



# **BEA WebLogic Server<sup>®</sup> and WebLogic Express<sup>™</sup>**

## **Release Notes**



# Contents

## 1. What's New in WebLogic Server 9.2

Security .....	1-2
Support for Custom XACML Roles and Policies.....	1-2
Customizable Security Roles and Policies for WebLogic Server MBeans.....	1-3
WebLogic Server Security for Custom MBeans.....	1-3
Bulk Access Versions of Authorization, Adjudication, and Role Mapping Providers	1-4
Policy and Role Consumer SSPI .....	1-4
Administration Console Extensions.....	1-5
JDBC and JTA .....	1-6
Dynamic Multi Data Sources .....	1-6
LLR Support for Transaction Recovery Service Migration .....	1-6
Improved Data Source Controls .....	1-6
Microsoft SQL Server 2005 Support .....	1-6
Diagnostics.....	1-6
WebLogic Diagnostic Framework .....	1-6
WebLogic Diagnostic Framework Console Extension.....	1-7
Messaging .....	1-7
Client-Side Store-and-Forward for Reliable JMS Messaging .....	1-8
Automatic Failover for JMS Message Consumers .....	1-8
Unit-of-Work Message Groups.....	1-8
WebLogic Store Administration Utility .....	1-8
Improved Non-Persistent Messaging Performance with One-Way Message Sends ..	1-9

JMS Delivery Count Message Property . . . . .	1-9
Deprecation of Messaging Bridge Adapter for WebLogic Server 5.1 . . . . .	1-9
Web Services . . . . .	1-9
Implementation of the WS-SecureConversations Specification . . . . .	1-10
Callbacks . . . . .	1-10
SOAP 1.2 Support . . . . .	1-11
Web Services Invocation Using Multiple Transports . . . . .	1-11
Streaming SOAP Attachments . . . . .	1-11
Configurable Server Addresses in the Dynamic WSDL . . . . .	1-11
Web Services Testing Feature in Administration Console . . . . .	1-12
WSSecurityInfo API for Viewing Security-Related SOAP Headers . . . . .	1-12
JWS Annotation Additions and Changes . . . . .	1-12
Web Services Ant Task Changes . . . . .	1-13
Miscellaneous Web Services Changes and Additions . . . . .	1-14
Deprecated Web Services Features . . . . .	1-15
Deployment . . . . .	1-16
Filtering Classloader . . . . .	1-16
Migration and Clustering . . . . .	1-16
Server Migration . . . . .	1-17
Automatic Singleton Service Migration . . . . .	1-17
Independent Support for Leasing in a Cluster . . . . .	1-17
Job Scheduler . . . . .	1-17
Web Applications, Servlets, and JSPs . . . . .	1-17
WebApp Libraries . . . . .	1-18
JSF and JSTL Libraries . . . . .	1-18
Using Two-Way SSL with Generic Proxy Servlets . . . . .	1-18
Servlet Authentication Fallback Mechanism . . . . .	1-19
Future Response Model for HTTP Servlets . . . . .	1-19

Limiting Number of Concurrent Requests for a Session . . . . .	1-19
Deprecated Servlet Features . . . . .	1-19
Spring Framework Support . . . . .	1-19
WebLogic Tuxedo Connector . . . . .	1-20
Dynamic Configuration Support for WTC Queuing Bridge . . . . .	1-20
WTC Performance Enhancements . . . . .	1-20
weblogic.apache Classes . . . . .	1-21
Standards Support . . . . .	1-21



# What's New in WebLogic Server 9.2

**Welcome to BEA WebLogic Server 9.2.** The following sections describe new and changed functionality in this WebLogic Server™ release.

**Note:** WebLogic Server changed substantially in release 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](#).
- For a description of new and changed functionality in WebLogic Server 9.1, see [What's New in WebLogic Server 9.1](#).
- “Security” on page 1-2
- “Administration Console Extensions” on page 1-5
- “JDBC and JTA” on page 1-6
- “Diagnostics” on page 1-6
- “Messaging” on page 1-7
- “Web Services” on page 1-9
- “Deployment” on page 1-16
- “Filtering Classloader” on page 1-16
- “Migration and Clustering” on page 1-16

- [“Web Applications, Servlets, and JSPs” on page 1-17](#)
- [“Spring Framework Support” on page 1-19](#)
- [“WebLogic Tuxedo Connector” on page 1-20](#)
- [“weblogic.apache Classes” on page 1-21](#)
- [“Standards Support” on page 1-21](#)

## Security

The following features are new to WebLogic security in this release:

- [“Support for Custom XACML Roles and Policies” on page 1-2](#)
- [“Customizable Security Roles and Policies for WebLogic Server MBeans” on page 1-3](#)
- [“WebLogic Server Security for Custom MBeans” on page 1-3](#)
- [“Bulk Access Versions of Authorization, Adjudication, and Role Mapping Providers” on page 1-4](#)
- [“Policy and Role Consumer SSPI” on page 1-4](#)

