Oracle® Fusion Middleware

Administrator's Guide for Oracle Application Development Framework

11*g* Release 2 (11.1.2.2.0)

E16179-03

April 2012

Documentation for Oracle Application Development Framework (Oracle ADF) administrators that describes how to deploy, monitor, and configure ADF applications.

ORACLE

Oracle Fusion Middleware Administrator's Guide for Oracle Application Development Framework 11*g* Release 2 (11.1.2.2.0)

E16179-03

Primary Authors: Peter Jew (Lead), Liza Rekadze

Contributing Author: Odile Sullivan-Tarazi, Himanshu Marathe, Cammy Moore

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

Contents

3 Monitoring and Configuring ADF Applications

4 WLST Command Reference for ADF Applications

Preface

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

Audience

This document is intended for system administrators who need to deploy, manage, monitor, and configure Oracle ADF applications using the Oracle Application Development Framework (Oracle ADF).

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents:

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

Oracle Fusion Middleware 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.

What's New in This Guide in Release 11.1.2.2.0

For Release 11.1.2.2.0, this guide has been updated in several ways. The following table lists the sections that have been added or changed.

For changes made to Oracle JDeveloper and Oracle Application Development Framework (Oracle ADF) for this release, see the What's New page on the Oracle Technology Network at `http://www.oracle.com/technetwork/developer-tools/jdev/documentation/index.html`.

Sections	Change Description
Chapter 3 Monitoring and Configuring ADF Applications	
Section 3.3.1, "How to Modify ADF Business Components Parameters."	Added best practice to `Dofailover` element about enabling failover passivation that involves configuring a pool parameter for Oracle WebLogic Server.

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).

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 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, WLST commands, the `ojdeploy` command, scripts, or the WebLogic Administration 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.

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 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.

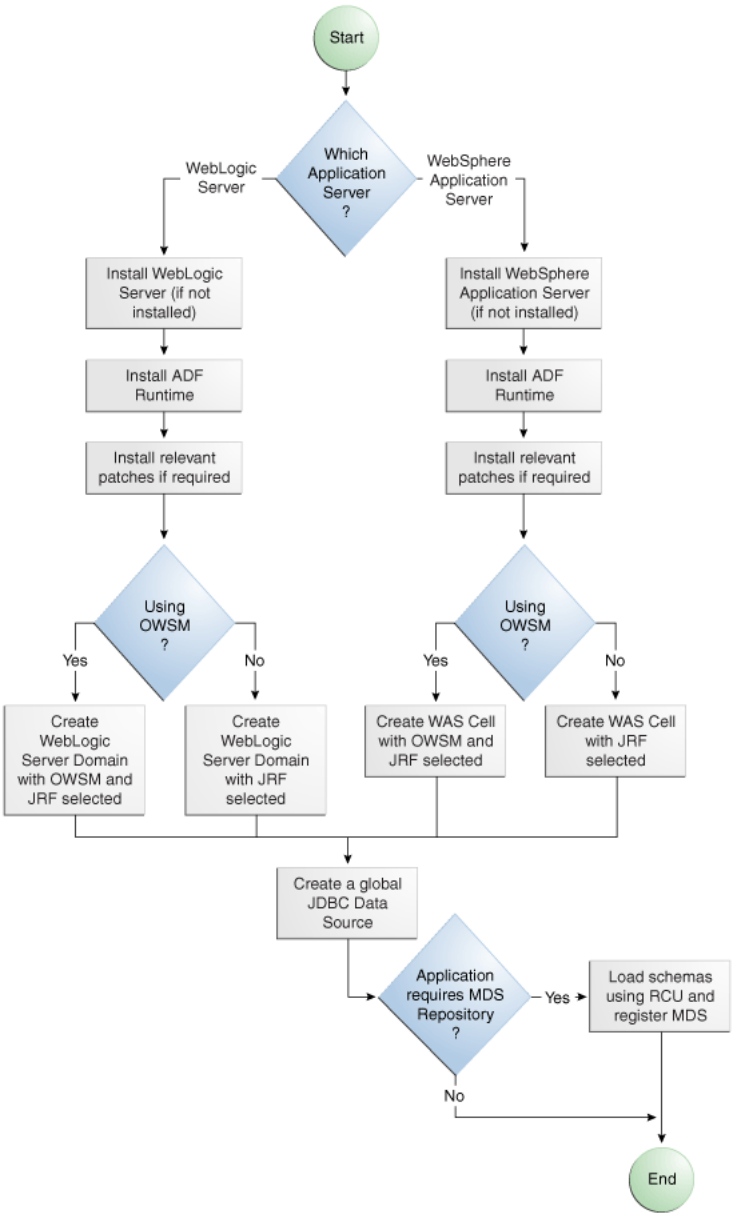
2.2 Preparing the Standalone Application Server for Deployment

