Tree as Independent Objects

Version 1.0 resource adapters are identified by their ConnectionFactory objects bound in the JNDI tree. Version 1.5 resource adapters are now bound in the JNDI tree as independent objects, making them available as system resources in their own right or as message sources for MDBs. See "Connection Management" in *Programming WebLogic Resource Adapters*.

Connection Factory-Specific and Adapter-Scoped Properties

All transaction and security property settings apply to all outbound connection factories. BEA has extended this functionality to allow per-connection-factory settings as well as resource adapter-scoped properties. This includes a credential map per connection factory. See "Security" in *Programming WebLogic Resource Adapters*.

Viability Test (ping) for Single and Pooled Connections

You can now test either a specific outbound connection or the entire pool of outbound connections for a particular ManagedConnectionFactory. Testing the entire pool will test each connection in the pool individually. See "Connection Management" in *Programming WebLogic Resource Adapters*.

Configurable Connection Proxy Generation for 1.0 Adapters

If you already know whether a connection proxy can be used in the resource adapter, you can avoid a proxy test by explicitly setting the `use-connection-proxies` element in the WebLogic Server 8.1 version of `weblogic-ra.xml` to `true` or `false`. This is applicable to resource adapters based on the J2EE 1.0 Connector Architecture, which are still supported in this release. See "Connection Management" in *Programming WebLogic Resource Adapters*.

Deprecation of link-ref Mechanism

The link-ref mechanism is a 1.0 resource adapter feature that is deprecated but still supported in the current release. The feature allows a base resource adapter to be referenced by a child resource adapter, giving the child access to the base resource adapter's classes and configuration. For 1.5 resource adapters, this mechanism is replaced by the federated application, which is a stand-alone module that can be accessed by any application. See "Configuration" in *Programming WebLogic Resource Adapters*.

JDBC

The following sections describe new features and changes for JDBC in WebLogic Server 9.0:

For more information about these and other new features and changes in WebLogic JDBC, see "New and Changed Features in WebLogic JDBC" in *Configuring and Managing WebLogic JDBC*.

JDBC 3.0 Support

WebLogic Server 9.0 is compliant with the JDBC 3.0 specification. For more information about JDBC 3.0 features, see the Java JDBC technology page on the Sun Web site at http://java.sun.com/products/jdbc/. For more information about WebLogic JDBC, see *Configuring and Managing WebLogic JDBC*.

RowSet Extensions

The WebLogic RowSet implementation complies with and extends the new JDBC RowSet Implementations Specification (JSR-114). See the Java JDBC technology page on the Sun Web site at http://java.sun.com/products/jdbc/.

WLCachedRowSets extends and can be used interchangeably with several standard RowSet types. It also includes methods for setting optimistic concurrency options and data synchronization options. SharedRowSet extends CachedRowSets so that additional CachedRowSets can be created for use in other threads based on data in an original CachedRowSet. SortedRowSets extends CachedRowSets so that rows in a CachedRowSet can be sorted in memory rather than depending on the database management system for sort processing. SharedRowSets and SortedRowSets increase performance by reducing the number of database round-trips required by an application.

See "Using RowSets with WebLogic Server" in *Programming WebLogic JDBC*.

Support for the J2EE Management Model (JSR-77)

WebLogic Server 9.0 JDBC fully supports JSR-77, which defines the J2EE Management Model. You access the J2EE Management Model to monitor resources, including the WebLogic JDBC system as a whole, JDBC drivers loaded into memory, and JDBC data sources.

See *Configuring and Managing WebLogic JDBC*.

Modular Configuration and Deployment of JDBC Resources

JDBC configurations in WebLogic Server 9.0 are now stored in XML documents that conform to the new `weblogic-jdbc.xsd` schema. You create and manage JDBC resources either as system modules, similar to the way they were managed prior to version 9.0, or as application modules. JDBC application modules are a WebLogic-specific extension of J2EE modules and can be deployed either within a J2EE application or as stand-alone modules. For more details, see "JMS and JDBC Deployable Resource Configuration" on page 1-46.

In support of the new deployment model for JDBC application modules in WebLogic Server 9.0, BEA now provides a schema for WebLogic JDBC modules. Each JDBC system module or application module that you create must conform to the schema. IDEs and other tools can validate JDBC modules based on the schema.

For more information about WebLogic JDBC resources, see *Configuring and Managing WebLogic JDBC*.

Fewer Resource Types for Simpler JDBC Configuration

Fewer JDBC resource types simplify JDBC configuration and reduce the likelihood of configuration errors. Instead of configuring a JDBC connection pool and then configuring a data source or transactional (tx) data source to point to the connection pool and bind to the JNDI tree, you configure a data source that encompasses a connection pool.

Also, MultiDataSources replace MultiPools. A MultiDataSource does not require a separate data source to bind it to the JNDI tree. If you use the Administration Console, you can configure the MultiDataSource and all encompassed data sources in one step.

See *Configuring and Managing WebLogic JDBC*.

JDBC Monitoring and Diagnostics Enhancements

The following enhancements facilitate JDBC monitoring and diagnostics.

New Monitoring Statistics and Profile Information for JDBC Resources

New data source usage information is available through the Administration Console and JMX:

- Profile information—Applications can get detailed usage profiles of JDBC resources. The data includes current applications that are using pooled connections, current applications waiting for a connection, current entries in the prepared statement cache, and more.

- Monitoring and tuning statistics —Total number of connections used over the life of a connection pool, the number of applications waiting for a connection from the connection pool, average time an application waits for a connection, average time a connection is in use by an application, and more.

Callbacks for Monitoring Driver-Level Statistics

Callbacks for methods called on a JDBC driver enable you to monitor and profile JDBC driver usage, including methods being executed, exceptions thrown, and the time spent executing driver methods.

JDBC Debugging Enhancements

The JDBC subsystem uses the new system-wide WebLogic Diagnostic Service for centralized debugging access and logging.

See *Understanding the WebLogic Diagnostic Service*.

Optimized Performance for Non-XA Resources in Global Transactions

You can enable Logging Last Resource (LLR) transaction optimization for a data source, which enables one non-XA resource to participate in a global transaction with improved performance and with the same ACID (atomic, consistent, isolated, and durable) guarantee as XA.

LLR transaction optimization confers these advantages:

- No need for XA processing at the database level if the database is the one non-XA resource.
- No need for an XA JDBC driver to connect to the database. XA JDBC drivers are typically inefficient compared to non-XA JDBC drivers.
- Fewer transaction processing steps, which reduces network traffic and the number of disk I/Os.

See "Understanding the Logging Last Resource Transaction Option" in *Configuring and Managing WebLogic JDBC*.

Credential Mapping of WebLogic and Database User IDs

Credential mapping for a JDBC data source enables WebLogic Server to set a mapped database user ID as a light-weight client ID on a database connection. This can reduce the need for creating new connections with a specific database user ID or may enable your application to take advantage of connection pooling performance advantages.

See "Configuring Credential Mapping for a Data Source" in *Configuring and Managing WebLogic JDBC*.

Multi Data Sources Supported for XA Transactions

If you configure the data sources used by a multi data source to use XA JDBC drivers, you can take advantage of the failover features in multi data sources for your applications that include global transactions.

See "XA Support in Multi Data Sources" in *Configuring and Managing WebLogic JDBC*.

MySQL Support

WebLogic Server 9.0 is supported for use with a MySQL database, a popular open-source production-ready database management system. The MySQL Connection/J JDBC driver is installed with WebLogic Server. MySQL is certified for use with JMS, JDBC, and EJB container-managed persistence. MySQL does not support XA or JTA.

Updated WebLogic Type 4 JDBC Drivers

Updates to the WebLogic Type 4 JDBC drivers resolve important issues and include some notable enhancements. See *WebLogic Type 4 JDBC Drivers* for more information.

Deprecated JDBC Features, Methods, Interfaces, and MBeans

Removed from WebLogic Server 9.0:

- WebLogic JDriver for Oracle. Replaced by the WebLogic Type 4 JDBC Driver for Oracle.
- WebLogic JDriver for MS SQL Server. Replaced by the WebLogic Type 4 JDBC Driver for MS SQL Server.

Deprecated in WebLogic Server 9.0:

- Data source factories and legacy application-scoped connection pools. Replaced by JDBC modules. See "Modular Configuration and Deployment of JDBC Resources" on page 1-29.
- Legacy driver-level logging. Replaced by DebugJDBCDriverLogging. See "JDBC Debugging Enhancements" on page 1-31.
- JDBCXADebugLevel of the JDBCConnectionPoolMBean. Replaced with the JTAJDBC debugging scope on the ServerDebugMBean.

- Legacy connection leak profiling, statement cache profiling, and statement usage profiling. Replaced with new JDBC resource profiling. See "New Monitoring Statistics and Profile Information for JDBC Resources" on page 1-30.
- JDBCConnectionPoolMBean, JDBCDataSourceMBean, JDBCTxDataSourceMBean, and JDBCMultiPoolMBean. Replaced by JDBCSystemResourceMBean.
- JDBCDataSourceFactoryMBean. No replacement.
- JDBCConnectionPoolRuntimeMBean. Replaced by JDBCDataSourceRuntimeMBean.

Enterprise JavaBeans

WebLogic Server 9.0 complies with the EJB 2.1 specification and includes many EJB usability improvements, as described in the following sections.

For more information about all these features, see *Programming WebLogic Server EJBs*.

New Features to Support the EJB 2.1 Specification

The following sections describe new WebLogic Server features that comply with the J2EE EJB 2.1 Specification.

EJB Timer Service for Modeling Business Processes

In compliance with EJB 2.1, the WebLogic Server EJB Timer Service enables you to schedule a notification at a particular time, at the end of an elapsed period of time, or at recurring intervals.

EJB-QL Changes

In compliance with the EJB 2.1 specification, the following Enterprise JavaBeans query language (EJB-QL) functions have been added or changed in WebLogic Server 9.0:

- MOD arithmetic function (new)
- Aggregate functions (enhanced)
- Order By clause (enhanced)

Message Destination References

WebLogic Server 9.0 satisfies the EJB 2.1 requirement that logical message destinations can be declared in a deployment descriptor and mapped to an actual message destination, such as a JMS queue. EJBs can look up the message destination by performing a JNDI lookup using the logical message destination name.

Stateless Session Beans Exposed as Web Services

WebLogic Server 9.0 complies with EJB 2.1 requirements related to declaring and accessing external Web Services and exposing stateless session EJBs as Web Services. These features make it easier to develop and maintain EJBs that access Web Services. Both Java and non-Java clients can access stateless session beans as Web Services.

Message-Driven Bean Enhancements

Several enhancements make MDBs more powerful, and easier to use and support.

- MDBs can receive messages from JCA 1.5-compliant adapters, thus facilitating the integration of WebLogic Server applications with Enterprise Information Systems.
- A unique `client-id` for can be created for every instance of an MDB, making it easier to deploy durable MDBs to multiple servers instances in a WebLogic Server cluster.
- The EJB container suppresses repetitive exceptions that can occur as it repeatedly tries and fails to deliver a JMS message. The container prints out the exception and the number of times it has been thrown, rather than printing the exception message each time it occurs.
- You can automatically pause message processing for a configurable time period when MDB applications throw exceptions.
- You can administratively pause and resume message delivery without undeploying.

