BEA WebLogic Server® and WebLogic Express™

Release Notes

Version 9.1
Document Revised: December 15, 2005

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

What's New in WebLogic Server 9.1

What's New in WebLogic Server 9.1

Welcome to BEA WebLogic Server 9.1. The following sections describe new and changed functionality in this release.

Note: WebLogic Server changed substantially in version 9.0, and these changes apply to later releases as well. For a detailed description of features and functionality introduced in WebLogic Server 9.0, see What's New in WebLogic Server 9.0.

Security

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

New Security Providers Support XACML 2.0 from OASIS

WebLogic Server includes two new security providers, the XACML Authorization provider and the XACML Role Mapping provider. Previous releases of WebLogic Server used an authorization provider and a role mapping provider based on a proprietary security policy language. These new XACML security providers support the eXtensible Access Control Markup Language (XACML) 2.0 standard from OASIS. These providers can import, export, persist, and execute policy expressions using all standard XACML 2.0 functions, attributes, and schema elements.

WebLogic Server domains created using WebLogic Server 9.1 include the new XACML providers by default. The new XACML providers are fully compatible with policies and roles that have been created with the WebLogic Authorization provider (DefaultAuthorizer) and WebLogic Role Mapping provider (DefaultRoleMapper). Existing WebLogic Server 9.0 domains that you upgrade to 9.1 will continue to use the originally specified authorization and role mapping providers, such as third-party partner providers or the original WebLogic Authorization and Role Mapping providers. However, you can choose to replace the WebLogic Server proprietary providers with the XACML providers, and perform bulk imports of existing policies. You can replace a third-party provider with a XACML provider, but bulk imports of existing policies may not be possible unless the third-party provider can export policies in an XACML format.

Enhanced SAML

The WebLogic Server implementation of Security Assertion Markup Language (SAML) includes several enhancements:

- You configure a server's Intersite Transfer Service, Assertion Retrieval Service, and Assertion Consumer Service at the server level, rather than in the SAML Identity Assertion provider and SAML Credential Mapping provider.

- You can now configure separate aliases and passphrases for three distinct credentials: an assertion signing key, a protocol signing key, and an SSL client identity. Each of these credentials defaults to the server's SSL identity if no separate alias is specified.
- Trust management is enhanced to require that each partner configuration specify the aliases of the certificates trusted for particular purposes – assertion signing, SAML protocol element signing, and SSL client authentication. Registered certificates are now trusted only for the specified purpose.
- The Assertion Retrieval Service now supports multiple authentication methods for destination sites (SSL client certificate, username/password), and verifies that the site requesting an assertion is the site to which the corresponding artifact was sent.
- SAML partner configuration now includes the ability to specify parameters to be appended as query parameters when redirecting, or included as POST form variables, during execution of the of SSO profiles. This feature permits users to include partner IDs as request parameters.

New and Deprecated SAML Security Providers

The SAML Credential Mapping V2 provider and SAML Identity Assertion V2 provider are new in WebLogic Server 9.1. The SAML Credential Mapping V1 provider and SAML Identity Assertion V1 provider are deprecated; you should use the V2 versions these providers. Although the version number of the providers is incremented to V2, the new SAML security providers implement the SAML 1.1 standard, as did the V1 providers.

JDBC

The following features are new to WebLogic JDBC in this release.

Updated WebLogic Type 4 JDBC Drivers

The WebLogic Type 4 JDBC drivers were updated and include:

- Certification with J2SE 5.0.
- Enhanced `ResultSet` metadata support.
- Connection retry feature enhancements.
- General performance improvements

For detailed information on changes to supported drivers, see *WebLogic Type 4 JDBC Drivers*.

Identity-Based Connection Pooling

This feature provides identity-based connection pooling for a data source. When an application requests a database connection, the WebLogic Server instance selects or creates a physical connection with requested DBMS identity based on a map of WebLogic user IDs and database IDs.

See "Identity-Based Connection Pooling" in *Configuring and Managing WebLogic JDBC*.

Improved Configuration of JDBC Data Sources in Console

The Administration Console has been improved to simplify the configuration of JDBC data sources:

- New transaction, diagnostic, and security tabs.
- Reorganization of attributes into more useful groupings.

