# Oracle® Fusion Middleware

Administrator's Guide for Oracle Application Development Framework

11*g* Release 1 (11.1.1.5.0)

**E15470-05**

April 2011

ORACLE®

Oracle Fusion Middleware Administrator's Guide for Oracle Application Development Framework 11*g* Release 1 (11.1.1.5.0)

E15470-05



Primary Authors: Peter Jew, Liza Rekadze

Contributing Author: Odile Sullivan-Tarazi, Himanshu Marathe

Contributors: Lynn Munsinger, Duncan Mills, Dipankar Bajpai, Harry Hsu, Ray Maslinski, Ricky Frost

# Contents

## 3 Monitoring and Configuring ADF Applications



## 4 WLST Command Reference for ADF Applications

## Part III Administering ADF Mobile Client Applications

## 5 Deploying ADF Mobile Client Applications



## 6 Deploying and Configuring ADF Mobile Transaction Replay Service

## C ADF Mobile Transaction Replay Service Tables

# Preface

Welcome to the *Administrator's Guide for Oracle Application Development Framework*.

## Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

### Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

### Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

### Access to Oracle Support

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

## Audience

This document is intended for system administrators who need to deploy, manage, monitor, and configure the following:

- Oracle ADF applications using the Oracle Application Development Framework (Oracle ADF)

- Oracle ADF Mobile Client applications

## Related Documents

For more information, see the following documents:

*Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*

*Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*

*Oracle Fusion Middleware Administrator's Guide*

*Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*

*Oracle Fusion Middleware Mobile Browser Developer's Guide for Oracle Application Development Framework*

*Oracle Fusion Middleware Desktop Integration Developer's Guide for Oracle Application Development Framework*

*Oracle Fusion Middleware Application Security Guide*

*Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

*Oracle Fusion Middleware High Availability Guide*

*Oracle Fusion Middleware Third-Party Application Server Guide*

*Oracle JDeveloper 11g Online Help*

*Oracle JDeveloper 11g Release Notes*, included with your JDeveloper 11*g* installation, and on Oracle Technology Network

*Oracle Fusion Middleware Java API Reference for Oracle ADF Model*

*Oracle Fusion Middleware Java API Reference for Oracle ADF Controller*

*Oracle Fusion Middleware Java API Reference for Oracle ADF Lifecycle*

*Oracle Fusion Middleware Java API Reference for Oracle ADF Faces*

*Oracle Fusion Middleware JavaScript API Reference for Oracle ADF Faces*

*Oracle Fusion Middleware Java API Reference for Oracle ADF Data Visualization Components*

*Oracle Fusion Middleware Java API Reference for Oracle ADF Share*

*Oracle Fusion Middleware Java API Reference for Oracle ADF Business Components Browser*

*Oracle Fusion Middleware Java API Reference for Oracle Generic Domains*

*Oracle Fusion Middleware interMedia Domains Java API Reference for Oracle ADF Business Components*

*Oracle Fusion Middleware Java API Reference for Oracle Metadata Service (MDS)*

## Conventions

The following text conventions are used in this document:

| Convention | Meaning |
| --- | --- |
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

# Part I

# Understanding Oracle ADF

Part I contains the following chapters:

# 1

# Introduction to Oracle ADF Administration

This chapter describes the administrative tasks you can perform and the tools you can use to deploy, manage, monitor, and configure applications developed for the Oracle Application Development Framework (Oracle ADF) and Oracle ADF Mobile client (the mobile client).

This chapter includes the following sections:



## 1.1 Introducing Oracle ADF

The Oracle Application Development Framework (Oracle ADF) builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to provide a complete framework for implementing service-oriented applications. You can use this framework to provide enterprise solutions across different platforms. You can build applications that search, display, create, modify, and validate data for web, web services, desktop, or mobile interfaces.

You use Oracle JDeveloper 11*g* with Oracle ADF to develop applications with an environment that supports the full development lifecycle of design, test, and deployment. For more information about ADF development, see *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

After you have developed and tested your ADF application in test environments, you can deploy your application to production environments using the tools described in this book. You can monitor the performance of applications as they are running. You can also manage and configure properties and attributes.

## 1.2 Oracle ADF Architecture

Oracle ADF supports the industry-standard model-view-controller architecture to achieve separation of business logic, navigation, and user interface. The MVC architecture provides:

- A model layer that represents the data values
- A view layer that contains the UI components

- A controller layer that handles input and navigation
- A business service layer that encapsulates business logic

The Fusion web application technology stack components are:

- ADF Model, for accessing declarative data binding metadata
- ADF Business Components, for building business services
- ADF Faces rich client, for AJAX-enabled UI components for web applications built with JavaServer Faces (JSF)
- ADF Controller, for input processing, navigation, and reusable task flows

### 1.2.1 ADF Business Components

ADF Business Components are application objects you can use to implement service-oriented Java EE applications. You implement ADF Business Components for clients to query, insert, update, and delete business data. You can apply business rules to the Business Components to enforce proper usage. The key components of ADF Business Components are the entity object, the view object, and the application module.

An *entity object* represents a row in a database table. It uses data manipulation language (DML) operations to modify data. Entity objects are used with others to reflect relationships in the database schema.

A *view object* represents a SQL query. You use the SQL Language to query the database to obtain the results. You can also link a view object with other entity objects to create master-detail hierarchies.

An *application module* is the transactional component that allows UI components to access data. It presents a data model and methods to perform certain tasks.

### 1.2.2 ADF Model

ADF Model implements a service abstraction called *data control*. Data control uses metadata interfaces to abstract business services. This metadata is used to describe data collections, properties, methods, and types. In JDeveloper, data controls appear in the Data Controls panel. When you drag and drop attributes, collections, and methods onto a page, JDeveloper automatically creates the bindings from the page to the associated services.

### 1.2.3 ADF Controller

ADF Controller provides a navigation and state management model that works with JSF. You can create navigational flows called task flows that encapsulate a specific task sequence.

### 1.2.4 ADF Faces Rich Client

ADF Faces provides over 100 rich components that can be used out of the box to create web applications. ADF Faces components provide built-in AJAX functionality to allow requests to be sent to the server without fully rendering the page. JSF provides server-side control to reduce the dependency on JavaScript. The components support skinning, internationalization, and accessibility options.

ADF Faces has a large set of components, including tables, trees, dialogs, accordions, and a variety of layout components. It also includes ADF Data Visualization

components, which are Flash- and SVG-enabled, for displaying graphs, charts, and gauges.

## 1.3 Introducing Oracle ADF Mobile Client

Oracle ADF Mobile client lets you develop mobile applications that do not require a browser and that provide full functionality even when disconnected from the server.

The mobile client applications use Oracle Database Lite Mobile Server and Oracle ADF Mobile transaction replay service to synchronize data between back-end business data sources and the mobile device or smartphone.

For more information, see "Introduction to ADF Mobile Client" chapter of *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

After you have developed and tested your mobile client application in test environments, you can deploy your application to production environments using the tools described in this book. You can also manage and configure properties and attributes.

## 1.4 Administering Oracle ADF Applications

You can perform a variety of administration tasks on ADF applications. You can deploy ADF applications using Enterprise Manager Fusion Middleware Control, `ojdeploy` command, Ant, or scripts. For WebLogic Server, you can also use WLST commands or the WebLogic Administration Console. For IBM WebSphere Application Server, you can use wsadmin commands or the IBM WebSphere Administrative Console.

After the ADF application has been deployed, you can configure application properties using Enterprise Manager Fusion Middleware Control. You can also configure some properties using the MBean Browser to change values in the ADF MBeans. For example, you can use Enterprise Manager Fusion Middleware Control to change the URL connection or WebService connection endpoints or seed the production credentials.

When you run the application, you can monitor performance data on the application modules, application module pooling, and task flows.

## 1.5 Administering Oracle ADF Mobile Client Applications

You can perform a number of administration tasks on the mobile client applications. You can deploy applications to BlackBerry smartphones using BlackBerry Desktop Manager, BlackBerry Enterprise Server or Web server. You can also deploy to Windows Mobile devices using Microsoft ActiveSync or Microsoft Windows Mobile Device Center. For more information, see Chapter 5, "Deploying ADF Mobile Client Applications." After the mobile client application has been deployed, you can configure its properties.

Deployment and configuration of Oracle ADF Mobile transaction replay service is also performed during production. For more information, see Chapter 6, "Deploying and Configuring ADF Mobile Transaction Replay Service."

# Part II

# Administering ADF Applications

Part II contains the following chapters:

# 2

# Deploying ADF Applications

This chapter describes how to deploy Oracle ADF applications packaged as an EAR file to a target application server. It also describes how to use scripts and Ant to automate the deployment process. This chapter focuses on deploying ADF applications for production and later stage testing. For information about deploying ADF applications for development, see the *Oracle Fusion Middleware Developer's Guide for Oracle Application Development Framework*.

For deploying to third-party application servers, such as IBM WebSphere Application Server, see the *Oracle Fusion Middleware Third-Party Application Server Guide*.

This chapter includes the following sections:



## 2.1 Introduction to Deploying ADF Applications

Deployment is the process of packaging application files and artifacts and transferring them to a target application server to be run. During application development using JDeveloper, developers can test the application using the Integrated WebLogic Server that is built into the JDeveloper installation, or they can use JDeveloper to directly deploy to a standalone application server.

After the application has been developed, administrators can deploy the application to production application servers. The tools that the administrators use for production-level deployment are:

- Oracle Enterprise Manager Fusion Middleware Control
- WebLogic Scripting Tool (WLST) commands or WebSphere Application Server (wsadmin) commands
- Command scripts and Ant scripts
- Oracle WebLogic Administration Console or IBM WebSphere Administrative Console

This chapter describes the tools and methods that administrators use to deploy ADF applications. For information about deploying ADF applications for development and testing purposes using JDeveloper, see the *Oracle Fusion Middleware Developer's Guide for Oracle Application Development Framework*.

If your application uses customization, you may need to set up the MDS repository in the application server. For more information about MDS, see the *Oracle Fusion Middleware Administrator's Guide*.

---

**Note:** Developers, Test, and QA personnel may also use these tools and the methods in this chapter to deploy ADF applications to staging application servers.

---

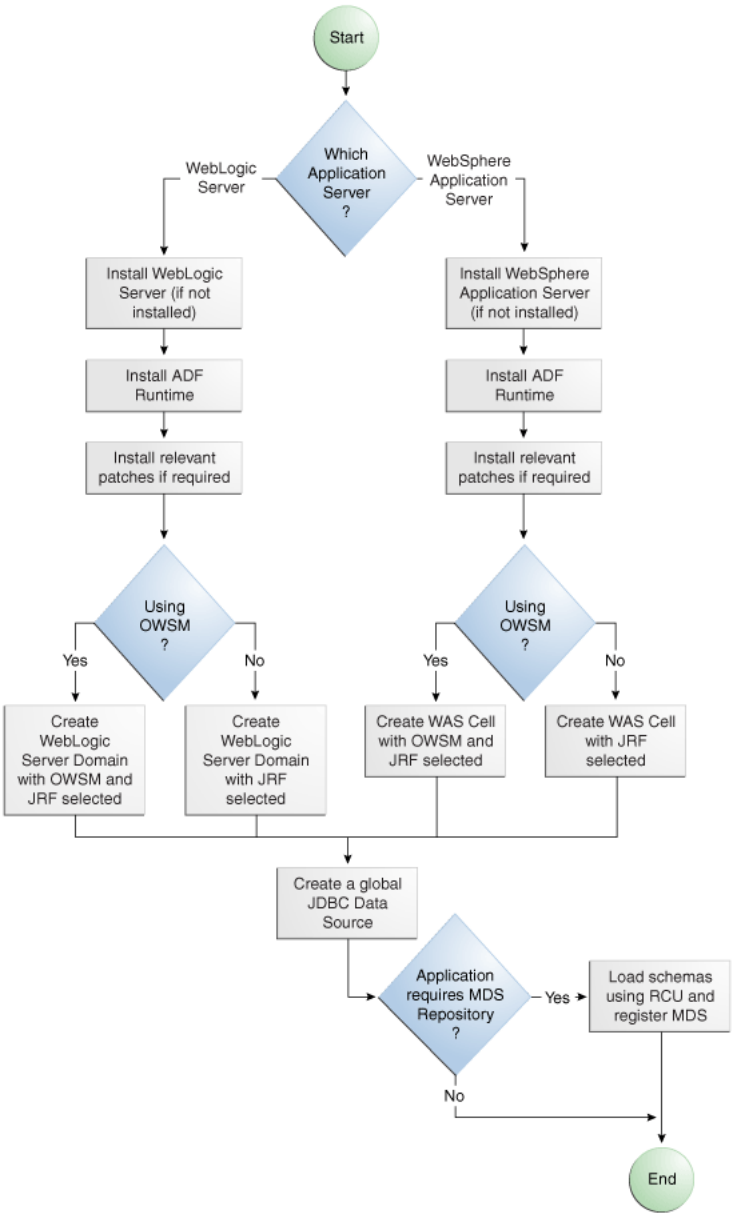
## 2.2 Preparing the Standalone Application Server for Deployment

To run ADF applications, you must install the standalone application server with the ADF runtime. You can include the ADF runtime during a new application server installation or you can install the ADF runtime into an existing application server installation.

Figure 2–1 shows the flow diagram for preparing a standalone application server for deployment. Note the following definitions used in the diagram:

- OWSM: Oracle Web Services Manager
- JRF: Java Required Files
- RCU: Repository Creation Utility
- MDS: Metadata Store

***Figure 2–1 Preparing the Application Server Flow Diagram***




For WebLogic Server, the following points apply:

- After WebLogic Server has the ADF runtime installed, you can create a new WebLogic Server domain or you can extend an existing WebLogic Server domain for Oracle ADF.
- If the Managed Servers are on a different host than the Administration Server, you must perform additional configuration tasks for the Managed Servers to enable them to host ADF applications.
- An ADF application will use either a JDBC data source or a JDBC URL to access its data. You can configure WebLogic Server with the data source using the Oracle WebLogic Server Administration Console.

For WebSphere Application Server, the following points apply:

- After WebSphere Application Server has the ADF runtime installed, you can create a new WebSphere cell or you can extend an existing WebSphere cell for ADF.
- If the servers are on a different node than the Deployment Manager, you must perform additional configuration tasks for the servers to enable them to host ADF applications.
- An ADF application will use either a JDBC data source or a JDBC URL to access its data. You can configure WebSphere Application Server with the data source using the WebSphere Administrative Console.

## 2.2.1 How to Install the ADF Runtime to the Application Server Installation

The application server requires the ADF runtime to run ADF applications.

Installing the ADF runtime is not required if you are using JDeveloper to run applications in Integrated WebLogic Server.

For WebLogic Server, you can install the ADF runtime using the following installers:

- Oracle Fusion Middleware 11*g* Application Developer Installer: Installs the ADF runtime and Oracle Enterprise Manager. You should use the Oracle Fusion Middleware 11*g* Application Developer Installer if you want to use Oracle Enterprise Manager to manage standalone ADF applications (without Oracle SOA Suite or Oracle WebCenter components). You must have already installed Oracle WebLogic Server before you can use this installer.

> **Note:** The Oracle 11*g* Installer for JDeveloper can also be used to install the ADF runtime to the application server installation. However, it does not include all the components that are typically needed for production and full test environments. Therefore, this installer should not be used for anything other than for development purposes.

For WebSphere Application Server, you can install the ADF runtime using the following installer:

- Oracle Fusion Middleware 11*g* Application Developer Installer: Installs the ADF runtime and Oracle Enterprise Manager. You must have already installed WebSphere Application Server before you can use this installer. For more information, see the *Oracle Fusion Middleware Third-Party Application Server Guide*.

### 2.2.1.1 Installing the ADF Runtime into an Existing WebLogic Server Installation Using the Oracle Fusion Middleware Application Developer Installer

You can use the Oracle Fusion Middleware 11*g* Application Developer Installer to install the ADF runtime and Enterprise Manager.

Install Oracle WebLogic Server. You must also have obtained the Oracle Fusion Middleware 11*g* Application Developer Installer.

Use the instructions in the *Oracle Fusion Middleware Installation Planning Guide* to obtain the software and start the installer,

In the installer you will perform several tasks including:

- Adding any software updates
- Selecting the WebLogic Server directory for installation

- Verifying installation information

After you have installed the ADF runtime, follow the instructions in Section 2.2.2, "How to Create and Extend Oracle WebLogic Server Domains," to use the Oracle Fusion Middleware Configuration Wizard to create or extend the Oracle WebLogic Server domain.

#### 2.2.1.2 Installing the ADF Runtime into an Existing WebSphere Application Server Installation Using the Oracle Fusion Middleware Application Developer Installer

You can use the Oracle Fusion Middleware 11*g* Application Developer Installer to install the ADF runtime and Enterprise Manager.

Before you begin, you must already have a WebSphere Application Server installation.

Use the instructions in the *Oracle Fusion Middleware Installation Planning Guide* to obtain the software and start the installer.

In the installer you will perform several tasks including:

- Adding any software updates
- Selecting the WebSphere directory for installation
- Verifying installation information

### 2.2.2 How to Create and Extend Oracle WebLogic Server Domains

You need to create and configure the Oracle WebLogic Server domain to accept ADF applications. If you do not already have a domain, you need to create one. If you already have a domain, you must extend the domain before it can run ADF applications.

If you are using Managed Servers to run your applications, you may need to configure your Managed Server. For more information about configuring a Managed Server on Oracle WebLogic Server, see *Oracle Fusion Middleware Creating Domains Using the Configuration Wizard*.

If you are setting up Managed Servers for ADF where the Managed Servers are on the same host as the Administration Server, follow the instructions described in this section.

If you are setting up to deploy to Managed Servers that are on a different host than the Administration Server, perform the additional steps described in Section 2.2.2.3, "Setting Up Remote WebLogic Managed Servers for Oracle ADF."

#### 2.2.2.1 Creating an Oracle WebLogic Server Domain for Oracle ADF

You must create an Oracle WebLogic Server domain if it does not already exist.

**To create a new Oracle WebLogic Server domain:**

1. Start the Oracle Fusion Middleware Configuration wizard as described in the "Configuring Application Developer" chapter of the *Oracle Fusion Installation Guide for Application Developer*.

   Follow the directions as described in that guide but consider the following steps.

2. In the Welcome page, select **Create a New WebLogic Domain** and click **Next**.

3. In the Select Domain Source page, select **Generate a domain configured automatically to support the following products**.

   The option **Basic WebLogic Server Domain (Required)** is already selected.

Select **Oracle JRF**. If you are using Oracle Web Services, select **Oracle WSM Policy Manager** and click **Next**.

#### 2.2.2.2 Extending the Oracle WebLogic Server Domain for Oracle ADF

**Before you begin:**

You must already have an existing Oracle WebLogic Server domain with the ADF runtime installed.

**To extend an Oracle WebLogic Server domain for ADF:**

1. Start the Oracle Fusion Middleware Configuration wizard as described in the "Configuring Application Developer" chapter of the *Oracle Fusion Installation Guide for Application Developer*.

   Follow the directions as described in that guide but consider the following steps.

2. In the Welcome page, select **Extend an existing WebLogic domain** and click **Next**.
3. In the Select a WebLogic Domain Directory page, select the location of the domain you want to configure for Oracle ADF, and click **Next**.
4. In the Select Extension Source page, select **Extend my domain automatically to support the following added products**.

   The option **Basic WebLogic Server Domain (Required)** is already selected.

   Select **Oracle JRF**. If you are using Oracle Web Services, select **Oracle WSM Policy Manager** and click **Next**.

This configures the rest of the runtime `.jar` files using the `manifest` file.

> **Note:** Your application's EAR file must have a `weblogic-application.xml` file containing a reference to the `adf.oracle.domain` shared library.

You can now start Oracle WebLogic Server by running the command-line script `ORACLE_HOME\user_projects\domains\`*`domain_name`*`\bin\startWebLogic.cmd`, and you can stop the server using the `stopWebLogic.cmd` script in the same directory. For Linux platforms, use `\bin\startWebLogic.sh` and `stopWebLogic.sh` respectively.

Access the Oracle WebLogic Server Administration Console using the URL `http://localhost:7001/console`.

#### 2.2.2.3 Setting Up Remote WebLogic Managed Servers for Oracle ADF

If the WebLogic Managed Servers are on a different host than the Administration Server, you need to perform additional steps.

You will need to set up Managed Servers for Oracle ADF on the host with the Administration Server, pack the JRF template, copy it to the remote host, and unpack the template.

**To set up remote Managed Servers for Oracle ADF:**

1. Use the Oracle Installer for JDeveloper to install Oracle WebLogic Server installations on both the local and remote hosts, if not already installed. If you are not installing JDeveloper Studio, you need to select the **Application Development**

**Framework Runtime** option in the installer. The local host is the host with the Administration Server.

Or, if there are existing Oracle WebLogic Server installations, use the Oracle Installer for JDeveloper to install the ADF runtime into the WebLogic Server installations on both hosts by selecting the **Application Development Framework Runtime** option. For more information on installation, see Section 2.2.1, "How to Install the ADF Runtime to the Application Server Installation."

2. Run the Oracle Fusion Middleware Configuration Wizard to create a new Oracle WebLogic Server domain. In the wizard, select the **Oracle JRF** option, as described in Section 2.2.2.1, "Creating an Oracle WebLogic Server Domain for Oracle ADF."
3. On the local host, run the Oracle Fusion Middleware Configuration Wizard to create Managed Servers.
4. On the local host, start the Administration Server and the Managed Server.

   For example,

   ```
   cd ORACLE_HOME/user_projects/domain/base_domain/bin
   ./startWeblogic.sh
   ./startManagedWebLogic.sh ManagedServer_1 http://localhost:7001
   ```

5. On the local host, pack the Managed Server configuration information into a JAR and then copy the JAR to the remote host. This JAR contains the JRF template information.

   For example,

   ```
   cd ORACLE_HOME/oracle_common/common/bin
   ./pack.sh -managed=true -domain=../../../user_projects/domains/base_domain
        -template=../../../base_domain_managed.jar -template_name=
        "Base Managed Server Domain"
   cp ../../../base_domain_managed.jar remote_machine_ORACLE_HOME/
   ```

6. On the remote host, unpack the Managed Server configuration JAR.

   For example,

   ```
   cd ORACLE_HOME/oracle_common/common/bin
   ./unpack.sh -domain=../../../user_projects/domains/base_domain
            -template=../../../base_domain_managed.jar
   ```

   If the Managed Server was created after the domain was, you must delete the entire domain configuration directory of the Managed Server before running `unpack`.

7. On the remote host, start the Node Manager.

   For example,

   ```
   cd ORACLE_HOME/wlserver_10.3/server/bin
   ./startNodeManager.sh
   ```

8. On the remote host, if the Managed Server was not created with the JRF template applied, run the `applyJRF` WLST command to extend the Managed Server with the JRF template.

   Also, if the Managed Server was created after the domain was, you must delete the entire domain configuration directory of the Managed Server before running `applyJRF`.

9. On the both hosts, start the Managed Servers.

   For example,

   ```
   cd ORACLE_HOME/user_projects/domains/base_domain/bin
   ./startManagedWebLogic.sh ManagedServer_2 http://<adminServerHost>:7001
   ```

### 2.2.3 How to Create a JDBC Data Source for Oracle WebLogic Server

Use the Oracle WebLogic Server Administration Console to set up a JDBC data source in the WebLogic Server instance for your applications.