- Additional statistics are available, including last exception thrown by the application, which helps in diagnosing application failures.
- A new API extension enables non-transactional applications to force message redelivery without also forcing the MDB instance to be destroyed and recreated.
- A new performance extension supports processing multiple messages under a single transaction, rather than one message per transaction.
- Load-balancing capabilities are improved for MDB consumers from remote distributed destinations.

Improved EJB Caching and Pooling

Administrators can exercise more control over how EJBs are cached and pooled. For more information on all the following features, see *Programming WebLogic Server EJBs*.

Dynamic Entity Cache and EJB Pool to Match Demand

An administrator can configure the entity cache to release unused memory when demand increases, and to release pooled memory when demand decreases.

EJB Cache and Pool Initialization and Re-Initialization on Demand

This feature and the configurable option described in "Dynamic Entity Cache and EJB Pool to Match Demand" enable customers to take advance of the response time benefits that caching and pooling offer, without continuing to consume memory as load decreases.

Passivation During a Transaction

The EJB container can sustain a greater load before throwing a `CacheFullException`. In some circumstances, non-idle EJBs in cache can be passivated, so that a new request can be served.

Application developers can passivate non-idle EJB instances with the new `operationsComplete` method of `weblogic.ejb.interfaces.EJBLocalObject` This method indicates that all operations on that bean are complete for the current transaction. The container uses this information when evaluating beans for passivation.

Query Caching

Read-only entity EJBs can be cached at the query level and can be accessed in cache by any finder. This capability improves performance by avoiding database access.

Query caching is done implicitly by the EJB container; no application code is required. The behavior can be configured for a bean-level or application-level cache, using the `max-queries-in-cache` element in the `entity-cache` element of `weblogic-ejb-jar.xml`.

Concurrency Options for Optimistic Beans

Container-managed persistence (CMP) entity EJBs that use optimistic concurrency have new configuration options:

- Explicit invalidation—A developer can explicitly invalidate optimistic entity EJBs when the underlying data is updated by a program running outside the EJB container. Previously, this functionality was supported only for read-only entity EJBs.
- Time-to-live—Reduces the possibility that optimistic data will become out-dated.
- Database triggers—Maintains the version or timestamp value for versioned optimistic data. This new feature is useful in legacy environments for maintaining version information.

Automatic Retry of Rolled Back Transactions

Automatic retry of container-managed transactions can improve run-time performance in environments in which transactions are expected to roll back occasionally or regularly. Automatic transaction retry is supported for session and entity beans that use container-managed transaction demarcation.

SQL Query Support

In addition to EJB-QL queries, EJB developers can now specify SQL queries in `weblogic-cmp-jar.xml`.

Coding SQL queries is useful when the query logic cannot be expressed in EJB-QL, or if a database vendor-specific query is not supported by EJB-QL. Developers can use both EJB-QL and SQL in combination, taking advantage of EJB-QL portability benefits for most queries, and using SQL for those that cannot be accomplished with EJB-QL.

Trimming String-Type Values

The EJB container trims trailing blanks at the end of string type values in primary key and other container-managed persistence fields. Trimming string-type values in this fashion avoids comparison errors and portability issues that might otherwise arise.

Improved Operation Ordering for CMP Entities

The EJB container detects symmetric and circular relationships between container-managed persistence (CMP) entities when batching and ordering database operations.

In previous versions of WebLogic Server, batch database operations on entities in symmetrical or circular relationships could result in a foreign key constraint or non-null exception.

Web Applications, JSPs, and Servlets

The following sections describe new features in Web applications, JSPs, and servlets:

Web Applications

- The `weblogic.xml` deployment descriptor is now schema based rather than document type definition (DTD) based. See "weblogic.xml Schema" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*
- You can configure virtual hosts for Web applications. These can be configured as channel based or host based. Channel-based virtual hosting is a new feature with this release. See "Creating and Configuring Web Applications" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*
- The `<filter-dispatched-requests-enabled>` element controls whether or not filters are applied to dispatched requests. The default value is `false`. Because 2.4 servlets are backward compatible with 2.3 servlets (per the 2.4 specification), when 2.3 descriptor elements are detected by WebLogic Server, the `<filter-dispatched-requests-enabled>` element defaults to `true`. See "weblogic.xml Schema" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*

Servlet 2.4 Implementation

WebLogic Server now supports the Servlet 2.4 Specification from Sun Microsystems, which is downloadable at `http://java.sun.com/products/servlet/download.html#specs`. The following changes to WebLogic Server 9.0 support the new specification.

- The Servlet 2.3 Specification from Sun Microsystems did not specify whether filters should be applied on forwards and includes. The Servlet 2.4 specification clarifies this by introducing a new `dispatcher` element in the `web.xml` deployment descriptor. Using this `dispatcher` element, you can configure a `filter-mapping` to be applied on `REQUEST/FORWARD/INCLUDE/ERROR`. In WebLogic Server 8.1, the default was `REQUEST+FORWARD+INCLUDE`. For the old DTD-based deployment descriptors, the default value has not been changed in order to preserve backward compatibility. For the new descriptors (schema based) the default is `REQUEST`.

  See "Servlet Programming Tasks" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*

- According to the Servlet 2.4 specification, the ServletContextListener is now called prior to servlet initialization. See "Templates for Event Listening Classes" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*

- The `WEB-INF` directory is not part of the public document tree of the application. No file contained in the `WEB-INF` directory can be served directly to a client by the container. However, the contents of the `WEB-INF` directory are visible to servlet code using the `getResource` and `getResource-AsStream` method calls on the `ServletContext`, and may be exposed using the `RequestDispatcher` calls. See "Creating and Configuring Web Applications" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*

- Welcome files can now be servlets, as well as JSPs and static pages. See "Creating and Configuring JSPs" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*

- The new `disable-implicit-servlet-mappings` flag, when set to `true`, prevents the Web application container from creating implicit mappings for internal servlets. See "weblogic.xml Schema" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*

- Two new servlet request events (`ServletRequestListener` and `ServletRequestAttributeListener`) can manage state across the lifecycle of servlet requests and the methods invoked when the request events occur. See "Application Events and Event Listener Classes" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server.*

JSP 2.0 Implementation

WebLogic Server now supports the JSP 2.0 Specification from Sun Microsystems. The following changes to WebLogic Server 9.0 support the new specification.

- JSP 2.0 simplifies the creation of tag extensions. See *Programming WebLogic JSP Tag Extensions*.
- A new `SimpleTag` tag handler API exists for JSP provide a much simpler invocation protocol than do classic tag handlers. See "Implementing the Tag Handler" in *Programming WebLogic JSP Tag Extensions*.
- Any tag handler can optionally extend the `DynamicAttributes` interface to indicate that it supports dynamic attributes. In addition to implementing the `DynamicAttributes` interface, tag handlers that support dynamic attributes must declare that they do so in the Tag Library Descriptor (TLD). For more information on dynamic attributes, see "Creating a Tag Library Descriptor" in *Programming WebLogic JSP Tag Extensions*.
- The new JSP expression language (EL) is inspired by both ECMAScript and the XPath expression languages. The EL is available in attribute values for standard and custom actions and within template text. In both cases, the EL is invoked consistently by way of the construct ${expr}. For more information on the JSP EL, see "WebLogic JSP Reference" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.
- A JSP property group is a collection of properties that apply to a set of files representing JSP pages. You define these properties in one or more subelements of the `jsp-property-group` element in the `web.xml` deployment descriptor. For more information on this feature, see "Creating and Configuring JSPs" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.
- You can now configure implicit includes at the beginning and end of a JSP using the `include-prelude` and `include-coda` elements of the `jsp-property-group` respectively. For more information, see "Creating and Configuring JSPs" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.

Deprecated and Obsolete Web Applications Features

- The Servlet 2.4 Specification has deprecated `javax.servlet.SingleThreadModel` instance pools. WebLogic Server still supports `SingleThreadModel`; however, its use is not recommended. See "weblogic.xml Schema" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.
- The `<redirect-content-type>` element is an obsoleted element in this release. See "weblogic.xml Schema" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.
- The `<redirect-content>` element is obsoleted element in this release. See "weblogic.xml Schema" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.

Application Development

The following sections describe new features and functionality in application development for WebLogic Server.

J2EE Library Support for Easier Sharing of J2EE Modules

J2EE library support in WebLogic Server 9.0 makes it easier to share J2EE modules among multiple applications. A J2EE library is a J2EE module or collection of modules packaged in an Enterprise application (EAR) that is first deployed and then registered with the J2EE application container. After a library is registered with the container, you can deploy Enterprise applications that reference the library. Each deployed Enterprise application that references a J2EE library can use the library modules as if they were packaged within that particular EAR.

You can, for example, deploy and register a single Web application module as a J2EE library that is used in multiple Enterprise applications in a domain. If the Web application requires updates, you can modify the code in a single place and redeploy the J2EE library. Referencing applications can then be redeployed, if necessary, to use the latest version of the Web application.

See Creating Application Libraries and Optional Packages in *Developing Applications for WebLogic Server* for information about creating libraries and referencing libraries in J2EE Applications. See Deploying Applications in *Deploying Applications to WebLogic Server* for information about registering libraries and deploying referencing applications.

Optional Package Support for Sharing JAR Files

WebLogic Server supports optional packages as described in the J2EE 1.4 Specification, Section 8.2 Optional Package Support, with versioning described in Optional Package Versioning. Optional packages allow you to share the contents of a JAR file with multiple J2EE modules deployed either as stand-alone modules or as part of an Enterprise application. For example, third-party Web application framework classes needed by multiple Web applications can be packaged and deployed in a single JAR file, and shared among multiple applications in the domain.

In general, you use J2EE library support when you need to share one or more J2EE modules among different Enterprise applications, and use optional packages when you need to share one or more JAR files among different J2EE modules. See Creating Application Libraries and Optional Packages in *Developing Applications for WebLogic Server* and Deploying Applications in *Deploying Applications to WebLogic Server*.

Split Development Directory Support for Deployment Plans

The WebLogic split development directory environment and associated Ant tasks now support deployment plans and the new application installation root, as described in "Deploying Applications" on page 1-20.

`wldeploy` includes a new argument for specifying a deployment plan. `appc` has been updated to perform deployment descriptor validation for applications that include a deployment plan.

New Features in Medical Records Sample Application

The Avitek Medical Records sample application now implements EJB entity-relationship caching, Log4j integration, custom diagnostics logging and log access through the WebLogic Diagnostic Service, JDBC RowSets, Struts SSL, DBA authentication, security realm extension using JMX, and XMLBeans.

For more information about how these features are implemented, see the New Features topic for the Avitek Medical Records Sample Application in the *WebLogic Server Code Examples* documentation. This documentation is provided as a collection of HTML files installed in the WebLogic Server installation directory at *`WL_HOME`*`/samples/server/docs/core/index.html`

Startup and Shutdown Classes Deprecated

WebLogic Server startup and shutdown classes are deprecated in WebLogic Server 9.0, in favor of applications that respond to application lifecycle events. See Programming Application Lifecycle Events in *Developing Applications with WebLogic Server*.

Application Deployment

The following sections describe new application deployment functionality for WebLogic Server:

- "WebLogic Deployment API Implements and Extends JSR-88" on page 1-43
- "Application Configurations Deployable to Multiple Domains" on page 1-44

WebLogic Deployment API Implements and Extends JSR-88

In J2EE 1.4, the JSR-88 specification defines a standard API application configuration and deployment to a target application server environment. WebLogic Server 9.0 implements the JSR-88 Service Provider Interface (SPI) plug-in and model plug-in to comply with the J2EE 1.4 deployment specification. You can use a basic JSR-88 deployment tool with the WebLogic Server plug-ins (with no extensions) to configure, deploy, and redeploy J2EE applications and modules to WebLogic Server.