## Support for Custom XACML Roles and Policies

If you need to create security roles or policies that are more complex than can be created with the WebLogic Server Administration Console, or if you are required to express your security roles and policies in a standard language, you can create roles and policies in an XML document and then use WebLogic Scripting Tool to add them to your security realm. Note the following requirements:

- Your roles and policies must be expressed in eXtensible Access Control Markup Language (XACML) 2.0.
- If your XACML document describes authorization policies, your security realm must use either the WebLogic Server XACML Authorization Provider or another provider that implements the `weblogic.management.security.authorization.PolicyStoreMBean` interface.
- If your XACML document describes role assignments, your security realm must use either the WebLogic Server XACML Role Mapping Provider or another provider that



implements the  
`weblogic.management.security.authorization.PolicyStoreMBean` interface.

For information about creating XACML roles policies and adding them to your realm, see [Using XACML Documents to Secure WebLogic Resources](#) in *Securing WebLogic Server Resource with Roles and Policies*.

## Customizable Security Roles and Policies for WebLogic Server MBeans

To secure the WebLogic Server MBeans that are registered in an MBean server, WebLogic Server provides a default set of security roles and policies. Prior to WebLogic Server 9.2, you could not modify the default policies for WebLogic Server MBeans. As of this release, you can use the WebLogic Server Administration Console to change the default access permissions. For example, you can create roles for specific applications and allow only specific roles to access the MBean instances that are associated with specific applications. See [Create JMX Policies](#) in the Administration Console Online Help.

You can also create custom roles and policies based on XACML 2.0 and use WebLogic Scripting Tool to add them to your security realm. See [“Support for Custom XACML Roles and Policies” on page 1-2](#).

## WebLogic Server Security for Custom MBeans

If you register custom MBeans in a WebLogic Server MBean server, you can use roles and policies along with the WebLogic Security Service to protect your MBeans. Note the following restrictions:

- Your MBean object name must include a “`Type=value`” key property.
- You must describe your roles and policies in an XACML 2.0 document and then use the WebLogic Scripting Tool to add the data to your realm.
- If your XACML document describes authorization policies, your security realm must use either the WebLogic Server XACML Authorization Provider or another provider that implements the  
`weblogic.management.security.authorization.PolicyStoreMBean` interface.
- If your XACML document describes role assignments, your security realm must use either the WebLogic Server XACML Role Mapping Provider or another provider that implements the  
`weblogic.management.security.authorization.PolicyStoreMBean` interface.

For information about creating XACML roles policies and adding them to your realm, see [Creating Roles and Policies for Custom MBeans](#) in *Securing WebLogic Server Resource with Roles and Policies*.

## Bulk Access Versions of Authorization, Adjudication, and Role Mapping Providers

WebLogic Server includes bulk access versions of the following Authorization, Adjudication, and Role Mapping provider SSPI interfaces:

- BulkAuthorizationProvider
- BulkAccessDecision
- BulkAdjudicationProvider
- BulkAdjudicator
- BulkRoleProvider
- BulkRoleMapper

The bulk access SSPI interfaces allow Authorization, Adjudication, and Role Mapping providers to receive multiple decision requests in one call rather than through multiple calls, typically in a 'for' loop. This feature enables provider implementations to take advantage of internal performance optimizations, such as detecting that many passed-in `Resource` objects are protected by the same policy and will generate the same decision result.

See the [Bulk Authorization Providers](#), [Bulk Adjudication Providers](#), and [Bulk Role Mapping Providers](#) sections in *Developing WebLogic Security Providers* for additional information.

## Policy and Role Consumer SSPI

WebLogic Server implements a policy consumer for JMX (MBean) default policies and WebService annotations, and a role consumer for WebService annotations. This release of WebLogic Server includes an SSPI that Authorization and Role Mapping providers can use to obtain the policy and role collections.

The `PolicyConsumer` and `RoleConsumer` SSPI is optional; only those Authorization and Role Mapping providers that implement the SSPI are called to consume a policy or role collection.

The SSPI supports both the delivery of initial policy and role collections and the delivery of updated policy and role collections.

If you want your custom Authorization provider to support the delivery of policy collections, you must implement three interfaces:

- `weblogic.security.providers.spi.PolicyConsumerFactory`
- `weblogic.security.providers.spi.PolicyConsumer`
- `weblogic.security.providers.spi.PolicyCollectionHandler`

If you want your custom Role Mapping provider to support the delivery of role collections, you must implement three interfaces:

- `weblogic.security.providers.spi.RoleConsumerFactory`
- `weblogic.security.providers.spi.RoleConsumer`
- `weblogic.security.providers.spi.RoleCollectionHandler`

See the [Policy Consumer](#) and [Role Consumer](#) sections in *Developing WebLogic Security Providers* for additional information.