Support for Sybase JConnect 6.0

The third-party Sybase JConnect 6.0 (JDBC 2.0) driver is installed with WebLogic Server. See "Using Third-Party JDBC Drivers with WebLogic Server" in *Configuring and Managing WebLogic JDBC*.

Diagnostics

The following features are new to the WebLogic Diagnostic Framework in this release.

Hot-Swap Support for Deployment Plans

WebLogic Server supports deployment plans, as specified in the J2EE Deployment Specification API (JSR-88). With deployment plans, you can modify an application's configuration after the application is built, without having to modify the application archives.

In release 9.0, you had to redeploy an application after making changes to its diagnostic configuration, even when using a deployment plan.

In WebLogic Server 9.1, WLDF supports a feature called "hot swap." If you enable hot swap and then deploy your application with a deployment plan, you can dynamically update diagnostic instrumentation settings without redeploying the application.

See Using Deployment Plans for Dynamically Controlling Instrumentation Configuration" in *Configuring and Using the WebLogic Diagnostic Framework*.

WLDF Console Extension

The WebLogic Diagnostic Framework (WLDF) Console Extension is an extension that you can add to the WebLogic Server Administration Console. It provides views and tools for graphically presenting ("visualizing") diagnostic data in charts and graphs. See *Using the WebLogic Diagnostic Framework Console Extension*.

EJB

The following new attributes were added to EJBGen annotations in this release:

- `clusterInvalidationDisabled`, was added to the `@Entity` EJBGen annotation. EJBGen can now generate the `cluster-invalidation-disabled` element into the `weblogic-cmp-rdbms-jar.xml` file.
- `remoteClientTimeout`, was added to the `@Entity`, `@Session`, and `@MessageDriven` EJBGen annotations. EJBGen can now generate the `remote-client-timeout` element into the `weblogic-ejb-jar.xml` file.

Messaging (JMS and SAF)

WebLogic Server 9.1 includes the following improvements in the configuration, administration, availability, and performance of WebLogic Server JMS and the Store-and-Forward service.

- "Simplified Configuration and Targeting of JMS Resources in System Modules" on page 6
- "Message Life Cycle Logging for JMS SAF Destinations" on page 6
- "Enhanced Run-time Management for Durable Subscribers and Distributed Queues" on page 6
- "Automatic, Transparent JMS Client Failover" on page 7
- "Message Prefetching Available for Synchronous Message Consumers" on page 7
- "Improved Tuning of Message Latency and Message Throughput on JMS Destinations" on page 8

Simplified Configuration and Targeting of JMS Resources in System Modules

JMS configurations in this release are stored as modules, which are defined by an XML file that conforms to the `weblogic-jmsmd.xsd schema`. When you use the Administration Console to configure JMS resources in a globally-available system module, you can choose whether to simply accept pre-selected targets for a resource type or proceed to an advanced targeting page where you can either select an existing subdeployment or create a new one. A subdeployment is a mechanism by which JMS module resources (such as queues, topics, and connection factories) are grouped and targeted to a server resource (such as JMS servers, server instances, or cluster). A module-level subdeployment management page has also been added to allow you to manage the configured subdeployments for JMS system modules.

See "Targeting JMS Modules and Subdeployment Resources" in *Configuring and Managing WebLogic JMS.*

Message Life Cycle Logging for JMS SAF Destinations

The Store-and-Forward service now includes message life cycle logging for JMS SAF destinations. The Message Life Cycle Logging feature provides an administrator with better transparency into JMS messages from a JMS server's 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 of time. It can be also be used in real-time mode while the JMS server is running, or off-line when a JMS server is down.

See Troubleshooting WebLogic Store-and-Forward in *Configuring and Managing WebLogic Store-and-Forward.*

Enhanced Run-time Management for Durable Subscribers and Distributed Queues

Message administration enhancements allow administrators to manage durable topic subscribers and distributed queues using either the Administration Console or through public runtime APIs. This functionality also enables you to view and browse *all* messages, and to manipulate *most* messages on durable subscribers and distributed queues. These message management enhancements include message browsing (for sorting), message manipulation (such as move and delete), and message import and export.

See "Managing JMS Messages" in *Configuring and Managing WebLogic JMS.*

Automatic, Transparent JMS Client Failover

The JMS client reconnect feature provides transparent failover by making JMS client objects individually and collectively refreshable on a network failure. The network connection failure could be due to transient (temporary interruption in the network connection) or non-transient (server bounce or network failure) reasons. For WebLogic Server 9.1, refreshable client objects include ConnectionFactories, Destinations, Connections, Sessions, and Producers.