WebLogic Server's new deployment API implements and extends JSR-88 to support the advanced deployment capabilities described in the sections that follow. All WebLogic Server deployment tools, including the Administration Console, `weblogic.Deployer`, `wldeploy` Ant task, and the WebLogic Scripting Tool, are used in conjunction with the deployment API to configure, deploy, and redeploy applications in a domain. You can use the deployment API to build your own WebLogic Server deployment tools, or to integrate WebLogic Server configuration and deployment operations with an existing JSR-88-compliant tool.

See the J2EE v1.4 Documentation for more information about the JSR-88 deployment specification. For information about extensions to JSR-88 deployment, see Introduction to WebLogic Server Deployment in *Deploying WebLogic Server Applications* and WebLogic Server 9.0 API Reference.

For information about configuring applications for deployment, see Configuring Applications for Deployment in Deploying WebLogic Server Applications.

Application Configurations Deployable to Multiple Domains

The basic JSR-88 configuration process does not support migration of an application configuration from one environment to another within an organization. WebLogic Server extends JSR-88 with a command-line tool, `weblogic.Configure`, whereby you can export an application configuration into a deployment plan for deployment to multiple domains.

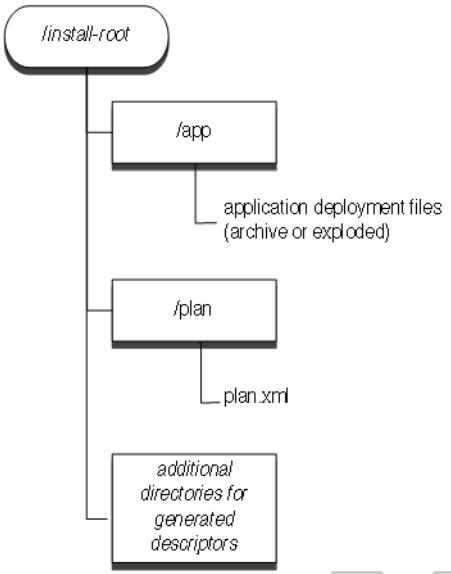
You select WebLogic Server deployment descriptor properties that you know will need to change, such as global resource names and tuning parameters, before the application is deployed into another environment. The export process creates variable definitions in the deployment plan to act as placeholders for the selected descriptor properties. Through the Administration Console, a deployer can easily assign values to those descriptor properties for different server domains, without changing the actual deployment descriptors files in the application itself.

See Exporting a Deployment Configuration in *Developing Applications for WebLogic Server* and weblogic.Configure Command Line Reference in *Developing Applications for WebLogic Server*.

New Directory Structure for Easier Production Deployment

A new application directory structure separates generated configuration files from core application files, so that configuration files can be easily changed or replaced without disturbing the application itself. Figure 1-1 shows the directory hierarchy for storing a deployable application or module.

Figure 1-1 Application Installation Root



You can easily deploy applications by specifying only the installation root. Also, the Administration Console automatically creates the above directory structure for deployments that are staged on target servers, so that deployment configuration files are available in a well-known location. BEA recommends this directory structure for all production deployments.

Production Redeployment and Maintaining Availability

You can redeploy a revised version of a production application alongside the older version of the application, without affecting existing clients to the application, and without interrupting the availability of the application to new client requests. WebLogic Server automatically manages client connections so that existing clients continue to use the older application, while new client requests are directed to the newer application. The older version is undeployed after all current clients complete their work; the Administrator explicitly undeploys the older version, or a configured timeout is reached. A rollback capability allows you to stop the redeployment process if, for example, problems are detected in the newer application version.

Production redeployment can also be used with Administration mode (see "Sanity Checking in Production with No Disruptions to Clients" on page 1-46) to isolate the new application in the production environment for testing purposes.

See Redeploying Applications in a Production Environment in *Deploying Applications to Weblogic Server*.

Version Designation to Support Production Redeployment

To support the production redeployment strategy, WebLogic Server now recognizes a unique version string entry in an Enterprise Application `MANIFEST.MF` file. When a redeployment operation is requested, WebLogic Server checks the version string to determine whether to deploy a new version of the application.

Production redeployment is performed automatically if an application supports production redeployment and its deployment configuration is updated with changes to resource bindings. This occurs even if no version string is specified in the application's manifest file. See Redeploying Applications in a Production Environment in *Deploying Applications to Weblogic Server*.

Sanity Checking in Production with No Disruptions to Clients

Using Administration mode, you can now deploy applications into a production environment (or redeploy a new version of an application, as described in "Production Redeployment and Maintaining Availability" on page 1-45) while isolating the application from external client connections. Administration mode restricts access to the application to a configured Administration channel. You can perform final ("sanity") checking of the application directly in the production environment without disrupting clients.

See Using Production Redeployment with Administration Mode in *Deploying Applications to WebLogic Server*.

JMS and JDBC Deployable Resource Configuration

JMS and JDBC configurations in WebLogic Server 9.0 are now stored in XML documents that conform to the appropriate WebLogic Server schema for the resource: `weblogic-jmsmd.xsd` or `weblogic-jdbc.xsd`. You create and manage JMS and JDBC resources either as system modules, similar to the way they were managed prior to version 9.0, or as application modules, similar to standard J2EE modules.

A JMS or JDBC application module can be deployed as a stand-alone resource, in which case the resource is available to the servers or cluster targeted during the deployment process, or as part of an

Enterprise application. An application module deployed as part of an Enterprise Application is available only to the enclosing application (an *application-scoped resource*). Using application-scoped resources ensures that an application always has access to required resources, and simplifies the process of deploying the application into new environments.

In contrast to system modules, application modules are owned by the developer who created and packaged the module, rather than the Administrator who deploys the module. This means that the Administrator has more limited control over JDBC and JMS application modules. When deploying an application module, an Administrator can change resource properties that were specified in the module, but cannot add or delete resources. System modules are created by the Administrator via the WebLogic Administration Console, and can be changed or deleted as necessary by the Administrator.

For more information, see:

- *Configuring and Managing WebLogic JMS*
- *Configuring and Managing WebLogic JDBC*
- Deploying Applications in *Deploying Applications to WebLogic Server*

Deployment Targets in WebLogic Server 9.0

WebLogic Server 9.0 introduces a new JMS Server deployment target that you can select when deploying JMS application modules to a domain. The following table describes all valid deployment targets and lists the types of modules that you can deploy to them.

Table 1-1 WebLogic Server 9.0 Deployment Targets

Target Type	Description	Valid Deployments
WebLogic Server	A single WebLogic Server instance, such as an Administration Server or a Managed Server	J2EE Applications J2EE modules JMS or JDBC application modules J2EE Libraries and optional packages
Cluster	A configured cluster of multiple WebLogic Server instances	J2EE Applications J2EE modules JMS or JDBC application modules J2EE Libraries and optional packages

Target Type	Description	Valid Deployments
Virtual host	A configured host name that routes requests for a particular DNS name to a WebLogic Server instance or cluster.	Web Applications
JMS server	A JMS server configured in a Weblogic Server domain.	A JMS queue, topic, or connection factory defined within a JMS application module

New Deployment Configuration Tools

WebLogic Server includes these new deployment tools:

- `weblogic.Configure`—Generates WebLogic Server deployment plans and deployment descriptors as necessary to configure an application. It also exports deployment descriptor properties to deployment plan variables using the Formal Classification for Deployment Descriptor Properties. See the weblogic.Configure Command Line Reference in *Developing Applications for WebLogic Server*.
- WebLogic Scripting Tool (WLST)— Provides commands for configuring applications and automated deployment, as well as other WebLogic Server administration tasks. See"WebLogic Scripting Tool Automates Domain Configuration Tasks" on page 1-8.

Enhanced Deployment Configuration Editing Extends JSR-88

Prior to WebLogic Server 9.0, in the Administration Console you could dynamically edit selected deployment descriptors for applications that were deployed using an exploded archive directory. Changes to the deployment configuration were persisted directly to the application's WebLogic Server deployment descriptor files.

The Administration Console in WebLogic Server 9.0 enables you to edit the WebLogic Server deployment configuration for any deployed application or module, regardless of whether the application was deployed using an archive file or an exploded archive directory. Changes to the deployment configuration are now persisted to a WebLogic Server deployment plan, leaving the original deployment descriptors untouched. Using a deployment plan preserves the original deployment files and makes use of WebLogic Server extensions to the JSR-88 deployment specification introduced in J2EE 1.4.

See Configuring Applications for Deployment in *Deploying Applications to WebLogic Server*.

Deprecated and Unsupported Deployment Features

The WebLogic Server 6.x single-phase deployment protocol, deprecated in versions 7.x and 8.x, is no longer supported in WebLogic Server 9.0. All deployments now use the two-phase deployment protocol.

The `weblogic.management.runtime.DeployerRuntimeMBean` API is deprecated in this release. Use the new `weblogic.deploy.api` packages for all application configuration and deployment tasks.

The use of alternate deployment descriptors for deploying applications modules is deprecated in this release. Use JSR-88-style deployment configuration and deployment plans, instead of alternate deployment descriptors, to maintain custom configurations outside of a packaged application.

The use of the `weblogic.Deployer -redeploy` *`module-uri`* syntax for redeploying a single module in a deployment is deprecated. Instead, use production redeployment or use the `-redeploy -targets` *`module@target`* syntax.

XML

WebLogic Server 9.0 implements the following new XML standards:

- "StAX Implementation" on page 1-49
- "JAX-P 1.2 Implementation" on page 1-50
- "JAX-R 1.0 Implementation" on page 1-50

See "Deprecated XML Features" on page 1-50 for information about the deprecated XML features in this release.

See "Changed Feature: Parsing XML Documents in a Servlet" on page 1-51 for information about the changed XML feature in this release.

StAX Implementation

The *Streaming API for XML* (StAX) is a Java Community Process specification that describes a bi-directional API for reading and writing XML. StAX gives parsing control to the programmer by exposing a simple iteration-based API and an underlying stream of events.

See Using the Streaming API for XML at http://e-docs.bea.com/wls/docs90/xml/stax.html.

JAX-P 1.2 Implementation

Version 1.2 of the *Java API for XML Processing* (JAX-P) specification includes an XSLT framework based on TrAX (Transformation API for XML), plus some updates to the parsing API to support DOM Level 2 and SAX version 2.0 and an improved schema to locate pluggable implementations. JAXP provides support for XML schema and an XSLT compiler (XSLTC).

See Developing XML Applications with WebLogic Server at http://e-docs.bea.com/wls/docs90/xml/programming.html.

JAX-R 1.0 Implementation

Version 1.0 of the *Java API for XML Registries* (JAX-R) specification documents a standard means of accessing different kinds of XML registries.

See Using the Java API for XML Registries at http://e-docs.bea.com/wls/docs90/xml/api.html#jaxr.

Deprecated XML Features

The following WebLogic XML features have been deprecated in this release.

WebLogic XML Streaming API

Although the WebLogic XML Streaming API is still accessible in this release of WebLogic Server, its functionality has been deprecated. Programmers should use StAX instead.

See Using the Streaming API for XML at http://e-docs.bea.com/wls/docs90/xml/stax.html.