## Administration Console Extensions

Prior to this release, you could use Administration Console extensions to import a set of third-party JSP tag libraries by specifying a pathname to the tag library file. For example,

```
<%@ taglib uri="/WEB-INF/bee hive-netui-tags-template.tld"
prefix="bee hive-template" %>
```

As of this release, Administration Console extensions that use these third-party JSP tag libraries from the WebLogic Server installation must use predefined, absolute URIs to specify the tag libraries. For example:

```
<%@ taglib uri="http://bee hive.apache.org/netui/tags-template-1.0"
prefix="bee hive-template" %>
```

The Administration Console `web.xml` file maps these URIs to tag libraries within the WebLogic Server installation. This mapping facility enables BEA to reorganize its installation directory without requiring you to change your JSPs.

Any Administration Console extensions that use the old pathname syntax to import Apache Struts, Apache Beehive, or the JSTL tag libraries must update all pathnames to the new URIs.

The URI for the WebLogic Server Console Extension tag library (`console-html.tld`) remains unchanged: `/WEB-INF/console-html.tld`.

See [JSP Tag Libraries](#) in *Extending the Administration Console*.

## JDBC and JTA

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

### Dynamic Multi Data Sources

This feature allows environments such as Oracle real application clusters (RAC) installations to add and remove RAC nodes and the corresponding data sources without having to untarget and redeploy the Multi Data Source. See [Multi Data Source Features](#) in *Configuring and Managing WebLogic JDBC*.

### LLR Support for Transaction Recovery Service Migration

This feature allows the Logging Last Resource (LLR) transaction optimization to use transaction recovery service migration after a failover. See [Failover Considerations for LLR](#) in *Programming WebLogic JTA*.

### Improved Data Source Controls

A new data source control enables you to start individual instances of a data source manually from the Administration Console. See [Starting a Data Source](#) in *Configuring and Managing WebLogic JDBC*.

### Microsoft SQL Server 2005 Support

The BEA WebLogic Type 4 JDBC MS SQL Server driver supports Microsoft SQL Server 2005 database management system. See [SQL Server Database Version Support](#) in *Type 4 JDBC Drivers*.

## Diagnostics

The WebLogic Diagnostics Framework (WLDF) and the WLDF Console Extension both have new features.

### WebLogic Diagnostic Framework

A new standard application-scoped monitor, HttpSessionDebug, enables you to inspect an HTTP session object.

## WebLogic Diagnostic Framework Console Extension

- **Custom Metrics.** You can define custom metrics on an MBean type. Once defined, a custom metric attribute for an MBean instance can be graphed just like other attributes. See [Working with Metrics Charts and Graphs](#) in *Using the WebLogic Diagnostic Framework Console Extension*.
- **Cut, Copy, and Paste.** You can cut, copy, and paste graphs and charts. This gives you more flexibility in how you organize graphs and charts and allows you to copy and move graphs and charts from one view to another. See [Working with All Charts and Graphs](#) in *Using the WebLogic Diagnostic Framework Console Extension*.
- **Updated Visual and Interaction Design.** The WLDF Console Extension has new icons, plus various improvements to the visual tools, such as additional functionality available in the context menus.
- **Additional Global Property Settings.** On the Global Properties tab, you can now set a default viewport size (the time interval displayed in a chart) and a default zoom percentage for zooming in or out of a chart (thereby changing the viewport size). See [Working with All Charts and Graphs](#) in *Using the WebLogic Diagnostic Framework Console Extension*.

## Messaging

WebLogic Server 9.2 includes the following improvements in WebLogic Server JMS, Messaging Bridge, and the Store-and-Forward service:

- [“Client-Side Store-and-Forward for Reliable JMS Messaging” on page 1-8](#)
- [“Automatic Failover for JMS Message Consumers” on page 1-8](#)
- [“Unit-of-Work Message Groups” on page 1-8](#)
- [“WebLogic Store Administration Utility” on page 1-8](#)
- [“Improved Non-Persistent Messaging Performance with One-Way Message Sends” on page 1-9](#)
- [“JMS Delivery Count Message Property” on page 1-9](#)
- [“Deprecation of Messaging Bridge Adapter for WebLogic Server 5.1” on page 1-9](#)

## Client-Side Store-and-Forward for Reliable JMS Messaging

The WebLogic Store-and-Forward (SAF) service introduced in release 9.0 enables WebLogic Server to reliably deliver JMS messages between server-side JMS applications distributed across WebLogic Server clusters, domains, and server instances. In release 9.2, the JMS SAF Client provides a store-and-forward mechanism whereby standalone JMS clients can reliably send messages to server-side JMS destinations, even when a JMS client cannot temporarily reach a destination (for example, due to a network connection failure). While disconnected from the server, messages sent by a JMS SAF client are stored locally on the client and are forwarded to server-side JMS destinations when the client reconnects.