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, start the installer, and to complete the installation.

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 to 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

After you have installed the ADF runtime, configure the cells and perform other tasks as described in the *Oracle Fusion Middleware Third-Party Application Server Guide* and the *Oracle Fusion Middleware Configuration Guide for WebSphere*.

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 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_home/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 Third-Party Application Server Guide*.

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) 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 using Fusion Middleware Control. It also describes configuration tasks required for applications deployed to IBM WebSphere Application 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"
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"
Active Data Service (ADS)		Section 3.4.5, "How to Modify Active Data Service 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`. 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.

Element	Description
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. 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. 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
AmpoolMaxinactiveage	Enter the maximum amount of time (in milliseconds) that an application module may remain inactive before it is removed from the pool. 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. 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. 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.DefaultConnection Strategy`. This is the class that implements the connection strategy.

Element	Description
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. 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. 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.
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. 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. 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. 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.

Element	Description
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`. 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`). **Best Practice**: When enabling application module state passivation, a failure can occur when Oracle WebLogic Server is configured to forcibly release connection back into the pool. A failure of this type produces a SQLException (Connection has already been closed) that is saved to the server log. The exception is not reported through the user interface. To ensure that state passivation occurs and users' changes are saved, set an appropriate value for the `weblogic-application.xml` deployment descriptor parameter `inactive-connection-timeout-seconds` on the `<connection-check-params>` pool-params element. Setting the deployment descriptor parameter to several minutes, in most cases, should avoid forcing the inactive connection timeout and the resulting passivation failure. Adjust the setting as needed for your environment.
poolClassName	Enter the custom application pool implementation class.
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. The default is an initial size of 0 instances.

Element	Description
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. 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. 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. 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. 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.
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. 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. 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. 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.

Element	Description
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. `database` is the default choice. While it may be a little slower than passivating to file, it is by far the most reliable choice. `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.
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)

Element	Description
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.

Element	Description
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 +`). `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. 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. 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`. 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.

Element	Description
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`. 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: **None** - No authentication. **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. **Must** - Like Test, but if authentication fails, the application module will not be instantiated. Instead, you will get an exception. **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. 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. For example: `http://discuss-server.com:8888/owc_discussions`
Administrator User Name	Enter the user name of the discussion server administrator. 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). The default is 0.
ache Expiration Time (in Minutes)	Specify a suitable expiration period for the cache. This is the maximum length of time (in minutes) that cached content is valid. The default is 0.
Connection Timeout (in Seconds)	Specify a suitable timeout for the connection. 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. 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` **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.
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. 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. 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. 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. Use the format: `http://<host>:<port>/search/query/OracleSearch` For example: `http://myHost:7777/search/query/OracleSearch`
Application User Name	Enter the name of a valid user. 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. 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.
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 MBean
- Active Data Service (ADS) MBean

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.