Default XML Parser Based on Apache Xerces

The default parser in Versions 8.1and previous of WebLogic Server was one that was `based` on Apache's Xerces parser and whose package name started with `weblogic.apache.xerces.*`. In Version 9.0 of WebLogic Server, this parser has been deprecated. Instead, the default parser is the one that is shipped in JDK 5.0.

See Difference In Default Parsers Between Versions 8.1 and 9.0 of WebLogic Server at http://e-docs.bea.com/wls/docs90/xml/overview.html#deprecated_xerces.

Changed Feature: Parsing XML Documents in a Servlet

Parsing XML documents from within a servlet using the `setAttribute` and `getAttribute` methods no longer works by default. You must now first configure a WebLogic Server servlet filter to use this feature.

See Parsing XML Documents in a Servlet at http://e-docs.bea.com/wls/docs90/xml/programming.html#apps005.

Configuration Wizard

A redesigned, easy-to-use Configuration Wizard gives you more flexibility and leverage in creating, customizing, packaging, and delivering WebLogic domains.

Changed and new features in the Configuration Wizard include:

- Simplified procedure for creating and extending domains

  The Configuration Wizard has been completely redesigned. The user interface is simpler and a larger set of defaults is used so that the amount of information you need to provide is reduced.

- Silent-mode configuration is deprecated.

  Silent-mode operation of the Configuration Wizard is deprecated in WebLogic Platform 9.0. BEA recommends that you use the BEA WebLogic Scripting Tool in its place. For more information, see *WebLogic Scripting Tool*.

For complete details about the Configuration Wizard, see *Creating WebLogic Domains Using the Configuration Wizard*.

Simplified Domain Template Builder

The Domain Template Builder improves on the WebLogic Configuration Template Builder with the following new features:

- Two new commands, `pack` and `unpack`, for creating domain templates and domains, respectively, from the command line.Previously this functionality was available only through the Domain Template Builder graphical interface. These commands enhance and simplify the process of abstracting a domain. They are particularly useful in making a domain that has been customized through the Administration Console or WebLogic Scripting Tool available as a portable configuration template that can be used with a script.

  - The `pack` command creates a template archive (`.jar`) file that contains a snapshot of either an entire domain or the part of a domain needed to create a Managed Server domain directory hierarchy on a remote machine in the domain.
  - The `unpack` command creates a domain directory or Managed Server domain directory using a template created with the `pack` command.

  For more information about the `pack` and `unpack` commands, see "Pack/Unpack Command Descriptions" in *Creating Templates Using the Domain Template Builder*.

- Simplified building of templates

  In 9.0, the Domain Template Builder has been simplified and streamlined to reduce the number of steps required to build a domain template.

For more information about the Domain Template Builder, see *Creating Templates Using the Domain Template Builder*.

Installation Changes

Note the following changes to WebLogic Server installation:

- The WebLogic Web server plug-ins are no longer installed by default. In 9.0, you must choose Custom installation to install the Web server plug-ins.
- Prior to 9.0, only one instance of Node Manager could be installed on a machine. In 9.0, Node Manager is installed as a Windows service by default, and it is registered with the product installation directory in which you are installing the software; for example, `C:_bea_weblogic90b`. Node Manager can then be used to manage communication with all domains associated with that installation directory.

  Note: If you do not want Node Manager installed, perform a custom installation, and deselect Node Manager Service in the **Install Windows Service** window.

For more information about installing WebLogic Server 9.0 Beta, see the *WebLogic Platform Installation Guide*.

WebLogic Upgrade Wizard

The WebLogic Upgrade Wizard enables you to upgrade your application environment from a WebLogic Server 7.0 or 8.1 release to WebLogic Server 9.0 Beta. Specifically, you can use the WebLogic Upgrade Wizard to upgrade the following:

- **7.0 or 8.1 WebLogic Server domain** so that it runs in WebLogic Server 9.0 Beta environment.

- **Node Manager,** if used in your current environment to provide high-availability to Managed Servers.
- **Custom security provider,** if you wish to use the same provider in the new environment.

You can also use the Upgrade Wizard to perform a class compatibility inspection of the binary classes and identify WebLogic APIs that have been deprecated or removed in WebLogic Server 9.0 Beta.

For more information about the WebLogic Upgrade Wizard, see *Upgrading WebLogic Domains Using the Upgrade Wizard*.

Java Management Extensions (JMX)

Release 9.0 introduces several important changes to the WebLogic Server JMX implementation:

For more information about these changes, see "New and Changed JMX Features in This Release" in *Developing Manageable Applications with JMX*.

JMX 1.2 and JMX Remote API (JSR-160)

The WebLogic Server implementation of Java Management Extensions (JMX) is upgraded from 1.0 to 1.2. JMX 1.2 includes a new group of APIs that enable JMX components to communicate across JVMs (JSR-160).

With the introduction of these APIs, WebLogic Server deprecates its proprietary `weblogic.management.RemoteMBeanServer` interface.

Revised Model for Distributing Domain Configuration Data

All changes to a domain configuration now follow a process that resembles a transaction. The Administration Server hosts a group of Edit MBeans, which are the in-memory representation of all

pending changes to a domain's configuration. Changes in the Edit MBeans must be explicitly distributed to server instances in a domain. If any server is unable to consume a change, the entire set of changes in a distribution process is rolled back.

For more information, see "Predictable Distribution of Domain Configuration Changes" on page 1-7.

Changes to the MBean Hierarchy

WebLogic Server MBeans exist within a hierarchical data model, and as of 9.0, JMX clients must navigate this hierarchy to create, access, or destroy instances of WebLogic Server MBeans. (JMX clients can no longer create or access a child WebLogic Server MBean without first accessing the parent MBean; however, they no longer need to construct or parse WebLogic Server MBean object names.)

Instead of organizing all MBeans in a domain into a single, large hierarchy, WebLogic Server divides its MBeans into different MBean trees: MBeans that represent the current state of a server instance, MBeans that configure domain-wide services, and a tree of editable configuration MBeans.

Simplified and Secure Access to WebLogic Server MBeans

A WebLogic Server domain maintains three types of MBean servers, each of which provides access to different MBean hierarchies. The Edit MBean server provides access to the hierarchy of editable configuration MBeans; the Domain Runtime MBean server provides federated access to all runtime MBeans and read-only configuration MBeans in the domain; and the Runtime MBean server provides access only to the runtime and read-only configuration MBeans on a specific server instance.

JMX clients use the standard javax.remote.access (JSR-160) APIs to access and interact with MBeans registered in the MBean servers.

New Reference Document for Public WebLogic Server MBeans

All public WebLogic Server MBeans are described in a new document, WebLogic Server MBean Reference. For each MBean, the document describes:

- MBean factory methods and other points of access within WebLogic Server MBean trees
- Data type, read-write privileges, and other information for each attribute
- Parameters, signature, and other information for each operation

New and Deprecated MBeans and Interfaces

Many subsystems, such as logging, JMS, JDBC, and deployment, have deprecated all or part of their old JMX interfaces and replaced them with new or updated MBeans.

For details, see WebLogic Server MBean Reference.

Also, prior to 9.0, applications could access type-safe stub interfaces for WebLogic Server MBeans through the `weblogic.management.MBeanHome` interface. As of 9.0, the `MBeanHome` interface is deprecated. Instead of using this typed API layer, all JMX applications should use the standard JMX programming model. The WebLogic Server 9.0 API Reference no longer describes the `MBeanHome` interface or any of the MBeans that can be accessed through it. Instead, this deprecated access is described in a separate reference document, Type-Safe Access to WebLogic Server 9.0 MBeans.

If any of your classes import the type-safe interfaces (which are under `weblogic.management`), BEA recommends that you update to using the standard JMX programming model.

No Support for Registering Custom MBeans

WebLogic Server does not currently support registering an MBean that you have created to manage your applications or resources. WebLogic Server does not expose its `MBeanServer` interface through the JNDI tree; only the service interfaces are exposed, and these services only provide access to WebLogic Server MBeans.

BEA will enable you to register custom MBeans in a subsequent release during the Beta phase of 9.0.

J2EE Management APIs

The J2EE Management APIs enable a software developer to create a single Java program that can discover and browse resources, such as JDBC connection pools and deployed applications, on any J2EE Web application server. The APIs are part of the J2EE Management Specification, which requires all J2EE Web application servers to describe their resources in a standard data model.

WebLogic Server 9.0 implements the required features of the J2EE Management Specification (JSR-077), version 1.0.

See Monitoring and Managing with the J2EE Management APIs.

Security

The following sections describe new functionality in the WebLogic Security Service. See Understanding WebLogic Security for detailed information.

Support for Java Authorization Contract for Containers (JACC)

The Java Authorization Contract for Containers (JACC) Standard can replace the EJB and Servlet container deployment and authorization provided by WebLogic Server.

When JACC is configured for use in a WebLogic Server domain, EJB and servlet authorization decisions are made by the classes in the JACC framework. All other authorization decisions within WebLogic Server are still determined by the WebLogic Security Framework.

Single Sign-On Capabilities

Single sign-on (SSO) requires a user to sign on to an application only once to access many different application components, even though these components may have their own authentication schemes. This release of WebLogic Server supports SSO with Security Assertion Markup Language (SAML) and Windows desktops with the Simple and Protected Negotiate (SPNEGO) protocol.

Support for Certificate Lookup and Validation

The WebLogic Security Service finds and validates X509 certificate chains for inbound 2-way SSL, outbound SSL, application code, and WebLogic Web Services. The security framework extends and completes the JDK CertPath functionality.

SSL Features

The following SSL features have been added:

- SSL configuration options for network channels — You can specify identity certificates, private key information, and one- and two-way SSL options for individual channels. In previous releases, network channels used the SSL attributes defined for the SSL port of the server.

- Dynamic SSL attributes for the server—Changes made to the SSL attributes for a particular server through the WebLogic Server Administration Console now take effect without rebooting the server.
- SSL server channels can now be restarted using the WebLogic Server Administration Console when changes were not made through the Console.

New WebLogic Security Providers

The following sections describe the new security providers available in this release. For more detailed information, see Understanding WebLogic Security.

Authentication Providers

- **Database Base Management System (DBMS) providers**— Access user, password, group, and group membership information in databases for authentication purposes and can be used to upgrade from the RDBMS security realm. These providers include SQL Authentication, Read-only SQL Authentication, and Custom DBMS Authentication.
- **Windows NT** — Enables the use of Windows NT users and groups for authentication purposes.

Identity Assertion Providers

- **New LDAP X509 Identity Assertion** —Receives an X509 certificate; looks up the LDAP object for the associated user; ensures that the certificate in the LDAP object matches the presented certificate; and retrieves user name from the LDAP object for the purpose of authentication.
- **Negotiate Identity Assertion** —Decodes Simple and Protected Negotiate (SPNEGO) tokens to provide SSO with the Windows desktop by obtaining Kerberos tokens; validates the Kerberos tokens; and maps Kerberos tokens to WebLogic users.

SAML Providers

The SAML providers use OpenSAML 1.0 to support SAML assertion generation and consumption.

- **SAML Identity Assertion** —Consumes and validates SAML assertion tokens and determines whether the assertion is to be trusted (using either the proof material available in the SOAP message, the client certificate, or some other configuration indicator).
- **SAML Credential Mapping** —Generates SAML assertion for users. The SAML assertion contains the name of the requesting user as well as any groups to which the user belongs.