See [Reliably Sending Messages Using the JMS SAF Client](#) in *Programming Stand Alone Clients*.

## Automatic Failover for JMS Message Consumers

If a server or network failure occurs, some JMS client objects will transparently fail over to use another server instance, if one is available. In release 9.1 or later, WebLogic JMS message producer applications automatically attempt to reconnect to an available server instance without any manual configuration or changes to existing client code. For release 9.2, you can use the Administration Console or WebLogic JMS APIs to configure WebLogic JMS message consumer applications to automatically reconnect to an available server instance.

See [Automatic Failover for JMS Clients](#) in *Programming WebLogic JMS*.

## Unit-of-Work Message Groups

WebLogic Server 9.0 introduced the Message Unit-of-Order feature, which groups messages by delivering messages within the unit one at a time. However, many applications need an even more restricted notion of a group. Therefore, release 9.2 introduces the Unit-of-Work feature, which enables applications to send JMS messages, identifying some of them as a group and allowing the consumer to process them as such. For example, an application can designate a set of messages that need to be delivered to a single client without interruption, so that the messages can be processed as a unit.

See [Unit-of-Work Message Groups](#) in *Programming WebLogic JMS*.

## WebLogic Store Administration Utility

The WebLogic Store administration utility allows you to troubleshoot a WebLogic Store or to extract its data. The most common use cases for store administration are compacting a file store

to reduce its size and dumping the contents of a file store or JDBC store to an XML file for troubleshooting purposes. This utility can be run from a Java command line or from WLST.

See [Administering a Persistent Store](#) in *Configuring WebLogic Server Environments*.

## Improved Non-Persistent Messaging Performance with One-Way Message Sends

You may greatly improve the performance of typical non-persistent messaging by using one-way message sends. By enabling the “One-Way Send Mode” option on your connection factory, its associated producers can send messages without internally waiting for a response from the target destination’s host JMS server. You can choose to allow queue senders and topic publishers to do one-way sends, or limit this capability to topic publishers only. You can also configure a one-way window size to determine when a two-way message is required to regulate producers before they can continue making additional one-way sends.

See [Using One-Way Message Sends For Improved Non-Persistent Message Performance](#) in *WebLogic Server Performance and Tuning*.

## JMS Delivery Count Message Property

JMSXDeliveryCount is a system generated property that specifies the number of message delivery attempts, where the first attempt is 1, the second is 2, and so on. WebLogic Server makes a best effort to persist the delivery count, so that the delivery count does not reset back to one after a server reboot. See [Message Property Fields](#) in *Programming WebLogic JMS*.

## Deprecation of Messaging Bridge Adapter for WebLogic Server 5.1

BEA WebLogic Server 5.1 is retired and is no longer supported. In this release, the Messaging Bridge adapter used to interoperate with WebLogic Server 5.1 is deprecated and will be removed in the next major release. See [WebLogic Server 5.1 End-of-Life Announcement](#) in *Supported Configurations*.

## Web Services

Web Services include new and changed features, as described in the following sections:

- [“Implementation of the WS-SecureConversations Specification” on page 1-10](#)

- [“Callbacks” on page 1-10](#)
- [“SOAP 1.2 Support” on page 1-11](#)
- [“Web Services Invocation Using Multiple Transports” on page 1-11](#)
- [“Streaming SOAP Attachments” on page 1-11](#)
- [“Configurable Server Addresses in the Dynamic WSDL” on page 1-11](#)
- [“Web Services Testing Feature in Administration Console” on page 1-12](#)
- [“WSSecurityInfo API for Viewing Security-Related SOAP Headers” on page 1-12](#)
- [“JWS Annotation Additions and Changes” on page 1-12](#)
- [“Web Services Ant Task Changes” on page 1-13](#)
- [“Miscellaneous Web Services Changes and Additions” on page 1-14](#)
- [“Deprecated Web Services Features” on page 1-15](#)

## Implementation of the WS-SecureConversations Specification

Version 9.2 of WebLogic Web Services includes an implementation of the [Web Services Secure Conversation Language](#) (WS-SecureConversations, February 2005) specification.

The WS-Security 1.0 specification, implemented as of WebLogic Server 8.1, provides the basic mechanisms on top of which secure messaging semantics can be defined for multiple message exchanges. The WS-SecureConversation specification defines extensions to allow security context establishment and sharing, and session key derivation. Potentially more efficient keys or new key material can be exchanged, thereby increasing the overall performance and security of the subsequent exchanges.