**To configure Oracle WebLogic Server for a JDBC data source:**

1. Start Oracle WebLogic Server (if not already started) by choosing **Oracle Fusion Middleware > User Projects >** *Domain* **> Start Admin Server for WebLogic Server Domain** from the Windows **Start** menu.

   For Linux, log in as the root user and navigate to:

   ```
   <ORACLE_HOME>/user_projects/domains/MYSOADomain/bin
   ```

   Run the following command:

   ```
   ./startWebLogic.sh
   ```

   Or, from the Application Server Navigator, right-click an Oracle WebLogic Server instance and choose **Launch Admin Console**.

2. Start the Oracle WebLogic Server Administration Console by choosing **Oracle Fusion Middleware > User Projects >** *Domain* **> Admin Server Console** from the Windows **Start** menu.
3. Log in to the Oracle WebLogic Server Administration Console.
4. In the WebLogic Server Administration Console page, select **JDBC > Data Sources**.
5. Click **New**.
6. In the JDBC Data Source Properties page:
   - In the **Name** field, enter the name of the JDBC data source.
   - In the **JNDI** field, enter the name of the connection in the form `jdbc/`*`connection`*`DS`.
   - For the **Database Type**, select **Oracle**.
   - For the **Database Driver**, select **Oracle Driver (thin)**, and click **Next**.
7. In the Transactions Options page, accept the default options and click **Next**.
8. In the Connection Properties page:
   - For **Database Name**, enter the Oracle SID. For example, `orcl`.
   - For **Host Name**, enter the machine name of the database.
   - Enter the port number used to access the database.
   - Enter the user name and password for the database and click **Next**.
9. In the Test Database Connection page, click **Test Configuration** to test the connection.

10. In the Select Targets page, select the server for which the JDBC data source is to be deployed.
11. Click **Finish**.

Once the data source has been created in Oracle WebLogic Server, it can be used by an application module.

### 2.2.4 How to Create a JDBC Data Source for IBM WebSphere Application Server

To configure a JDBC data source for WebSphere Application Server, see the *Oracle Fusion Middleware Configuration Guide for WebSphere*.

## 2.3 Deploying Using Oracle Enterprise Manager Fusion Middleware Control

You can use Oracle Enterprise Manager Fusion Middleware Control to deploy the EAR file created in JDeveloper. Fusion Middleware Control is a Web browser-based, graphical user interface that you can use to monitor and administer a farm. For more information about deploying using Fusion Middleware Control, see the *Oracle Fusion Middleware Administrator's Guide*.

## 2.4 Deploying Using Scripting Commands

Applications or modules can be deployed from JDeveloper without starting the JDeveloper IDE. You can run WLST commands (for WebLogic Server) or wsadmin commands (for WebSphere Application Server) from the command line or sequence them in scripts to run as a batch.

Before deploying from the command line, there must be deployment profiles for the application (EAR) or project (JAR or WAR). JDeveloper creates these deployment profiles automatically for certain types of applications, but before using commands for deployment, it is important to verify that the deployment profile(s) exist. To verify that the profiles exist, choose the **Deployment** node from either the Application Properties or Project Properties dialogs in JDeveloper. For more information about deployment profiles, see the *Oracle Fusion Middleware Developer's Guide for Oracle Application Development Framework*.

JDeveloper can also be used to deploy an application's EAR, WAR, or JAR files. The same scripts that are used for deployment via a command line are also used to deploy via JDeveloper, but JDeveloper creates the syntax and provides a user interface for the deployment.

There are specific WLST commands (WebLogic) for working with ADF applications. For a list of these commands, see Chapter 4, "WLST Command Reference for ADF Applications."

For more information about using WLST scripts, see the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

There are specific wsadmin commands (WebSphere Application Server) for working with ADF applications. For a list of these commands, see Appendix B, "wsadmin Command Reference for ADF Applications."

## 2.5 Deploying Using Scripts and Ant

You can deploy the application using commands and scripts. You create a script to deploy the application using the `ojdeploy` command and use the `ojaudit` command to audit projects, workspaces, or source files of the application. You can also set up the script to run automatically, for instance, whenever a developer checks in new changes.

`ojdeploy` scripts and Ant scripts can be used together or separately:

1. Create an `ojdeploy` script to compile, package, and deploy the application.
2. Create an `ojdeploy` script to compile and package the application. Then use an Ant script (such as `WLDeploy`) to deploy the application.
3. Create an Ant script to compile, package, and deploy the application. The Ant does not need to use `ojdeploy`.

For more information about the `ojdeploy` and `ojaudit` commands, see the JDeveloper online help.

You can deploy to most application servers from JDeveloper, or use tools provided by the application server vendor. You may also use Ant to package and deploy applications. The `build.xml` file, which contains the deployment commands for Ant, may vary depending on the target application server.

For deployment to other application servers, see the application server's documentation. If your application server does not provide specific Ant tasks, you may be able to use generic Ant tasks. For example, the generic `ear` task creates an EAR file for you.

For information about Ant, see `http://ant.apache.org`.

## 2.6 Deploying Using the Application Server Administration Tool

For WebLogic, you can use the Oracle WebLogic Server Administration Console to deploy the EAR file created in JDeveloper. For more information, see *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*.

For WebSphere Application Server, you can use the IBM WebSphere Administrative Console to deploy the EAR file created in JDeveloper. For more information, go to the WebSphere Application Server Information Center at: `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.home.doc/welcome.html`.

# 3

# Monitoring and Configuring ADF Applications

This chapter describes how to monitor ADF application performance. It also describes how to configure an ADF application's properties after it has been deployed to Oracle WebLogic Server.

This chapter includes the following sections:



## 3.1 Introduction to ADF Application Monitoring and Configuration

After you have deployed an ADF application to Oracle WebLogic Server, you can monitor the application performance and configure application properties on the server. You can use Enterprise Manager Fusion Middleware Control to perform these tasks.

Enterprise Manager Fusion Middleware Control offers a user interface for the performance tasks. Some configuration tasks can be performed either from a user interface or by configuring an MBean, as listed in Table 3–1.

***Table 3–1 Configuration Tasks Using Fusion Middleware Control***

| Configuration tasks | Fusion Middleware Control UI | Fusion Middleware Control MBean Browser |
|---|---|---|
| ADF Business Components | Section 3.3.1, "How to Modify ADF Business Components Parameters" | Section 3.4.3, "How to Modify ADF Business Components Configuration Using MBeans" |

*Table 3–1 (Cont.) Configuration Tasks Using Fusion Middleware Control*

| Configuration tasks | Fusion Middleware Control UI | Fusion Middleware Control MBean Browser |
|---|---|---|
| ADF connections | Section 3.3.2, "How to Modify Connection Configurations" | Section 3.4.2, "How to Modify ADF Connections Using MBean" |
| ADF application configuration | | Section 3.4.1, "How to Modify ADF Application Configuration Using MBean" |
| Metadata Services (MDS) | | Section 3.4.4, "How to Modify MDS Configuration Using MBean" |

By default, the post-deployment changes made using MBeans are stored in MDS with a layer name of `adfshare` and a layer value of `adfshare`. You can provide a specific layer name by specifying the `adfAppUId` property in the application's `adf-config.xml`.

Example 3–1 shows the `adf-properties-child` code in `adf-config.xml`.

***Example 3–1 MDS Layers in the adf-config.xml File***

```
<adf:adf-properties-child xmlns="http://xmlns.oracle.com/adf/config/properties">
     <adf-property name="adfAppUID" value="DeptApp.myApp"/>
</adf:adf-properties-child>
```

If you are moving data between MDS repositories (for example, from a test to a production system), use the MDS `exportMetadata` and `importMetadata` commands as described in the chapter on managing the Oracle metadata repository in the *Oracle Fusion Middleware Administrator's Guide* and in the chapter on Metadata Services custom WLST commands in the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

## 3.2 Monitoring Performance Using Fusion Middleware Control

You can monitor the performance of Oracle ADF applications using the Enterprise Manager Fusion Middleware Control.

You can:

- View application module performance
- View application module pool performance
- View task flow performance

### 3.2.1 How to View Application Module Performance

You can view performance information about application modules. Application module components can be used to support a unit of work which spans multiple browser pages.

**Before you begin:**

You must already have deployed an Oracle ADF application and have Enterprise Manager Fusion Middleware Control available to access the application.

**To view ADF application module performance:**

1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.

2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.
4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.
5. Click **Application Deployment** and select **ADF > ADF Performance** from the dropdown menu.

   The ADF Performance page displays. It contains subtabs for viewing performance information about active application module pools and task flows.

## 3.2.2 How to view Application Module Pool Performance

An *application module pool* is a collection of instances of a single application module type which are shared by multiple application clients. One application module pool is created for each root application module used by an ADF web application (ADF Business Components, ADF Controller, or ADF Faces) in each Java virtual machine where a root application module of that type is used by the ADF Controller layer.

**Before you begin:**

You must already have deployed an Oracle ADF application and have Enterprise Manager Fusion Middleware Control available to access the application.

**To view application module pooling performance:**

1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.
2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.
4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.
5. Click **Application Deployment** and select **ADF > ADF Performance** from the dropdown menu.

   The ADF Performance page displays. It contains subtabs for viewing performance information about active Application Module Pools and Task Flows.
6. Click the **Application Module Pools** tab.
7. In the **Module** column, select an application module to display its details in the Application Module Pools table.

   `No Data Available` displays in the Module column if an application has never run.
8. Click a module to display additional informations about the module, for example, Lifetime, State Management, Pool Use, and Application Module Pools Page.

   Use the Application Module Pools page to display active application module pools, a collection of application module instances of the same type. The Application Module Pools page:

   - Displays size and performance information about pool connections
   - Specifies settings that affect how application module pools behave

- Specifies credential information for the application module pools

| Element | Description |
|---|---|
| Module | Displays the active application module pool name, for example, `model.BugTest5PM`.<br>Click a module to display additional information about it, for example, Lifetime, State Management, Pool Use, Application Module Pools page. |
| Requests | Displays the number of requests that were made for the application during the selected time interval. |
| Average Creation Time (ms) | Displays the average time (in milliseconds) required to complete a request for the application module pool. |
| Maximum Creation Time (ms) | Displays the longest time (in milliseconds) required to complete any of the requests for the application module pool. |
| Free Instances | Displays the number of available instances of the application module pool. |

### 3.2.3 How to View ADF Task Flow Performance

You can view performance information about task flows. Task flows provide a modular and transactional approach to navigation and application control. Task flows mostly contain pages that will be viewed, but they also can contain activities that call methods on managed beans, evaluate an EL expression, or call another task flow, all without invoking a particular page.

**Before you begin:**

You must already have deployed an Oracle ADF application and have Enterprise Manager Fusion Middleware Control available to access the application.

**To view task flow performance:**

1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.
2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.
4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.
5. Click **Application Deployment** and select **ADF > ADF Performance** from the dropdown menu.

   The ADF Performance page displays. It contains subtabs for viewing performance information about active application module pools and task flows.
6. Click the **Task Flows** tab.

   By default, Task Flow Performance charts on the tab display data for the preceding 15 minutes. To set a different interval, click the time at the top of the page or move the slider to another interval, for example, from 08:00 AM to 08:30 AM.
7. Click **TF Charts**.

- **Request Processing Time** displays the average request processing time for all ADF task flows that execute during the selected interval.
- **Active Task Flows** displays the number of active instances of each ADF task flow during the selected interval.

## 3.3 Configuring Application Properties Using Fusion Middleware Control

You can use Enterprise Manager Fusion Middleware Control to configure ADF application configuration parameters. These configuration parameters are stored in ADF MBeans. Fusion Middleware Control provides a user interface to configure the ADF Business Components and ADF Connections MBeans. You can also use the System MBean Browser to directly access the underlying MBeans and configure their values. For more information about accessing the underlying MBeans, see Section 3.4, "Configuring Application Properties Using the MBean Browser."

Fusion Middleware Control provides a user interface for you to:

- Configure ADF Business Component parameters
- Configure connection parameters

### 3.3.1 How to Modify ADF Business Components Parameters

You control the runtime behavior of an application module pool by setting appropriate configuration parameters. Fusion Middleware Control provides a UI to configure ADF Business Components, as described in this section. You can also configure the ADF Business Components MBeans directly using the generic MBean Browser, as described in Section 3.4.3, "How to Modify ADF Business Components Configuration Using MBeans."

**Before you begin:**

You must already have deployed an Oracle ADF application and have Enterprise Manager Fusion Middleware Control available to access the application.

**To modify business components parameters:**

1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.
2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.
4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.
5. Click **Application Deployment** and select **ADF > Configure ADF Business Components** from the dropdown menu.
6. Click an **Application Module.**
7. Click the **Pooling and Scalability**, **Core**, **Database**, or **Security** tabs to update configuration parameters.

   If the application module uses data sources, you can configure the data sources by clicking **Edit Datasource** from the **Core** tab.

The ADF Business Components configurations page is arranged with the following sections or tabs:

- Application Modules section
- Pooling and Scalability tab - Application Pool Properties
- Pooling and Scalability tab - Connection Pool Properties
- Core tab
- Database Properties tab
- Security Properties tab

### Application Modules Section

In the Application Modules section, select the application module you want to configure.

| Element | Description |
|---|---|
| Application Modules | Displays the active application module name. Click the module name to display the applications in the module. |

### Pooling and Scalability Tab - Application Pool Properties

In the Pooling and Scalability tab, select the application pool properties you want to configure.

| Element | Description |
|---|---|
| AmpoolDoampooling | Select to enable application module pooling by default. Whenever you deploy your application in a production environment the default setting of `jbo.ampool.doampooling` is `true` and is the way you will run your application. But, as long as you run your application in a test environment, setting the property to `false` can play an important role in your testing. When this property is `false`, there is effectively no application pool. |
| AmpoolWritecookietoclient | Select to write the `SessionCookie` value to the client browser. |
| AmpoolMaxavailablesize | Enter the maximum number of available application modules that should be referenced by an application pool. This is the ideal maximum number of available application module instances in the pool when not under abnormal load.<br><br>When the pool monitor wakes up to do resource cleanup, it will try to remove available application module instances to bring the total number of available instances down to this ideal maximum. Instances that have been not been used for a period longer than the idle instance timeout will always get cleaned up at this time, and then additional available instances will be removed, if necessary to bring the number of available instances down to this size.<br><br>The default maximum available size is 25 instances. Configure this value to leave the maximum number of available instances desired after a resource cleanup. A lower value generally results in more application module instances being removed from the pool on a cleanup. |
| AmpoolSessioncookiefactoryclass | Enter a custom session cookie factory implementation. This class creates the session cookies that allow clients to retrieve application modules in stateful mode |

| Element | Description |
|---|---|
| AmpoolMaxinactiveage | Enter the maximum amount of time (in milliseconds) that an application module may remain inactive before it is removed from the pool.<br><br>The default is 600000 milliseconds of idle time (which is 600 seconds, or ten minutes). A lower value results in more application module instances being marked as candidates for removal at the next resource cleanup. A higher value results in fewer application module instances being marked as candidates for removal at the next resource cleanup. |
| AmpoolMinavailablesize | Enter the minimum number of available application modules that should be referenced by an application pool. This is the minimum number of available application module instances that the pool monitor should leave in the pool during a resource cleanup operation.<br><br>Set to 0 (zero) if you want the pool to shrink to contain no instances when all instances have been idle for longer than the idle timeout after a resource cleanup.<br><br>The default is 5 instances. |
| Doconnectionpooling | Select if the application pool should release the application module connection upon checkin. This forces the application module pool to release the JDBC connection used each time the application module is released to the pool. |
| Recyclethreshold | Enter the maximum number of application module instances in the pool that attempt to preserve session affinity for the next request made by the session. This session used them last before releasing them to the pool in managed-state mode. |
| AmpoolConnectionstrategyclass | Enter a custom connection strategy implementation, for example `oracle.jbo.common.ampool.DefaultConnectionStrategy`. This is the class that implements the connection strategy. |
| Maxpoolcookieage | Enter the maximum browser cookie age for pooled application module sessions. This is the maximum age of the browser cookies used to help clients retrieve stateful application modules. If these cookies do not time out, the value is -1. It is recommended that the maximum cookie age be always set less than or equal to the session cookie age. It is set that way by default (both are -1). If you change the maximum cookie age, then you must also change the session cookie age to the same value. |
| AmpoolInitpoolsize | Enter an initial number of application module instances to be created in a pool. This is the number of application module instances to created when the pool is initialized.<br><br>The default is 0 (zero) instances. A general guideline is to configure this value to 10% more than the anticipated number of concurrent application module instances required to service all users.<br><br>Creating application module instances during initialization takes the CPU processing costs of creating application module instances during the initialization instead of on-demand when additional application module instances are required. |

| Element | Description |
|---|---|
| AmpoolDynamicjdbccredentials | Select if an application pool may support multiple JDBC users. This property enables additional pooling lifecycle events to allow developer-written code to change the database credentials (username/password) each time a new user session begins to use the application module.<br><br>This feature is enabled by default (`true`); however this setting is a necessary but not sufficient condition to implement the feature. The complete implementation requires additional developer-written code. |
| AmpoolIsuseexclusive | Select if application module use is exclusive. |
| AmpoolResetnontransactionalstate | Select if the nontransactional application module state should be reset upon an unmanaged checkin. This forces the application module to reset any nontransactional state like view object runtime settings, JDBC prepared statements, bind variable values, and so on. when the application module is released to the pool in unmanaged, or "stateless," mode.<br><br>This feature is enabled by default (`true`). Disabling this feature can improve performance; however, since it does not clear bind variable values, your application needs to ensure that it systemically sets bind variable values correctly. If your application does not do so, and this feature is disabled, then it is possible for one user to see data with another user's bind variable values. |
| AmpoolMaxpoolsize | Enter the maximum number of application module instances that the pool can allocate. The pool will never create more application module instances than this limit imposes.<br><br>The default is 5000 instances. A general guideline is to configure this value to 20% more than the initial pool size to allow for some additional growth. If the value is set too low, then some users may see an error when they tries to access the application if no application module instances are available. |
| AmpoolTimetolive | Enter the connection pool time to live for connection instances. This is the number of milliseconds after which an application module instance in the pool is considered as a candidate for removal during the next resource cleanup, regardless of whether it would bring the number of instances in the pool below `minavailablesize`.<br><br>The default is 3600000 milliseconds of total time to live (which is 3600 seconds, or one hour). The default value is sufficient for most applications. |
| AmpoolMonitorsleepinterval | Enter the length of time (in milliseconds) between pool resource cleanups. |
| Dofailover | Select if failover should occur upon checkin to the application module pool. This feature enables eager passivation of pending transaction state each time an application module is released to the pool in managed state mode. Web applications should set enable failover (`true`) to allow any other application module to activate the state at any time. This feature is disabled by default (`false`). |
| poolClassName | Enter the custom application pool implementation class. |

| Element | Description |
|---|---|
| Show Connection Pool Properties | Expand to display fields containing current advanced connection pool properties, or enter new values in the fields. |
| Hide Connection Pool Properties | Click to hide all Connection Pool Properties fields. |

### Pooling and Scalability Tab - Connection Pool Properties

In the Pooling and Scalability tab, select the connection pool properties you want to configure.

| Element | Description |
|---|---|
| Initpoolsize | Enter the initial size of a JDBC connection pool. This is the number of JDBC connection instances created when the pool is initialized.<br>The default is an initial size of 0 instances. |
| Maxpoolsize | Enter the maximum size of a JDBC connection pool. This is the maximum number of JDBC connection instances that the pool can allocate. The pool will never create more JDBC connections than this allows.<br>The default is 5000 instances. |
| Poolmaxinactiveage | Enter the maximum amount of time (in milliseconds) that a connection may remain inactive before it is removed from the pool. This is the number of milliseconds after which to consider an inactive application module instance in the pool as a candidate for removal during the next resource cleanup.<br>The default is 600000 milliseconds of idle time (which is 600 seconds, or ten minutes). A lower value results in more application module instances being marked as candidates for removal at the next resource cleanup. A higher value results in fewer application module instances being marked as candidates for removal at the next resource cleanup. |
| Poolmaxavailablesize | Enter the maximum number of available connections that should be referenced by a connection pool. This is the ideal maximum number of JDBC connection instances in the pool when not under abnormal load.<br>When the pool monitor wakes up to do resource cleanup, it will try to remove available JDBC connection instances to bring the total number of available instances down to this ideal maximum. Instances that have been not been used for a period longer than the idle instance timeout will always get cleaned up at this time, and then additional available instances will be removed, if necessary, to bring the number of available instances down to this size.<br>The default is an ideal maximum of 25 instances (when not under load). |
| Poolrequesttimeout | Enter the time (in milliseconds) that a request should wait for a JDBC connection to be released to the connection pool. |

| Element | Description |
|---|---|
| Poolminavailablesize | Enter the minimum number of available connections that should be referenced by a connection pool. This is the minimum number of available JDBC connection instances that the pool monitor should leave in the pool during a resource cleanup operation.<br>Set to zero (0) if you want the pool to shrink to contain no instances when instances have been idle for longer than the idle time-out.<br>The default is to not let the minimum available size drop below 5 instances. |
| Poolmonitorsleepinterval | Enter the time (in milliseconds) that the connection pool monitor should sleep between pool checks. This is the length of time in milliseconds between pool resource cleanup.<br>While the number of application module instances in the pool will never exceed the maximum pool size, available instances which are candidates for getting removed from the pool do not get "cleaned up" until the next time the application module pool monitor wakes up to do its job. |
| ConnectionPoolManager | Enter the implementation of the connection pool manager which will be used. |
| Pooltimetolive | Enter the application pool time to live (in milliseconds) for application module instances. |

### Core Tab

Use the core tab to view or edit core properties for the application module.

| Element | Description |
|---|---|
| DefaultLanguage | Enter the default business components session language, which is part of the locale. |
| Passivationstore | Enter the type of store, file, or database file that should be used for application module passivation.<br>`database` is the default choice. While it may be a little slower than passivating to file, it is by far the most reliable choice.<br>`file` may offer faster performance because access to the file is faster then access to the database. |
| Default Country | Enter the default business components session country, which is part of the Locale. |
| AssocConsistent | Select if entity row set associations have been kept consistent. |
| XmlValidation | Select to determine the validation mode for the XML parser. If selected, the XML parser uses strict XML validation. |
| DatabaseConfig | Database Configuration. |
| Name | Enter the name of the application module. |
| OracleSchema | Enter the name of the schema in which the business components runtime libraries are deployed. |
| Show Advanced Properties | Expand to display fields containing current advanced core properties, or enter new values in the fields. |

| Element | Description |
|---|---|
| PersMaxRowsPerNode | Enter the maximum size of a node for view row spillover. |
| PassivationTrackInsert | If selected when an application module is activated, it will be updated to include rows inserted into the database while it was passive. |
| ApplicationPath | For EJB deployment, enter the JNDI path to the business components. |
| ViewlinkConsistent | If selected, the view object row sets retrieved through view link accessors will include rows that have been added, even if these changes have not been posted to the database. |
| ConnectionMode | Deprecated property, formerly used for deployment to VisiBroker. VisiBroker deployment is no longer supported. |
| Maxpassivationstacksize | Enter the maximum size of the passivation stack (default is 10) |
| TxnHandleafterpostexc | Select to cause ADF Business Components to take a transaction snapshot before beginning a commit operation. If an exception is thrown after changes have been posted to the database, ADF Business Components will use this snapshot to roll back the in-memory state of your application module to the point before commit operation began. |
| SnapshotstoreUndo | Enter the target for undo snapshots {transient \| persistent} |
| Project | Enter the name of the project containing extended business components to be substituted for base ones, if Factory-Substitution-List is not empty. |
| Tmpdir | Enter the directory for temporary Oracle ADF Business Components files. |
| DeployPlatform | The deployment platform: select LOCAL, EJB_IAS (for an EJB deployed to Oracle Application Server), or WLS (for an EJB deployed to Oracle WebLogic Server). |
| PersMaxActiveNodes | Enter the maximum number of nodes that will be cached in memory for view row spillover. |
| Saveforlater | Select Save snapshots for the lifetime of the transaction. |
| ViewCriteriaAdapter | Enter a custom class that will be used by view objects to convert between view criteria and view object SQL. |
| Connectfailover | Select business components transparent JDBC connection failover |
| Hide Advanced Properties | Click to hide all Connection Pool Properties fields. |