Certificate Lookup and Validation Providers

- **WebLogic CertPath** —Completes certificate paths and validates certificates using the trusted CA configured for a particular server instance. It also checks signatures in the chain; ensures that the chain has not expired; and checks that one certificate in the chain is issued by one of the trusted CAs configured for the server. If any checks fail, the chain is not valid.
- **Certificate Registry** —Supports the Certificate lookup and validation framework. The registry allows the system administrator to explicitly configure a list of trusted CA certificates that are allowed access to the server. The Certificate Registry provides an inexpensive mechanism for performing revocation checking. An administrator revokes a certificate by removing it from the certificate registry. The registry is stored in the embedded LDAP server.

Enhancements to WebLogic Security Providers

The following enhancements have been made to the WebLogic security providers:

- You can adjust the performance characteristics of the WebLogic Authentication provider, LDAP Authentication providers, and DBMS providers by configuring options that yield a faster group lookup response; optimize group membership caches; expose the internal PrincipalValidator cache; and increase its thresholds.
- The WebLogic Auditing provider can audit data for many types of context data. A set of supported context data (such as HTTP servlet requests or EJB parameters) is defined. The WebLogic Auditing provider also supports a new management interface, `weblogic.management.security.audit.ContextHandler`, which indicates whether the provider supports auditing context elements. Custom auditing providers can also implement this interface.
- The WebLogic Credential Mapping provider stores, retrieves, and manages credentials based on an alias and credential type.
- The WebLogic Authentication provider now supports password digests. WebLogic Web Services use this functionality to support the username token profile of a Web Service, including the password digest.
- The WebLogic Authorization provider supports new default security predicates for accessing HTTP Servlet requests, HTTP Session attributes, and any element passed to the ContextHandler. In addition, new Date and Time predicates are available.

Enhancements to the Security Service Programming Interfaces (SSPIs)

The following enhancements were made to the SSPIs:

- A context handler contains additional context and container-specific information from the resource container, and provides that information to the security provider making the access or role mapping decision. Context handler support is now available for the following methods:
  - `Adjudicator.adjudicate()`
  - `LoginModule.login()`
  - `IdentityAsserter.assertIdentity()`
  - `AuditAtnEvent()`
  - `CredentialMapper.getCredentials()`
- Servlet authentication filters perform pre- and post-processing for identity assertion and authentication functions. You can use them to encapsulate recurring tasks in reusable units and transform the response from a servlet or JSP page.

WebLogic Tuxedo Connector

WebLogic Tuxedo Connector is enhanced as follows:

- Dynamic Domain Administration utility

  This dmadmin-like facility enables WebLogic Tuxedo Connector users to start and stop individual connections between WebLogic Server applications and remote Tuxedo domains. You can shut down individual connections to remote Tuxedo domains without affecting communication between a WebLogic Server application and other Tuxedo domains. You can then modify the configuration for the domains and then re-establish the connection to effect the new configuration, without affecting other connected domains.

- MBSTRING support and codeset encoding updates

  WebLogic Tuxedo Connector now supports the new Tuxedo MBSTRING buffer type as well as the MBSTRING field in FML32. MBSTRING support is provided for the conversion of the codesets that the underlying JVM can handle. This feature also provides aliasing capability to support mapping of the various names for a given encoding that are used with different platforms and standards.

Examples

The server now ships with a DBMS pluggable run-time authentication provider that replaces the custom authetication provider used by the Medical Records sample application.

All EJB examples and other examples that use EJBs are now modified to use EJBGen for EJB 2.0 code and descriptor generation.

Certifications

For the latest information about supported WebLogic Server configurations, see *Supported Configurations*.

Standards Support

The following tables provide information about J2EE and other standards supported in WebLogic Server 9.0:

- Java Standards Support: Table 1-2
- Web Services Standards Support: Table 1-3
- Other Standards Support: Table 1-4

Table 1-2 Java Standards Support

Standard	Version
J2EE	1.4, 1.3
JDKs	5.0 (aka 1.5), 1.4
J2EE Enterprise Web Services	1.1
J2EE EJB	2.1, 2.0, and 1.1
J2EE JMS	1.1, 1.0.2b
J2EE JDBC (with third-party drivers)	2.0
MS SQL jDriver	1.0
Oracle OCI jDriver	1.0 and some 2.0 features (batching)
J2EE JNDI	1.2

Table 1-2 Java Standards Support (Continued)

OTS/JTA	1.2 and 1.0.1b
J2EE Servlet	2.4, 2.3, and 2.2
J2EE JSP	2.0, 1.2, and 1.1
RMI/IIOP	1.0
JMX	1.2, 1.0
JavaMail	1.2
JAAS	1.0 Full
J2EE CA	1.5, 1.0
JCE	1.4
Java RMI	1.0
JAX-P	1.2, 1.1
JAX-RPC	1.1, 1.0
JAX-R	1.0
SOAP Attachments for Java (SAAJ)	1.2

Table 1-3 Web Services Standards Support

Standard	Version
Enterprise Web Services	1.1
SOAP	1.1
WSDL	1.1
UDDI	2.0
WS-Security	1.0

Table 1-4 Other Standards Support

Standard	Version
SSL	v3
X.509	v3
LDAP	v3
TLS	v1
HTTP	1.1
SNMP	--

Deprecated APIs

For a list of deprecated WebLogic Server classes, see http://e-docs.bea.com/wls/docs81/javadocs/deprecated-list.html.

Third-Party JAR Files

Table 1-5 lists third-party JAR files that are part of WebLogic Server:

Table 1-5 Third-Party JAR Files

JAR File	Description
J2EE, javax, and subdirectories	• com/sun/activation/ • com/sun/mail • corba idl • javax/activation • javax/connector • javax/ejb • javax/jms • javax/jts • javax/mail • javax/management • javax/net • javax/servlet • javax/transaction • javax/xml/messaging • javax/xml/soap • javax/xml/rpc • jta • jts

Table 1-5 Third-Party JAR Files (Continued)

JAR File	Description
Libraries	• Ant 1.6.2 • Antlr 2.7.1 (MageLang Institute) • Cert-J 2.0.2 from certicom • Certicom SSL 3.1.14 • Crypto-J 3.3.1 from certicom • Netscape LDAP 3.1 • Oracle Thin JDBC driver 9.2.0.3 • AdventNet SNMP 3.2 SP1 • JavaScript 1.5 from Mozilla • JCom from J-Integra • PointBase 4.3 • Octetstring 1.5
XML	• Acumen UDDI • Apache Xerces DOM

CHAPTER 2

WebLogic Server 9.0 Known Issues

The following sections describe known issues in WebLogic Server 9.0 BETA:

Administration Console Known Issues

Change Request Number	Description
CR043645	The Administration Console does not work properly if you make it the default Web application in a Managed Server environment.
CR063594	Currently, information about cached JDBC statements is not displayed on the JDBC Monitoring pages.
CR075845	Deploying a Web application requires multiple steps.Configuration actions must be performed separately from deployment actions.
CR080476	You can not set a default Web application for a server from the Administration Console.
CR082949	When you use the FullSecurityDelegation setting, an empty `<auth-constraint>` or an unchecked method in a deployment descriptor does not delegate security to the Administration Console.
CR087951	The Administration Console does not create a new `ldift` file when creating a new security realm. If a new security realm is created and then set as the default security realm, the server may not reboot because there are no Admin accounts for the new security realm.
CR089480	Modifications to the Monitoring tables are not persisted.
CR194454	The DefaultWebAppContextRoot option for Web servers is not yet implemented.
CR200229	Currently, you can not change the order of security providers using the Administration Console.
CR202076	The Start/Stop option in the Web application assistant produces a page flow error.
CR202168	The Administration Console currently recognizes EARs that are deployed as Web Services, but not individual EJB JARs or WARs.In the Deployment Control table, EJBs and Web applications with a Web Service descriptor appear twice - once with the EJB or Web application module type and once with a Web Service module type.
CR202364	On the Domain Logging page, if the limit of 100 logs is reached, additional logs are not visible.

Change Request Number	Description
CR202430	A predicate for a security policy cannot be edited through the Administration Console.
CR203422	A JMS Bridge destination requires a Connection Factory JNDI Name and a Destination JNDI Name. If the Destination JNDI Name field is not specified, all bridge destinations are considered one single physical destination, and the following message is displayed: `"Caused by: weblogic.descriptor.DescriptorValidateException: The following failures occurred: -- Messaging Bridge "Bridge-2": source and target destinations cannot be the same physical destination."`
CR204630	An EJB-style Web service can not be tested through the Administration Console. The EJB Module-->Testing tab cannot be used to test the deployment of an EJB Module (JAR file).
CR204751	Starting or restarting a Web application does not appear in the Change List in the Change Center.
CR204886	Currently, no more than 50 users or groups from an external LDAP server or database can be displayed in the Administration Console.
CR204887	Currently, you can not use wildcards in the Administration Console to restrict the search of users and groups.
CR206426	After obtaining a lock and installing an application, starting the application does not work. There are no messages in the log and the Administration Console does not report any errors.
CR207640	A server reboot is required after adding security providers to a security realm using the Administration Console; however, there is no notification of this requirement.

Change Request Number	Description
CR207995	When you upgrade a 7.x `config.xml` file with SSL enabled to this release of WebLogic Server, the Administration Console must be accessed over the non-secure port (for example, `http://localhost:7001`). If you try to use the secure port (for example, `http://localhost:7002`), errors are recorded in the server log; a page flow error occurs when accessing the home.jsp for the Adminstration Console; and the following message is displayed: `server is currently making changes.`
CR208141	The Administration Console can not be used to delete a certificate from the certificate registry.
CR208157	Using the Copy button when copying a registered certificate to a file causes an error message to be displayed in the Administration Console.
CR208168	The information displayed in the Change Center after a deployment action (install, start, stop, delete) is confusing and incorrect.
CR208683	Cannot view or edit bindables in a deployment plan through the Administration Console.

Browser Support

The Administration Console works best with Internet Explorer version 6.0 SP1. Limited testing has been done with Netscape version 7.0.

Deployment Plans

Currently, the Administration Console cannot be used to view or edit deployment plans. Use the weblogic.Configure tool as described in Exporting an Application for Deployment to New Environments in *Developing Applications with WebLogic Server*.

Planned Console Functionality Not Implemented for WebLogic Server 9.0 Beta

The following table lists the planned features that are not implemented in the Administration Console for this Beta release of WebLogic Server 9.0.

Table 2-1 Planned Console Functionality Not Implemented for WebLogic Server 9.0 Beta

WebLogic Server Subsystem	Feature Not Implemented
Administration Console	• Console extensions • Security sensitive views • Redesigned Home page • Persistent preferences • Support for WebLogic Tuxedo Connector (WTC) • Support for WebLogic jCOM • Support for Jolt • Viewing and editing of security roles and policies for the JNDI tree • Table preferences • System status (currently these areas are greyed out)
Resource Adapters	• Updated terminology • Creation of outbound connection pools and administered topics such as JMS topics and queues
Web Services	• Configuration of Web service policies • Monitoring for Web services
J2EE and Deployment	• Application libraries • Page flow and control annotation configuration • EJB timers • Dynamic specification of Automatic Retry of Transactions for EJBs • On-demand re-initialization of EJB caches and pools • Dynamic specification of idle timeouts for EJB cache • Selection of deployment plans • Deployment of multiple Web applications by using the Activate option in the Change Center of the Console • Support for current annotation configurations as well as generic bindables in a deployment plan

Table 2-1 Planned Console Functionality Not Implemented for WebLogic Server 9.0 Beta

Diagnostics	• Watch notifications • Configuration of the Instrumentation service • Server Debug service • Logging configuration for Web applications and resource adapters • Support for current annotation configurations as well as generic bindables in a deployment plan
JDBC	• JDBC data source profiling pages • JDBC debug data source • Configuration of application-scoped JDBC functionality • Configuration of security roles and policies
JMS	• Configuration of application-scoped JMS functionality • Configuration of security roles and policies • Monitoring of JMS servers
Security	• Security Assertion Markup Language (SAML) • Alias Credential Mapping provider • Ability to unlock a user in a security realm
Core	• Self-tuning overload protection • Monitoring of queues • Monitoring of work instances • Configuration and monitoring of application-scoped work management • JNDI links
Online Help	• Task help • Page help

Configuration Wizard Known Issues

Change Request Number	Description
CR208178 CR208402	If, after creating a domain, you do not exit from the Configuration Wizard, but instead navigate backward to the initial window and extend the newly created domain, the Configuration Wizard subsequently terminates with a fatal error message. **Workaround:** After you create a domain, restart the Configuration Wizard before extending that domain.