See [Configuring Message-Level Security \(Digital Signatures and Encryption\)](#), an existing section in *Programming Web Services for WebLogic Server* that has been expanded to include WS-SecureConversations information and procedures.

## Callbacks

Callbacks notify a client of your Web Service that some event has occurred. For example, you can notify a client when the results of that client's request are ready, or when the client's request cannot be fulfilled.



See [Using Callbacks to Notify Clients of Events](#) in *Programming Web Services for WebLogic Server*.

## SOAP 1.2 Support

WebLogic Web Services now support SOAP 1.2, in addition to SOAP 1.1. Use the `@weblogic.jws.Binding` JWS annotation to specify that you want the SOAP messages of your Web Service to use SOAP 1.2.

See [@weblogic.jws.Binding](#) in *Programming Web Services for WebLogic Server*.

## Web Services Invocation Using Multiple Transports

You can enable your Web Service to be invoked using a variety of transports (HTTP, HTTPS, or JMS) by specifying multiple `@WLXXXTransport` annotations in your JWS file or `<WLXXXTransport>` child elements of the `jws` Ant task. Prior to WebLogic Server 9.2, you could specify only one transport per Web Service.

See [jws](#) and [JWS Annotation Reference](#) in *Programming Web Services for WebLogic Server*.

## Streaming SOAP Attachments

Using the new `@weblogic.jws.StreamAttachments` JWS annotation, you can now specify that a Web Service use a streaming API when reading inbound SOAP messages that include attachments, rather than the default behavior in which the service reads the entire message into memory. This feature increases the performance of Web Services with particularly large SOAP messages.

See [Streaming SOAP Attachments](#) in *Programming Web Services for WebLogic Server*.

## Configurable Server Addresses in the Dynamic WSDL

Using the Administration Console, you can now configure the server address (protocol and host information) that WebLogic Server publishes in the dynamic WSDL of a deployed Web Service. The server address is the first part of the URL pointed to by the `<address>` element of a particular Web Service port in the WSDL, such as `http://myhost:7101`.

See [Configuring the Server Address Specified in the Dynamic WSDL](#) in *Programming Web Services for WebLogic Server*.

## Web Services Testing Feature in Administration Console

You can now use the Web Services Test Client, included in the WebLogic Administration Console, to test your Web Service without writing any client code.

See [Test a Web Service](#) in *Programming Web Services for WebLogic Server*.

## WSSecurityInfo API for Viewing Security-Related SOAP Headers

Version 9.2 of WebLogic Web Services includes a new API, `WSSecurityInfo`, which enables you to view the security headers of the SOAP messages. This information includes the parts of the SOAP message that were digitally signed or encrypted, and the username/password that were presented.

See [WSSecurityInfo](#) in *Programming Web Services for WebLogic Server*.

## JWS Annotation Additions and Changes

The following JWS annotations are new in this release:

- `@weblogic.jws.Binding`
- `@weblogic.jws.Callback`
- `@weblogic.jws.CallbackMethod`
- `@weblogic.jws.CallbackService`
- `@weblogic.jws.ReliabilityErrorHandler`
- `@weblogic.jws.StreamAttachments`
- `@weblogic.jws.Types`
- `@weblogic.jws.WildcardBinding`
- `@weblogic.jws.WildcardBindings`
- `@weblogic.jws.security.CallbackRolesAllowed`
- `@weblogic.jws.soap.SOAPBinding`

The existing `@weblogic.jws.Transactional` JWS annotation has a new attribute called `timeout`. The existing `@weblogic.jws.WLJmsTransport` annotation has a new attribute called `connectionFactory`.

See [JWS Annotation Reference](#) in *Programming Web Services for WebLogic Server*.

## Web Services Ant Task Changes

The WebLogic Web Services Ant tasks have been updated, mostly to support the upgrade of 8.1 WebLogic Workshop Web Services, as described in the following sections.

### jwsc

The `jwsc` Ant task, used to compile a JWS file into a deployable Web Service, has changed as follows:

- The `jwsc` task generates and packages a JWS file into a Web application WAR file. The only exception occurs if your JWS file explicitly implements `javax.ejb.SessionBean`; in this case the Ant task generates and packages the Web Service in an EJB JAR file. Prior to WebLogic Server 9.2, the `jwsc` Ant sometimes generated an EJB when processing certain types of JWS files.
- You can now use the standard Ant `<FileSet>` task to include additional files in the generated Web Service module.
- The `jwsc` Ant task has the following new child elements:
  - `<module>`—Groups two or more JWS files (specified with the `<jws>` element) into a single module, such as the Web application WAR file. If you do not group the `<jws>` elements under a `<module>`, then `jwsc` generates a separate Web application WAR file for each JWS file.
  - `<clientgen>`—Generates the client-side JAX-RPC stubs for any Web Service that is being invoked in the JWS file that the `jwsc` Ant task is compiling.
  - `<jwsfileset>`—Specifies one or more directories in which `jwsc` searches for JWS files to compile.
  - `<descriptor>`—Specifies an existing `web.xml` or `weblogic.xml` file to which the `jwsc` Ant task adds information, in place of the Ant task generating a new file.
- The main `jwsc` Ant task has the following new attributes:
  - `dotNetStyle`
  - `keepTempFiles`

See [jwsc](#) in *Programming Web Services for WebLogic Server*.