### Database Properties Tab

If you are using a JDBC URL for your connection information so that the ADF database connection pool is used, then the configuration parameters listed here can be used to tune the behavior of the database connection pool.

| Element | Description |
|---|---|
| MaxCursors | Enter the maximum number of cursors to be used by the session. This is the maximum number of cursors the business components may have open. The framework will clean up free JDBC statements as the number of cursors approaches this number. |
| Sql92DbTimeQuery | Enter the database system time SQL query string. |
| SQLBuilder | Enter the SQLBuilder implementation (Oracle, OLite, DB2, or SQL92 for other SQL92-compliant databases). |
| Sql92LockTrailer | Enter the SQL statement trailer clause for locking. |
| JdbcTrace | Select to trace all JDBC activity with lines flagged by `+ PropertyConstants.JDBC_MARKER +` |
| oracleDefineColumnLength | Enter the column length for all JDBC CHAR or VARCHAR2 columns. Use `as_bytes` to make column precision specifications in bytes. Use `as_chars` to make column precision specifications in characters. This is important for larger character sets, such as Unicode. |
| Sql92JdbcDriverClass | Enter the name of the class implementing JDBC Driver, for example, `sun.jdbc.odbc.JdbcOdbcDriver`. |
| TypeMapEntries | Enter the type map implementation. This specifies a custom type map between Java types and SQL types. |
| ControlTableName | Enter the persistent collection control table name. |
| FetchMode | Enter the control fetch behavior of View Objects (`+ PropertyConstants.ENV_FETCH_AS_NEEDED + "\|" + PropertyConstants.ENV_FETCH_ALL +`).<br>`AS.NEEDED` causes view objects to fetch rows only when they are requested. `ALL` causes them to fetch the entire results of their queries. |
| LockingMode | Enter the default locking mode for an application module. This prevents the application module pool from creating a pending transaction state on the database with row-level locks each time the application module is released to the pool.<br>Fusion web applications should set the locking mode to optimistic to avoid creating the row-level locks. |
| JdbcBytesConversion | Indicate whether to use JDBC default bytes conversion or to perform such conversion in the framework. |
| Show Advanced Properties | Expand to display fields containing current advanced database properties, or enter new values in the fields. |
| TxnSeqInc | Select persistent transaction sequence increment. |
| UsePersColl | Select enable view row spillover to help manage large rowsets. |
| TxnSeqName | Enter persistent transaction sequence name. |
| Hide Advanced Properties | Click to hide all advanced property fields. |

### Security Properties Tab

Use the Security Properties tab to configure application module security information.

| Element | Description |
|---|---|
| SecurityContext | Enter the JAAS context. This element specifies a particular JAAS implementation.<br>The default is JAZN. |
| Show Advanced Properties | Expand to display fields containing current advanced security properties, or enter new values in the fields. |
| UserPrincipal | Enter the authenticated user principal name. |
| SecurityConfig | Enter the complete path and file name of JAZN configuration, for example, `k:\j2ee\home\config\jazn.xml`.<br>If this property value is null or length 0, runtime will assume that `jazn.xml` is in the same path as `jazn.jar` and append `/config/jazn.xml` before it accesses login module or gets the JAZN context for getting permission manager. |
| javaNamingSecurityCredentials | For EJB deployment, enter the password for the application server connection. |
| AppModuleJndiName | For EJB deployment, enter the JNDI name used to look up the application module factory. |
| SecurityLoginmodule | Enter a custom login module for authentication, for example, `oracle.security.jazn.realm.RealmLoginModule`.<br>The default is the JAZN login module. |
| ServerUseNullDbTransaction | Use 9.0.2 compatible `oracle.jbo.server.NullDbtransactionImpl` when not connected to the database. |
| SecurityEnforce | Enter one of the following values:<br>**None** - No authentication.<br>**Test** - Requires authentication. If using the tester or ADF Swing, a dialog will prompt for login. If authentication fails, the application module is still instantiated.<br>**Must** - Like Test, but if authentication fails, the application module will not be instantiated. Instead, you will get an exception.<br>**Auth** - Like Must, but in addition, if you have used the Entity Wizard Authorization editor to define entity or attribute permissions, the permissions will be checked. For example if the permission on `Dept.Deptno` was granted `update_while_new` to role `users`, then the `users` role can set the `Deptno` value only when the row is new. Otherwise, it is not editable.<br>Note that even if there are permissions granted via the wizard, they will not be enforced unless `jbo.security.enforce` is set to `Auth`. |
| javaNamingSecurityPrincipal | For EJB deployment, enter the password for the application server connection. |
| Hide Advanced Properties | Click to hide all advanced property fields. |

## 3.3.2 How to Modify Connection Configurations

A connection configuration contains information that a client application uses to identify the ADF application module's deployment scenario. You use Oracle Enterprise Manager Fusion Middleware Control to:

- Register and manage back-end services such as mail, discussion forums servers, and so on
- Register and manage external applications that users need access to while working with applications
- Register and manage any portlet producers that the application uses or that users may need access to

Fusion Middleware Control provides a UI to configure ADF connections, as described in this section. You can also configure the ADF connections MBean directly using the generic MBean Browser, as described in Section 3.4.2, "How to Modify ADF Connections Using MBean."

**Before you begin:**

You must already have deployed an Oracle ADF application and have Enterprise Manager Fusion Middleware Control available to access the application.

You must have MDS configured in your application before you can modify the ADF application and connection configurations. ADF connection attributes are persisted to MDS.

If you deployed an application to several nodes within a cluster, any ADF connection changes to a single node will be propagated to all the other nodes. MDS will store a single set of connection information for all versions of an application.

**To modify connection configurations:**

1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.
2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.
4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.
5. Click **Application Deployment** and select **ADF > Configure ADF Connections** from the dropdown menu.
6. In the **Connection Type** drop-down list, choose the type of connection you want to configure:
   - ADF BC Service
   - Discussions and Announcements
   - File System
   - Mail Server
   - Secure Enterprise Search
   - URL
   - Web Service

You cannot create an Essbase connection, however, you can edit an existing Essbase connection that was deployed with the application.

7. In the **Connection Name** field, enter a unique name for the connection configuration.
8. Click **Create Connection**.

   The Connection Configuration page updates with a section where you can specify options for the connection type you chose.

The following connection types are supported:

- ADF Business Components Service connection
- Essbase connection
- Discussions and Announcements connection
- File system connection
- Mail server connection
- Secure enterprise search connection
- URL connection
- Web Service connection

### ADF Business Components Service Connection

Use the ADF Business Components Service connection page to create a new ADF Business Components Service connection or to modify existing connection details.

| Element | Description |
|---|---|
| serviceEndpointProvider | Enter the provider of the service endpoint. Valid types are `ADFBC`, `Fabric`, `SOAP`. |
| serviceConnectionName | Enter the service connection name. |
| jndiName | Enter the JNDI name of the EJB that implements the service interface. Applicable when the endpoint is `ADF BC`. |
| jndiFactoryInitial | Enter the class name of initial context factory for JNDI lookup. Applicable when the endpoint is `ADF BC`. |
| jndiProviderURL | Enter configuration information for the JNDI lookup. Applicable when endpoint is `ADF BC`. |
| jndiSecurityPrincipal | Enter the identity of the principal (e.g. user) for the JNDI lookup. Applicable when the endpoint is `ADF BC`. |
| jndiSecurityCredentials | Enter the principal's credentials for JNDI lookup. Applicable when the endpoint is `ADF BC`. |
| fabricAddress | Enter the service name of the SOA composite. Applicable when the endpoint is `Fabric`. |
| serviceInterfaceName | Enter the class name of the service endpoint interface. |
| serviceSchemaName | Enter the name of the service schema file. |
| serviceSchemaLocation | Enter the relative path of the service schema file. |

### Essbase Connection

You cannot create an Essbase connection; however, you can edit an existing Essbase connection that was deployed with the application.

| Element | Description |
|---|---|
| Host | Enter the host that this connection represents. |
| Port | Displays the default port that this connection uses to connect to Essbase. Clear the Default option to enter a port other than the default. |
| Username | Enter the user name authorized to connect to Essbase during design time. This user name is replaced at runtime with the user name specified by the application. |
| Password | Enter the password of the user. An asterisk (*) is displayed for each character you enter in this field. |

### Discussions and Announcements Connection

Use the Discussion Forum Connection pages to connect to a new discussions server connection or to modify existing connection details. Forum Connections configuration includes configurations for name, connection details, and advanced.

#### Discussions and Announcements Connection - Name

| Element | Description |
|---|---|
| Name | Enter a unique name for the connection. |

#### Discussions and Announcements Connection - Connection Details

| Element | Description |
|---|---|
| Server URL | Enter the URL of the discussion server hosting the discussion forums.<br>For example: `http://discuss-server.com:8888/owc_discussions` |
| Administrator User Name | Enter the user name of the discussion server administrator.<br>Administrative privileges are required for this connection so that operations can be performed on behalf of WebCenter users. |
| Connection Timeout (in Seconds) | .Enter the connection timeout in seconds. The default is -1. |
| Connection Secured | Indicate whether or not the discussion server connection is secure. |

#### Discussions and Announcements Connection - Advanced Configuration

| Element | Description |
|---|---|
| Cache Size (in MB) | Specify the amount of space reserved for the cache (in MB).<br>The default is 0. |
| Cache Expiration Time (in Minutes) | Specify a suitable expiration period for the cache.<br>This is the maximum length of time (in minutes) that cached content is valid.<br>The default is 0. |

| Element | Description |
|---|---|
| Connection Timeout (in Seconds) | Specify a suitable timeout for the connection.<br>This is the length of time (in seconds) that the WebCenter application waits for a response from the discussion server before issuing a connection timeout message.<br>The default is 60 seconds. |

### File System Connection

Use the Add/New Content Repository Connection pages to connect to a new content repository or to modify existing connection details.

> **Note:** All configuration changes are stored in the MDS repository.

### File System Connection Details - File System

| Element | Description |
|---|---|
| Root Path | Enter the full path to a folder on a local file system in which your content is placed. For example: `C:/MyContent`<br>**Caution:** File system content *must not* be used in production or enterprise application deployments. This feature is provided for development purposes only. |

### Mail Server Connection

Use the Mail Server connection pages to configure LDAP and advanced mail server configurations

| Element | Description |
|---|---|
| IMAP Host | Enter the host name of the machine where the IMAP service (Internet Message Access Protocol) is running. |
| IMAP Port | Enter the port on which the IMAP service listens. |
| IMAP Secured | Indicate whether a secured connection (SSL) is required for incoming mail over IMAP. |
| SMTP Host | Enter the host name of the machine on which the SMTP service (Simple Mail Transfer Protocol) is running. |
| SMTP Port | Enter the port on which the SMTP service listens. |
| SMTP Secured | Indicate whether a secured connection (SSL) is required for outgoing mail over SMTP. |
| Associated External Application | Associate the mail server with an external application. External application credential information is used to authenticate users against the IMAP server. |

### Mail Server Connection - LDAP Configuration

| Element | Description |
|---|---|
| LDAP Domain | Enter the LDAP domain. |
| LDAP Host | Enter the host name of the LDAP (Lightweight Directory Access Protocol) server. |

| Element | Description |
|---|---|
| LDAP Port | Enter the port on which the LDAP service listens. |
| LDAP Secured | Indicate whether a secured connection (SSL) is required for the LDAP connection. |
| LDAP Administrator User Name | Enter the user name of the LDAP server administrator. |
| LDAP Administrator Password | Enter the password for the LDAP server administrator. |
| LDAP Base DN | Enter the base-distinguished name for the LDAP schema. |
| LDAP Default User | Enter the LDAP default user. |

#### Mail Server Connection - Advanced Configuration

| Element | Description |
|---|---|
| Connection Timeout (in Seconds) | Specify a suitable timeout for the connection.<br>This is the length of time (in seconds) that the WebCenter application waits for a response from the mail server before issuing a connection timeout message. The default is 60 seconds. |
| Cache Expiration Time (in Minutes) | Specify a suitable expiration period for the cache.<br>This is the maximum length of time (in minutes) that cached content is valid. The default value (-1) means that the cache never expires. |

### Secure Enterprise Search Connection

Use the Secure Enterprise Search Connection pages to connect the WebCenter application to a new Oracle Secure Enterprise Search server or to modify existing connection details.

Secure Enterprise Search Connection Provider configuration includes configurations for name, connection details, and advanced configurations.

#### Secure Enterprise Search Connection - Name

| Element | Description |
|---|---|
| Connection Name | Enter a unique name for the connection. |
| Active Connection | Select to use this connection for search-related services in the WebCenter application.<br>You can register multiple search connections through Oracle Enterprise Manager Fusion Middleware Control, but only one connection is active at a time. |

### Secure Enterprise Search Connection - Connection Details

| Element | Description |
|---|---|
| SOAP URL | Enter the Web Service URL that Oracle Secure Enterprise Search exposes to enable search requests.<br>Use the format:<br>`http://<host>:<port>/search/query/OracleSearch`<br>For example: `http://myHost:7777/search/query/OracleSearch` |
| Application User Name | Enter the name of a valid user.<br>You can specify the name of any user in the identity store. The user must be present in both the Oracle Identity Management server configured for your WebCenter application and the Oracle Identity Management server configured for Oracle SES.<br>The WebCenter application must authenticate itself as a trusted application to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter users. |
| Application Password | Enter the appropriate user password. |

### Secure Enterprise Search Connection - Advanced Configuration

| Element | Description |
|---|---|
| Oracle Secure Enterprise Search Data Group | Enter the Secure Enterprise Search data group in which to search. |
| Execution Timeout | Enter the search execution timeout in milliseconds. |
| Executor Preparation Timeout | Enter the search executor preparation timeout in milliseconds. |
| Number of Saved Searches | Enter the number of saved searches displayed. |
| Simple Search Result Rows | Enter the number of results displayed from a simple search, for each service. |
| Search Result Rows | Enter the number of search results displayed for each service. |
| Global Search Result Rows | Enter the number of search results displayed (on the toolbar) for each service. |

### URL Connection

Use the URL Connection pages to configure URL connections.

| Element | Description |
|---|---|
| URL | Enter the URL of the desired data stream, but omit any URL parameters. |
| Username | Enter the username require to enter the web site. |
| Password | Enter the password required to enter the web site. |

| Element | Description |
|---|---|
| AuthenticationRealm | Defines the Realm as in HTTP authentication. Defined by the server hosting the protected resources. |
| Proxy | Defines the proxy to be used for connecting to HTTP/HTTPS resources. Specifies the host/port and any authentication details needed to authenticate against the proxy itself. |
| ProxyUseDefault | Uses the default proxy at the system level instead of the connection level at both DT or RT, or wherever the connection instance is active. At design time, the default proxy will be the JDeveloper IDE proxy settings, at runtime, it will be the one configured for WLS. |
| ConnectionClassName | Indicates the type of challenge authentication. The two supported modes are Basic and Digest authentication (HTTP basic & digest). |
| ChallengeAuthenticationType | The class name of the connection that gets loaded into the reference to be used by the factory to construct the connection instance. |

#### Web Service Connection

Use the Web Service Connection page to configure a connection using the WebService MDDS model based on the service WSDL to call and invoke the WebService.

Use the **Configure Web Service** dropdown list to configure the Web Service Client, including attaching and detaching policy. After you have finished the configuration in the web services page, you can use the breadcrumbs to navigate back to the ADF Connections page.

| Element | Description |
|---|---|
| Model | Enter the WebService MDDS model elements generated based on the service WSDL. |
| WsdlUrl | Enter the WebService service WSDL URL. |
| DefaultServiceName | Enter the default service Name of the service WSDL. |

## 3.4 Configuring Application Properties Using the MBean Browser

You can use the Enterprise Manager Fusion Middleware Control MBean Browser to access and modify the values in ADF MBeans deployed with the ADF application into Oracle WebLogic Server.

You can view and modify:

- ADFcConfiguration MBean
- ADF Connections MBean
- ADF Business Components BC4J MBeans
- MDS Configuration MBeans

### 3.4.1 How to Modify ADF Application Configuration Using MBean

You can modify ADF application configurations MBeans using the MBean Browser.

**Before you begin:**

You must have MDS configured in your application before you can modify the ADF application and connection configurations. ADF application attributes are persisted to MDS.

If you deployed an application to several nodes within a cluster, any ADF application configuration changes to a single node via an MBean will be propagated to all the other nodes. MDS will store a single set of ADF application configuration information for all versions of an application.

**To modify ADF application configuration using MBean Browser:**

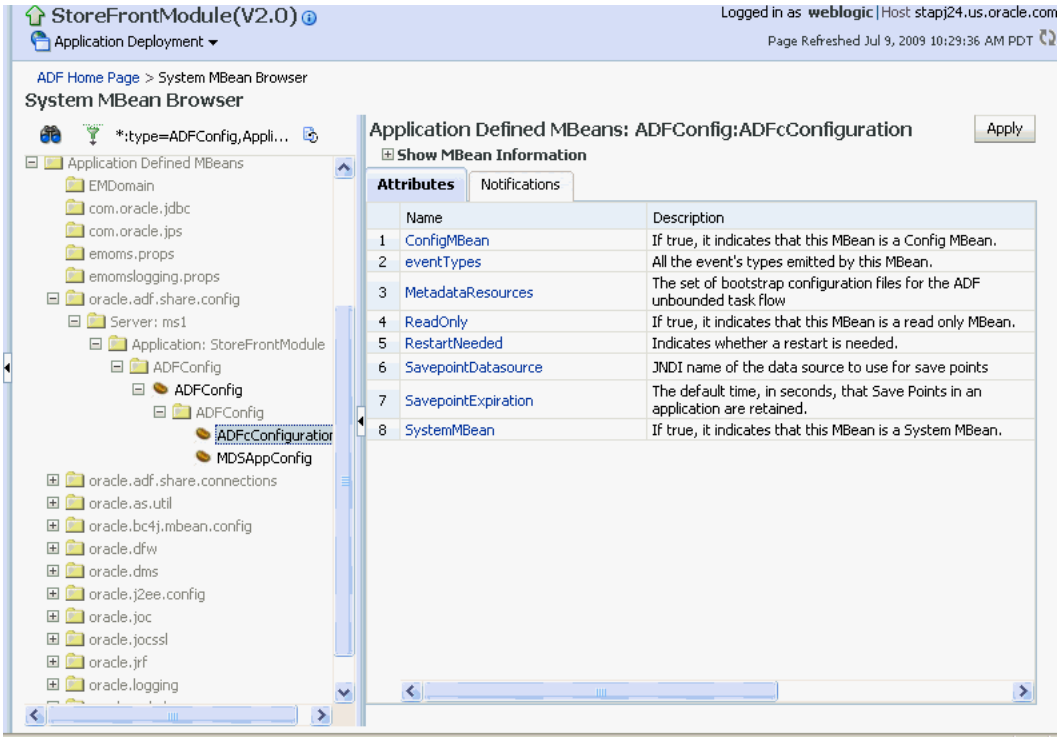
1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.
2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.
4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.
5. Click **Application Deployment** and select **ADF > Configure ADF (adf-config)** from the dropdown menu.
6. In the left pane of the System MBean Browser, expand the parent ADF MBean **ADFConfig** and then the **ADFConfig** folder to expose the child ADF MBeans.

   You may see the child ADF MBeans **ADFcConfiguration** and **MDSAppConfig**.
7. In the left pane, select the **ADFcConfiguration** MBean, and in the right pane, select the attribute you want to view or modify.

   Figure 3–1 shows an ADF Configuration MBean in the Fusion Middleware Control MBean Browser.

***Figure 3–1 ADF Configuration MBean***




8. Change the attribute value and click **Apply**.
9. In the left pane, select the parent ADF MBean **ADFConfig**.
10. In the right pane, click the **Operations** tab and click **save**.

    The new values you have edited are written to MDS after you click **save** from the parent MBean.

## 3.4.2 How to Modify ADF Connections Using MBean

You can modify ADF connection configurations MBean using the MBean Browser.

You can also modify ADF connections using the Fusion Middleware UI described in Section 3.3.2, "How to Modify Connection Configurations."

**Before you begin:**

You must have MDS configured in your application before you can modify the ADF application and connection configurations. ADF application attributes are persisted to MDS.

If you deployed an application to several nodes within a cluster, any ADF connection changes to a single node via an MBean will be propagated to all the other nodes. MDS will store a single set of ADF application configuration information for all versions of an application.

**To modify ADF application configuration using MBean Browser:**

1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.
2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.

4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.

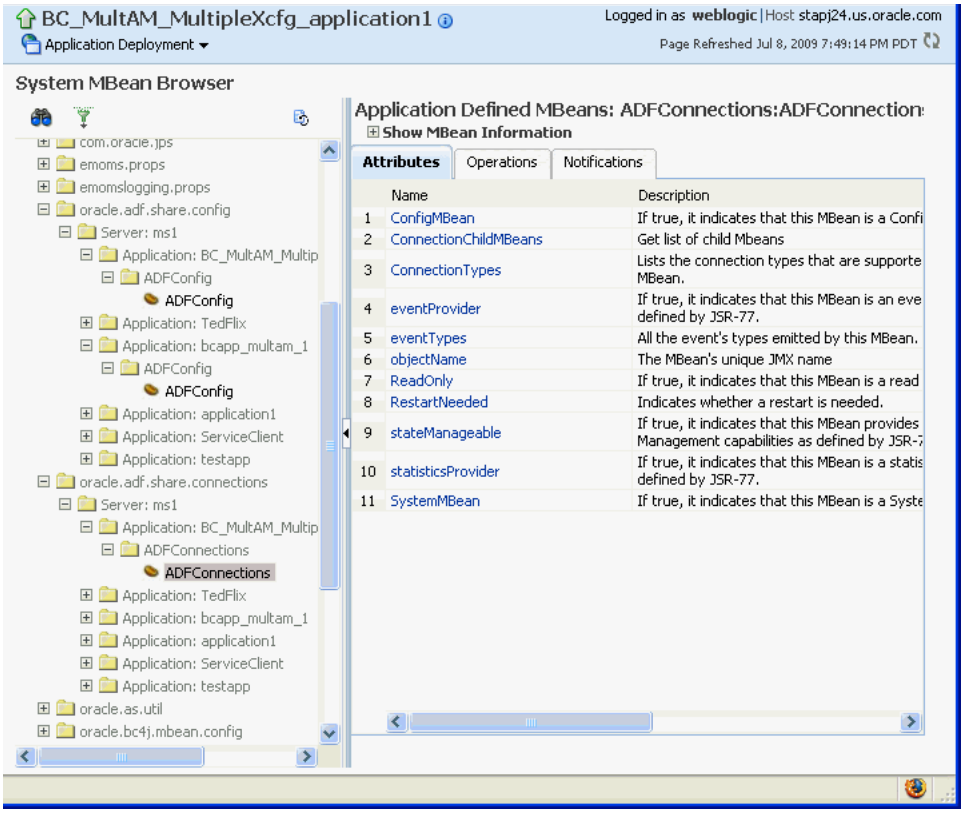
5. Click **Application Deployment** and select **System MBean Browser** from the dropdown menu.