For example, a JMS destination (queue or topic) looked up via JNDI is re-usable after a network failure without requiring another lookup. This capability applies to network failures between the JMS client JVM and the remote WebLogic Server instance to which it is connected as part of the JNDI lookup, and between the JMS client JVM and any remote WebLogic Server instance in the same cluster to which the client subsequently connects.

See Automatic Failover for JMS Clients in *Programming WebLogic JMS*.

Message Prefetching Available for Synchronous Message Consumers

Prior to WebLogic Server 9.1, synchronous consumers required a two-way network call for each message. This model is inefficient because the synchronous consumer cannot retrieve multiple messages, which increases use of network traffic resources because synchronous consumers continually poll the server for available messages. With an asynchronous consumer model, messages are pushed unidirectionally and are pipelined to a message listener. Moreover, asynchronous pipelining supports the aggregation of multiple messages into a single network call.

In WebLogic Server 9.1, synchronous consumers can use the same efficient behavior as asynchronous consumers by enabling the Prefetch Mode for Synchronous Consumers option on the consumer's JMS connection factory via the Administration Console or the `JMSClientParamsBean` MBean. Similar to the asynchronous message pipeline, when the Prefetch Mode for Synchronous Consumers is enabled on a connection factory, its targeted JMS server will proactively push batches of unconsumed messages to synchronous message consumers, using the connection factory's Messages Maximum per Session parameter to define the maximum number of prefetched messages per batch.

This feature may improve performance because messages are ready and waiting for synchronous consumers when the consumers are ready to process more messages, and it may also reduce network traffic by reducing synchronous calls from consumers that must otherwise continually poll for messages.

See Receiving Messages Synchronously in *Programming WebLogic JMS*.

Improved Tuning of Message Latency and Message Throughput on JMS Destinations

The Messaging Performance Preference tuning option on JMS destinations enables you to fine-tune message handling by a destination. JMS destinations include internal algorithms that attempt to automatically optimize performance by grouping messages into batches for delivery to consumers. In response to changes in message rate and other factors, these algorithms change batch size and delivery times. However, it is impossible for the algorithms to optimize performance for every messaging environment. The Messaging Performance Preference tuning option enables you to modify how these algorithms react to changes in message rate and other factors so that you can fine-tune performance for your system.

See Tuning WebLogic JMS in *Configuring and Managing WebLogic JMS*.

Window Interval for SAF Agents

SAF sending agents now have a Window Interval parameter for JMS SAF messages, which is the maximum amount of time, in milliseconds, that a SAF sending agent will wait before forwarding JMS messages in a single batch.

Window intervals take effect and improve performance only in cases where the forwarder is already able to forward messages as fast as they arrive. In this case, instead of immediately forwarding newly arrived messages, the forwarder pauses in an attempt to accumulate more messages and forward them as a batch. The resulting larger batch sizes may improve forwarding throughput and reduce overall system disk and CPU usage, but may increase message latency.

For more information about SAF agent parameters, see "Store-and-Forward Agents: Configuration: General" in the *Administration Console Online Help*.

Resource Adapters

The pages for configuring a resource adapter's Resource Adapter Bean, Outbound Connection, and Admin Object properties in the WebLogic Administration Console have been enhanced for greater ease of use.

See Configure resource adapters in the Administration Console online help.

Web Services

The following sections describe new functionality in WebLogic Web Services:

Security-Related JWS Annotations

WebLogic Web Services 9.1 include the following new security-related JWS annotations. Use these annotations in your JWS file to configure access-control security, specify that HTTPS is required when invoking a Web Service, and specify the user associated with the Web Service:

- `weblogic.jws.security.RolesAllowed`
- `weblogic.jws.security.SecurityRole`
- `weblogic.jws.security.RolesReferenced`
- `weblogic.jws.security.SecurityRoleRef`
- `weblogic.jws.security.RunAs`
- `weblogic.jws.security.UserDataConstraint`

See Configuring Access Control: Main Steps for procedural information about using these annotations and JWS Annotations Reference for reference information.