## clientgen

The `clientgen` Ant task has changed as follows:

- You can now use the standard `<sysproperty>` child element of the `java` Ant task to specify properties required by the Web Service for which you are generating client-side artifacts.
- The `clientgen` Ant task has the following new attributes:
  - `destFile`
  - `failonerror`
  - `includeGlobalTypes`

See [clientgen](#) in *Programming Web Services for WebLogic Server*.

## wsdlc

The `wsdlc` Ant task has the following new attributes:

- `explode`
- `srcPortName`
- `srcServiceName`

See [wsdlc](#) in *Programming Web Services for WebLogic Server*.

# Miscellaneous Web Services Changes and Additions

Additional changes to Web Services in this release include the following:

- The namespaces used in the WS-Policy files for reliable messaging have changed in this release. Additionally, the order in which the reliable messaging assertions are listed in the files is important; in previous releases it was not. For the correct order of assertions, namespace declarations, and examples, see [Web Service Reliable Messaging Policy Assertion Reference](#) in *Programming Web Services for WebLogic Server*.
- WebLogic Web Services now support collection data types, such as `java.util.Collection` and `java.util.List`, as operation parameters or return values. The Apache XMLBeans (in package `org.apache.xmlbeans`) data type is also supported. For the full list of supported data types, see [Data Types](#) in *Programming Web Services for WebLogic Server*.
- Production redeployment is now fully supported for WebLogic Web Services. In *Programming Web Services for WebLogic Server*, see [Client Considerations When](#)

[Redeploying a Web Service](#) for Web Services-specific information and [Updating Applications in a Production Environment](#) for general information about production redeployment.

- In the Administration Console, monitoring pages can periodically poll the monitored resource and refresh the data displayed on the page. You can start or stop polling a resource by clicking a Refresh icon. To see an example, view the Server: Monitoring: Performance page.
- The `jwsc` and `wsdlc` Ant tasks now support both the `<xsd:any>` and `<xsd:anyType>` XML Schema data types.
- The new `@weblogic.jws.ReliabilityErrorHandler` annotation enables you to add error handling to your reliable Web Service.
- The `WebserviceTimestampMBean` has a new property: `Clock Skew`.

## Deprecated Web Services Features

The following Web Services features have been deprecated in this release:

- The `@weblogic.jws.WLHttpsTransport` annotation and the `<WLHttpsTransport>` element of the `jwsc` Ant task.

Instead, you should use the `@weblogic.jws.WLHttpTransport` annotation or `<WLHttpTransport>` element because they now support both the HTTP and HTTPS protocols. If you want client applications to access the Web Service using *only* the HTTPS protocol, then you must also specify the `@weblogic.jws.security.UserDataConstraint JWS` annotation in your JWS file.

- The `Clock Precision` and `Lax Precision` properties of the `WebserviceTimestampMBean` have been deprecated. Use the new `Clock Skew` property instead.
- The `weblogic.webservice.async.KernelFeederImpl.ExecuteTask` interface has been removed from WebLogic Server.
- The `srcBindingName` attribute of the `wsdlc` Ant task. Use `srcServiceName` or `srcPortName` instead.
- The following classes and methods in the `weblogic.xml.crypto.wss` and `weblogic.xml.crypto.wss.provider` packages:
  - `weblogic.xml.crypto.wss.SecurityTokenContextHandler` class

- `java.lang.Object getCredential()` method of `weblogic.xml.crypto.wss.provider.SecurityToken`
- `java.security.Key getSecretKey()` method of `weblogic.xml.crypto.wss.provider.SecurityToken`
- `java.security.PrivateKey getPrivateKey()` method of `weblogic.xml.crypto.wss.provider.SecurityToken`
- `java.security.PublicKey getPublicKey()` method of `weblogic.xml.crypto.wss.provider.SecurityToken`
- `void setId(java.lang.String)` method of `weblogic.xml.crypto.wss.provider.SecurityToken`

## Deployment

The following deployment features were added in this release:

- Production redeployment is now fully supported for Web Services, both stateless and stateful services that use more advanced features such as conversations and reliable messaging. See [Updating Applications in a Production Environment](#) in *Deploying Applications to WebLogic Server*.
- You can specify a grace period (in seconds) for processing of RMI client requests when retiring or gracefully shutting down an application. The work manager of a server instance accepts and schedules RMI calls until the grace period expires without a receiving new RMI client request. See [Graceful Shut Down of RMI Client Request Processing](#) in *Deploying Applications to WebLogic Server*.