Core Server Known Issues

Change Request Number	Description
CR055352	WebLogic Server does not enforce unique names for configured components. It is possible to configure two different components of the same type that have the same name. The `ObjectName` associated with configured components must be unique. In order to ensure uniqueness, configuration MBeans that are of the same type and have the same parent must have unique names. **Workaround:** Use unique names for components you create. Consider adopting a naming convention for new components.
CR205123	A native library (`terminalio`) is used to prevent passwords typed on the command line from being echoed to the terminal. This is a requirement for production mode. In a few cases, WebLogic Server must prompt for a password before the production mode is known. In these cases, production mode is assumed to be enabled and the server will not boot if the `terminalio` native library is not available. **Workaround:** If you are on a platform that does not support `terminalio`, then you can use the `-Dweblogic.management.allowPasswordEcho=true` option to start WebLogic Server.

Change Request Number	Description
CR206938	Large scale deployments on clusters with 10 or more nodes may cause out-of-memory errors.
CR205275	The WebLogic Scripting Tool (WLST) does not enforce serialized configuration changes. Each configuration change made using the same username is treated as part of the same edit session. As a result of this problem, large scale deployments on clusters with 10 or more nodes may cause runtime exceptions on the Administration Server. **Workaround:** If possible, avoid having two processes with the same username access an Administration Server. Use weblogic.Admin (deprecated) rather than WLST. Alternatively, when you use WLST, use the `config` command rather than the `edit` command and avoid using the `startEdit`, `save`, or `activate` commands. These two alternatives will serialize the configuration changes and avoid collisions between the changes.
CR208815	jCom has not been updated to the most recent version and has not been tested for this Beta release.
CR208161	The Node Manager logger intermittently throws an `IOException: Bad file number` (When running on Solaris only).
CR208240	The `RestartDelaySeconds` does not function correctly when using the SSH version of Node Manager.
CR194490	The Upgrade Wizard does not complete the database configuration when upgrading singleton service migration to server migration. Use the Administration Console to complete this configuration. See Server Migration.
CR185847	Specifying a server start script used by Node Manager to start a server is supported only for the Java version of Node Manager, not the shell script version of Node Manager.
CR208661	Node Manager may not be able to find the `nodemanager.domains` file. To correct this, edit the `wlscontorl.sh` file and edit the definition of `DomainDir` to point to your domain directory.

Change Request Number	Description
CR208664	The value of the `Interface` variable in `wlscontrol.sh` is not set correctly. When configuring server migration, set the Interface variable in `wlscontrol.sh` to the name of the network interfaces on your machines. The Interface variable usually looks something like: `'hme0'. Running 'ifconfig -a'`
CR206130	When you configure cross-cluster session replication, setting the Session Flush Interval and Session Flush Session attributes to very low values can cause poor performance when the cluster attempts to send the replication data to an unresponsive remote cluster.

Deployment Known Issues

Change Request Number	Description
CR071138	The `weblogic.Deployer` tool interprets any extra string values between command-line arguments as a file specification. For example, if you enter the command: `java weblogic.Deployer -activate -nostage true -name myname -source c:\myapp\mymodule` the tool attempts to activate a file specification named "true", because the `-nostage` option takes no arguments and "true" is an extraneous string value.
CR091020	If you deploy an application to a cluster and one or more clustered servers are unavailable (for example, servers partitioned from the cluster due to a network outage), the deployment operation may appear to hang. In addition, the partitioned servers may not deploy the application even after they successfully rejoin the cluster. **Workaround:** Reboot the partitioned servers after they rejoin the cluster.
CR100540	ComponentMBean names generated during deployment are inconsistent between modules within an Enterprise Application and stand-alone modules. Modules within an Enterprise Applications receive a component name with a full file extension, while stand-alone modules receive a component name without a file extension. For example, if you deploy an Enterprise Application containing an EJB named `ejb.jar`, the component is named "ejb.jar" and you can refer to the component using "ejb.jar" in subsequent deployment commands. However, if you deploy a stand-alone EJB named `ejb.jar`, the component is named "ejb" and you must use "ejb" to refer to that component in subsequent deployment commands.
CR136717	This release of `weblogic.Deployer` does not support the `-sourcerootforupload` option. (This option enables you to specify a remote root directory or archive file to upload to the Administration Server upload directory for redeployment operations.)
CR177695	The deployment API does not provide an implementation for the JSR-88 `TargetModuleID.getWebUrl()` method. `getWebUrl()` always returns null.

Change Request Number	Description
CR179465, CR202600	In this Beta release, a call to the JSR-88 `DeploymentManager.getRunningModules()` method returns a list of all configured modules, rather than just those modules that are currently active. Because of this problem, the Administration Console may show deployed modules as "Running" when in fact they were not successfully deployed.
CR203630	When you use the Administration Console to deploy multiple applications within a single change list, clicking Activate Changes deploys only the first application installed. The remaining applications are listed as "Distribute Initializing." To deploy the remaining applications, select the application names on the Deployments page and use the Start/Restart button.
CR208271	If a configuration change is made to a DTD-based application, the change is persisted to the deployment plan, as it is for schema-based applications. However, the application will not pick up the new values from the deployment plan, either dynamically, after a reboot of the server, or after a redeploy of the application. The Administration Console displays a message after a configuration save to indicate to the user that the persisted changes will not be used for DTD-based applications. The example applications installed in this Beta release use older, DTD-based WebLogic Server deployment descriptors. Configuring Applications for Production Deployment in *Deploying Applications to WebLogic Server* provides a link to an updated sample application that uses newer schema-based deployment descriptors, and that can be configured using a deployment plan. The Single Threaded Servlet Pool Size option on the Web Application configuration page in the Administration Console is deprecated and will be removed for the GA of this release. Currently, changing this option in the Administration Console has no effect on the Web application whether it is DTD or schema-based.

Diagnostics Known Issues

Change Request Number	Description
CR209137	Currently, dynamic updates to diagnostic information do not work. To modify the configured diagnostic information, edit the `weblogic-diagnostics.xml` descriptor in the application or deployment plan and redeploy the application.
CR209086	If an application contains diagnostic information defined in the `weblogic-diagnostics.xml` descriptor, the weblogic.Configure tool may fail with the following message: `Failed to parse descriptor at META-INF/weblogic-diagnostics.xml for module` **Workaround:** Specify the CLASSPATH on the weblogic.Configure command-line. When you deploy the application using the weblogic.Configure tool, the diagnostics information is modified during the initial deployment. However, subsequent updates to the plan do not take effect.

Documentation Known Issues

Change Request Number	Description
CR196210	Previous versions of WebLogic Server documentation erroneously described weblogic.management.Admin.getInstance().getAdminMBeanHome() as a means of looking up the MBeanHome interface on the Administration Server. However, the weblogic.management.Admin class is not public. Instead of using this non-public class, use JNDI to retrieve MBeanHome. See Determining the Active Domain and Servers in *Programming WebLogic Server JMX Services*.

Domain Template Builder Known Issues

Change Request Number	Description
CR206572	The **Select a Template Domain Source** window displays two templates named WebLogic Server Examples. Although both templates have the same name, they are different. The first template creates a basic WebLogic Server domain, including a Web application with product information and a set of example applications that run out-of-the-box. The second template creates the WebLogic Server Examples domain that includes the complete set of WebLogic Server sample applications and tutorials.
CR208014	When creating a template from a domain or domain template, the text in the **Add and Omit Applications** window that explains the information you need to provide may not be completely visible. **Workaround:** The following text is missing: • The field labeled **Current Application Path** should have the following descriptive text: "This is the path for the selected application as it appears in your current configuration. The application is copied into the template from this location." The application selected from the list on the left resides in the directory specified in this field, and is then copied into your template from this directory. • The field labeled **Internal Application Path** should have the following descriptive text: "When you create a WebLogic configuration using this template, this path will point to the files contained in your installation directory. These files will not be imported into the template." *Internal applications* are common WebLogic application files contained in your WebLogic Server installation directory, and are distinct from applications in the domain that you are adding to the template. Because internal applications already exist on your system, they are not copied into the template.

EJB Known Issues

Change Request Number	Description
CR202400, CR203119	Using EJBGen to generate XML Schema Definition-based (XSD) deployment descriptors is not supported in this Beta release; you can use EJBGen to generate Document Type Definition-based (DTD) deployment descriptors only. **Workaround:** Use DDConverter to convert your DTD-based descriptors to XSD-based and then, as necessary, manually edit or create descriptors in the deployment descriptor files. You can use an XML editor to edit deployment descriptor files.

Examples Known Issues

Change Request Number	Description
CR191354	Medical Records does not yet use deployment plans.
CR200319	Examples descriptor definitions need to be updated from version 8.1 and J2EE 1.3 to version 9.0 and J2EE 1.4.
CR208963	README.TXT instructs you to run setup\config to build the domain. This is no longer necessary because the Medical Records domain template does that.
CR208965	The medrec.wls.config target in SAMPLES_HOME/server/medrec/setup/build.xml has a known issue with respect to security configuration. The target's usage of wlconfig in the Ant scripting environment needs to be replaced with theWebLogic Scripting Tool command.

JDBC Known Issues

Change Request Number	Description
--	Dynamic redeployment of JDBC modules using JSR-88 is not available at Beta. If you deploy a JDBC application module either as a packaged module (within an application) or as a stand-alone module, to make changes you must undeploy the application or module, make your changes, and then redeploy the application or module.
--	In previous releases of WebLogic Server, you could create more than one data source or tx data source, each with different transaction options, and point them to a single JDBC connection pool. In WebLogic Server 9.0, each JDBC data source contains its own pool of database connections rather than pointing to a separately configured connection pool. Because of the new configuration design, you can no longer have multiple data sources that point to a single connection pool. If you require multiple data sources, each with different transaction attributes, you will need to configure multiple data sources, each with the desired transaction attributes. Because each data source has its own connection pool, your configuration may require a greater total number of JDBC database connections. If you require multiple data sources simply as multiple entry points to the connection pool (and data sources are identical except for the data source names), you can bind a single data source to the JNDI tree with multiple names. See "Configuring and Deploying WebLogic JDBC Resource" and "Binding a Data Source to the JNDI Tree with Multiple Names" in *Configuring and Managing WebLogic JDBC* for more information.