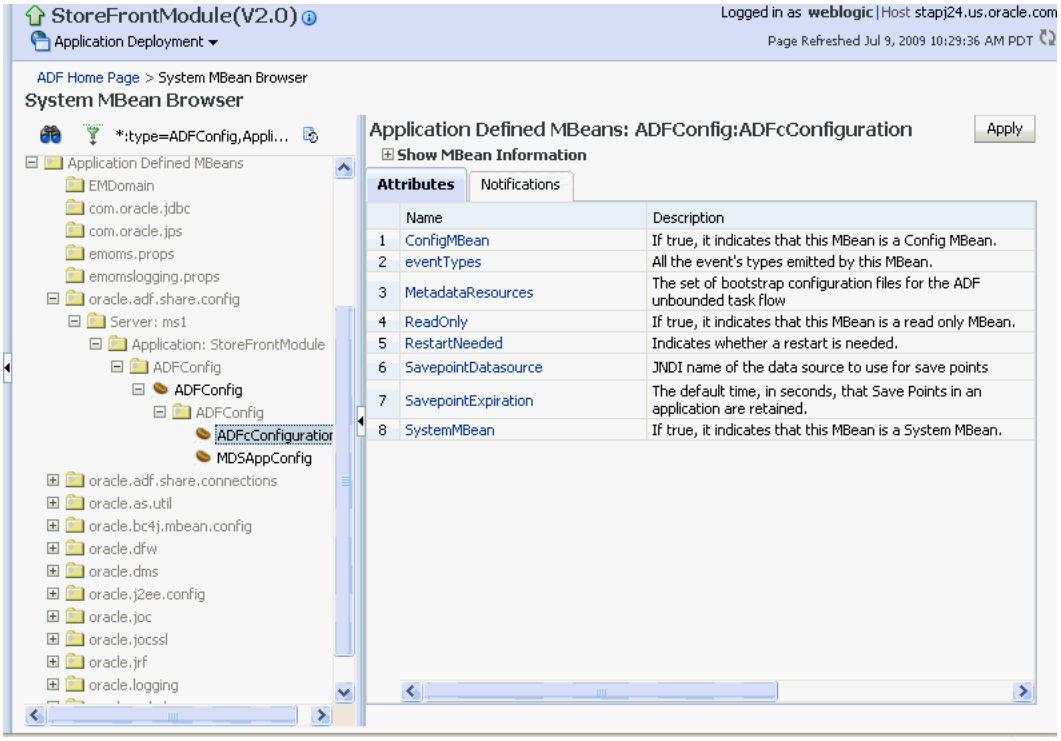
6. In the left pane of the System MBean Browser, expand the parent ADF MBean **ADFConfig** and then the **ADFConfig** folder to expose the child ADF MBeans.

   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.

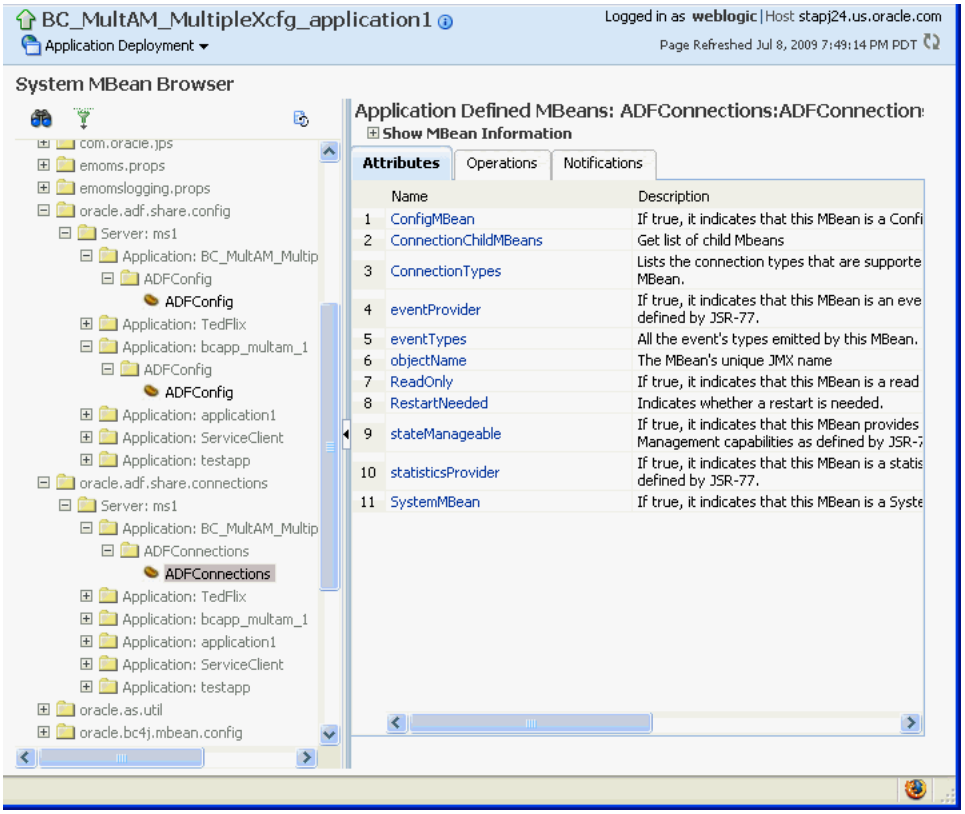
To modify ADF application configuration using MBean Browser:

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.

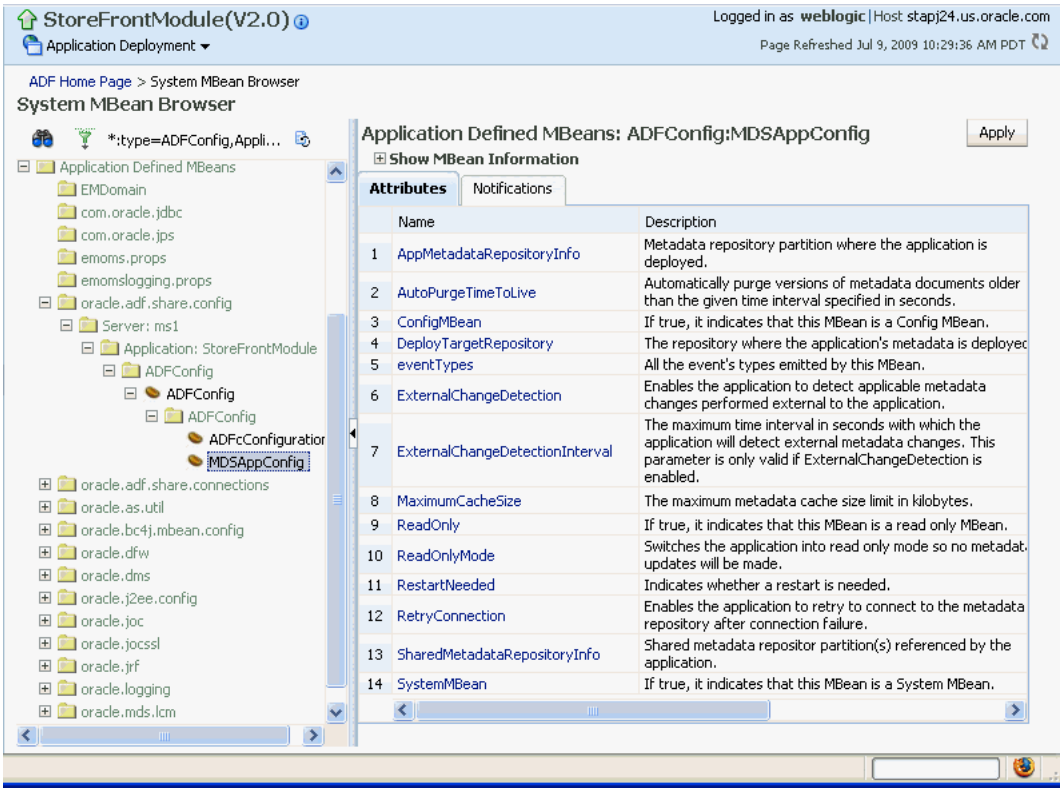
To modify MDS configuration using MBean Browser:

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**.