## Filtering Classloader

The WebLogic FilteringClassLoader enables users to configure deployment descriptors to explicitly specify packages that are always loaded from the application, rather than being loaded using the system classloader. See [Using a Filtering Class Loader](#) in *Developing Applications with WebLogic Server*.

## Migration and Clustering

The following features related to migration and clustering were added in this release:

## Server Migration

**Note:** Server Migration is not supported on all platforms. See [Server Migration](#) in *WebLogic Platform 9.2 Supported Configurations*.

This release provides server migration support for HP-UX platform configurations.

## Automatic Singleton Service Migration

Automatic singleton service migration allows the automatic health monitoring and migration of singleton services. A singleton service is a user-defined service operating within a cluster that is available on only one server at any given time.

When a migratable service fails or become unavailable for any reason (for example, because of a bug in the service code, server failure, or network failure), it is deactivated at its current location and activated on a new server.

For more information, see [Automatic Singleton Service Migration](#) in *Using WebLogic Server Clusters*.

## Independent Support for Leasing in a Cluster

A database is no longer required to store leasing information that is used during server migration. For more information, see [Server Migration](#) in *Using WebLogic Server Clusters*.

## Job Scheduler

The Job Scheduler functionality is an implementation of the `commonj.timer` API that can be used within a clustered environment. Job Schedulers allow cluster-wide timers to be aware of the other JVMs containing each server within the cluster and is therefor able to perform load balancing and failover. For information on using the Job Scheduler, see [Using the Timer API Within a Cluster](#) in *The Timer and Work Manager API (CommonJ)*.

## Web Applications, Servlets, and JSPs

Web applications, servlets, and JSPs include new and changed features, as described in the following sections:

- [“WebApp Libraries” on page 1-18](#)
- [“JSF and JSTL Libraries” on page 1-18](#)

- “Using Two-Way SSL with Generic Proxy Servlets” on page 1-18
- “Servlet Authentication Fallback Mechanism” on page 1-19
- “Future Response Model for HTTP Servlets” on page 1-19
- “Limiting Number of Concurrent Requests for a Session” on page 1-19
- “Deprecated Servlet Features” on page 1-19

## WebApp Libraries

Just as standard shared J2EE applications can be deployed to WebLogic Server as application-libraries, a standard Web application can be deployed to WebLogic Server as a webapp-library so that other Web applications can refer to these libraries. For information on referencing these WebApp libraries with your Web applications, see [Using WebApp Libraries with Web Applications](#) in *Developing Applications with WebLogic Server*.

## JSF and JSTL Libraries

Three JSF (JavaServer™ Faces) and JSTL (JSP™ Standard Tag Library) packages are bundled with WebLogic Server as WebApp libraries. These libraries can be referenced by standard Web applications that use JSF or JSTL functionality.

The following three packages are available as WebApp libraries in release 9.2:

- MyFaces JSF library – <http://myfaces.apache.org>
- Sun JSF RI library – <https://javaserverfaces.dev.java.net>
- JSTL library – <http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html>

## Using Two-Way SSL with Generic Proxy Servlets

When using generic proxy servlets, you can define the `keyStore` initialization parameters to use two-way SSL with your own identity certificate and key. For more information, refer to the following documents:

- [Proxying Requests to Another Web Server](#) in *Programming WebLogic HTTP Servlets*
- [Configuring Proxy Plug-Ins](#) in *Application Development Guide*
- [Setting Up a Proxy to a Secondary Web Server](#) in *Using Web Server Plug-Ins with WebLogic Server*



## Servlet Authentication Fallback Mechanism

The [Servlet 2.4 specification](#) allows you to define the authentication method (BASIC, FORM, and so on) to be used in a Web application. WebLogic Server 9.2 provides an `auth-method` security module that allows you to define multiple authentication methods (as a comma separated list), so the container can provide a fallback mechanism. Authentication will be attempted in the order the values are defined in the `auth-method` list.

See [Providing a Fallback Mechanism for Authentication Methods](#) in *Introduction to WebLogic Security*.

## Future Response Model for HTTP Servlets

In general, WebLogic Server processes incoming HTTP requests and the response is returned immediately to the client. Such connections are handled synchronously by the same thread; however, some HTTP requests may require longer processing time. Handling these requests synchronously causes the thread to be held, waiting until the request is processed and the response sent. To avoid this scenario, WebLogic Server provides two classes that handle HTTP requests asynchronously by de-coupling the response from the thread that handles the incoming request.