Change Request Number	Description
CR100625	Several `weblogic.Admin` JDBC commands behave differently than documented in "Commands for Managing JDBC Connection Pools" in the *WebLogic Server Command Reference*. The documentation describes the desired behavior, rather than the actual behavior. Commands with behavior other than what is documented: `SUSPEND_POOL` - This command does not take `true` or `false` parameters. Instead, it defaults to the behavior for the `false` option. Syntax: `java [`*`SSL trust options`*`] weblogic.Admin [ [-url | -adminurl] `*`URL`*`] -username `*`username`*` -password `*`password`*` SUSPEND_POOL -poolName `*`connection_pool_name`* `SHUTDOWN_POOL` - This command does not take `true` or `false` parameters. Instead, it defaults to the behavior for the `false` option. Syntax: `java [`*`SSL trust options`*`] weblogic.Admin [ [-url | -adminurl] `*`URL`*`] -username `*`username`*` -password `*`password`*` SHUTDOWN_POOL -poolName `*`connection_pool_name`*
CR188442	The Oracle Thin JDBC driver has not been certified with JDK 5.0. In internal testing, BEA has noted test failures with the DECIMAL data type (TAR 4019650.995).
CR206152	In order for multi data sources to deploy correctly on server startup, data sources listed in a multi data source must be deployed before the multi data source is deployed. Objects with the same MBean type are deployed in alphabetical order. Multi data sources and data sources are the same MBean type; therefore, multi data source names must start with a letter that is later in the alphabet than the first letter of every data source listed within the multi data source.

Change Request Number	Description
CR207566	When creating a JDBC data source through the Administration Console, you cannot save the data source configuration if the database server is not accessible, even if no target servers or clusters are selected. This problem occurs because the change management functionality in the Administration Console tests the data source when you attempt to save your changes. **Workaround:** Make sure the database is available or use the WebLogic Scripting Tool to configure the JDBC data source.

JMS Known Issues

Change Request Number	Description
CR174763	During an attempt to dynamically deploy and undeploy a JMS server that is hosting a JMS module, the module's destinations do not go away nor come back properly. **Workaround:** Redeploy every JMS module associated and let the module determine its new set of deployments.
CR189696	Upgrading a previous version of the JMS JDBC Store to the 9.0 WebLogic JDBC Store may not work if the customer is using a non-supported database version.
CR200510	JMS compatibility with previous releases does not persist a JMSdestinationMBean erroneously parented by the DomainMBean instead of the JMSServerMBean. This errors occurs because a JMSDestinationMBean that is created under the DomainMBean is not persisted anywhere. Therefore, if an error causes the configuration to be rolled back, then the JMSDestinationMBean is removed when the edit tree is compared with the runtime tree. In addition, starting and stopping the Administration Server also causes these JMSDestinationMBeans to be removed.
CR200572	A message-driven bean (MDB) cannot reach a destination configured in a JMS module that is packaged in the same EAR as the MDB unless the MDB uses the global JNDI name to find the destination.
CR202787	Dynamic attributes for SAF Imported Destinations in a JMS module cannot be changed dynamically.
CR202808	The recommended way to create JMS application modules (stand-alone or packaged) for WebLogic Server 9.0 Beta is to use the WebLogic Scripting Tool (WLST) or the Administration Console to create a JMS system module, and then copy and rename the resulting descriptor file in the *domain*`\config\jms` directory. However, when using WLST, the resulting descriptor file may not have the required `-jms.xml` suffix since it is not added by default. Descriptor files without a `-jms.xml` suffix will still work for the Beta release because they are not being validated.

Change Request Number	Description
CR205307 CR207946	A deployed Uniform Distributed Destination (UDD) cannot be dynamically undeployed using the `weblogic.deployer` utility. **Workaround:** Untarget the UDD and then restart the server.
CR206495	You cannot create a distributed destination member and its physical destination in the same edit session with the Administration Console. **Workaround:** 1. "Lock " the Administration Console. 2. Create the distributed destination. 3. Activate the distributed destination, and then lock the Console again. 4. Add the member to the distributed destination and activate it again.
CR206824	When the Administration Console is used to recreate and activate a deleted JMS destination, WebLogic Server throws the following exception: `weblogic.application.ModuleException: The Durable flag may only be set before activation` However, the reactivation of a recreated destination does succeed in the Administration Console.
CR207765	When configuring JMS Store-and-Forward (SAF) through the Administration Console, you must configure a SAF Error Handling Name for a SAF Imported Destinations. Otherwise, the Console sets the Error Handling Name value to None, which causes the activation to fail and can cause problems at server reboot. **Workaround:** Always configure Error Handling for SAF Imported Destinations and its SAF Queues and SAF Topics, even if you do not require error handling. In addition, the creation of SAF Imported Destinations or SAF Queues/Topics and the configuration of Error Handling must be activated in the same "Lock and Edit" session.
N/A	When using the Administration Console to configure SAF Error Handling in a JMS module, you can only configure either a "Principal Name" value or the "Username and Password" values. If you enter all these values, only the Principal Name value is used.

JMX Known Issues

Change Request Number	Description
CR173079 CR193100 CR193105 CR192450 CR194792 CR206474	The following public, non-deprecated APIs have been removed in either 8.1 or 9.0: • `weblogic.management.DistributedManagementException int MAX_EXCEPTIONS` • `weblogic.management.WebLogicObjectName WebLogicObjectName(java.lang.String, java.lang.String)` • In `weblogic.management.configuration.WTCServerMBean`: all add operationsk such as `addExport()` all remove operations, such as `removeExport()` `settBridgeRedirects(weblogic.management.configuration.WTCtBridgeRedirectMBean[])` `setExports(weblogic.management.configuration.WTCExportMBean[])` `setImports(weblogic.management.configuration.WTCImportMBean[])` `setLocalTuxDoms(weblogic.management.configuration.WTCLocalTuxDomMBean[])` `setPasswords(weblogic.management.configuration.WTCPasswordMBean[])` `setRemoteTuxDoms(weblogic.management.configuration.WTCRemoteTuxDomMBean[])` • In `weblogic.management.configuration.XMLRegistryMBean`: all add operations such as `addEntitySpecRegistryEntry()` all remove operations, such as `removeParserSelectRegistryEntry()` • In `JMSServerMBean`, `removeSessionPool()` and `addSessionPool()` • In `ServerRuntimeInfo`, `getJVMID()` Publication of these APIs in prior releases was erroneous. The APIs are part of BEA's internal implementation and are subject to change. In most cases, the APIs are not useful to external clients.

Change Request Number	Description
CR174782 CR185687	WebLogic Server APIs and MBean operations now return `javax.management.ObjectName` instead of `WebLogicObjectName`. The WebLogic Server 9.0 JMX implementation is based on JMX 1.2, which eliminates the ability to subclass `javax.management.ObjectName`. The class `weblogic.management.WebLogicObjectName` is a subtype of `ObjectName` and is therefore subject to this JMX restriction. **Workaround:** If you wrote JMX clients to interact with prior WebLogic Server releases, and if these clients used methods such as `RemoteMBeanServer.queryMBeans()` or `MBeanHome.getObjectName()`, update the clients so that either they do not use the deprecated `RemoteMBeanServer` or `MBeanHome` interfaces, or they do not cast the return values as `WebLogicObjectName`.
CR176977	If a JMX application runs in a WebLogic Server 8.1 server instance or runs in a separate JVM but refers to WebLogic Server 8.1 classes in its classpath, the application throws exceptions if it uses JMX to connect to a WebLogic Server 9.0 server instance. The exceptions are a result of major changes in the JMX specification between Java 5.0 and previous versions. **Workaround:** Use a JMX application running in a WebLogic Server 9.0 environment.

Jolt Known Issues

Change Request Number	Description
CR206564	Jolt session pools fail to start up in WebLogic Server 9.0 Beta.

JSP and Servlet Known Issues

Change Request Number	Description

N/A	For this Beta release, the JSP 2.0 servlet class to be used for compiling JSPs on the server side is `weblogic.servlet.JavelinxJSPServlet`. Custom patterns other than `*.jsp` or `*.jspx` should be mapped to this servlet.
	
	For example:
	`<servlet>`
	`<servlet-name>JSPServlet</servlet-name>`
	`<servlet-class>weblogic.servlet.JavelinxJSPServlet</servlet-class>`
	`</servlet>`
	`<servlet-mapping>`
	`<servlet-name>JSPServlet</servlet-name>`
	`<url-pattern>*.psp</url-pattern>`
	`</servlet-mapping>`

Change Request Number	Description

CR203749	Some JSP code may contain variants of syntax that are not commonly supported by standard JSP compilers. The WebLogic Server 9.0 release has stricter syntactic rules than prior releases, and may reject some JSP code that previously compiled but does not adhere to the JSP 2.0 specification. For example, the following JSP syntax yields compilation errors in WebLogic Server 9.0:
	`<%@ page import="`
	`import javax.naming.Context;`
	`import javax.naming.InitialContext;`
	`import weblogic.management.MBeanHome;`
	`import java.lang.System;`
	`" %>`
	To address this, an attribute in weblogic.xml has been added to enable backwards compatibility in JSPs. The syntax to enable backwards compatibility is as follows:
	`<jsp-descriptor>`
	`<jsp-param>`
	`<param-name>backward-compatible</param-name>`
	`<param-value>true</param-value>`
	`</jsp-param>`
	`</jsp-descriptor>`
	If a domain is automatically upgraded from a prior WebLogic Server release to WebLogic Server 9.0, then the backwardCompatible flag is automatically set, and there is no need to revise the weblogic.xml file. Code that does not adhere to the JSP 2.0 specification will not compile out-of-the-box.

RMI-IIOP Known Issues

Change Request Number	Description
CR113155	A CORBA client may hang if a server crashes during a the commit phase of an Object Transaction Service (OTS) managed transaction.
CR204705	WebLogic Server IIOP does not currently support the use of the JMX remote API (JSR 160) to call JMX remotely from a Java client. **Workaround:** If possible, use the WebLogic Thin client or the WebLogic Fat client.
CR206436	JDK-IIOP client performance degrades slowly as the number of client calls processed by a server increases. Eventually, data transmission becomes slow enough to cause a message timeout. It is most noticeable when extremely large objects are being transmitted. **Workaround:** If possible, use the WebLogic Thin client or the WebLogic Fat client.

Security Known Issues

Change Request Number	Description
CR200229	Currently, you can not change the order of security providers using the Administration Console.
CR204707	After you upgrade from Compatibility security to this release of WebLogic Server, the users and groups from the Compatibility realm are not displayed in the Administration Console.
CR204887	Currently, you can not use wildcards in the Administration Console to restrict the search of users and groups.

Change Request Number	Description
CR205481	Prior to this release, if you did not define a credential for the domain, the startup process for a WebLogic Server instance would create one for you. In WebLogic Server 9.0, you must define a credential for the domain before you can start any servers in the domain. A credential is generated when you create a domain. A credential is also generated by the upgrade process if one has not already been defined when you upgrade a domain from 8.1 to 9.0.
CR205481	Update security clients to use the `SecurityConfigurationMBean generateCredential()` operation instead of setting the SecurityConfigurationMBean `CredentialGenerated` attribute to `true`. Prior to WebLogic Server 9.0, security clients caused WebLogic Server to generate a credential for the domain by setting the `SecurityConfigurationMBean CredentialGenerated` attribute to `true`. While this behavior is maintained (and deprecated) for backwards compatibility, BEA recommends that you update your clients to use the new technique: Use the `generateCredential()` operation to generate and set a credential. Alternatively, define your own credential and set it as the value of the `SecurityConfigurationMBean Credential` attribute.
CR205933, CR208534	Currently, the server cannot be locked down due to the following known problems: • A user account can not be unlocked using the Administration Console. • Security policies can not be set on a Server resource using the Administration Console.
CR207444	When you use the Administration Console to define security roles and policies on an individual bean in an EJB module, you cannot define credential mappings for the bean.