6. In the left pane of the System MBean Browser, navigate to the **ADFConnections** MBean. The MBean should be in **oracle.adf.share.connections** > *server name* > *application name*.

7. In the left pane, select the ADF Connections MBean, and in the right pane, select the attribute you want to view or modify.

   Figure 3–2 shows an ADF Connections MBean displayed in the Fusion Middleware Control MBean Browser.

***Figure 3–2 ADF Connections MBean***




8. Change the attribute value and click **Apply**.

9. In the right pane, click the **Operations** tab and click **save**.

   The new values you have edited are written to MDS after you click **save**.

## 3.4.3 How to Modify ADF Business Components Configuration Using MBeans

You can modify ADF Business Components configurations MBeans using the MBean Browser. ADF Business Component configuration information are stored in MBeans that are specific for each application. Unlike ADF connections and ADF application configuration information which you can configure once for all versions of the same

application, you will need to configure ADF Business Components for each version of the application.

You can also modify ADF Business Components configuration information using the Fusion Middleware UI described in Section 3.3.1, "How to Modify ADF Business Components Parameters."

**Before you begin:**

You must have MDS configured in your application before you can modify the ADF application and connection configurations. ADF application attributes are persisted to MDS.

If you deployed an application to several nodes within a cluster, any ADF Business Components changes to a single node via MBeans will be propagated to all the other nodes. MDS will store a single set of ADF application configuration information for all versions of an application.

**To modify ADF application configuration using MBean Browser:**

1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.
2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.
4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.
5. Click **Application Deployment** and select **System MBean Browser** from the dropdown menu.
6. In the left pane of the System MBean Browser, navigate to the BC4J MBeans. These MBeans should be in **oracle.bc4j.mbean.share** > ***server name*** > ***application name***.
7. In the left pane, select the ADF Connections MBean, and in the right pane, select the attribute you want to view or modify.
8. Change the attribute value and click **Apply**.

## 3.4.4 How to Modify MDS Configuration Using MBean

You can use the MBean Browser to perform advanced configuration of MDS parameters. For more information about configuring MDS using MBeans, see the *Oracle Fusion Middleware Administrator's Guide*.

**Before you begin:**

You must already have deployed an Oracle ADF application and have Enterprise Manager Fusion Middleware Control available to access the application.

**To modify MDS configuration using MBean Browser:**

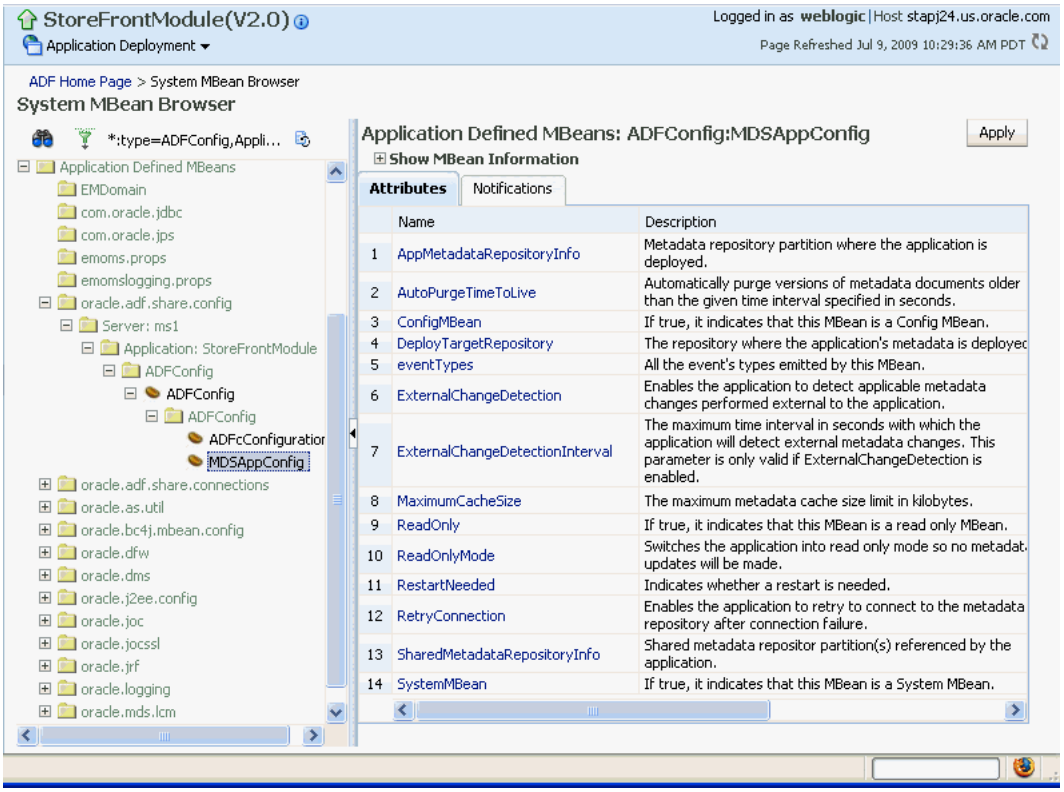
1. Log in to an Oracle Fusion Middleware farm using Fusion Middleware Control.
2. Click the **Farm** tab.
3. Expand the ***Farm_domain*** node.
4. Expand the **Application Deployments** node and click a J2EE application deployment, for example, `StoreFrontModule (AdminServer)`.

   After you select an application, the Application Deployment page displays.

5. Click **Application Deployment** and select **MDS Configuration** from the dropdown menu.
6. Click **Configuration MBean Browser** or **Runtime MBean Browser.**
7. Select the MBean and the attribute you want to view or modify.

   Figure 3–3 show an MDS MBean in the Fusion Middleware Control MBean Browser.

***Figure 3–3 MDS MBean***




8. Change the value and click **Apply**.

## 3.5 How to Edit Credentials Deployed with the Application

You can use Enterprise Manager Fusion Middleware Control to edit credentials that were deployed with an ADF application to the credential store. You can also create new credentials and delete existing credentials.

For ADF applications, the following considerations apply:

- The **Map** name is typically the `adfAppUId` property defined in the application's `adf-config.xml` file.
- The **Key** name is typically in the format `anonymous#`*`connection`*, where *`connection`* is the connection name.
- The **Credential Type** is **Generic** and it is modeled as a hash map of key-value pairs.

For more information, see the "Managing Credentials with Fusion Middleware Control" section of the *Oracle Fusion Middleware Application Security Guide*.

## 3.6 Diagnosing Problems using the Diagnostic Framework

Oracle Fusion Middleware provides a Diagnostic Framework to help you detect, diagnose, and resolve problems with your application

When a critical error occurs, the Diagnostic Framework immediately captures diagnostic data and associates the data and error with an incident number. Using this number, you can retrieve the data for analysis from the Automatic Diagnostic Repository (ADR).

Oracle ADF provides an ADFConfig dump which will execute when an INCIDENT_ERROR message is logged. You can also add code to invoke the dump in the application exception handlers. Example 3–2 show a sample code you can add to your exception handler to invoke the ADFConfig dump.

***Example 3–2 Sample Code for Invoking ADFCcnfig Diagnostic Dump in Exception Handler***

```
IllegalArgumentException e = new IllegalArgumentException("test exception");
LoggerFactory.getFrameworkLogger().log(ODLLevel.INCIDENT_ERROR,
 "Test error message", e);
```

For more information about the Diagnostic Framework, see the chapter on diagnosing problems in the *Oracle Fusion Middleware Administrator's Guide*.

If you are using the Diagnostic Framework on an IBM WebSphere application server, you need to perform additional tasks. For more information, see the *Oracle Fusion Middleware Third-Party Application Server Guide*.

# 4

# WLST Command Reference for ADF Applications

This chapter describes the WLST commands you can use to deploy, manage, and configure Oracle ADF applications.

For wsadmin commands reference for the IBM WebSphere application server, see Appendix B, "wsadmin Command Reference for ADF Applications."

This chapter includes the following sections:

- Section 4.1, "Overview of Custom WSLT Commands for Oracle ADF"
- Section 4.2, "ADF-Specific WLST Commands"

## 4.1 Overview of Custom WSLT Commands for Oracle ADF

Use the ADF-based URL Connections WLST commands to navigate the hierarchy of configuration or runtime beans and control the prompt display. Use the `getADFMArchiveConfig` commands to manage the `ADFMArchiveConfig` object.

To use the custom WLST commands for Oracle ADF, you must invoke the WLST script from the Oracle Common home. For more information about other WLST commands, such as custom Metadata Services (MDS) commands, see the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

## 4.2 ADF-Specific WLST Commands

Use the commands in Table 4–1 for ADF applications.

***Table 4–1 Browse Commands for WLST Configuration***

| Use this command... | To... | Use with WLST... |
|---|---|---|
| `adf_createFileUrlConnection` | Create a new ADF file connection. | Online or Offline |
| `adf_createHttpUrlConnection` | Create a new ADF URL connection. | Online or Offline |
| `adf_setURLConnectionAttributes` | Set or edit the attributes of a newly created or existing ADF connection. | Online or Offline |
| `adf_listUrlConnection` | List a new URL connection. | Online or Offline |

*Table 4–1 (Cont.) Browse Commands for WLST Configuration*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| `getADFMArchiveConfig` | Returns a handle to the `ADFMArchiveConfig` object for the specified archive. | Online or Offline |

## 4.2.1 adf_createFileUrlConnection

Use with WLST: Online or Offline.

### 4.2.1.1 Description

Use this command to creates a new connection based on the `oracle.adf.model.connection.url.FileURLConnection` connection class.

### 4.2.1.2 Syntax

```
adf_createFileURLConnection(appName, name, URL)
```

| Argument | Definition |
|---|---|
| `appName` | Application name for which the connection will be created. |
| `name` | The name of the new connection. |
| `URL` | The URL associated with this connection. |

### 4.2.1.3 Example

```
adf_createFileURLConnection('myapp','tempDir','/scratch/tmp')
```

## 4.2.2 adf_createHttpUrlConnection

Use with WLST: Online or Offline.

### 4.2.2.1 Description

Use this command to create a new connection based on the `oracle.adf.model.connection.url.HttpURLConnection` connection `type` class.

### 4.2.2.2 Syntax

```
adf_createHttpURLConnection (appName, name, [URL], [authenticationType], [realm],
[user], [password])
```

| Argument | Definition |
|---|---|
| *`appName`* | Application name for which the connection will be created. |
| *`name`* | The name of the new connection. |
| *`url`* | (Optional) The URL associated with this connection. |
| *`authenticationType`* | (Optional) The default is basic. |
| *`realm`* | (Optional) If this connection deals with authentication, then this should be set. The default is basic. |

| Argument | Definition |
|---|---|
| *user* | (Optional) |
| *password* | (Optional) |

#### 4.2.2.3 Example

```
adf_createHttpURLConnection('myapp','cnn','http://www.cnn.com')
```

### 4.2.3 adf_setURLConnectionAttributes

Use with WLST: Online or Offline.

#### 4.2.3.1 Description

Use this command to set or edit the attributes of a newly created or existing ADF connection.

#### 4.2.3.2 Syntax

```
adf_setURLConnectionAttributes(appname, connectionname, attributes)
```

| Argument | Definition |
|---|---|
| appname | Application name. |
| connectionname | The name of the connection. |
| *attributes* | The array containing attributes to set in key/value pairs. |

#### 4.2.3.3 Example

```
adf_setURLConnectionAttributes
('myapp','cnn','ChallengeAuthenticationType:digest',
'AuthenticationRealm:XMLRealm'
```

### 4.2.4 adf_listUrlConnection

Use with WLST: Online or Offline.

#### 4.2.4.1 Description

Use this command to list the connections of the application.

#### 4.2.4.2 Syntax

```
adf_listURLConnection(appname)
```

| Argument | Definition |
|---|---|
| appname | Application name. |

#### 4.2.4.3 Example

```
adf_listURLConnection ('myapp')
```

## 4.2.5 getADFMArchiveConfig

Use with WLST: Online or Offline.

### 4.2.5.1 Description

Returns a handle to the `ADFMArchiveConfig` object for the specified archive. The returned `ADFMArchiveConfig` object's methods can be used to change application configuration in an archive.

The `ADFMArchiveConfig` object provides the following methods:

- `setDatabaseJboSQLBuilder([value])` - Sets the Database `jbo.SQLBuilder` attribute.
- `getDatabaseJboSQLBuilder()` - Returns the current value of the `jbo.SQLBuilder` attribute.
- `setDatabaseJboSQLBuilderClass([value])` - Sets the Database `jbo.SQLBuilderClass` attribute. Value is the full name of the custom builder class.
- `getDatabaseJboSQLBuilderClass()` - Returns the current value of the `jbo.SQLBuilderClass` attribute.
- `setDefaultRowLimit([value])` - Sets the defaults `rowLimit` attribute. Value is a long specifying the row limit (Default -1).
- `getDefaultRowLimit()` - Returns the current value of the `rowLimit` attribute.
- `save([toLocation])` - If you specify the `toLocation`, then the changes will be stored in the target archive file and the original file will remain unchanged. Otherwise, the changes will be saved in the original file itself.

### 4.2.5.2 Syntax

```
archiveConfigObject = ADFMAdmin.getADFMArchiveConfig(fromLocation)
```

| Argument | Definition |
|---|---|
| *fromLocation* | The name of the ear file, including its complete path. |

The syntax for `setDatabaseJboSQLBuilder([value])` is:

```
archiveConfigObject.setDatabaseJboSQLBuilder([value])
```

| Argument | Definition |
|---|---|
| *value* | The value of the `jbo.SQLBuilder` attribute. Valid values are: 'Oracle' (Default), 'OLite', 'DB2', 'SQL92', 'SQLServer', or 'Custom. If 'Custom' is specified, then the `jbo.SQLBuilderClass` attribute should also be set. |

The syntax for `getDatabaseJboSQLBuilder()` is:

```
archiveConfigObject.getDatabaseJboSQLBuilder()
```

The syntax for `setDatabaseJboSQLBuilderClass([value])` is:

```
archiveConfigObject.setDatabaseJboSQLBuilderClass([value])
```

| Argument | Definition |
|---|---|
| *value* | The value of the jbo.SQLBuilderClass attribute. |

The syntax for getDatabaseJboSQLBuilderClass) is:

```
archiveConfigObject.getDatabaseJboSQLBuilderClass()
```

The syntax for setDefaultRowLimit([value]) is:

```
archiveConfigObject.setDefaultRowLimit([value])
```

| Argument | Definition |
|---|---|
| *value* | The value of the rowLimit attribute. |

The syntax for getDefaultRowLimit() is:

```
archiveConfigObject.getDefaultRowLimit([value])
```

The syntax for save([toLocation]) is:

```
archiveConfigObject.save([toLocation])
```

| Argument | Definition |
|---|---|
| *toLocation* | The file name along with the absolute path to store the changes. |

### 4.2.5.3 Example

In the following example. the jbo.SQLBuilder attribute is set to 'DB2'.

```
wls:/offline> archive =
               ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder(value='DB2')
wls:/offline> archive.save()
```

In the following example, the jbo.SQLBuilder attribute is removed so that application default is used.

```
wls:/offline> archive =
              ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder()
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

In the following example, the jbo.SQLBuilder attribute is set to 'Custom', and the jbo.SQLBuilderClass attribute is set to the class 'com.example.CustomBuilder'.

```
wls:/offline> archive =
              ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder('Custom')
wls:/offline> archive.setDatabaseJboSQLBuilderClass('com.example.CustomBuilder')
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