See [A Future Response Model for HTTP Servlets](#) in *Programming WebLogic HTTP Servlets*.

## Limiting Number of Concurrent Requests for a Session

The `weblogic.http.session.maxConcurrentRequest` property has been added to limit the number of concurrent requests for a session. If the number of concurrent requests for a given session exceeds the specified value, the servlet container will start rejecting requests. By default, this property is set to -1, which indicates the servlet container does not impose any restrictions.

## Deprecated Servlet Features

The `docHome` parameter for `FileServlet` has been deprecated. Use virtual directories as an alternative.

## Spring Framework Support

WebLogic Server supports the deployment and use of applications that use the following two versions of the Spring Framework:

- 1.2.8

- 2.0

For details, see [Spring Applications Reference](#).

## WebLogic Tuxedo Connector

WebLogic Tuxedo Connector includes new and changed features, as described in the following sections:

- “[Dynamic Configuration Support for WTC Queuing Bridge](#)” on page 1-20
- “[WTC Performance Enhancements](#)” on page 1-20

### Dynamic Configuration Support for WTC Queuing Bridge

The Tuxedo Queuing Bridge is a part of the WebLogic Tuxedo Connector that provides a bi-directional JMS interface for your WebLogic Server applications to communicate to Tuxedo application environments. The transfer of messaging between the environments consists of JMS- based messages containing text, Byte, or XML data streams used to invoke services on behalf of the client application.

Dynamic configuration is now supported. If WTC is activated and the Tuxedo Queuing Bridge is deactivated, the following additions and modifications are possible.

- Add WTC Queuing Bridge if there is no WTC Queuing Bridge configuration
- Modify WTC Queuing Bridge
- Delete WTC Queuing Bridge
- Add WTC Redirections
- Modify WTC Redirections
- Delete WTC Redirections

For more information about the WTC Queuing Bridge, refer to [How To Configure the Tuxedo Queuing Bridge](#) in the *WebLogic Tuxedo Administration Guide*.

### WTC Performance Enhancements

The following WTC performance enhancements are available in this release.

- WTC performance enhancements are provided through FML buffer handling to improve tpcall performance for non-transactional calls.
- One-phase optimization improves WTC transaction performance.

See [Tuning WebLogic Tuxedo Connector](#) in *WebLogic Server Performance and Tuning*.

## weblogic.apache Classes

The `weblogic.apache.xerces.*` classes have been deprecated since WebLogic Server release 9.1. Since WebLogic Server release 9.2, all classes in all `weblogic.apache.*` packages are deprecated.

Instead of using these proprietary BEA classes, use the equivalent `org.apache.*` classes, which you can download from <http://xerces.apache.org/xerces-j/>. These classes are also included in the JDK in the `com.sun.org.apache.*` packages.

## Standards Support

This release of WebLogic Server supports the following standards.

**Table 1 Java Standards Support**

Standard	Version
J2EE	1.4, 1.3
JDKs	5.0 (aka 1.5), 1.4 (clients only)
J2EE Enterprise Web Services	1.1 (JSR-921)
Web Services Metadata for the Java Platform	1.0 (JSR-181)
J2EE EJB	2.1, 2.0, and 1.1
J2EE JMS	1.1, 1.0.2b
J2EE JDBC (with third-party drivers)	3.0
MS SQL jDriver	1.0
Oracle OCI jDriver	1.0 and some 2.0 features (batching)
J2EE JNDI	1.2

**Table 1 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 2 Web Services Standards Support**

<b>Standard</b>	<b>Version</b>
J2EE Enterprise Web Services	1.1 (JSR-921)
Web Services Metadata for the Java Platform	1.0 (JSR-181)
SOAP	1.1, 1.2
WSDL	1.1
JAX-RPC	1.1

**Table 2 Web Services Standards Support (Continued)**

<b>Standard</b>	<b>Version</b>
SAAJ	1.2
WS-Security	1.0
WS-Policy	1.0
WS-PolicyAttachment	1.0
WS-Addressing	1.0
WS-ReliableMessaging	1.0
WS-Trust	1.0
WS-SecureConversation	1.0
UDDI	2.0
JAX-R	1.0

**Table 3 Other Standards**

<b>Standard</b>	<b>Version</b>
SSL	v3
X.509	v3
Security Assertion Markup Language (SAML)	1.0, 1.1
LDAP	v3
TLS	v1
HTTP	1.1
SNMP	SNMPv1 and SNMPv2

**Table 3 Other Standards**

<b>Standard</b>	<b>Version</b>
xTensible Access Control Markup Language (XACML)	2.0
Partial implementation of Core and Hierarchical Role Based Access Control (RBAC) Profile of XACML	2.0