3.4.5 How to Modify Active Data Service Configuration Using MBean

You can use Active Data Service (ADS) framework to control the runtime behavior of an Oracle ADF application and qualifying ADF Faces components so that whenever data changes on the server, the ADF Model layer notifies the component and the component rerenders the changed data.

Before you begin:

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

Note that the ADF Faces components of your application must be configured to use ADS. Additionally, if your application services do not support ADS, then your application must define a service proxy so that the components can display the data as it updates in the source. For details about ADS, see Chapter 45, "Using the Active Data Service" in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

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, `application1 (AdminServer)`.

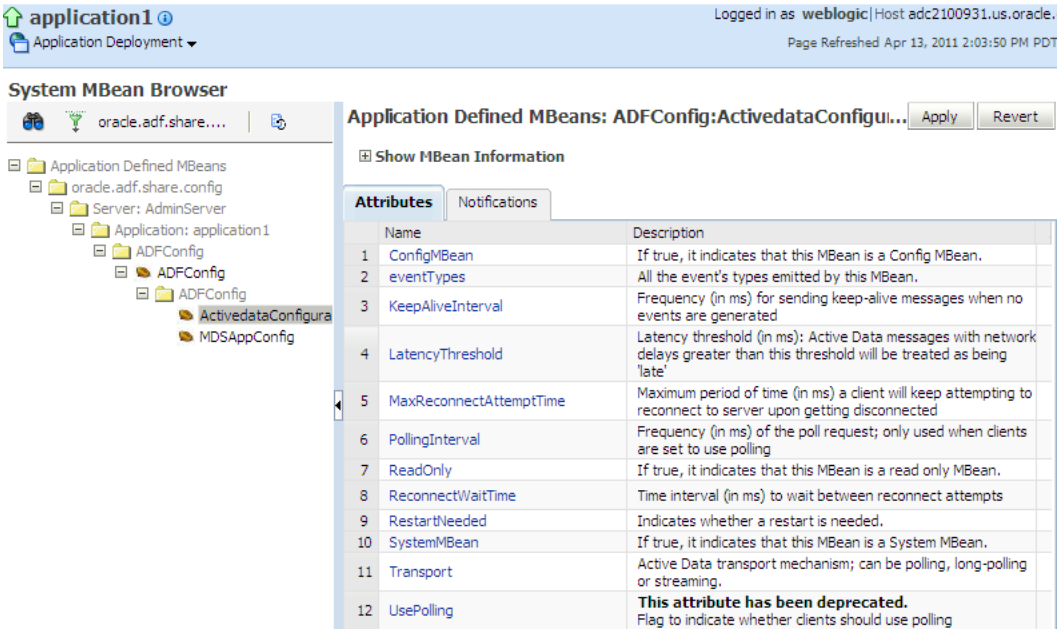
   After you select an application, the Application Deployment page displays.
5. Click **Application Deployment** and select **ADF > Configure ADF (adf-config)** from the dropdown menu.
6. In the left pane of the System MBean Browser, expand the parent ADF MBean **ADFConfig** and then the **ADFConfig**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

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

Attribute	Description
Transport	The method by which data will be delivered to the client. Value values are: ▪ **streaming (default)** ▪ **polling** ▪ **long-polling** For more information, see the "What You May Need to Know About Transport Modes" section of the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.
LatencyThreshold	Latency threshold in milliseconds. Active data messages with network delays greater than this threshold will be treated as being "late".
KeepAliveInterval	Frequency in milliseconds for sending keep-alive messages when no events are generated.
PollingInterval	Frequency in milliseconds of the poll request; only used when clients are set to use polling
MaxReconnectAttemptTime	Maximum period of time in milliseconds a client will keep attempting to reconnect to server upon getting disconnected
ReconnectWaitTime	Time interval in milliseconds to wait between reconnect attempts.