In the following example, the rowLimit attribute is set to 100.

```
wls:/offline> archive = getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDefaultRowLimit(100)
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

# Part III

# Administering ADF Mobile Client Applications

Part III contains the following chapters:

# 5

# Deploying ADF Mobile Client Applications

This chapter explains how to deploy ADF Mobile client applications to BlackBerry smartphones and Windows Mobile devices during production and later-stage testing. For information about deploying the mobile client applications for development, see *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

This chapter includes the following sections:



## 5.1 Introduction to Deploying ADF Mobile Client Applications

After the mobile client application has been developed using Oracle JDeveloper, you can deploy this application to the production environment. You use the following tools for production-level deployment:

- Oracle Database (Standard or Enterprise Edition) with administrative login credentials.
- Oracle Database Lite Mobile Server (Mobile Server) with administrative login credentials, configured with the mobile client application database synchronization.
- Mobile Sync Client (mSync)
- Packaged mobile client application to be deployed
- Windows Mobile device or BlackBerry smartphone

For information about deploying the mobile client applications for development and testing purposes using JDeveloper, see the following:

- "Deploying ADF Mobile Client Components" chapter of *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*
- *Oracle Database Lite Developer's Guide*

## 5.2 Deploying Applications to BlackBerry Smartphones

You use one of the following to deploy a mobile client application as a packaged COD file that includes platform-specific deployment profiles:

- BlackBerry Desktop Manager (see Section 5.2.1, "How to Deploy Using BlackBerry Desktop Manager")
- BlackBerry Enterprise Server (see Section 5.2.2, "How to Deploy Using BlackBerry Enterprise Server")
- Web server (see Section 5.2.3, "How to Deploy Using a Web Server")

The packaged application files are located in directories that were specified at development time during the deployment phase. For more information on the packaged files and their locations, see the following:

- Section 5.2.5, "What You May Need to Know About BlackBerry Deployment Files"
- "Deploying BlackBerry Applications" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*

For more information and details on production deployment to BlackBerry smartphones, see *How to Deploy and Distribute Applications* tutorial on BlackBerry forums.

### 5.2.1 How to Deploy Using BlackBerry Desktop Manager

You use BlackBerry Desktop Manager to deploy a packaged mobile client application to a single smartphone.

BlackBerry Desktop Manager uses BlackBerry ALX loader program to load COD files.

---

**Note:** BlackBerry Desktop Manager is included in the BlackBerry Desktop Software installation. For more information, see "How to Install BlackBerry Desktop Software" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

---

**To deploy the application using BlackBerry Desktop Manager:**

1. Ensure that you have both the application COD and ALX files available. These files were placed by JDeveloper in the `deploy` directory of your application during the deployment stage of development.
2. Connect the smartphone to the PC with the USB cable or Bluetooth.
3. Open BlackBerry Desktop Manager.
4. Click **Applications** tab. All applications installed on the smartphone are listed in the Applications pane.
5. To add an application, click **Import Files**.
6. In the dialog that appears, browse and select the ALX file of the mobile client application, and then click **OK**.
7. Click **Apply** to complete the deployment.

### 5.2.2 How to Deploy Using BlackBerry Enterprise Server

You can use BlackBerry Enterprise Server to deploy a packaged mobile client application (COD file) to a smartphone (see *How to Deploy and Distribute Applications* BlackBerry tutorial).

For more information on BlackBerry Enterprise Server and its usage, see the following BlackBerry Enterprise Server documentation:

- http://us.blackberry.com/apps-software/server
- http://docs.blackberry.com/en/admin/?userType=2

### 5.2.3 How to Deploy Using a Web Server

A packaged mobile client application can be distributed to the production environment using a Web server. This deployment method enables the end users to download the application from the provided URL and install it on their smartphone.

The process of uploading the application to the Web server varies depending on the following:

- The type of the Web server
- Access configurations

Typically, you distribute the application as follows:

- Place both the JAD file that references the application COD file, as well as the COD file, on the Web server.
- Point the BlackBerry smartphone to the location of the JAD file.

For more information, see BlackBerry documentation.

### 5.2.4 How to Run an Application on a BlackBerry Smartphone

After successful deployment, the mobile client application appears on the BlackBerry smartphone as an application, or an icon to run the application. To run the application, you would require ALX application loader file to load COD files.

For more information about COD and ALX files, see BlackBerry support forums at http://us.blackberry.com/support.

**To run the Application on BlackBerry Smartphone:**

1. Install the client database for the BlackBerry smartphone. For more information, see "About Using a Client Database" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.
2. Open the smartphone and load the application. The application would be available in the **Downloads** folder:
   - In the **Mobile Sync** page, enter the required details.
   - Navigate the selection to the **Sync** button.
   - Press **Enter** to run the application.

### 5.2.5 What You May Need to Know About BlackBerry Deployment Files

Table 5–1 lists files that you deploy to a BlackBerry smartphone.

*Table 5–1 BlackBerry Deployment Files*

| File Name | Location | Description | Used with |
|---|---|---|---|
| COD | Specified during deployment phase at development time. | A generic executable file that contains all application artifacts, including platform-specific deployment profiles.<br>Can be uploaded through a USB link or wirelessly. Appears on the smartphone as an application or icon. | 1. BlackBerry Desktop Manager<br>2. BlackBerry Enterprise Server<br>3. Web server |
| ALX | The location of the COD file. | The Application Loader XML file that communicates the location (on your computer) of the application to be installed. | BlackBerry Desktop Manager |
| JAD | There is no default location | Java Application Descriptor file that contains information about the application and the location of the application's COD file. | Web server |

## 5.3 Deploying Applications to Windows Mobile Devices

Deploying a packaged mobile client application to a Windows Mobile device is different from deploying to a BlackBerry smartphone: applications developed for Windows Mobile platform are packaged as CAB (cabinet) files and require usage of Windows Mobile-specific tools to enable deployment.

The CAB file includes the application JAR file, the launcher executable file, the options file, and platform-specific deployment profiles. For more information, see the following:

- "About the Launcher Executable File" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*
- Section 5.3.4, "What You May Need to Know About Windows Mobile Deployment Files"

For information on how to create an application package at development time, see "Deploying a Windows Mobile Application" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

You use one of the following to deploy a packaged mobile client application:

- Microsoft ActiveSync (see Section 5.3.1, "How to Deploy Using Microsoft ActiveSync")
- Oracle Database Lite Mobile Server (the Mobile Server) (see Section 5.3.2, "How to Deploy Using the Mobile Server")

### 5.3.1 How to Deploy Using Microsoft ActiveSync

You can use Microsoft ActiveSync 4.5 or Microsoft Windows Mobile Device Center 6.1 or later to deploy a packaged mobile client application.

For more information on Microsoft ActiveSync 4.5 and Microsoft Windows Mobile Device Center 6.1, see "How to Connect the Mobile Device or Emulator" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

Before you begin, ensure that you have deployed the Java Virtual Machine to the Windows Mobile device. For more information, see "Deploying the ADF Mobile Client Runtime" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

**To deploy the application using Microsoft ActiveSync:**

1. Connect the Windows Mobile device to the PC with the USB cable or Bluetooth.
2. Open Microsoft ActiveSync or Microsoft Windows Mobile Device Center.
3. Click **Explore** to open the Mobile Device directory.
4. Copy the application's CAB file to the Mobile Device directory, and then close the window.
5. Navigate to the CAB file on the device and click on it to trigger the execution, and then follow the installer prompts.

### 5.3.2 How to Deploy Using the Mobile Server

You can deploy a packaged mobile client application to the Mobile Server. For more information, see "Publishing Applications to the Mobile Server Repository" section in *Oracle Database Lite Administration and Deployment Guide*.

For additional information, see the following:

- *Oracle Database Lite SQLite Mobile Client Guide*
- *Oracle Database Lite Administration and Deployment Guide*

Once the application is deployed to the Mobile Server, it is available for download from a specific URL (see "Selecting Application Files for Public Use" section in *Oracle Database Lite Administration and Deployment Guide*).

Once the application is installed on the user device, you can perform a variety of administrative tasks, such as the following:

- Trigger the Mobile Server's device management function to push a new version of the application or a patch to the device. For more information, see "Managing Device Software Updates" section in *Oracle Database Lite Administration and Deployment Guide*.
- Introduce the device management functionality in the synchronization client and use the synchronization client running on the device. For more information, see "Managing Your SQLite Mobile Client" chapter of *Oracle Database Lite SQLite Mobile Client Guide*.
- Troubleshoot and tune the application. For more information, see *Oracle Database Lite Administration and Deployment Guide.*

### 5.3.3 How to Run an Application on a Windows Mobile Device

After successful deployment, the mobile client application is ready to run on the Windows Mobile device.

**To run the application on a Windows Mobile device:**

1. Install the client database for the Windows Mobile device. For more information, see "About Using a Client Database" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.
2. In the Windows Mobile device, open File Explorer, and then run the application as follows:
   - In the **Mobile Sync** page, enter the required details.
   - Navigate the selection to the **Sync** button.
   - Press **Enter** to run the application.

### 5.3.4 What You May Need to Know About Windows Mobile Deployment Files

Table 5–2 lists files that you deploy to a Windows Mobile device.

***Table 5–2 Windows Mobile Deployment Files***

| File Name | Location | Description | Used with |
|---|---|---|---|
| CAB | Specified during deployment phase at development time. | The cabinet file is a library of compressed files stored as a single file. Cabinet files are used to organize installation files that are copied to the user's system.<br>This file includes an application JAR file, a launcher executable file, and an options file. | 1. Microsoft ActiveSync<br>2. Microsoft Windows Mobile Device Center<br>3. Oracle Database Lite Mobile Server |

## 5.4 Configuring Applications for Client Database Connection

For information, see the following:

- "About Synchronizing Data with Oracle Mobile Server" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*
- "Specifying the Client Database Location for an Application" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*
- "Working Directly with the Database" chapter in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*
- *Oracle Database Lite Administration and Deployment Guide*

## 5.5 Monitoring Data Synchronization

When the end user synchronizes data from the mobile device or smartphone, all changes made by this user are updated on Oracle Database Lite Mobile Server (the Mobile Server). The Mobile Server also updates itself with any changes made on the base database server by the base ADF application. For more information, see *Oracle Database Lite Administration and Deployment Guide*.

# 6

# Deploying and Configuring ADF Mobile Transaction Replay Service

This chapter describes how to deploy and configure ADF Mobile transaction replay service during production. For information about using the transaction replay service during development, see *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

This chapter includes the following sections:



## 6.1 Introduction to ADF Mobile Transaction Replay Service

ADF Mobile transaction replay service (the transaction replay service) enables mobile client applications running on clients to access and execute business logic components on the server or on the base ADF application. For more information, see "Transaction Synchronization" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

The transaction replay service functionality is enabled by the interaction of the following elements:

- ADF Mobile Transaction Replay Service Schema
- Replay Items
- ADF Mobile Transaction Replay Service Components

**Note:** To use the transaction replay service, it must be enabled and configured at development time. For more information, see "Configuring ADF Mobile Transaction Replay Service in JDeveloper" section in *Oracle Fusion Middleware Installation Guide for ADF Mobile Trans action Replay Service*.

### 6.1.1 ADF Mobile Transaction Replay Service Schema

The transaction replay service schema is composed of the following database tables:

- `REPLAY_TYPE`
- `REPLAY_ITEM`
- `REPLAY_STATUS`
- `REPLAY_USERS`
- `REPLAY_JOBS`

A database trigger is also set as part of the `REPLAY_ITEM` table, which notifies the transaction replay service infrastructure that a new transaction from a disconnected client requires processing. The `REPLAY_USERS` and `REPLAY_JOBS` tables are internal tables used by components of the transaction replay service architecture during processing of transactions, storing content on disconnected clients, and storing transactions to be replayed, respectively.

The `REPLAY_ITEM` and `REPLAY_TYPE` tables contain content visible to administrators and represent disconnected transactions from identified clients (replay items) and the transactions mapped business service (replay types).

The database trigger is set on the replay items table and is triggered by the insertion of a new record; that is, a new transaction entry from a disconnected client. The database trigger sets a flag in the `REPLAY_USERS` table for the client posting the new transaction from a client.

For more information about schema tables, see Appendix C, "ADF Mobile Transaction Replay Service Tables."

### 6.1.2 Replay Items

A *replay item* represents an action created by a disconnected client that is to be replayed on the enterprise server. Each replay item is a single record stored in the `REPLAY_ITEM` table of the transaction replay service schema. The attributes for the replay item record contain information that describes the disconnected transaction, including any parameters, and also references an entry in the `REPLAY_TYPE` table that maps the transaction to respective business service information.

The replay items record also contains identification information on the record itself, the disconnected client creating the transaction, status of the transaction replay, and authentication data.

A replay item is a logical entity that describes the client events and activities that modify the domain state of the client. The form of a replay item differs depending on the location, context, and the state of the data. A replay item contains sufficient information to ensure that server applications maintain domain write consistency. During a synchronization session, a row that has been updated by a source other than the current synchronization node since it was downloaded, cannot be updated or deleted.

The sync node repository stores the previous synchronization time. The user application maintains the time stamps for the last updates on each record.

#### 6.1.2.1 Replay Item Dependency

The `REPLAY_ITEM` table has a column called `DEPENDENCY`. For each replay item, this column stores the ID of another replay item (parent) on which it has a dependency. The ID value stored in the `DEPENDENCY` column is `null` when a replay item does not have a parent replay item. To ensure data integrity, the transaction replay service does not process a replay item if its parent replay item's status is not successfully applied.

### 6.1.3 Parameter Name-Value Pair Format

The parameter-name value pair of the mobile client application can be logged using various methods, such as store the parameter information on the client in XML format, or create a parent-child relationship and store the pair information in parent-child tables.

#### 6.1.3.1 Storing the Name-Value Pair on the Client in XML Format

Storing the parameter list as an XML node provides the advantages of XML. However, it can also result in slower parsing and larger data.

The XML schema (see Example 6–1) reconstructs the method signature and therefore enables support for method overloading.

***Example 6–1 XML Schema***

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://tempuri.org/XMLSchema.xsd"
           elementFormDefault="qualified"
           xmlns="http://tempuri.org/XMLSchema.xsd"
           xmlns:mstns="http://tempuri.org/XMLSchema.xsd"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="Parameter">
      <xs:complexType>
         <xs:sequence />
         <xs:attribute name="name" type="xs:string" />
         <xs:attribute name="type" type="xs:string" use="required" />
         <xs:attribute name="value" type="xs:string" use="required" />
         <xs:attribute name="order" type="xs:positiveInteger" use="required" />
      </xs:complexType>
   </xs:element>
</xs:schema>
```

The self-defining parameter list in the XML document (see Example 6–2) supports method overloading. The parameter description field contains the parameter name, its data type (such as `String` or `Date`) and associated value, the parameter's order and the actual data to be passed to the target method.

***Example 6–2 The Self-Defining Parameter List***

```
<?xml version="1.0"?>
<note xmlns="http://tempuri.org/XMLSchema.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://tempuri.org/XMLSchema.xsd xmfile1.xsd">
   <Parameter Name="SR_ID" Type="string" Value="102032" Order="1" />
   <Parameter Name="OWNER_ID" Type="string" Value="100" Order="2" />
   <Parameter Name="TIME_CR" Type="date" Value="03/22/2006 10:23:33" Order="3" />
</note>
```

**Note:** You can add the `Old Value` attribute to the parameter description field as `oldValue`.

## 6.1.4 ADF Mobile Transaction Replay Service Components

The transaction replay service incudes the following components:

- TRS Dispatcher Component
- TRS TxnReplayer Component
- TRS Secure Web Service Component

### 6.1.4.1 TRS Dispatcher Component

The TRS Dispatcher component is a Java EE servlet implemented as a polling thread in the Web server. The TRS Dispatcher batches new transactions submitted by a disconnected client and forwards these transactions to the TRS TxnReplayer Component for replaying on the enterprise server. In the overall transaction replay service architecture, the TRS Dispatcher functions as a singleton component.

In further detail, the TRS Dispatcher component monitors the TRS Users table (`REPLAY_USERS`) for a flag (`MORE_FLAG`) set by the database trigger indicating that a new transaction is ready for processing. The TRS Dispatcher component creates a new row in the TRS Jobs table (`REPLAY_JOBS`), and submits the job to the TRS TxnReplayer component using `HTTP GET` commands. Submitting jobs as `HTTP GET` commands allows the use of the transaction replay service with any third-party Web server load balancing options. The TRS Dispatcher component then sets the status of the job to `new`.

### 6.1.4.2 TRS TxnReplayer Component

The TRS TxnReplayer component is a Java EE servlet that starts an asynchronous thread to process replay items. The TRS TxnReplayer receives a request to process replay items through a job submitted from the TRS Dispatcher Component. A job processes one or more replay items for the associated user. In the overall transaction replay service architecture, there can be several instances of the TRS TxnReplayer component to assist with load balancing.

In further detail, the TRS TxnReplayer component receives a request for transaction replay from a job submitted by the TRS Dispatcher component. The TRS TxnReplayer component queries the TRS Replay Items table (`REPLAY_ITEMS`) using information recorded in the submitted job for that client, and parses the replay items for any necessary parameters required by the method calls associated with the particular transaction type. The method calls are made one by one and in the order that they are received; that is, replay items from the same client are processed in order. Replay items for different clients are processed concurrently.

The TRS TxnReplayer component updates the status of replay items during processing. The replay item status is also synchronized back to the disconnected client from where it originated to complete the cycle of the transaction replay.

### 6.1.4.3 TRS Secure Web Service Component

The TRS Secure Web Service is a JAX-WS Web service, which can be implemented as one solution to post disconnected client transactions to the enterprise server for replaying through the transaction replay service architecture. For other transaction posting solutions and further information on the disconnected client's integration with

the transaction replay service architecture, see Section 6.1.5, "Disconnected Client Interaction With ADF Mobile Transaction Replay Service Architecture".

The TRS Secure Web Service is deployed to the main enterprise server and receives calls from disconnected clients that require access to the transaction replay service architecture. The disconnected client provides user credentials, and the Web service authenticates and authorizes users for each method accessed by the disconnected client. For further details on security with the TRS Secure Web Service, see *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

TRS Secure Web Service issues `@SecurityPolicy` annotation to attach security policies. The security policy is an XML file that contains security rules. For example, in the case of a user name token policy, the user name and password requirement is a security rule. The authentication policy and the authorization policy are attached to the TRS Web service.

To access the Web service, the mobile client application provides user name and password. After the user credentials are verified, the user is verified against the authorization policy to confirm the permissions granted for the requested operation.

The TRS Secure Web Service has three public methods for use by disconnected clients:

- `getPublicKey`: Retrieves TRS public key in clear text for encryption.
- `insertReplayItem`: Inserts a replay item into the `REPLAY_ITEM` table on the enterprise application server. This method uses `TRSConnectionMgr` to insert the replay item into the database.
- `hasOutstandingReplayItems`: Checks if a given user's replay items have completed processing.
- `validateCredential`: Validates the user name and password for a given replay type.

Using these methods, the TRS Secure Web Service can post disconnected client transactions to the transaction replay service for replaying at the enterprise server. The service can also query these replay items for their status. If replay items do not complete, error messages are logged on the enterprise server side for review.

#### 6.1.4.4 ADF Mobile Transaction Replay Service Failover and Scalability

The transaction replay service architecture is scalable and designed for high-availability and failover. The transaction replay service can be deployed using various configurations to make sure the transaction replay service architecture is highly available for use by disconnected clients. For example, using third-party clustering services, the TRS Dispatcher Component, which can only have one instance active per node, can reside on both a primary and secondary node, and if the primary dispatcher fails, the secondary or passive node TRS Dispatcher component can be brought online to maintain availability of the architecture.

Another example of high-availability design is the deployment of the TRS TxnReplayer Component, which can have several instances running simultaneously. It maintains high-availability by use of a third-party HTTP load balancer, which balances the job requests between available components. These scenarios are examples of high-availability solutions; your individual deployment may differ.

### 6.1.5 Disconnected Client Interaction With ADF Mobile Transaction Replay Service Architecture

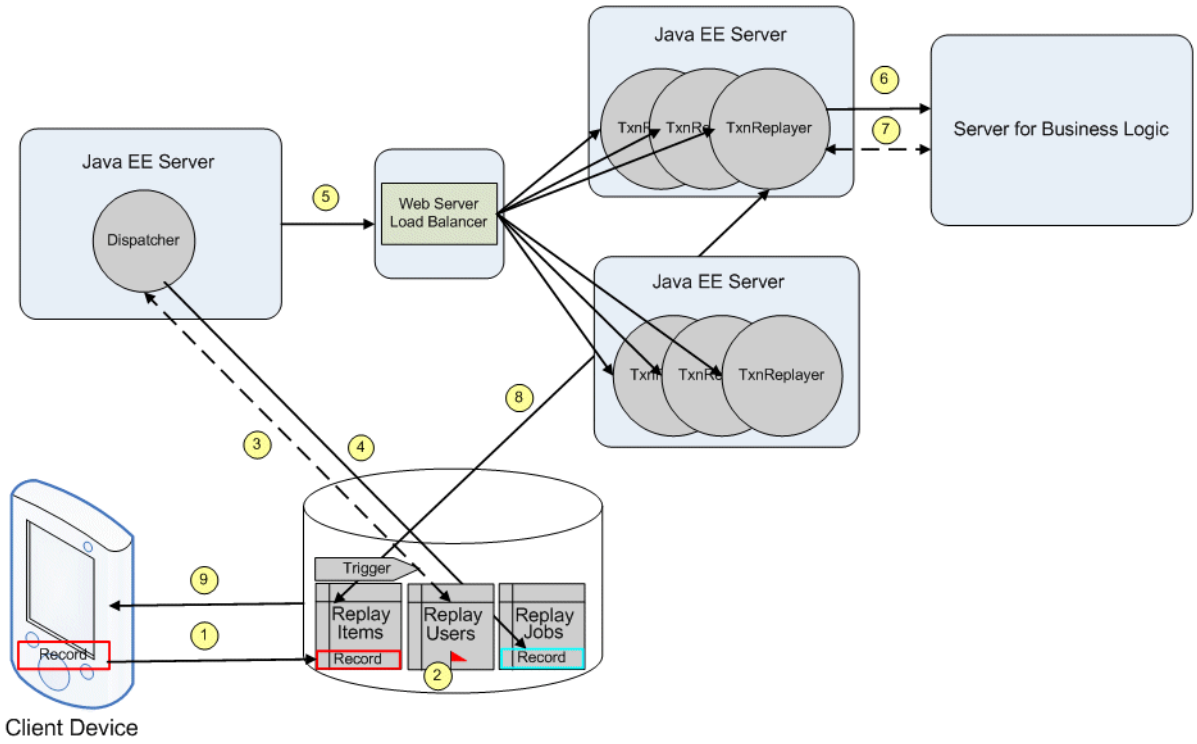
The transaction replay service architecture is independent of the type of disconnected client accessing the replay service on the enterprise server side of the application.

Any disconnected client configuration with the transaction replay service must be able to populate the `REPLAY_ITEM` table in the transaction replay service schema, as well as receive status or logging updates produced by the transaction replay service architecture.

### 6.1.6 Transaction Process Flow

The high-level interaction between various components of the transaction replay service architecture during a transaction replay is illustrated Figure 6–1.

***Figure 6–1 ADF Mobile Transaction Replay Service Transaction Process***




1. The synchronization service runs, adding a record to the `REPLAY_ITEM` table. This record represents a transaction from a disconnected client requiring replay on the enterprise application server.
2. The new `REPLAY_ITEM` record triggers a flag to be set on the `REPLAY_USER` table.
3. The TRS Dispatcher Component, which periodically reviews the `REPLAY_USER` table, detects the flag.
4. After detecting the flag, the TRS Dispatcher component creates a row in the `REPLAY_JOBS` table
5. The TRS Dispatcher submits the job to the TRS TxnReplayer Component, usually through a third-party load balancer.

6. The TRS TxnReplayer processes the items in the REPLAY_ITEM table with information from the REPLAY_TYPE table (not shown). At this stage of the process, the transaction is replayed on the server containing the application business logic.
7. The TRS TxnReplayer component monitors the progress of the transaction replay.
8. TRS TxnReplayer component updates the status of each replay item during processing.
9. After the completion of the transaction on the enterprise server, the status of the replay item is updated and synchronized back to the disconnected client.

## 6.2 Configuring ADF Mobile Transaction Replay Service Security

The transaction replay service takes business service method calls from a disconnected mobile device and replays the calls on the server side. The mobile client application authenticates users without any direct connection to the server, and then synchronizes a batch of business service calls to the REPLAY_ITEM table on the server. The transaction replay service reads data from the REPLAY_ITEM table to get the user identity. The replay service replays the business service calls asynchronously without any user interface. If user authentication is required, the transaction replay service obtains a security token for that given user and propagates it to the business service.

The security in the transaction replay service is configured in two parts:

- Inbound security (see Section 6.2.1, "How to Configure Inbound Security")
- Outbound security (see Section 6.2.2, "How to Configure Outbound Security")

### 6.2.1 How to Configure Inbound Security

Inbound security is responsible for the integrity of data being entered into the REPLAY_ITEM table on the server. The server-side REPLAY_ITEM table is protected by server database access control. Only the administrator has privilege to access the REPLAY_ITEM table. Additionally, transaction replay service requires that the synchronization tools selected by end users must implement proper access control to allow only authorized users to synchronize data to the REPLAY_ITEM table.

Transaction replay service also requires that client applications implement authentication and authorization to allow only trustworthy data being entered into the device database.

Under these requirements, the transaction replay service assumes that the data in the REPLAY_ITEM table is always trustworthy.

### 6.2.2 How to Configure Outbound Security

Outbound security is responsible for establishing the user's credential for authentication during replaying a business service call. The transaction replay service supports credentials consisting of a user name and a password, and secures the password using credential store framework or public key cryptography.

#### 6.2.2.1 Securing Password Using Credential Store Framework

The Credential Store Framework (CSF) provides a mechanism for secure storage and management of credentials for Java-based applications. The CSF API supports two types of storage:

- File-based credentials, which are stored in the Oracle wallet.

- Repository-based credentials, which are stored in the LDAP directory.

Using CSF, transaction replay service obtains a user's user name and password from the server storage without having the mobile client application sending the password, and hence avoiding the encryption-decryption during transmission.

For more information, see "Developing with the Credential Store Framework" section in *Oracle Fusion Middleware Application Security Guide*.

One limitation of using a credential store is that it relies on a server-side password storage and it may require extra effort to integrate with existing user administration.

#### 6.2.2.2 Securing Password Using Public Key Cryptography

Public key cryptography, also known as asymmetric cryptography, is a form of cryptography in which a user has a pair of cryptographic keys–a public key and a private key. The private key is kept secret, while the public key may be widely distributed. The keys are related mathematically, but the private key cannot be practically derived from the public key. A message encrypted with the public key can be decrypted only with the corresponding private key.

To use public key cryptography to secure user passwords, the transaction replay service generates a pair of a public and a private key. It then publishes or distributes the public key. The client application uses the public key to encrypt its password and stores the encrypted password in the replay item. The password can then be transmitted safely. Only the transaction replay service can decrypt it using its private key.

To implement public key cryptography, the following steps are involved:

1. Creating a keystore:

   According to Java Cryptography Architecture (JCA), a keystore is a database of keys. Private keys in a keystore have a certificate chain associated with them, which authenticates the corresponding public key. A keystore also contains certificates from trusted entities.

   The transaction replay service supports two types of keystores, the Java Key Store and the Oracle Wallet. For more information about creating and using key stores for message protection, see the section about managing keystores, wallets, and certificates in the *Oracle Fusion Middleware Administrator's Guide*.

2. Creating a certificate and a key pair:

   In cryptography, a public key certificate (or identity certificate) is an electronic document which incorporates a digital signature to bind a public key with an identity—information such as the name of a person or an organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual.

   To use the public key cryptography in the transaction replay service, you create a certificate in a keystore. The certificate does not necessary belongs to an individual. In Java Key Store, the `keytool` command creates a certificate with an alias. In Oracle wallet, the alias is the Distinguished Name (DN) in a directory service of the certificate. The transaction replay service can retrieve a certificate's public key and private key from the keystore by its alias.

   The transaction replay service requires the following parameters:

   - `Keystore URL`: the file location of the keystore.
   - `KeystoreType`: JKS for Sun's Java Key Store, PKCS12 for Oracle's `ewallet.p12` file, or SSO for Oracle's `cwallet.sso`.

- KeystorePassword: The password to open the keystore.
- CertificateAlias
- keyPass (optional): The key created by Sun's keytool can have a password to protect each individual key. Therefore, the transaction replay service allows for configuration of a keyPass password to open the private key.
- Provider (optional): To open Oracle wallet, a security provider of value OraclePKI is needed in the keyStore.getInstance method.

The following code shows how to open a key store to retrieve a certificate and its key pair.

```
KeyStore ks = KeyStore.getInstance(KeystoreType, Provider);
char[] kspwd = KeystorePassword.toCharArray();
char[] keypwd = keyPass.toCharArray();
URL url = new URL(KeyStoreFile);
ks.load(url.openStream(), kspwd);
Certificate cert = ks.getCertificate(CertificateAlias);
PublicKey pubKey = cert.getPublicKey();
PrivateKey privKey = ks.getKey(CertificateAlias,keypwd);
```

Due to the secrecy of the KeystorePassword and keyPass, these parameters are not stored and configured directly. Instead, you must provision two password credentials in CSF for these two passwords. The following four parameters are needed for the transaction replay service to locate them in CSF:

- Keystore Password CSF Map
- Keystore Password CSF Key
- Key Password CSF Map
- Key Password CSF Key

3. Decrypting the password:

   The transaction replay service uses private key to decrypt the password.

4. Applying PKE to the USER_PASSWORD column in REPLAY_ITEM:

   The user passwords are transmitted from client to server in the USER_PASSWORD column in the REPLAY_ITEM table. The USER_PASSWORD's type is VCHAR2(256). The maximum length of user password is 117 characters. The encrypted text is 128 bytes of binary data. To ease the transmission, transaction replay service requires the encrypted text to be stored using base64 encoding in the USER_PASSWORD column. The encoded string has a length of 172 bytes.