API to Access Secured WSDL and Use Proxy Server

You can now use the `weblogic.wsee.connection.transport.http.HttpTransportInfo` API in your client application to specify a username and password when you access the dynamic WSDL of a Web Service configured for basic authentication. You can also use this API to specify a proxy server when invoking a Web Service.

See Setting the Username and Password When Creating the JAX-RPC Service Object and Using a Proxy Server When Invoking a Web Service.

Associate Multiple Client-Side WS-Policy Files

You can now associate multiple client-side WS-Policy files when invoking a Web Service; previously you could associate only one.

See Using a Client-Side Security WS-Policy File.

Turn Off X509 Validation When Using SAML Holder-Of-Key

By default, the WebLogic Web Services runtime always validates the X.509 certificate specified in the `<KeyInfo>` assertion of any associated WS-Policy file. In this release, however, you can disable this validation when using SAML holder-of-key assertions.

See Disable X.509 Certificate Validation When Using SAML Holder-Of-Key Assertions.

Failonerror Attribute of wsdlc Ant Task

The `wsdlc` Ant task has a new attribute: `failonerror`.

See wsdlc.

Web Services Testing Tab in Administration Console

You can now test a deployed WebLogic Web Service using the Administration Console.

See Test a Web Service.

Deprecated Features in WebLogic Web Services

The following Web Service features are deprecated as of release 9.1 of WebLogic Server:

- Using a `com.bea.xml.XMLBeans` data type as a parameter or return type of a WebLogic Web Service.
- The following two security-related JWS annotations:
  - `weblogic.jws.security.SecurityRole`
  - `weblogic.jws.security.SecurityIdentity`

WebLogic Tuxedo Connector

The following sections describe new functionality in WebLogic Tuxedo Connector.

Application KeepAlive

A firewall may time out the WTC TDomain connection. When the firewall connection timeout occurs, communication between WTC and the Tuxedo TDomain gateway stops. However, the TCP ABORT event may not be generated by the firewall, which may produce undesirable results. Enabling the new application keepalive feature keeps alive the connection between WTC and the Tuxedo TDomain gateway even if no user-level network activities occur for a long period of time.

See "Avoiding Hung Threads" in *Configuring WebLogic Tuxedo Connector*.

VIEW32 Support in FML32 Buffers

Enhanced support in the FML32 buffer type enables you to embed the VIEW32 buffer type inside an FML32 buffer. This feature also provides more flexibility in designing an application around the FML32 buffer.

To take advantage of this feature, you must modify your applications to use embedded VIEW32 inside an FML32 buffer. However, you do not have to add new or modify existing configuration attributes.

Note: It is not necessary to change existing applications if you do not want to take advantage of this feature.

See "How to Get VIEW32 Data In and Out of FML32 Buffers" in *WebLogic Tuxedo Connector Programming Guide*.

Suspend and Resume WTC Services

Administrators can suspend and resume a service on a per WTC server basis. The ability to suspend and resume WTC services is also available dynamically for imported services.

See Controlling WebLogic Tuxedo Connector Connections and Services in *WebLogic Tuxedo Connector Administration Guide*.

Web Applications and Servlets

WebLogic Server provides two new classes that enable you to avoid hung threads when you use an HTTP servlet.

- Future Response Servlet:

  This servlet enables you to handle a servlet response and the incoming request with different threads.

  You can enable this servlet by extending weblogic.servlet.FutureResponseServlet.java.

- Abstract Asynchronous Servlet:

  This servlet also allows you to handle servlet responses using a different thread than the one use to handle the incoming request. Unlike the Future Response Servlet, this class explicitly provides more of the general framework for handling the response including thread handling.

  You can implement the Abstract Asynchronous Servlet by extending weblogic.servlet.http.AbstractAsynchServlet.java.

See "Avoiding Hung Threads" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.

Deployment Plans in the Administration Console

The deployment plan for an Enterprise Application, when viewed in the Administration Console, now includes an additional page that shows the resource dependencies for the application. Any missing or unresolved dependencies are highlighted by a red icon.

J2EE Libraries in the Administration Console

The Administration Console pages that deploy and monitor applications and J2EE libraries now allow you to:

- View references between a library and an application that references it.
- Upgrade versioned libraries. Any referencing applications are also automatically redeployed.
- View and configure the modules within a library from the perspective of the referencing application.