Figure 3–4 shows an ActiveDataConfiguration MBean in the Fusion Middleware Control MBean Browser.

Figure 3–4 ADF Active Data 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.

3.5 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 ADFConfig 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*.

3.7 Viewing Application Metric Information with DMS SPY

You can use the DMS Spy servlet to view application metric information in a web browser.

For more information, see the ""Monitoring Oracle Fusion Middleware" section in the *Oracle® Fusion Middleware Performance and Tuning Guide*

3.8 Configuring WebSphere Application Server

You use the WebSphere Application Server administrative console to configure WebSphere Application Server. 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`.

3.8.1 How to Configure WebSphere to Allow Reuse of Query Result Sets

WebSphere Application Server closes shared database connections between application generated requests. You need to set two properties in WebSphere to allow reuse of result sets.

Use the WebSphere Application Server administrative console to set the `non-transactional datasource` and `DisableMultiThreadedServletConnectionMgmt` properties.

To set properties in WebSphere to reuse results sets:

1. Start WebSphere Application Server administrative console.
2. Navigate to **Data sources > DB2 Universal JDBC Driver XA DataSource > WebSphere Application Server data source properties** and set **Non-transactional data source** to **enabled**.
3. Save the configuration.

4. Navigate to **Application servers > server_name > Web Container > Custom Properties** and **set DisableMultiThreadedServletConnectionMgmt** to **true**.
5. Save the configuration.
6. Restart WebSphere Application Server.

Setting these two properties will enable your deployed application to reuse result sets across requests.

4

WLST Command Reference for ADF Applications

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

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.
`user`	(Optional)

Argument	Definition
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

Appendices

Part III contains the following chapters:

- Appendix A, "JDeveloper Runtime Libraries"
- Appendix B, "wsadmin Command Reference for ADF Applications"

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:

A.1 Using JDeveloper to Find the JDeveloper Runtime Library

In addition to the listings in this appendix, you can also use JDeveloper to find a JAR's corresponding JDeveloper runtime library.

To find the JDeveloper library for a JAR:

1. In JDeveloper, select **Tools > Manage Libraries**.
2. In the Manage Libraries dialog Libraries tab, click the **Search** icon and select **Jar name** from the dropdown list.
3. In the search field, enter the name of the JAR and click the search icon.

A.2 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

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

JAR	JDeveloper Library
oracle.adf.pageflow_11.1.1/adf-pageflow-dtrt.jar	ADF Page Flow Runtime 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 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
oracle.adf.view_11.1.1/adf-faces-databinding-dt-core.jar	ADF Designtime API
oracle.adf.view_11.1.1/adf-faces-databinding-rt.jar	Trinidad Databinding Runtime ADF Faces Databinding Runtime
oracle.adf.view_11.1.1/adf-faces-templating-dt-core.jar	ADF Designtime API
oracle.adf.view_11.1.1/adf-faces-templating-dtrt.jar	ADF Designtime API
oracle.adf.view_11.1.1/adf-richclient-api-11.jar	Trinidad Databinding Runtime ADF Faces Runtime 11
oracle.adf.view_11.1.1/adf-richclient-automation-11.jar	Oracle Extended Selenium (deprecated) Oracle Extended Selenium (Server excluded)
oracle.adf.view_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_11.1.1/adf-view-databinding-dt-core.jar	ADF Designtime API
oracle.adf.view_11.1.1/adf.constants.jar	NA
oracle.adf.view_11.1.1/batik-anim.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-awt-util.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-bridge.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-codec.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-css.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-dom.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-ext.jar	ADF DVT Faces Runtime