### 6.2.3 How to Enable Authentication and Authorization

When the user starts a mobile application for the first time after installation, they are prompted by the login page to enter the login credentials (user name and password) to establish a connection to the TRS Secure Web Service. The server performs a synchronous authentication at this time: if the credentials are correct, the TRS Secure Web Service returns its public key to the client.

The transaction replay service requires a user name and password be provided for each replay item (see Section 6.1.2, "Replay Items"). To avoid sending password in the REPLAY_ITEM table in clear text, the transaction replay service implements public key encryption to securely exchange passwords from the mobile client to the server. The mobile client obtains the transaction replay service public key in the authentication process. The client then encrypts the user password using the public key.

The `getPublicKey` method is implemented by the TRS Secure Web Service. It provides two web service calls for clients to use. At the time of client login, the following happens:

1. First, the mobile client issues the `getPublicKey` to get the `publicKey` as a `String`
2. The mobile client encrypts password using `publicKey` and makes the second call `validateCredential` to the same web service. The values of replay type name, user name, and public key encrypted password are passed as parameters. The method returns 1 for the valid user and the encrypted password, or 0 for the invalid user.

For more information about setting public key encryption in the transaction replay service, see "Configure Public Key Encryption for ADF Mobile Transaction Replay Service" section in *Oracle Fusion Middleware Installation Guide for ADF Mobile Trans action Replay Service*.

When the transaction replay service replays transactions, the replayer component honors access control and permissions associated with the user credential of the transaction. For example, if the user that generated a transaction has read-only access to Entity Object A, and the transaction requires update of Entity Object A, then the replay will fail and an error will be returned to the user.

The transaction replay service supports ADF authorization and access control. For more information, see the following:

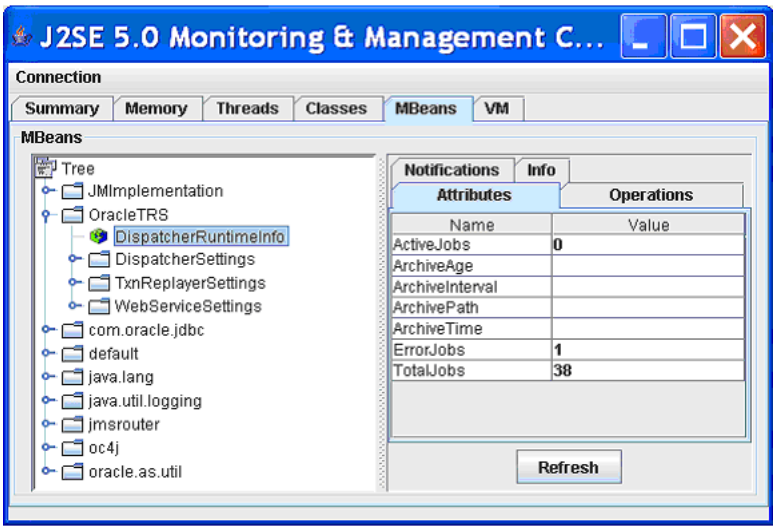
- "Enabling ADF Security" section in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
- "How to Define Policies for Data" section in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*

When the end user fails to authenticate, an error message is displayed. You can configure the error message using EL expressions. For more information, see "Authentication" section in *Oracle Fusion Middleware Mobile Client Developer's Guide for Oracle Application Development Framework*.

## 6.3 Managing ADF Mobile Transaction Replay Service Using MBeans

You configure and manage the Dispatcher and the transaction replay service using standard MBeans that contain interfaces for both the MBean implementation (inputs and outputs) and behavior. All of the transaction replay service-related MBeans are in the OracleTRS domain. Figure 6–2 illustrates the `OracleTRS` domain within the navigation tree in JConsole, a JMX-compliant console that you can use to manage transaction replay service. You can also manage the transaction replay service using MBean operations exposed through the Oracle Enterprise Manager Fusion Middleware Control console.

*Figure 6–2 The OracleTRS Domain in JConsole*




## 6.3.1 How to Manage the Dispatcher

Table 6–5 lists attributes of the `DispatcherSettings` MBean that allow you to configure the Dispatcher. You can also monitor the dispatcher using the `CurrentJobs`, `JobStatus`, and `JobInfo` operations of the `DispatcherSettings` MBean.

*Table 6–1 Attributes of the DispatcherSettings MBean*

| Attribute | Description |
|---|---|
| `URL` | The URL used to submit a transaction replay job. Changes to this attribute take effect after the service restarts. |
| `DataSourceName` | The name of the data source that contains the transaction replay tables. Changes to this attribute take effect after the service restarts. |
| `BackupDataSourceName` | The name of the backup data source that contain the transaction replay table. Changes to this attribute take effect after the service restarts. |
| `NumberOfDatabaseRetries` | The maximum number of times that the transaction replay service attempts to establish a database connection if a database connection fails. Changes to this attribute take effect after the service restarts. |
| `DatabaseConnectionRetryInterval` | The retry interval (in milliseconds) to obtain a database connection. Changes to this attribute take effect after the service restarts. |
| `NumberOfDispatcherDBRetries` | The number of database operation retries attempted by the Dispatcher. Changes to this attribute take effect after the service restarts. |
| `DispatcherDBRetryInterval` | The retry interval (in milliseconds) for the Dispatcher database operations. Changes to this attribute take effect after the service restarts. |
| `LoopDelay` | The time (in milliseconds) to pause before reading and processing transaction replay jobs. Changes to this attribute take effect after the service restarts. |

*Table 6–1 (Cont.) Attributes of the DispatcherSettings MBean*

| Attribute | Description |
|---|---|
| `NumberOfJobRetries` | The number of consecutive retries for the Dispatcher to resubmit a job for a job with an error status. |
| `JobSubmittedTimeout` | The timeout interval (in minutes) before a job is timed-out (stuck at Status 1 or 2). |
| `JobProcessingTimeout` | The timeout interval (in minutes) before a job is timed out, when stuck at Status 3. |
| `ItemProcessingTimeout` | The timeout interval (in minutes) before a txnreplayer is timed out. |
| `URLConnectionTimeout` | The time-out value (in seconds). This time-out applies to the worker servlet URL through which the Dispatcher submits jobs to the worker servlet. The default value is 60 seconds. |
| `URLReadTimeout` | The time-out value (in seconds). This time-out applies to the worker servlet URL through which the Dispatcher submits jobs to the worker servlet. The default value is 60 seconds. |

## 6.3.2 How to Configure the Dispatcher Runtime

The `DispatcherRuntimeInfo` MBean lets you view and affect the run-time information from the transaction replay service system.

The `DispatcherRuntimeInfo` MBean extends the `Timer` MBean class to send regular timer notifications for automatic archival of replay items and replay jobs. It also has the inner `DispatcherArchiveListener` class to receive the timer notifications and perform the actual archiving process.

Table 6–2 lists attributes of the `DispatcherRuntimeInfo` MBean.

*Table 6–2 DispatcherRuntimeInfo MBean Attributes*

| Attributes | Description |
|---|---|
| `ArchiveTime` | Indicates the start date and time for the auto-archiving process. The format for the start time is `DD-MON-YY HH:mm`. For example, `14-Feb-08 13:00`. The time is in military time format and should be the local time of the host on which the Dispatcher runs. The start time of the auto-archival process is after the current date and time. |
| `ArchiveInterval` | Indicates the time interval (in minutes) between each auto archival notification. For example, setting the value to 1440 minutes would set the auto archiving process to once a day. A minimum interval could be 1 minute. |
| `ArchiveAge` | Indicates how old (in days) records must be to be removed. For example, if you set this value to 30, then all records older than 30 calendar days at the time of auto-archival are removed.<br>Note that for all time calculations, UTC time is used. |
| `ArchivePath` | An optional attribute that indicates the archival directory where archive files with deleted records from the `REPLAY_ITEM` and `REPLAY_JOBS` tables are stored. For example, `C:\logs`.<br>For more information, see Section 6.3.2.1, "Using the ArchivePath Attribute." |

#### 6.3.2.1 Using the ArchivePath Attribute

Consider the following when using the `ArchivePath` attribute of the `DispatcherRuntimeInfo` MBean:

- If the archive directory path is provided (that is, it is not represented by an empty string), then prior to removing the targeted records, each record that meets the target criteria is saved to an archive file in the directory defined by the `ArchivePath` attribute. The archive log file name starts with a date encoding of the removal date, which facilitates sorting of all files in the directory and allows to identify the archived table:

  `yyyymmddHHmm_TRS_Replay_Items.log` (for replay items)

  `yyyymmddHHmm_TRS_Replay_Jobs.log` (for replay jobs)

  where:

  - `yyyy` represents the year
  - `mm` represents the month
  - `dd` represents the date
  - `HH` represents hours (in military time)
  - `mm` represents minutes

  For example, log files named `200802142035_TRS_Replay_Items.log` and `200802142035_TRS_Replay_Jobs.log` would contain all replay items and replay jobs older than 8:35 pm of February 14, 2008.

  The archive files contain all column data (including the column names) for each replay item and replay job record. The data are delimited by the vertical bar (`'|'`) and can be examined using Microsoft Excel.

- If writing of the archive files fails for any reason (for example, the disk is full), then the auto-archiving process stops and records are not removed from the `REPLAY_ITEM` and `REPLAY_JOBS` tables. If the issue is resolved, these records are archived when the next notification is received.
- If the archive files were successfully created, records that meet the target criteria are removed from the `REPLAY_ITEM` and `REPLAY_JOBS` tables.
- If the archive directory path was not provided and the archive files were not created, records that meet the target criteria are still removed from the `REPLAY_ITEM` and `REPLAY_JOBS` tables.

#### 6.3.2.2 What You May Need to Know About the Replay Items Removal Criteria

All replay items whose processed date is earlier than the remove date and whose status matches one of the statuses shown in Table 6–3 are removed from the `REPLAY_ITEM` table during the auto-archive operation.

***Table 6–3 Replay Items Removal Status***

| Status ID | Status |
|---|---|
| 3 | Successfully Applied |
| 4 | Error Encountered |
| 5 | Parent Error |
| 6 | Invalid Parent |
| 7 | Credential Error |

*Table 6–3 (Cont.) Replay Items Removal Status*

| Status ID | Status |
|---|---|
| 8 | Parameter Parsing Error |
| 9 | Replay Type Error |
| 11 | Error – Reviewed, No Action |

Replay items whose status matches one of the statuses shown in Table 6–4 cannot be auto-archived.

*Table 6–4 Replay Items Status That Prevent Removal*

| Status ID | Status |
|---|---|
| 1 | Pending |
| 2 | Processing |
| 10 | To be reviewed by administrator |

#### 6.3.2.3 What You May Need to Know About the Replay Jobs Removal Criteria

Replay jobs that match all of the following three criteria are removed from the `REPLAY_JOBS` table during the auto-archive operation:

1. Status matches one of the statuses shown in the following table:

| Status ID | Status |
|---|---|
| 4 | Done |
| 5 | Error |

2. An updated date is earlier than the remove date.
3. Not referenced by a record in the `REPLAY_USERS` table.

#### 6.3.2.4 What You May Need to Know About the Attribute Persistence

Using the Preferences API defined in the `java.util.prefs` package, the transaction replay service persists the auto archive attributes to the `oracle/txnreplay/autoarchive` node. This ensures that the process of auto archiving resumes automatically if the server is restarted.

The transaction replay service accesses the same node when it needs to remove attributes.

### 6.3.3 How to Monitor the Dispatcher

The `DispatcherRuntimeInfo` MBean exposes methods listed in Table 6–5.

*Table 6–5 DispatcherRuntimeInfo MBean Methods*

| Method | Description |
|---|---|
| `ReplayItemList` | Returns a list of replay items currently in the system. |
| `ReplayJobList` | Returns a list of jobs currently in the system. |
| `ReplayTypeList` | Returns a list of replay types currently defined in the system. |

*Table 6–5 (Cont.) DispatcherRuntimeInfo MBean Methods*

| Method | Description |
|---|---|
| `ReplayUserList` | Returns a list of replay users currently in the system. |
| `AddReplayType` | Adds a replay type to the transaction replay service system. |
| `RemoveReplayType` | Removes a reply type from the system. |
| `UpdateReplayType` | Updates a replay type. |
| `RemoveReplayItems` | Removes a reply type. |
| `RemoveReplayJobs` | Removes reply items from the `REPLAY_ITEMS` table. |
| `RemoveReplayUsers` | Removes users from the `REPLAY_USERS` table based on updated dates. |
| `RemoveReplayUser` | Removes a specific user from the `REPLAY_USERS` table. |
| `GetReplayItemDependency` | Returns a replay item, which is a dependency for the supplied item ID. |
| `AutoArchive` | Sends a timer notification to start a regular automatic archival process of replay items and replay jobs from the `REPLAY_ITEM` and `REPLAY_JOBS` tables, respectively.<br>For more information, Section 6.3.3.1, "What You May Need to Know About AutoArchive and StopAutoArchive Operations." |
| `StopAutoArchive` | By removing the auto archival notification from the timer, stops the automatic archival process for replay items and replay jobs from the `REPLAY_ITEM` and `REPLAY_JOBS` tables, respectively.<br>For more information, Section 6.3.3.1, "What You May Need to Know About AutoArchive and StopAutoArchive Operations." |

#### 6.3.3.1 What You May Need to Know About AutoArchive and StopAutoArchive Operations

If the auto archive notification is successfully created during the `AutoArchive` operation, the auto archive attributes are saved to the `oracle/txnreplay/autoarchive` node that contains persisted preferences settings. If the server is restarted, these attributes are retrieved and the auto archive process is automatically restarted.

If the auto archive notification is successfully removed during the `StopAutoArchive` operation, the auto archive attributes are removed from the `oracle/txnreplay/autoarchive` node.

### 6.3.4 How to Configure and Administer ADF Mobile Transaction Replay Service

The `TransactionReplaySettings` MBean lets you configure and manage the transaction replay service. This MBean has the following two interfaces:

- `TxnReplayerSettingsMBean`
- `TxnReplayerSettings`

The attributes of `TxnReplayerSettingsMBean` that Table 6–6 lists allow you to configure the behavior of the transaction replay service.

*Table 6–6 Attributes of the TxnReplayerSettinsMBean Interface*

| Attribute | Description |
|---|---|
| NumThreads | The number of Transaction Replayer threads on the server. This change takes effect when the service restarts. |
| DataSourceName | The name of the data source that contains the transaction replay tables. Changes to this attribute take effect when the service restarts. |
| BackupDataSourceName | The name of the backup data source that contains the transaction replay tables. Changes to this attribute take effect when the service restarts. |
| DBRetries | The maximum number of times that the transaction replay service attempts to get a database connection if the database connection fails. Changes to this attribute take effect when the services restarts. |
| DatabaseConnectionRetryInterval | The retry interval (in milliseconds) to get the database connection. Changes to this attribute take effect after the service restarts. |
| NumberofTxnReplayerDBRetries | The number of database operation retries for TxnReplayer. Changes to this attribute take effect after the service restarts. |
| TxnReplayerDBRetryInterval | The retry level (in milliseconds) for TxnReplayer database operations. Changes to this attribute take effect after the service restarts. |
| XSDURL | The location of the XML schema that is used to validate the XML parameters for each replay item. Changes to this attribute take effect after the service restarts. |

#### 6.3.4.1 Configuring Replay Types

For information on how to create and configure the transaction replay service replay types, see "Creating Transaction Replay Type" section in *Oracle Fusion Middleware Installation Guide for ADF Mobile Transaction Replay Service*.

#### 6.3.4.2 Troubleshooting Transactions

Table 6–7 shows values from the REPLAY_STATUS table that you can use to monitor status of the transaction replay service transactions.

*Table 6–7 Replay Status Values*

| Status_Id | Status |
|---|---|
| 1 | Pending |
| 2 | Processing |
| 3 | Successfully Applied |
| 4 | Error Encountered |
| 5 | Parent Error |
| 6 | Invalid Parent |
| 7 | Credential Error |
| 8 | Parameter Parsing Error |
| 9 | Replay Type Error |

*Table 6–7 (Cont.) Replay Status Values*

| Status_Id | Status |
|---|---|
| 10 | To be reviewed by administrator |
| 11 | Error – Reviewed, No Action |

In addition, you can review the transaction replay service's server log files for error messages.

### 6.3.5 How to Publish Business Events

The transaction replay service enables publishing of a business event to invoke a SOA component, such as a BPEL process. You can use this functionality as follows:

1. Create a replay type that contains the following in the `REPLAY_TYPE` table:
   - `CLASSNAME_EVENTNAMESPACE`: specify the event namespace.
   - `METHODNAME_EVENTNAME`: specify the event name.
   - `INV_TYPE`: Business Event
2. Create replay items to use the created replay type. Event payload is supplied as `PARAMS` in the `REPLAY_ITEM` table. Ensure that the event payload is defined in a valid XML format and conforms to the event schema.

You can identify thetransaction replay service replay types to use a style called *Business Event*.

Business events are asynchronous in nature and do not retain information after having been fired. As an event publisher, the transaction replay service can only guarantee that the event is published to the Oracle Event Delivery Network (EDN): if a replay item is to invoke a business event, the replay item's Successfully Applied status (3) indicates that the event is delivered to EDN; it does not mean that the to-be-invoked actions have completed successfully.

## 6.4 Configuring Security for ADF Mobile Transaction Replay Service Authentication and Data Synchronization

The transaction replay service takes business service method calls from a disconnected mobile device and replays the calls on the server side. The mobile client authenticates users without any direct connection to the server. The mobile client then synchronizes a batch of business service calls to the `REPLAY_ITEM` table on the server. The transaction replay service reads from the `REPLAY_ITEM` table to get the user identity. The replay service replays the business service calls asynchronously without any user interface. If the business service requires authentication, the transaction replay service must obtain a security token for that given user and propagate it to the business service.

### 6.4.1 What You May Need to Know About User Credentials

A user credential comes from the base ADF application, but it is stored and accessed by the mobile client.

The ADF Mobile client application accesses this credential when the transaction replay service's transactions are logged; the credential is sent along with the transaction. This user credential is accessed by a login screen that checks the credential typed in against the stored credential.

### 6.4.2 How to Set Up Users and Subscriptions

To add and manage users and user access to applications, you use the Mobile Manager.

For more information, see the following:

- "Managing Users and Groups" section in *Oracle Database Lite Administration and Deployment Guide*
- "Grant or Revoke Application Access to Users" section in *Oracle Database Lite Administration and Deployment Guide*

### 6.4.3 How to Configure Application Deployment Packages

Mobile Database Workbench (MDW) lets you create deployment packages, which would allow synchronization rules (publications and publication items) created in the development environment to be migrated to the production environment. For more information, see "Using the Packaging Wizard" section in *Oracle Database Lite Developer's Guide*.

### 6.4.4 How to Modify Synchronization Rules Using Mobile Database Workbench

Mobile Database Workbench (MDW) lets you make changes to refine synchronization rules for your mobile client application:

- Define data filters (for example, filtering on a specific `OrderDate`).
- Configure database sequence windows that separate ranges of sequence numbers assigned to each client.
- Create subscription packages and assigning users. The subscription defines which data (that is, which publications) a user can access.

For more information, see *Oracle Database Lite Developer's Guide*.

### 6.4.5 How to Add Security and Grant Privileges

Oracle ADF Security framework provides authentication and authorization services to Oracle Fusion web applications.

The transaction replay service ADF entity event replay types support the security policy defined for ADF entity objects and attributes.

For information on how to grant privileges and add security, see "Enabling ADF Security in a Fusion Web Application" section in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

## 6.5 Monitoring Business Events and Entity Replay Items

Entity replay items record mobile client changes at the entity object level. At development time, business events can declaratively defined at the entity level and conditions under which those events should be raised can be specified. Business events that meet the specified criteria are raised upon successful commit of the changed data. A business event can be triggered on a successful create, update, or delete of an entity object. A business event is implemented during development by creating an event definition, mapping it to an event point, and then publishing that definition.

Event definition describes an event, which is published and raised within an event system. It is owned by an entity object and stored as part of the entity object's metadata. Event definitions consist of an event name and a payload, which is a list of attributes to be sent to the subscriber.

Event point is a place from which an event may be raised. There are three event points for an entity: create, update, and delete. At design time, event definitions are mapped to event points. Events are raised for each entity in the transaction after successful commit only.

### 6.5.1 How to Use Events To Specify Entity Replay Items

Mobile clients use three basic event names: `TRS_Create`, `TRS_Update`, and `TRS_Delete`.

- `TRS_Create`: This event is raised when a new entity record is created. Its event payload includes entity name and all attributes from the entity object.
- `TRS_Update`: This event is raised when an existing entity record is updated. Its event payload includes entity name, primary key and changed attributes, and the new and the old values.
- `TRS_Delete`: This event is raised when an existing record is deleted. Its event payload includes entity name, and the primary key.

The root element of each event payload is the name of the event point concatenated with `Info`, and the namespace value encodes the corresponding entity name with package information. The entity name can be derided by removing events or schema from the namespace attribute.

The child elements are attribute names and values. The `payload.xml` file can include a list of these three types of event information with elements `TRS_CreateInfo`, `TRS_UpdateInfo` or `TRS_DeleteInfo`.

### 6.5.2 What You May Need to Know About Authentication in Replay Items

The transaction replay service requires user name and password be provided for each replay item. To avoid sending the password in the `REPLAY_ITEM` table in clear text, the transaction replay service implements the public key encryption to securely exchange passwords from the mobile client to the server.

The mobile client gets the transaction replay service public key during authentication, which it uses to encrypt the password. For more information about security and authentication, see Section 6.2, "Configuring ADF Mobile Transaction Replay Service Security."

### 6.5.3 What You May Need to Know About Error Reporting

When an entity event fails to commit, the replay servlet receives exceptions. This is critical for error reporting back to the device. The transaction replay service stores the remote exception's error message in the `ERROR` column of the `REPLAY_ITEM` table and synchronizes back to the device.

## 6.6 Introduction to ADF Mobile Transaction Replay Service Error Handling

The transaction replay service errors are categorized into two groups:

- Data error (see Section 6.6.1, "Data Errors"): This is an error where the data itself prevents a transaction from committing. All other errors are process errors.
- Process error (see Section 6.6.2, "Process Error"): This includes server internal errors, time-out errors, network errors, any discrepancy in the replay list and replay item that is detected by the transaction replay service.

---

**Note:** The transaction replay service does not resolve the conflicts, but reports the error conditions and the required information to resolve the conflict.

---

### 6.6.1 Data Errors

Data-related errors can occur when replaying the replay items. These errors are categorized into two types:

1. Data conflict: can be either an update conflict or delete conflict.

   A data conflict could occur for various reasons, such as the following:

   - Attempt for updating the data results in a conflict error, as the data had been updated by another user after the last synchronization.
   - Attempt for updating the data results in a conflict error, as the data had been updated by another replay item in the same replay list.
   - Attempt for updating the data results in a conflict error, as the data had been updated by another DML induced within the same replay item.
   - Attempt for updating the data results in a conflict error, as the data had been deleted by another user after the last synchronization.
   - Attempt for deleting the data results in a conflict error, as the data that had been updated by another user after the last synchronization.
   - Attempt for deleting the data results in a conflict error, as the data that had been deleted by another user after the last synchronization.