Change Request Number	Description
CR207766	The intermediate files that the WebLogic Server 9.0 BETA MBeanMaker generates erroneously includes two types of constructors for you to complete. One type is based on `ModelMBean` and the other type is based on `RequiredModelMBean`. The constructors that are based on `ModelMBean` will no longer be generated or supported after this Beta release. **Workaround:** When using MBeanMaker to create MBeans for custom security providers, supply content only for the constructors that are based on `RequiredModelMBean`.

Structure of MBean Tree

The structure of the Security MBean tree is undergoing change. The intent is to restructure the MBean tree to make it consistent with the other WebLogic MBeans. The restructuring focuses on the Security Realm, Provider, and User Lockout MBeans. This work is incomplete for this Beta release. The following caveats apply to use of Security MBeans:

- Any domains created with this Beta release must be re-created with later Beta releases or the GA version of WebLogic Server 9.0.
- Use the Administration Console, 8.1 weblogic.Admin scripts, or 8.1 JMX applications to configure security. Note that the weblogic.Admin scripts and JMX applications can only be used to configure 8.1 attributes carried forward to this release.
- The security providers and User Lockout Managers appear in the `config.xml` file. Do not edit the file manually to change the security configuration, because the `config.xml` file format for them will change in the GA version of WebLogic Server 9.0.

  Note: The security providers appear as `orphaned-provider` and the User Lockout Manager appears as `orphaned-user-lockout-manager` in the `config.xml` file.
- Similarly, even though the security providers and User Lockout Managers appear in the WebLogic Scripting Tool (WLST), you should not use WLST to view or configure them because the WLST syntax will change in the GA version of WebLogic Server 9.0.

  Note: The security providers appear as `OrphanedProvider` and the User Lockout Manager appears as `OrphanedUserLockoutManager` in WLST.

Security Context Element Names

The WebLogic Auditing provider has a list of known context element names it displays as `Supported` in the WebLogic Server Administration Console. You can make these names active so that the WebLogic Auditing provider can audit their values if it sees them in Audit events. The names and/or meanings of the supported context elements may change after Beta.

Encrypted Attributes

To prevent unauthorized access to sensitive data such as passwords, some attributes in configuration MBeans are encrypted. The attributes persist their values in the domain's `config.xml` file as an encrypted string. For further security, the in-memory value is stored in the form of an encrypted byte array to help reduce the risk of the password being snooped from memory.

In previous releases when you edited a `config.xml` file manually, either a clear-text password or an encrypted password could be inserted into the file. Prior to WebLogic Server 8.1 SP4, there was no way to encrypt the password. The password had to be obtained by copying and pasting it from a `config.xml` file that already had an encrypted password. When a clear-text password was inserted, the password was encrypted the first time the server was booted. In WebLogic Server 8.1 SP4, a new command-line utility `weblogic.security.Encrypt` was added which encrypted a password that could be inserted into a `config.xml` file.

In this release of WebLogic Server, the behavior has changed:

- Non-encrypted attributes are stored in the `config.xml` file by their name. For example `MyAttribute` is stored as `my-attribute`.
- The names of encrypted attributes end with `Encrypted`. For example, the JDBCConnectionPoolMBean exposes the password that is used to access the database in an attribute named `PasswordEncrypted`.This attribute is stored in the `config.xml` file as a password-encrypted.

In production mode, the password of an encrypted attribute must be encrypted. In development mode, the password of an encrypted attribute can be either encrypted or clear-text.

The `weblogic.security.Encrypt` command-line utility can still be used to encrypt the passwords. This utility is not just used for passwords in the config.xml file. It can also be used to encrypt passwords in descriptor files (for example, a JDBC or JMS descriptor) and in deployment plans.

Start Script Known Issues

Change Request Number	Description

CR205438	If you execute the `startWebLogic` script located in the WebLogic Server example domains directory (*`WL_HOME`*`/samples/domains/wl_domain`), the incorrect page may be displayed in a browser after the server has been started.
	Workaround:
	The correct script for starting the server in this domain is the `startWebLogicEx` script. This script, which is used by the Start Menu and QuickStart shortcuts, starts the server in this domain and launches a browser to display a Web page that provides access to information about using WebLogic Server 9.0 Beta. This script is unique to the examples server and does not accompany all domain templates.
	Note: With some configurations, the informational Web page is not displayed. Instead, an error window containing the following message may be displayed:
	`Unable to open http://localhost:7001/index.jsp`
	If this message is displayed, close the error window, and enter the following URL in your preferred Web browser:
	`http://localhost:7001/index.jsp`

Transactions Known Issues

Change Request Number	Description
CR209155	Migrating the Transaction Recovery Service as an individual service is not available in the Beta release. Attempts to migrate the service result in a null pointer exception. **Workaround:** Configure a migrateable server in place of migrating the Transaction Recovery Service. See "Server Migration."

Upgrade Wizard Known Issues

Change Request Number	Description
CR207305	When you upgrade the WebLogic Server samples domain to 9.0, the `SAMPLES_HOME` environment variable is set incorrectly in the `setDomainEnv` script for that domain. After the samples domain is upgraded, this environment variable points to the wrong directory. **Workaround:** After the upgrade is complete, edit the `setDomainEnv` script in the samples domain to set the correct value for the `SAMPLES_HOME` variable. The `SAMPLES_HOME` variable should be set to the directory containing the upgraded sample application.

CR208018	The value of the `JAVA_VENDOR` environment variable is not retained after an upgrade.
	Workaround:
	To ensure that the correct JVM is used in a domain after it has been upgraded, make sure that the value for the `JAVA_VENDOR` variable is set to the correct default value in the `setDomainEnv` script.
	For example, the following code excerpt in the `setDomainEnv.cmd` script sets the Sun Java 2 SDK 1.5.0 as the default JVM:
	`if "%JAVA_VENDOR%"=="BEA" (`
	`  set JAVA_HOME=%BEA_JAVA_HOME%`
	`) else (`
	`  if "%JAVA_VENDOR%"=="Sun" (`
	`set JAVA_HOME=%SUN_JAVA_HOME%`
	`  ) else (`
	`  set JAVA_VENDOR=Sun`
	`  set JAVA_HOME=C:\bea\jdk150b`
	`  )`
	`)`

WebLogic Tuxedo Connector Known Issues

Change Request Number	Description
CR204712	If CLASSPATH is not set to include VIEWFILES, the Resource part of the WebLogic Server 8.1 SP4 configuration file fails to upgrade to WebLogic Server 9.0 Beta. **Workaround:** Set CLASSPATH appropriately to include VIEW classes.

CR207896	When you upgrade the configuration from WebLogic Server 8.1 SP4 to WebLogic Server 9.0, AppPassword in the Resource section may not upgrade correctly.
	Workaround:
	Modify the `config.xml` file using a text editor and add the WebLogic Tuxedo Connector resources manually. For example:
	`<wtc-resources>`
	`<app-password>VRh4Kg36/ABR4ZHlRMAxdupRF3AiE4515k3txgkR+bs=</app-password>`
	`<app-password-iv>3M5uwvhhWIU=</app-password-iv>`
	`<tp-usr-file>d:\tests\unit-tests\CR016921\dom1\tpusr</tp-usr-file>`
	`</wtc-resources>`

CR208052	A WebLogic Tuxedo Connector service throws exceptions on startup if the TBRIDGE is configured. This is due to an ordering issue between JMS and WebLogic Tuxedo Connector deployments. As a result, the TBRIDGE component of WTC cannot be used in WebLogic Server 9.0 Beta.

Web Services and XML Known Issues

Change Request Number	Description
CR135158	Use of the `@javax.jws.OneWay` annotation with WS-Security encryption or digital signatures is not supported.

Change Request Number	Description
CR179307	The `jwsc` Ant task returns an error if the Java Web Service (JWS) file does not contain a `package` statement. The error is: `[jwsc] java.lang.IllegalArgumentException: Endpoint interface does not contain a package name:SimpleImplPortType` **Workaround:** Include a `package` statement in the JWS file.
CR188080	The WebLogic Web Services run-time does not reuse existing security headers in the SOAP response message when adding additional security information as specified by an attached WS-Policy file. Rather, the runtime incorrectly creates a completely new security header.
CR189091	Reliable messaging XML policy files are not validated.
CR191716	The WebLogic Web Services run-time does not process the security header with the right role and does not reuse already existing security headers for the role.
CR200359	Web Services of type document-literal wrapped are not supported with XMLBeans data types as input parameters or return values.
CR200545	When you encrypt a SOAP message, an error is returned if you specify a base64 transform in an Integrity assertion by setting the URI attribute of the `Policy/Integrity/Target/Transform` element in a WS-Policy file to `http://www.w3.org/2000/09/xmldsig#base64",`
CR201674	The following exception occurs when you send a `null` value to a Web Service method whose input parameter is of type `Holder`: `java.rmi.RemoteException: Parameter "ParamName" is an in-out or out param. but the holder class is not passed in.`
CR201970	The `clientgen` Ant task returns an error when run against a Web Service of type document-literal and at least one of the input parameters is of the following data type: `Image`, `MimeMultipart`, `Source`, or `DataHandler`.
CR202377	The `jwsc` Ant task generates an incorrect XML Schema when generating the XML artifacts of a Web Service of type document-literal bare that uses arrays as input parameters.
CR203290	The `rolesReferenced` attribute of the JSR-181 `@javax.jws.SecurityRoles` JWS annotation has not been implemented.
CR203758	WebLogic Web Services do not support multi-dimensional arrays as input parameters.
CR203762	WebLogic Web Services do not support arrays of user-defined data types as input parameters.

Change Request Number	Description
CR203769	The `clientgen` Ant task returns an error when run against a Web Service that has two or more methods, both of which use certain types of arrays as input parameters. The returned error is: `weblogic.wsee.tools.WsBuildException: com.bea.xml.XmlException: C:/somepath/tempdir/WEB-INF/RpcArrayBasicJavaService.wsdl:0: error: cvc-identity-constraint.4.2.2: Duplicate key 'ArrayOfbyte' for key constraint 'type@http://www.w3.org/2001/XMLSchema'` If only one method of the Web Service uses arrays, the `clientgen` Ant task completes without error. The Array types that cannot be parameters for two different methods within the same Web Service are as follows: • `Float[]` & `float[]` --> Duplicate key 'ArrayOffloat' error • `Double[]` & `double[]` --> Duplicate key 'ArrayOfdouble' error • `Byte[]` & `byte[]` --> Duplicate key 'ArrayOfbyte' error • `Boolean[]` & `boolean[]` --> Duplicate key 'ArrayOfboolean' error • `Calendar[]` & `Date[]` --> Duplicate key 'ArrayOfdateTime' error
CR203868	Arrays as input parameters are not supported for document-style WebLogic Web Services.
CR204454	When you sign a SOAP message, an error is returned if you specify an xmldsig-filter2 transform in a Confidentiality assertion by setting the URI attribute of the `Policy/Confidentiality/Target/Transform` element in a WS-Policy file to `http://www.w3.org/2000/09/xmldsig#base64".`