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

JAR	JDeveloper Library
oracle.adf.view_11.1.1/batik-extension.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-gui-util.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-gvt.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-parser.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-script.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-svg-dom.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-svggen.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-swing.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-transcoder.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-util.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/batik-xml.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/bundleresolver.jar	Resource Bundle Variable Resolver
oracle.adf.view_11.1.1/dvt-basemaps.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/dvt-databinding-dt-core.jar	ADF Designtime API
oracle.adf.view_11.1.1/dvt-databindings.jar	BI Data Control Runtime ADF DVT Faces Databinding Runtime
oracle.adf.view_11.1.1/dvt-faces.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/dvt-facesbindings.jar	BI Data Control Runtime ADF DVT Faces Databinding Runtime
oracle.adf.view_11.1.1/dvt-jclient.jar	Oracle BI Graph ADF DVT Core Runtime ADF Swing Runtime ADF DVT Faces Runtime
oracle.adf.view_11.1.1/dvt-trinidad.jar	ADF DVT Core Runtime ADF DVT Faces Runtime
oracle.adf.view_11.1.1/dvt-utils.jar	BI Data Control Runtime Oracle BI Graph ADF DVT Core Runtime ADF Swing Runtime ADF DVT Faces Runtime
oracle.adf.view_11.1.1/inspect4.jar	Obsolete JDeveloper Extension SDK Oracle JEWT
oracle.adf.view_11.1.1/jewt4.jar	BC4J Tester Obsolete JDeveloper Extension SDK Oracle Help for Java Oracle JEWT

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

JAR	JDeveloper Library
oracle.adf.view_11.1.1/prefuse.jar	ADF DVT Faces Runtime
oracle.adf.view_11.1.1/trinidad-api.jar	ADF Faces Runtime 11 Trinidad Runtime 11
oracle.adf.view_11.1.1/trinidad-impl.jar	ADF Faces Runtime 11 Trinidad Runtime 11
oracle.adf.view_11.1.1/xml-apis-ext.jar	ADF DVT Faces Runtime
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
oracle.xdk_11.1.0/xml.jar	MDS Runtime Dependencies Oracle XML Parser v2 XSQL Runtime
velocity-dep-1.4.jar	ADF Designtime API

A.3 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 ADF Model Generic Runtime BC4J Runtime
oracle.adf.model_ 11.1.1/adf-controller-schema.jar	ADF Controller Schema
oracle.adf.model_ 11.1.1/adf-faces-registration.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-registration.jar	Kava SDO
oracle.adf.model_11.1.1/adfbcsvc-share.jar	BC4J Service Runtime BC4J Service Client BC4J System Catalog
oracle.adf.model_11.1.1/adfbcsvc.jar	BC4J Service Runtime
oracle.adf.model_11.1.1/adfdt_common.jar	ADF Model Runtime ADFm Designtime API
oracle.adf.model_11.1.1/adflibfilter.jar	ADF Common Web Runtime

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

JAR	JDeveloper Library
oracle.adf.model_11.1.1/adflibrary.jar	ADF Model Runtime ADFm Designtime API
oracle.adf.model_11.1.1/adfm-debugger.jar	BC4J Tester
oracle.adf.model_11.1.1/adfm-sqldc.jar	ADF SQL Data Control Runtime
oracle.adf.model_11.1.1/adfm.jar	BC4J EJB Client ADF Model Runtime BC4J Oracle Domains ADF Model Generic Runtime BC4J Runtime ADF Swing Runtime ADF Model API BC4J EJB Runtime BC4J Client 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 Model Runtime ADFm Designtime API 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
oracle.adf.model_11.1.1/bc4jimdomains.jar	Oracle Intermedia ADF Swing Oracle Intermedia
oracle.adf.model_11.1.1/bc4jsyscat.jar	BC4J System Catalog
oracle.adf.model_11.1.1/datatags.jar	NA
oracle.adf.model_11.1.1/db-ca.jar	BC4J EJB Client ADF Model Runtime BC4J Tester BC4J Runtime BC4J Client BC4J IAS Client DB Runtime (db-tests)
oracle.adf.model_ 11.1.1/dvt-databindings-mds.jar	ADF DVT Faces Databinding MDS Runtime