2. Constraint violation: can occur in any database manipulation operations.

   A constraint violation could occur for various reasons, such as the following:

   - Attempt for updating the data that causes a constraint violation, as the related data had been modified by another user after the last synchronization.
   - Attempt for inserting the data that causes a constraint violation.
   - Attempt for deleting the data that causes a constraint violation.

### 6.6.2 Process Error

All errors that do not occur due to data conflicts or constraint violations are process errors.

### 6.6.3 Error Information

The transaction replay service provides various information as part of the conflict resolution framework.

#### 6.6.3.1 Error Information on the Client

The transaction replay service provides the following information on the client to identify and resolve conflicts:

- Replay item identifier
- Replay item error code
- Replay item error details
- Replay item created date
- Replay item parameters with name-value pairs
- Replay item old values

#### 6.6.3.2 Error Information on the Server

The transaction replay service provides the following information on the server to identify and resolve conflict:

- Replay item identifier
- Replay item error code
- Replay item error details
- Replay item created date
- Replay item parameters with name-value pairs
- Job ID for the replay list
- Server ID where the replay list was processed
- Thread ID of the server
- Timestamp when the replay list was processed
- Job status
- Job details, if available

## 6.7 Configuring and Monitoring ADF Mobile Transaction Replay Service Logging

The transaction replay service uses the standard `java.util.logging` mechanism. The following loggers are available:

- `oracle.txnreplay.common`
- `oracle.txnreplay.dispatcher`
- `oracle.txnreplay.txnreplay`
- `oracle.txnreplay.webservice`

You can adjust the log level using the standard Java Server log management tools, such as JConsole or Oracle Enterprise Manager. For more information, see *Oracle Fusion Middleware Administrator's Guide*.

### 6.7.1 How to Enable Logging for Entity Replay Items

To enable logging for entity replay items, configure the logging levels as described in "Configuring ADF Mobile Transaction Replay Service" section of *Oracle Fusion Middleware Installation Guide for ADF Mobile Transaction Replay Service*. The following

settings in the `adf-config.xml` file of the mobile client enable entity replay item logging:

```
<amc:setting name="app-name" value="BB1" xmlns=""/>
<amc:setting name="replay-type-id" value="39"/>
<amc:setting name="replay-type-name" value="HR-Entity-Event-Test"/>
<amc:setting name="use-trs" value="true"/>
```

The `replay-type-id` and `replay-type-name` settings refer to the `REPLAY_TYPE` table that corresponds to the correct replay type on the server base application.

# Part IV

## Appendices

Part IV contains the following chapters:

- Appendix A, "JDeveloper Runtime Libraries"
- Appendix B, "wsadmin Command Reference for ADF Applications"
- Appendix C, "ADF Mobile Transaction Replay Service Tables"

# A

# JDeveloper Runtime Libraries

This appendix provides a reference of the contents of JDeveloper runtime libraries that are deployed into Oracle WebLogic Server to support ADF applications.

The following JDeveloper runtime libraries are described:

- Section A.1, "adf.oracle.domain.webapp.war Library"
- Section A.2, "adf.oracle.domain.ear Library"
- Section A.3, "System Classpath"

## A.1 adf.oracle.domain.webapp.war Library

Table A–1 lists the JAR files that are packaged into the `adf.oracle.domain.webapp.war` file and their corresponding JDeveloper runtime library.

***Table A–1 adf.oracle.domain.webapp.war Library***

| JAR | JDeveloper Library |
|---|---|
| `oracle.adf.controller_11.1.1\adf-controller-api.jar` | ADF Controller Runtime |
| `oracle.adf.controller_11.1.1\adf-controller-rt-common.jar` | ADF Controller Runtime |
| `oracle.adf.controller_11.1.1\adf-controller.jar` | ADF Controller Runtime |
| `oracle.adf.pageflow_11.1.1\adf-pageflow-dtrt.jar` | ADF Page Flow Runtime<br>ADF Designtime API |
| `oracle.adf.pageflow_11.1.1\adf-pageflow-fwk.jar` | ADF Page Flow Runtime |
| `oracle.adf.pageflow_11.1.1\adf-pageflow-impl.jar` | ADF Page Flow Runtime |
| `oracle.adf.pageflow_11.1.1\adf-pageflow-rc.jar` | ADF Page Flow Runtime |
| `oracle.adf.view_11.1.1\adf-dt-at-rt.jar` | ADF Model Runtime<br>ADF Designtime API |
| `oracle.adf.view_11.1.1\adf-dynamic-faces.jar` | ADF Faces Dynamic Components |
| `oracle.adf.view_11.1.1\adf-faces-changemanager-rt.jar` | ADF Faces Change Manager Runtime 11 |

***Table A–1 (Cont.) adf.oracle.domain.webapp.war Library***

| JAR | JDeveloper Library |
|---|---|
| oracle.adf.view_<br>11.1.1\adf-faces-databinding-dt-core.jar | ADF Designtime API |
| oracle.adf.view_<br>11.1.1\adf-faces-databinding-rt.jar | Trinidad Databinding Runtime<br>ADF Faces Databinding Runtime |
| oracle.adf.view_<br>11.1.1\adf-faces-registration.jar | NA |
| oracle.adf.view_<br>11.1.1\adf-faces-templating-dt-core.jar | ADF Designtime API |
| oracle.adf.view_<br>11.1.1\adf-faces-templating-dtrt.jar | ADF Designtime API |
| oracle.adf.view_<br>11.1.1\adf-richclient-api-11.jar | Trinidad Databinding Runtime<br>ADF Faces Runtime 11 |
| oracle.adf.view_<br>11.1.1\adf-richclient-automation-11.jar | Oracle Customized Selenium |
| oracle.adf.view_<br>11.1.1\adf-richclient-impl-11.jar | ADF Faces Runtime 11 |
| oracle.adf.view_11.1.1\adf-share-web.jar | NA |
| oracle.adf.view_<br>11.1.1\adf-view-databinding-dt-core.jar | ADF Designtime API |
| oracle.adf.view_<br>11.1.1\dvt-databindings.jar | BI Data Control Runtime<br>Essbase Data Control Runtime<br>ADF DVT Faces Databinding Runtime |
| oracle.adf.view_11.1.1\dvt-faces.jar | ADF DVT Faces Runtime |
| oracle.adf.view_<br>11.1.1\dvt-facesbindings.jar | BI Data Control Runtime<br>Essbase Data Control Runtime<br>ADF DVT Faces Databinding Runtime |
| oracle.adf.view_11.1.1\dvt-jclient.jar | Oracle BI Graph<br>ADF DVT Core Runtime<br>ADF Swing Runtime<br>ADF DVT Faces Runtime |
| oracle.adf.view_11.1.1\dvt-trinidad.jar | ADF DVT Core Runtime<br>ADF DVT Faces Runtime |
| oracle.adf.view_11.1.1\dvt-utils.jar | BI Data Control Runtime<br>Oracle BI Graph<br>ADF DVT Core Runtime<br>ADF Swing Runtime<br>Essbase Data Control Runtime<br>ADF DVT Faces Runtime |
| oracle.adf.view_<br>11.1.1\oracle-page-templates.jar | Oracle Page Templates |
| oracle.adf.view_11.1.1\trinidad-api.jar | ADF Faces Runtime 11<br>Trinidad Runtime 11 |

*Table A–1 (Cont.) adf.oracle.domain.webapp.war Library*

| JAR | JDeveloper Library |
|---|---|
| `oracle.adf.view_11.1.1\trinidad-impl.jar` | ADF Faces Runtime 11<br>Trinidad Runtime 11 |
| `oracle.facesconfigdt_ 11.1.1\facesconfigmodel.jar` | ADF Faces Change Manager Runtime 11 |
| `oracle.facesconfigdt_11.1.1\taglib.jar` | ADF Faces Change Manager Runtime 11 |
| `velocity-dep-1.4.jar` | ADF Designtime API |

## A.2 adf.oracle.domain.ear Library

Table A–2 lists the JAR files that are packaged into the `adf.oracle.domain.ear` file and their corresponding JDeveloper runtime library.

*Table A–2 adf.oracle.domain.ear Library*

| JAR | JDeveloper Library |
|---|---|
| `groovy-all-1.6.3.jar` | ADF Model Runtime<br>ADF Model Generic Runtime<br>BC4J Runtime |
| `oracle.adf.model_ 11.1.1\adf-controller-schema.jar` | ADF Controller Schema |
| `oracle.adf.model_ 11.1.1\adf-controller-system.jar` | NA |
| `oracle.adf.model_11.1.1\adf-runtime-mbean.jar` | NA |
| `oracle.adf.model_11.1.1\adf-sec-idm-dc.jar` | User and Role Data Control |
| `oracle.adf.model_11.1.1\adfbcsvc-client.jar` | BC4J Service Client |
| `oracle.adf.model_11.1.1\adfbcsvc-share.jar` | BC4J Service Runtime<br>BC4J Service Client |
| `oracle.adf.model_11.1.1\adfbcsvc.jar` | BC4J Service Runtime |
| `oracle.adf.model_11.1.1\adfdeployrt.jar` | NA |
| `oracle.adf.model_11.1.1\adfdt_common.jar` | ADF Model Runtime<br>ADFm Designtime API |
| `oracle.adf.model_11.1.1\adflibfilter.jar` | ADF Common Web Runtime |
| `oracle.adf.model_11.1.1\adflibrary.jar` | NA |
| `oracle.adf.model_11.1.1\adfm-debugger.jar` | BC4J Tester |

*Table A–2 (Cont.) adf.oracle.domain.ear Library*

| JAR | JDeveloper Library |
|---|---|
| oracle.adf.model_11.1.1\adfm.jar | BC4J EJB Client<br>ADF Model Runtime<br>BC4J Oracle Domains<br>ADF Model Generic Runtime<br>BC4J Runtime<br>ADF Swing Runtime<br>JSR-227 API<br>BC4J EJB Runtime<br>BC4J Client<br>BC4J IAS Client |
| oracle.adf.model_11.1.1\adfmportlet.jar | NA |
| oracle.adf.model_11.1.1\adfmweb.jar | ADF Web Runtime |
| oracle.adf.model_11.1.1\adftags.jar | Oracle ADF DataTag |
| oracle.adf.model_11.1.1\adftransactionsdt.jar | ADF Designtime API |
| oracle.adf.model_11.1.1\bc4j-mbeans.jar | BC4J Runtime |
| oracle.adf.model_11.1.1\bc4jhtml.jar | BC4J Struts Runtime<br>BC4J HTML |
| oracle.adf.model_11.1.1\bc4jimdomains.jar | Oracle Intermedia ADF Swing<br>Oracle Intermedia |
| oracle.adf.model_11.1.1\bc4jsyscat.jar | BC4J Tester |
| oracle.adf.model_11.1.1\bc4jwizard.jar | BC4J Tester<br>BC4J Runtime |
| oracle.adf.model_11.1.1\datatags.jar | BC4J HTML |
| oracle.adf.model_11.1.1\db-ca.jar | BC4J EJB Client<br>ADF Model Runtime<br>BC4J Tester<br>BC4J Runtime<br>BC4J Client<br>BC4J IAS Client |
| oracle.adf.model_11.1.1\jdev-cm.jar | BC4J EJB Client<br>ADF Model Runtime<br>BC4J Tester<br>BC4J Runtime<br>Obsolete JDeveloper Extension SDK<br>BC4J Client<br>BC4J IAS Client<br>Connection Manager |
| oracle.adf.model_11.1.1\jmxdc.jar | NA |
| oracle.adf.model_11.1.1\jr_dav.jar | Resource Catalog Service |

*Table A–2 (Cont.) adf.oracle.domain.ear Library*

| JAR | JDeveloper Library |
|---|---|
| oracle.adf.model_11.1.1\oicons.jar | ADFm Designtime API |
| oracle.adf.model_11.1.1\oraclexsql.jar | NA |
| oracle.adf.model_11.1.1\ordhttp.jar | Oracle Intermedia ADF Swing<br>Oracle Intermedia |
| oracle.adf.model_11.1.1\ordim.jar | Oracle Intermedia ADF Swing<br>Oracle Intermedia |
| oracle.adf.model_11.1.1\rcs-adflib-rt.jar | NA |
| oracle.adf.model_11.1.1\rcsrt.jar | Resource Catalog Service |
| oracle.adf.model_11.1.1\regexp.jar | NA |
| oracle.adf.model_11.1.1\xsqlserializers.jar | XSQL Runtime |
| oracle.bali.share_11.1.1\share.jar | MDS Runtime Dependencies<br>BC4J Tester<br>ADF Model Generic Runtime<br>Oracle Help for Java<br>Oracle JEWT |

## A.3 System Classpath

Table A–3 lists the JAR files that are loaded into the system classpath and their corresponding JDeveloper runtime library.

*Table A–3 System Classpath*

| JAR | JDeveloper Library |
|---|---|
| com.sun.msv.datatype.xsd_20030530.jar | NA |
| commonj.sdo_2.1.0.jar | BC4J Service Runtime<br>BC4J Service Client<br>Java EE 1.5<br>EJB SDO Client |
| features\adf.model.generic_11.1.1.jar | NA |
| features\adf.model_11.1.1.jar | NA |
| features\adf.share.ca_11.1.1.jar | NA |
| features\adf.share_11.1.1.jar | NA |
| groovy-all-1.6.0.jar | NA |
| jakarta.jstl_1.0\lib\jaxen-full.jar | JSTL 1.0 |
| jakarta.jstl_1.0\lib\jdbc2_0-stdext.jar | JSTL 1.0 |
| jakarta.jstl_1.0\lib\jstl.jar | JSTL 1.1<br>JSTL 1.0 |
| jakarta.jstl_1.0\lib\saxpath.jar | JSTL 1.0 |
| jakarta.jstl_1.0\lib\standard.jar | JSTL 1.1 Tags<br>JSTL 1.0 Tags |

***Table A–3 (Cont.) System Classpath***

| JAR | JDeveloper Library |
|---|---|
| jakarta.jstl_1.0\lib\xalan.jar | JSTL 1.0 |
| jakarta.jstl_1.0\lib\xercesImpl.jar | JSTL 1.0 |
| jakarta.jstl_1.0\lib\xml-apis.jar | JSTL 1.0 |
| jakarta.jstl_1.1\lib\jstl.jar | JSTL 1.1<br>JSTL 1.0 |
| jakarta.jstl_1.1\lib\standard.jar | JSTL 1.1 Tags<br>JSTL 1.0 Tags |
| jsf.facelets_1.1.14\jsf-facelets.jar | Facelets Runtime |
| oracle.adf.dconfigbeans_11.1.1.jar | NA |
| oracle.adf.management_11.1.1\adf-em-config.jar | ADF Management Pages |
| oracle.adf.model.generic_11.1.1\bc4jdomgnrc.jar | BC4J Generic Domains |
| oracle.adf.model_ 11.1.1\redist\adfbinding-samples.jar | System Classpath |
| oracle.adf.model_11.1.1\redist\graphtags.jar | BC4J HTML |
| oracle.adf.share.ca_11.1.1\adf-share-base.jar | ADF Common Web Runtime<br>MDS Runtime Dependencies<br>ADF Model Generic Runtime<br>BC4J Runtime<br>ADF Swing Runtime<br>BC4J Security<br>ADF Common Runtime |
| oracle.adf.share.ca_11.1.1\adf-share-ca.jar | MDS Runtime Dependencies<br>ADF Model Generic Runtime<br>BC4J Runtime<br>BC4J Security<br>ADF Common Runtime |
| oracle.adf.share_11.1.1\adf-share-support.jar | MDS Runtime Dependencies<br>ADF Model Generic Runtime<br>BC4J Runtime<br>BC4J Security<br>ADF Common Runtime |
| oracle.adf.share_11.1.1\adf-share-wls.jar | NA |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.adf.share_11.1.1\adfloggінghandler.jar | MDS Runtime Dependencies |
| | BC4J Tester |
| | ADF Model Generic Runtime |
| | BC4J Runtime |
| | ADF Common Runtime |
| oracle.adf.share_11.1.1\adfsharembean.jar | BC4J Runtime |
| | ADF Common Runtime |
| oracle.adf.share_11.1.1\commons-cli-1.0.jar | MDS Runtime Dependencies |
| oracle.adf.share_11.1.1\commons-el.jar | ADF Model Runtime |
| | MDS Runtime Dependencies |
| | ADF Model Generic Runtime |
| | BC4J Runtime |
| oracle.adf.share_11.1.1\jsp-el-api.jar | ADF Model Runtime |
| | MDS Runtime Dependencies |
| | ADF Model Generic Runtime |
| | BC4J Runtime |
| oracle.adf.share_11.1.1\oracle-el.jar | ADF Model Runtime |
| | MDS Runtime Dependencies |
| | ADF Model Generic Runtime |
| | BC4J Runtime |
| oracle.auditprovider_ 11.1.1\jps-wls-auditprovider.jar | System Classpath |
| oracle.bulkops_11.1.1\bulkoperationsmbean.jar | NA |
| oracle.classloader_11.1.1.jar | NA |
| oracle.dconfig-infra_11.1.1.jar | NA |
| oracle.dms_11.1.1\dms.jar | MDS Runtime Dependencies |
| | Java EE 1.5 |
| | Essbase Data Control Runtime |
| | J2EE 1.4 |
| | Oracle JDBC |
| | JAX-RPC Client |
| oracle.ejb_11.1.1\ejbsvc-share.jar | EJB SDO Client |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.fabriccommon_11.1.1\fabric-common.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.fabriccommon_11.1.1\policy-api.jar | NA |
| oracle.help_5.0\help-indexer.jar | NA |
| oracle.help_5.0\help-share.jar | BC4J Tester<br>Oracle Help for Java |
| oracle.help_5.0\help-wizard.jar | NA |
| oracle.help_5.0\ohj.jar | BC4J Tester<br>Oracle Help for Java |
| oracle.help_5.0\ohw-rcf.jar | NA |
| oracle.help_5.0\ohw-share.jar | NA |
| oracle.help_5.0\ohw-uix.jar | NA |
| oracle.help_5.0\oracle_ice.jar | BC4J Tester<br>Ice<br>Oracle Help for Java |
| oracle.http_client_11.1.1.jar | URL Data Control<br>Java EE 1.5<br>Oracle SOAP<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.iau_11.1.1\fmw_audit.jar | Java EE 1.5<br>J2EE 1.4<br>BC4J Security<br>JAX-RPC Client |
| oracle.iau_<br>11.1.1\reports\AuditReportTemplates.jar | NA |
| oracle.idm_11.1.1\identitystore.jar | ADF Model Runtime<br>ADF Model Generic Runtime<br>BC4J Runtime<br>BC4J Security<br>ADF Common Runtime |
| oracle.idm_11.1.1\identityutils.jar | NA |
| oracle.javacache_11.1.1\cache.jar | MDS Runtime Dependencies<br>Java Cache<br>ADF Common Runtime |
| oracle.javacache_11.1.1\jocconfmbean.jar | NA |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.javatools_11.1.1\javamodel-rt.jar | Java EE 1.5<br>J2EE 1.4<br>JAX-RPC Client |
| oracle.javatools_11.1.1\javatools-nodeps.jar | ADF Common Web Runtime<br>MDS Runtime Dependencies<br>Java EE 1.5<br>ADFm Designtime API<br>J2EE 1.4<br>JAX-RPC Client |
| oracle.javatools_11.1.1\resourcebundle.jar | ADF Desktop Integration Runtime<br>Resource Bundle Support<br>BC4J Runtime |
| oracle.jdbc_11.1.1\ojdbc6dms.jar | NA |
| oracle.jmx_11.1.1\jmxframework.jar | MDS Runtime Dependencies<br>Java EE 1.5<br>BC4J Tester<br>BC4J Runtime<br>J2EE 1.4<br>JAX-RPC Client |
| oracle.jmx_11.1.1\jmxspi.jar | MDS Runtime Dependencies<br>Java EE 1.5<br>BC4J Tester<br>BC4J Runtime<br>J2EE 1.4<br>JAX-RPC Client |
| oracle.jps_11.1.1\jacc-spi.jar | BC4J Security |
| oracle.jps_11.1.1\jps-api.jar | BC4J Tester<br>BC4J Security<br>JAX-RPC Client |
| oracle.jps_11.1.1\jps-common.jar | BC4J Security<br>JAX-RPC Client |
| oracle.jps_11.1.1\jps-ee.jar | JPS Designtime<br>BC4J Security |
| oracle.jps_11.1.1\jps-internal.jar | BC4J Security |
| oracle.jps_11.1.1\jps-manifest.jar | NA |
| oracle.jps_11.1.1\jps-mbeans.jar | NA |
| oracle.jps_11.1.1\jps-unsupported-api.jar | BC4J Security |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.jps_11.1.1\jps-upgrade.jar | NA |
| oracle.jps_11.1.1\jps-wls.jar | NA |
| oracle.jrf_11.1.1\jrf-api.jar | JRF API |
| oracle.jrf_11.1.1\jrf-client.jar | JRF Client |
| oracle.jrf_11.1.1\jrf.jar | JAX-WS Client<br>JRF Runtime |
| oracle.jsf_1.2.7.1\jsf-api.jar | JSF |
| oracle.jsf_1.2.7.1\jsf-ri.jar | NA |
| oracle.jsf_1.2.7.1\sun-commons-beanutils.jar | NA |
| oracle.jsf_1.2.7.1\sun-commons-collections.jar | NA |
| oracle.jsf_1.2.7.1\sun-commons-digester.jar | NA |
| oracle.jsf_1.2.7.1\sun-commons-logging.jar | NA |
| oracle.jsf_1.2.9\glassfish.jsf_1.2.9.0.jar | JSF 1.2 |
| oracle.jsf_1.2.9\glassfish.jstl_1.2.0.1.jar | JSF 1.2<br>JSTL 1.2<br>JSTL 1.2 Tags |
| oracle.jsf_1.2.9\javax.jsf_1.2.0.1.jar | JSF 1.2<br>Java EE 1.5 API |
| oracle.jsf_1.2.9\wls.jsf.di.jar | JSF 1.2 |
| oracle.jsp_11.1.1\ojsp.jar | NA |
| oracle.ldap_11.1.1\jremtool.jar | NA |
| oracle.ldap_11.1.1\ldapjclnt11.jar | BC4J Security |
| oracle.ldap_11.1.1\ojmisc.jar | BC4J EJB Client<br>ADF Model Runtime<br>BC4J Tester<br>ADF Model Generic Runtime<br>BC4J Runtime<br>J2EE 1.4<br>JAX-RPC Client<br>BC4J IAS Client |
| oracle.logging-utils_11.1.1.jar | NA |
| oracle.mds_11.1.1\mdslcm.jar | NA |
| oracle.mds_11.1.1\mdsrt.jar | MDS Runtime |
| oracle.mds_11.1.1\oramds.jar | MDS Runtime Dependencies |
| oracle.nlsrtl_11.1.0\orai18n-collation.jar | NA |
| oracle.nlsrtl_11.1.0\orai18n-mapping.jar | NA |
| oracle.nlsrtl_11.1.0\orai18n-servlet.jar | NA |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.nlsrtl_11.1.0\orai18n-translation.jar | NA |
| oracle.nlsrtl_11.1.0\orai18n-utility.jar | NA |
| oracle.nlsrtl_11.1.0\orai18n.jar | Java EE 1.5<br>J2EE 1.4<br>JAX-RPC Client |
| oracle.oamprovider_11.1.1\oamAuthnProvider.jar | NA |
| oracle.oamprovider_11.1.1\oamcfgtool.jar | NA |
| oracle.oc4j_10.1.3.4\oc4jclient.jar | NA |
| oracle.odl_11.1.1\ojdl-log4j.jar | NA |
| oracle.odl_11.1.1\ojdl.jar | MDS Runtime Dependencies<br>Java EE 1.5<br>Essbase Data Control Runtime<br>J2EE 1.4<br>Oracle JDBC<br>JAX-RPC Client |
| oracle.odl_11.1.1\ojdl2.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.oer_11.1.1\activation-1.0.2.jar | NA |
| oracle.oer_11.1.1\aler-axis-1.2.1.jar | NA |
| oracle.oer_11.1.1\aler-axis-jaxrpc-1.2.1.jar | NA |
| oracle.oer_11.1.1\client.rex-10.3.1.0.jar | NA |
| oracle.oer_11.1.1\commons-discovery-0.2.jar | NA |
| oracle.oer_11.1.1\commons-logging-1.0.4.jar | NA |
| oracle.oer_11.1.1\mail-1.2.jar | NA |
| oracle.oer_11.1.1\oracle.jdeveloper.oer.jar | NA |
| oracle.oer_11.1.1\soap-2.2.jar | NA |
| oracle.oer_11.1.1\wsdl4j-1.6.2.jar | NA |
| oracle.osdt_11.1.1\ojdigsig.jar | NA |
| oracle.osdt_11.1.1\osdt_cert.jar | Java EE 1.5<br>J2EE 1.4<br>BC4J Security<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.osdt_11.1.1\osdt_cmp.jar | NA |
| oracle.osdt_11.1.1\osdt_cms.jar | NA |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.osdt_11.1.1\osdt_core.jar | Java EE 1.5<br>J2EE 1.4<br>BC4J Security<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.osdt_11.1.1\osdt_core_fips.jar | Web Service Data Control |
| oracle.osdt_11.1.1\osdt_jce.jar | Web Service Data Control |
| oracle.osdt_11.1.1\osdt_ldap.jar | NA |
| oracle.osdt_11.1.1\osdt_lib_v11.jar | Web Service Data Control |
| oracle.osdt_11.1.1\osdt_lib_v12.jar | Web Service Data Control |
| oracle.osdt_11.1.1\osdt_ocsp.jar | NA |
| oracle.osdt_11.1.1\osdt_saml.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.osdt_11.1.1\osdt_saml2.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.osdt_11.1.1\osdt_smime.jar | NA |
| oracle.osdt_11.1.1\osdt_tsp.jar | NA |
| oracle.osdt_11.1.1\osdt_wss.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.osdt_11.1.1\osdt_xkms.jar | NA |
| oracle.osdt_11.1.1\osdt_xmlsec.jar | Java EE 1.5<br>J2EE 1.4<br>BC4J Security<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.osdt_11.1.1\ospnego.jar | NA |
| oracle.ossoiap_11.1.1\ossoiap.jar | NA |
| oracle.pki_11.1.1\oraclepki.jar | J2EE 1.4<br>BC4J Security<br>JAX-RPC Client |
| oracle.pki_11.1.1\owm-3_0.jar | NA |
| oracle.pki_11.1.1\owm-images.jar | NA |
| oracle.pki_11.1.1\owm_help.jar | NA |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.security-api_11.1.1.jar | NA |
| oracle.ssofilter_11.1.1\ssofilter.jar | NA |
| oracle.toplink_11.1.1\eclipselink-dbwsutils.jar | NA |
| oracle.toplink_11.1.1\toplink-grid.jar | NA |
| oracle.toplink_11.1.1\toplink-javadoc.jar | NA |
| oracle.toplink_11.1.1\toplink-oc4j.jar | NA |
| oracle.ucp_11.1.0.jar | MDS Runtime Dependencies |
| oracle.web-common_11.1.1.jar | NA |
| oracle.webservices_11.1.1\dbws.jar | NA |
| oracle.webservices_11.1.1\lwdom.jar | JAX-RPC Client |
| oracle.webservices_11.1.1\mdds.jar | J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.webservices_11.1.1\oc4j-ws-support.jar | NA |
| oracle.webservices_11.1.1\orasaaj.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.webservices_11.1.1\orawsdl.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client<br>Oracle JWSDL |
| oracle.webservices_11.1.1\orawsrm.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.webservices_11.1.1\testpage.jar | NA |
| oracle.webservices_11.1.1\wsa.jar | NA |
| oracle.webservices_11.1.1\wsclient.jar | BC4J Service Runtime<br>ADF Desktop Integration Runtime<br>Java EE 1.5<br>Oracle JAX-WS Client<br>JAX-RPC 11 Web Services<br>J2EE 1.4<br>Web Service Data Control<br>Oracle JAX-RPC Client<br>OWSM Policy Lib |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.webservices_11.1.1\wsif.jar | WSIF Client<br>Java EE 1.5<br>J2EE 1.4<br>JAX-RPC Client |
| oracle.webservices_11.1.1\wssecurity.jar | Java EE 1.5<br>J2EE 1.4<br>Web Service Data Control<br>JAX-RPC Client |
| oracle.webservices_11.1.1\wsserver.jar | Java EE 1.5<br>J2EE 1.4<br>JAX-RPC Client |
| oracle.webservices_11.1.1\ws_confmbeans.jar | NA |
| oracle.wsm.agent.common_<br>11.1.1\wsm-agent-core.jar | Web Service Data Control |
| oracle.wsm.agent.common_<br>11.1.1\wsm-agent-fmw.jar | NA |
| oracle.wsm.agent.common_<br>11.1.1\wsm-agent-wls.jar | NA |
| oracle.wsm.agent.common_11.1.1\wsm-agent.jar | Web Service Data Control<br>JAX-RPC Client |
| oracle.wsm.agent.common_11.1.1\wsm-pap.jar | Web Service Data Control<br>JAX-RPC Client |
| oracle.wsm.common_11.1.1\wsm-audit-core.jar | NA |
| oracle.wsm.common_11.1.1\wsm-dependencies.jar | NA |
| oracle.wsm.common_11.1.1\wsm-pmlib.jar | Web Service Data Control<br>JAX-RPC Client |
| oracle.wsm.common_11.1.1\wsm-policy-core.jar | Web Service Data Control<br>JAX-RPC Client |
| oracle.wsm.common_11.1.1\wsm-secpol.jar | Web Service Data Control<br>JAX-RPC Client |
| oracle.wsm.pm_11.1.1\wsm-pm-ejb-client-api.jar | NA |
| oracle.wsm.pm_11.1.1\wsm-pmserver.jar | NA |
| oracle.wsm.policies_11.1.1\wsm-policytool.jar | NA |
| oracle.wsm.policies_<br>11.1.1\wsm-seed-policies.jar | Web Service Data Control |
| oracle.xdb_11.1.0.jar | J2EE 1.4<br>JAX-RPC Client |
| oracle.xdk_11.1.0\xml.jar | MDS Runtime Dependencies<br>Oracle XML Parser v2<br>XSQL Runtime |