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

JAR	JDeveloper Library
oracle.adf.model_11.1.1/jdev-cm.jar	BC4J EJB Client ADF Model Runtime BC4J Tester BC4J Runtime Obsolete JDeveloper Extension SDK BC4J Client BC4J IAS Client Connection Manager
oracle.adf.model_11.1.1/jmxdc.jar	JMX Data Control
oracle.adf.model_11.1.1/jr_dav.jar	Resource Catalog Service
oracle.adf.model_11.1.1/mds-dc.jar	NA
oracle.adf.model_11.1.1/oicons.jar	ADFm Designtime API
oracle.adf.model_11.1.1/ordhttp.jar	Oracle Intermedia ADF Swing Oracle Intermedia
oracle.adf.model_11.1.1/ordim.jar	Oracle Intermedia ADF Swing 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	BC4J Tester
oracle.xdk_11.1.0/oraclexsql.jar	NA
oracle.xdk_11.1.0/xsqlserializers.jar	XSQL Runtime

A.4 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
features/adf.security_11.1.1.jar	NA
features/adf.share_11.1.1.jar	NA
features/adf.share.ca_11.1.1.jar	NA
oracle.adf.security_ 11.1.1/adf-controller-security.jar	ADF Model Runtime ADF Common Runtime
oracle.adf.security_ 11.1.1/adf-share-security.jar	ADF Model Runtime BC4J Security ADF Common Runtime

Table A–3 (Cont.) System Classpath

JAR	JDeveloper Library
oracle.adf.share_11.1.1/adf-share-support.jar	MDS Runtime Dependencies ADF Model Generic Runtime BC4J Runtime BC4J Security ADF Common Runtime
oracle.adf.share_11.1.1/adf-share-wls.jar	NA
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-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.adf.share.ca_11.1.1/adf-share-base.jar	ADF Common Web Runtime MDS Runtime Dependencies ADF Model Generic Runtime BC4J Runtime ADF Swing Runtime BC4J Security ADF Common Runtime

Table A–3 (Cont.) System Classpath

JAR	JDeveloper Library
oracle.adf.share.ca_11.1.1/adf-share-ca.jar	MDS Runtime Dependencies
	ADF Model Generic Runtime
	BC4J Runtime
	BC4J Security
	ADF Common Runtime
oracle.javatools_11.1.1/javamodel-rt.jar	Java EE 1.5
	J2EE 1.4
	JAX-RPC Client
oracle.javatools_ 11.1.1/javatools-jndi-local.jar	NA
oracle.javatools_11.1.1/javatools-nodeps.jar	ADF Common Web Runtime
	MDS Runtime Dependencies
	Java EE 1.5
	ADFm Designtime API
	J2EE 1.4
	JAX-RPC Client
	DB Runtime (db-tests)
oracle.javatools_11.1.1/resourcebundle.jar	ADF Desktop Integration Runtime
	Resource Bundle Support
	BC4J Runtime
oracle.mds_11.1.1/mdslcm-client.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.xmlef_11.1.1/xmlef.jar	MDS Runtime Dependencies
	ADF Faces Change Manager Runtime 11
	ADF Model Generic Runtime
	Obsolete JDeveloper Extension SDK
oracle.bali.share_11.1.1/share.jar	MDS Runtime Dependencies
	BC4J Tester
	ADF Model Generic Runtime
	Oracle Help for Java
	Oracle JEWT

A.5 adf.desktopintegration.war Library

Table A–4 lists the JAR files that are packaged into the `adf.desktopintegration.war` file and their corresponding JDeveloper runtime library.

Table A–4 adf.desktopintegration.war Library

JAR	JDeveloper Library
`oracle.adf.desktopintegration_11.1.1/adf-desktop-integration.jar`	ADF Desktop Integration Runtime

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:

- Section B.1, "Overview of Custom wsadmin Commands for Oracle ADF"
- Section B.2, "ADF-Specific WebSphere Commands"

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.

Table B–1 Browse Commands for wsadmin Configuration

Use this command...	To...	Use with wsadmin...
`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
`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)
`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 `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')
```