*Table A–3 (Cont.) System Classpath*

| JAR | JDeveloper Library |
|---|---|
| oracle.xdk_11.1.0\xmlparserv2.jar | BC4J EJB Client |
| | ADF Model Runtime |
| | MDS Runtime Dependencies |
| | Oracle XML Parser v2 |
| | BC4J Tester |
| | ADF Model Generic Runtime |
| | BC4J Runtime |
| | XSQL Runtime |
| | Oracle SOAP |
| | BC4J Security |
| | JAX-RPC Client |
| | BC4J IAS Client |
| oracle.xdk_11.1.0\xquery.jar | Oracle XQuery |
| oracle.xdk_11.1.0\xsu12.jar | XSQL Runtime |
| | Oracle XML SQL Utility |
| oracle.xds_11.1.1.jar | NA |
| oracle.xmlef_11.1.1\xmlef.jar | MDS Runtime Dependencies |
| | ADF Faces Change Manager Runtime 11 |
| | ADF Model Generic Runtime |
| | Obsolete JDeveloper Extension SDK |
| oracle.xqs-api_11.1.1.jar | NA |
| org.apache.bcel_5.1.jar | NA |
| org.apache.commons.beanutils_1.6.jar | Commons Beanutils 1.6.1 |
| | Commons Beanutils 1.6 |
| org.apache.commons.digester_1.7.jar | Java EE 1.5 |
| | Commons Digester 1.5 |
| | Commons Digester 1.7 |
| | J2EE 1.4 |
| | JAX-RPC Client |
| org.apache.commons.logging_1.0.4.jar | Commons Logging 1.0.3 |
| | Commons Logging 1.0.4 |
| org.dom4j_1.6.1.jar | NA |
| org.jaxen_1.1.1.jar | Java EE 1.5 |
| | J2EE 1.4 |
| | Web Service Data Control |
| | JAX-RPC Client |

***Table A–3 (Cont.) System Classpath***

| JAR | JDeveloper Library |
|---|---|
| `org.objectweb.asm.attrs_1.5.3.jar` | NA |
| `org.objectweb.asm_1.5.3.jar` | NA |
| `org.osoa.sca_2.0.jar` | NA |
| `org.springframework_2.0.jar` | Java EE 1.5<br>J2EE 1.4<br>JAX-RPC Client |
| `sun.tools_1.6.0.jar` | NA |
| `ws.databinding.plugins_1.0.0.0.jar` | NA |
| `ws.databinding_1.0.0.0.jar` | NA |

# B

# wsadmin Command Reference for ADF Applications

This chapter describes the wsadmin commands you can use to deploy, manage, and configure Oracle ADF applications. wsadmin commands are intended to be used with the IBM WebSphere Application Server.

This chapter includes the following sections:



## B.1 Overview of Custom wsadmin Commands for Oracle ADF

Use the ADF-based URL Connections wsadmin commands to navigate the hierarchy of configuration or runtime beans and control the prompt display. Use the `getADFMArchiveConfig` commands to manage the `ADFMArchiveConfig` object.

Each command must be qualified by the module name. For example, if the module is `URLConnection.py`, then the command can be invoked like this: `URLConnection.createFileUrlConnection`. An example for the module `ADFAdmin.py` would be `ADFAdmin.getADFArchiveConfig`.

## B.2 ADF-Specific WebSphere Commands

Use the commands in Table B–1 to manage ADF applications. Invocation of wsadmin commands need to include the module name where the method is defined. For example, `URLConnection.createFileURLConnection('myapp', 'tempDir', '/scratch/tmp')`.

***Table B–1 Browse Commands for wsadmin Configuration***

| Use this command... | To... | Use with WLST... |
|---|---|---|
| `createFileUrlConnection` | Create a new ADF file connection. | Online or Offline |
| `createHttpUrlConnection` | Create a new ADF URL connection. | Online or Offline |
| `setURLConnectionAttributes` | Set or edit the attributes of a newly created or existing ADF connection. | Online or Offline |
| `listUrlConnection` | List a new URL connection. | Online or Offline |

*Table B–1 (Cont.) Browse Commands for wsadmin Configuration*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| `getADFMArchiveConfig` | Returns a handle to the `ADFMArchiveConfig` object for the specified archive. | Online or Offline |

## B.2.1 createFileUrlConnection

Use with wsadmin: Online or Offline.

### B.2.1.1 Description

Use this command to creates a new connection based on the `oracle.adf.model.connection.url.FileURLConnection` connection class.

### B.2.1.2 Syntax

```
URLConnection.createFileURLConnection(appName, name, URL)
```

| Argument | Definition |
|---|---|
| `appName` | Application name for which the connection will be created. |
| `name` | The name of the new connection. |
| `URL` | The URL associated with this connection. |

### B.2.1.3 Example

```
URLConnection.createFileURLConnection('myapp','tempDir','/scratch/tmp')
```

## B.2.2 createHttpUrlConnection

Use with wsadmin: Online or Offline.

### B.2.2.1 Description

Use this command to create a new connection based on the `oracle.adf.model.connection.url.HttpURLConnection` connection `type` class.

### B.2.2.2 Syntax

```
URLConnection.createHttpURLConnection (appName, name, [URL], [authenticationType],
[realm], [user], [password])
```

| Argument | Definition |
|---|---|
| *`appName`* | Application name for which the connection will be created. |
| *`name`* | The name of the new connection. |
| *`url`* | (Optional) The URL associated with this connection. |
| *`authenticationType`* | (Optional) The default is basic. |
| *`realm`* | (Optional) If this connection deals with authentication, then this should be set. The default is basic. |
| *`user`* | (Optional) |

| Argument | Definition |
|---|---|
| *password* | (Optional) |

#### B.2.2.3 Example

```
URLConnection.createHttpURLConnection('myapp','cnn','http://www.cnn.com')
```

### B.2.3 setURLConnectionAttributes

Use with wsadmin: Online or Offline.

#### B.2.3.1 Description

Use this command to set or edit the attributes of a newly created or existing ADF connection.

#### B.2.3.2 Syntax

```
URLConnection.setURLConnectionAttributes(appname, connectionname, attributes)
```

| Argument | Definition |
|---|---|
| appname | Application name. |
| connectionname | The name of the connection. |
| *attributes* | The array containing attributes to set in key/value pairs. |

#### B.2.3.3 Example

```
URLConnection.setURLConnectionAttributes
('myapp','cnn','ChallengeAuthenticationType:digest',
'AuthenticationRealm:XMLRealm')
```

### B.2.4 listUrlConnection

Use with wsadmin: Online or Offline.

#### B.2.4.1 Description

Use this command to list the connections of the application.

#### B.2.4.2 Syntax

```
URLConnection.listURLConnection(appname)
```

| Argument | Definition |
|---|---|
| appname | Application name. |

#### B.2.4.3 Example

```
URLConnection.listURLConnection ('myapp')
```

## B.2.5 getADFMArchiveConfig

Use with wsadmin: Online or Offline.

### B.2.5.1 Description

Returns a handle to the `ADFMArchiveConfig` object for the specified archive. The returned `ADFMArchiveConfig` object's methods can be used to change application configuration in an archive.

The `ADFMArchiveConfig` object provides the following methods:

- `setDatabaseJboSQLBuilder([value])` - Sets the Database `jbo.SQLBuilder` attribute.
- `getDatabaseJboSQLBuilder()` - Returns the current value of the `jbo.SQLBuilder` attribute.
- `setDatabaseJboSQLBuilderClass([value])` - Sets the Database `jbo.SQLBuilderClass` attribute.
- `getDatabaseJboSQLBuilderClass()` - Returns the current value of the `jbo.SQLBuilderClass` attribute.
- `setDefaultRowLimit([value])` - Sets the defaults `rowLimit` attribute. Value is a long specifying the row limit (Default -1).
- `getDefaultRowLimit()` - Returns the current value of the `rowLimit` attribute.
- `save([toLocation])` - If you specify the `toLocation`, then the changes will be stored in the target archive file and the original file will remain unchanged. Otherwise, the changes will be saved in the original file itself.

### B.2.5.2 Syntax

```
archiveConfigObject = ADFMAdmin.getADFMArchiveConfig(fromLocation)
```

| Argument | Definition |
|---|---|
| *fromLocation* | The name of the ear file, including its complete path. |

The syntax for `setDatabaseJboSQLBuilder([value])` is:

```
archiveConfigObject.setDatabaseJboSQLBuilder([value])
```

| Argument | Definition |
|---|---|
| *value* | The value of the `jbo.SQLBuilder` attribute. Valid values are: 'Oracle' (Default), 'OLite', 'DB2', 'SQL92', 'SQLServer', or 'Custom. If 'Custom' is specified, then the `jbo.SQLBuilderClass` attribute should also be set. |

The syntax for `getDatabaseJboSQLBuilder()` is:

```
archiveConfigObject.getDatabaseJboSQLBuilder()
```

The syntax for `setDatabaseJboSQLBuilderClass([value])` is:

```
archiveConfigObject.setDatabaseJboSQLBuilderClass([value])
```

| Argument | Definition |
|---|---|
| *value* | The value of the `jbo.SQLBuilderClass` attribute. |

The syntax for `getDatabaseJboSQLBuilderClass()` is:

```
archiveConfigObject.getDatabaseJboSQLBuilderClass()
```

The syntax for `setDefaultRowLimit([value])` is:

```
archiveConfigObject.setDefaultRowLimit([value])
```

| Argument | Definition |
|---|---|
| *value* | The value of the `rowLimit` attribute. |

The syntax for `getDefaultRowLimit()` is:

```
archiveConfigObject.getDefaultRowLimit([value])
```

The syntax for `save([toLocation])` is:

```
archiveConfigObject.save([toLocation])
```

| Argument | Definition |
|---|---|
| *toLocation* | The file name along with the absolute path to store the changes. |

### B.2.5.3 Example

In the following example, if the adf-config.xml file in the archive does not have the application and shared metadata repositories defined, then you should provide the complete connection information.

```
# Open something.ear and return an object which can be used
# manipulate it
archive = ADFMAdmin.getADFMArchiveConfig('/path/to/something.ear')

# Return current JBO SQL Builder value
archive.getDatabaseJboSQLBuilder()

# Change JBO SQL Builder value to Oracle
archive.setDatabaseJboSQLBuilder('Oracle')

# Save the changes back to the original file
archive.save()

archive = ADFMAdmin.getADFMArchiveConfig('/path/to/something.ear')
archive.getDatabaseJboSQLBuilder()
```

In the following example. the `jbo.SQLBuilder` attribute is set to 'DB2'.

```
wsadmin> archive =
               ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wsadmin> archive.setDatabaseJboSQLBuilder(value='DB2')
wsadmin> archive.save()
```

In the following example, the `jbo.SQLBuilder` attribute is removed so that application default is used.

```
wsadmin> archive =
             ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wsadmin> archive.setDatabaseJboSQLBuilder()
wsadmin> archive.save(toLocation='/tmp/targetArchive.ear')
```

In the following example, the `jbo.SQLBuilder` attribute is set to 'Custom', and the `jbo.SQLBuilderClass` attribute is set to the class 'com.example.CustomBuilder'.

```
wsadmin> archive =
             ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wsadmin> archive.setDatabaseJboSQLBuilder('Custom')
wsadmin> archive.setDatabaseJboSQLBuilderClass('com.example.CustomBuilder')
wsadmin> archive.save(toLocation='/tmp/targetArchive.ear')
```

In the following example, the `rowLimit` attribute is set to 100.

```
wsadmin:/offline> archive =
getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wsadmin:/offline> archive.setDefaultRowLimit(100)
wsadmin:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

# C

# ADF Mobile Transaction Replay Service Tables

This appendix describes tables that form parts of ADF Mobile transaction replay service schema.

This appendix includes the following sections:

- Section C.1, "REPLAY_ITEM Table"
- Section C.2, "REPLAY_TYPE Table"
- Section C.3, "REPLAY_STATUS Table"

> **Note:** `REPLAY_JOBS` and `REPLAY_USERS` tables are internal to the transaction replay service and are not described in this appendix.

## C.1 REPLAY_ITEM Table

Table C–1 contains details on the `REPLAY_ITEM` table.

***Table C–1 REPLAY_ITEM Table Column Details***

| Column Name | Type | Length | Key | Comments |
|---|---|---|---|---|
| `ITEM_ID` | `NUMBER` | 15 | Primary Key | Primary KeyID and the primary key. Replay items are processed in the order of `ITEM_ID`. |
| `REPLAY_TYPE` | `NUMBER` | 10 | Foreign Key | Foreign key to the `REPLAY_TYPE` table that references details for replaying the replay item. |
| `STATUS` | `NUMBER` | 10 | Foreign Key | Numerical status of replay item during processing through the transaction replay service. |

***Table C–1 (Cont.) REPLAY_ITEM Table Column Details***

| Column Name | Type | Length | Key | Comments |
|---|---|---|---|---|
| CLIENT_ID | NUMBER | 10 | Primary Key, Foreign Key | A unique identifier for a disconnected client device. Items with the same CLIENT_ID are processed sequentially. Items with different CLIENT_ID are processed concurrently. |
| USER_NAME | VARCHAR2 | 2000 | NA | The user name for an enterprise server application. Used for authentication and authorization to the back-end server where items are replayed. |
| USER_ PASSWORD | VARCHAR2 | 4000 | NA | The user password for an enterprise server application. Used for authentication and authorization to the back-end server where items are replayed. |
| PARAMS | CLOB | | NA | An XML-formatted string containing input parameters to the corresponding replay type method call. Note that this is not a required column. |
| ERROR | VARCHAR2 | 4000 | NA | Detailed error information regarding an item during replay. |
| CREATED_ON | TIMESTAMP | | NA | The timestamp when the replay item was posted to the table. |
| PROCESSED_ON | TIMESTAMP | | NA | The timestamp when the replay item was replayed on the server. Also indicates the time when a replay item status is set to success or error. |
| DEPENDENCY | NUMBER | 15 | Foreign Key | Dependency of this item. It may contain the Item_ID of another replay item, which means the dependency replay item must be processed before this replay item. |

## C.2 REPLAY_TYPE Table

Table C–2 contains details on the REPLAY_TYPE table.

*Table C–2 REPLAY_TYPE Table Column Details*

| Column Name | Type | Length | Key | Comments |
|---|---|---|---|---|
| TYPE_ID | NUMBER | 10 | Primary Key | ID number and the primary key of the REPLAY_TYPE table. |
| TYPE_NAME | VARCHAR2 | 1000 | NA | Descriptive name for the replay type. |
| INV_TYPE | VARCHAR2 | 1000 | NA | The invocation type to be used for replaying the type.For example, Java Class, Web Service, or Oracle ADF Business Service. |
| ADF_CONFIG_NAME | VARCHAR2 | 1000 | NA | ADF application module runtime configuration name. |
| CLASS_NAME | VARCHAR2 | 1000 | NA | The full class name including the package name. |
| METHOD_NAME | VARCHAR2 | 1000 | NA | The method to be called on the application. |
| XSD_URL | VARCHAR2 | 2000 | NA | The location of the XSD URL used for validation. |
| XSL_URL | VARCHAR2 | 2000 | NA | The location of the XSLT URL that is to be used for transformation. |
| PASSWORD_TYPE | VARCHAR2 | 40 | NA | Password type. The supported values are CSF for credential store, and PKE for public key encryption. |
| CS_MAP_NAME | VARCHAR2 | 1000 | NA | Specific map name for this given replay type. |
| PORT_CLASS_NAME | VARCHAR2 | 1000 | NA | Port classname for JAX-WS proxy. |
| CLIENT_CONFIG_URL | VARCHAR2 | 2000 | NA | The client security policy configuration file for JAX-WS proxy. |

## C.3 REPLAY_STATUS Table

Table C–3 contains details on the REPLAY_STATUS table.

*Table C–3 REPLAY_STATUS Table Column Details*

| Column Name | Type | Length | Key | Comments |
|---|---|---|---|---|
| STATUS_ID | NUMBER | 10 | Primary Key | The primary key generated by sequence. |
| STATUS | VARCHAR2 | 1000 | N/A | Descriptive text to define the status. |